LDS – Labelled Deductive Systems
Volume 1 — Foundations

Dov M Gabbay

MPI–I–94–223 May 94

Author's Address

Department of Computing
Imperial College of Science, Technology & Medicine
180 Queen's Gate
London SW7 2BZ
Tel: 071-225 8447
E-mail: dg@doc.ic.ac.uk

Publication Notes

This is the third intermediate working partial draft of a book intended for Oxford University Press.

Earlier drafts seem to have attracted great interest among the international community. The 6th Draft was published as a report by CIS, University of Munich in February 1991. *LDS* is used as a framework in three major European Esprit basic research actions and its development is supported by several national projects. I am therefore responding by publishing what I currently have written, *as is*, labelling it the third intermediate version in the form of a technical report. I am happy to include a discussion of algebraiclabelled deductive systems and a prototype semantics for *LDS* in the current version. Many colleagues have been asking what the Fibred semantics for *LDS* looks like.

This version is currently being expanded and adjusted and it is expected that the final draft will be double in size. I am therefore labelling it as Volume 1.

1st Draft - September 1989
2nd Draft - May 1990
3rd Draft - July 1990
4th Draft - August 1990
5th Draft - December 1990
6th (Intermediate) Draft - February 1991
7th (Expanded) Draft - April 1992
8th (2nd Intermediate) Draft - October 1993
9th Expanded Draft - March 1994
10th (3rd Intermediate) Draft - May 1994

Acknowledgements

This research is continuing while the author is a SERC Senior Research Fellow, under grant GR/H01014.

I would like to dedicate this work to my Administrator and friend Mrs Jane Spurr, with respect and appreciation. Against all odds, this wonderful colleague has managed to produce several of my books and research papers in LaTeX, to organise and produce multiple volumes Handbooks and book series and to launch, produce and maintain the publication of an International Journal.

Contents

Preface

This is the third intermediate working partial draft of a book intended for Oxford University Press.

Earlier drafts seem to have attracted great interest among the international community. The 6th Draft was published as a report by CIS, University of Munich in February 1991. *LDS* is used as a framework in three major European Esprit basic research actions and its development is supported by several national projects. I am therefore responding by publishing what I currently have written, *as is*, labelling it the third intermediate version in the form of a technical report. I am happy to include a discussion of algebraiclabelled deductive systems and a prototype semantics for *LDS* in the current version. Many colleagues have been asking what the Fibred semantics for *LDS* looks like.

This version is currently being expanded and adjusted and it is expected that the final draft will be double in size. I am therefore labelling it as Volume 1. Among the shortcomings of the present version are the following:

- Some topics have been further developed than others. This is a bit of a distortion which may create a feeling of lack of thematic flow.
- Some concepts need further refinement and adjustments. For example, label dependent skolemising is done in several contexts; modal logic, metabox environment, substructural logics etc. The principles are the same but the local definitions need to be made more compatible. This takes time, and comments from readers are welcome.
- Chapters 4–6 are less complete than others.
- The references to works by colleagues are not complete, and proper credit to other authors is yet to come.
- The final definition of what a logical system is and of what an *LDS* system is may change slightly, as well as the fine tuning of the fibred semantics.

I hope this third intermediate version will be of value and I would greatly appreciate reader's comments and criticism and any corrections.

To the Reader

It is not easy writing this book. It is not only a matter of doing the research and trying to solve problems and seek the correct formulation of new concepts, it is also the problem of how to communicate and present the material to the reader. The framework of *Labelled Deductive Systems* is of interest to a large variety of readers. On the one extreme there is the pure mathematical logician who likes exact formal definitions and dry theorems, who probably specialises in one logic and methodology. On the other extreme there are also the practical consumers of logic, who like to absorb the intuitions and use labelling as needed to advance the cause of their aplications.

Between the two extremes there are the rest of the 'logic and computation' community. My problem is how to communicate with all these readers in one book.

My compromise was to divide the book into parts and thus allow for different emphasis in the presentation. The purpose of this overview to the reader is to explain what I had done.

The first step in reading this book is to understand its intuitive message. This message is very simple:

Traditional logics manipulate formulas. My message is to manipulate pairs; formulas and labels. The labels annotate the formulas. This sounds very simple but it turned out to be a big step, which makes a serious difference, like the difference between using one hand only or allowing for the coordinated use of two hands. Of course the idea has to be made precise, and its advantages and limitations clearly demonstrated. 'Precise' means a good mathematical definition and 'advantages demonstrated' means case studies and applications in pure logic and in AI.

To achieve that we need to address the following:

1. Define the notion of *LDS*, its proof theory and semantics and relate it to traditional logics.
2. Explain what form the traditional concepts of cut elimination, deduction theorem, negation, inconsistency, update, etc. take in *LDS*.
3. Formulate major known logics in *LDS*. For example, modal and temporal logics, substructural logics, default, nonmonotonic logics, etc.
4. Show new results and solve long-standing problems using *LDS*.
5. Demonstrate practical applications.

This is what I am trying to do in this book. Part I of the book is an intuitive presentation of *LDS* in the context of traditional current views of monotonic and nonmonotonic logics. It is less oriented towards the pure logician and more towards the practical consumer of logic. It has two tasks, addressed in two chapters. These are:

Chapter1: Formally motivate *LDS* by starting from the traditional notion of 'What is a logical system' and slowly adding features to it until it becomes essentially an *LDS*.

Chapter 2: Intuitively motivate *LDS* by showing many examples where labels are used, as well as some case studies of familiar logics (e.g. modal logic) formulated as an *LDS*.

The second part of the book presents the formal theory of *LDS* for the formal logician. I have tried to avoid the style of definition-lemma-theorem and put in some explanations. What is basically needed here is the formulation of the mathematical machinery capable of doing the following.

- Define *LDS* algebra, proof theory and semantics.
- Show how an arbitrary (or fairly general) logic, presented traditionally, say as a Hilbert system or as a Gentzen system, can be turned into an *LDS* formulation.
- Show how to obtain a traditional formulations (e.g. Hilbert) for an arbitrary *LDS* presented logic.

- Define and study major logical concepts intrinsic to *LDS* formalisms.
- Give detailed study of the *LDS* formulation of some major known logics (e.g. modal logics, resource logics) and demonstrate its advantages.
- Translate *LDS* into classical logic (reduce the 'new' to the 'old'), and explain *LDS* in the context of classical logic (two sorted logic, metalevel aspects, etc).

Chapter 3: Give fairly general definitions of some basic concepts of *LDS* theory, mainly to cater for the needs of the practical consumer of logic who may wish to apply it, with a detailed study of the metabox system.

The presentation of Chapter 3 is a bit tricky. It may be too formal for the intuitive reader, but not sufficiently clear and elegant for the mathematical logician. I would be very grateful for comments from the readers for the next draft.

Chapter 4: Presents the basic notions of algebraic *LDS*. The reader may wonder how come we introduce algebraic *LDS* in chapter 3 and then again in chapter 4. Our aim in chapter 3 is to give a general definition and formal machinery for the applied consumer of logic. Chapter 4 on the other hand studies *LDS* as formal logics. It turns out that to formulate an arbitrary logic as an *LDS* one needs some specific labelling algebras and these need to be studied in detail (chapter 4). For general applications it is more convenient to have general labelling algebras and possibly mathematically redundant formulations (chapter 3). In a sense chapter 4 continues the topic of the second section of chapter 3.

Chapter 5: Present the full theory of *LDS* where labels can be databases from possibly another *LDS*. It also presents Fibred Semantics for *LDS*.

Chapter 6: Presents a theory of quantifers for *LDS*. The material for this chapter is still under research.

Chapter 7: Studies structured consequence relations. These are logical system swhere the structure is not described through labels but through some geometry like lists, multisets, trees, etc. Thus the label of a wff A is implicit, given by the place of A in the structure.

Chapter 8: Deals with metalevel features of *LDS* and its translation into two sorted classical logic.

Parts 3 and 4 of the book deals in detail with some specific families of logics. Chapters 9–11 essentailly deal with substructural logics and their variants.

Chapter 9: Studies resource and substructural logics in general.

Chapter 10: Develops detailed proof theory for some systems as well as studying particular features such as negation.

Chapter 11: Deals with many valued logics.

Chapter 12: Studies the Curry Howard formula as type view and how it compres with labelling.

Chapter 13: Deals with modal and temporal logics.

Part 5 of the book deals with *LDS* metatheory.

Chapter 14: Deals with labelled tableaux.

Chapter 15: Deals with combining logics.

Chapter 16: Deals with abduction.

This concludes the current version of Volume 1.

Tentative chapters for Volume 2

Part I

Labelled Deduction in Context

Chapter 1

What is a Logical System?

1.1 Introduction

There is an increasing demand from computer science, linguistics and philosophy for a variety of logical systems. This is prompted by the extensive applications of logic in theoretical computer science, artificial intelligence and logic programming. In these fields there is a growing need for a diversity of semantically meaningful and algoirthmically presented logical systems which can serve various applications. Therefore renewed research activity is being devoted to analysing and tinkering with old and new logics.

This activity has produced a shift in the notion of a logical system. Traditionally a logic was perceived as a 'consequence relation' between *sets* of formulas. Problems arising in application areas have emphasized the need for consequnce relations between structures of formulas (such as multisets, sequences or even richer structures). This finer-tuned approach to the notion of a logical system introduces new problems which call for an improved general framework in which many of the new logics arising from computer science applications can be presented and investigated.

This chapter is a systematic study of the notion of what is a logical system. It will incrementally motivate a notion of logical system through the needs of various applications and applied logical activity, with a view of naturally leading to the notion of *Labelled Deductive Systems*, the topic of this book. We begin with the traditional concept of a logical system as a consequence relation between sets of formulas and slowly end up with the notion of a logical system as a *Labelled Deductive System.* Let us consider the initial position of a logical system as a consequence relation on sets of formulas. In this traditional view, any set theoretical binary relation of the form $\Delta \mid\!\sim \Gamma$ satisfying certain conditions (reflexivity, monotonicity and cut) is a logical system. Such a relation has to be mathematically presented. This can be done either semantically, or set theoretically or it can be algorithmically generated. There are several options for the latter. Generate first the $\{A \mid \varnothing \mid\!\sim A\}$ as a Hilbert system and then generate $\{(\Delta, \Gamma) \mid \Delta \mid\!\sim \Gamma\}$ or generate the pairs (Δ, Γ) directly (via Gentzen rules) or use any other means (other proof theories)?

The concepts of a *logical system, semantics* and *proof theory* are not sharp enough even in the traditional literature. There are no clear definitions of what is a proof theoretic formulation of a logic (as opposed to, e.g. a decision procedure algorithm) and what is e.g. a Gentzen formulation. Let us try here to propose a working definition, only for the purpose of making the reader a bit more comfortable and not necessarily for the purpose of giving a definitive formulation of these concepts.

- We start with the notion of a well formed formula of the language **L** of the logic.
- A consequence relation is a binary relation on finite sets of formulas, Δ, Γ written as $\Delta \mid\!\sim \Gamma$, satisfying certain conditions, namely reflexivity, monotonicity and cut.
- Such a relation can be defined in many ways. For example, one can list all pairs (Δ, Γ) such that $\Delta \mid\!\sim \Gamma$ should hold. Another way is to give Δ, Γ to some computer program and wait for an answer (which should always come).

- A semantics is an interpretation of the language **L** into some family of set theoretical structures, together with an interpretation of the consequence relation $\mid\!\sim$ in terms of the interpretation. What I have just said is not clear in itself because I have not explained what 'structures' are and what an interpretation is. Indeed, there is no clear definition of what is a semantics. In my book [Gabbay, 1981], following Scott I defined a *model* as a function **s** giving each wff of the language a value in {0,1}. A semantics $\mathcal{S}$ is a set of models, and $\Delta \mid\!\sim_{\mathcal{S}} \Gamma$ is defined as the condition:

$$(\forall \mathbf{s} \in \mathcal{S})[\forall X \in \Delta(\mathbf{s}(X) = 1) \to \exists Y \in \Gamma(\mathbf{s}(Y) = 1)]$$

- There can be algorithmic systems for generating $\mid\!\sim$. Such systems are not to be considered 'proof theoretical systems' for $\mid\!\sim$. They could be decision procedures or just optimal theorem proving machines.

- the notion of a proof system is not well defined in the literature. There are some recognised methodologies such as 'Gentzen formulations', 'tableaux', 'Hilbert style' but these are not sharply defined. For our purpose, let us agree that a *proof system* is any algorithmic system for generating $\mid\!\sim$ using rules of the form:

$$\frac{\Delta_1 \mid\!\sim \Gamma_1; \ldots; \Delta_n \mid\!\sim \Gamma_n}{\Delta \mid\!\sim \Gamma}$$

  and 'axioms' of the form:

$$\frac{\varnothing}{\Delta \mid\!\sim \Gamma}$$

  The axioms are initial list of $(\Delta, \Gamma) \in \mid\!\sim$ and the other rules generate more. So a proof system is a particular way of generating $\mid\!\sim$. Note that there need not be structural requirement on the rule (that each involves a main connective and some subformulas, etc.).
  A Hilbert formulation is a proof system where all the Δ's involved are $\varnothing$. A Gentzen formulation would be a proof system where the rules are very nicely structured (try to define something reasonable yourself, again, there is no clear definition!) A Gentzen system can be viewed as a higher level Hilbert system for the 'connective' '$\mid\!\sim$'.

  A tableaux formulation is a syntactical countermodel construction relative to some semantics. We have $\Delta \mid\!\sim \Gamma$ if the countermodel construction is 'closed', i.e. must always fail. It is also possible to present tableaux formulations for logics which have no semantics if the consequence $\mid\!\sim$ and the connectives satisfy some conditions.

The central role which proof theoretical methodologies play in generating logics compells us to put forward the view that a logical system is a pair $(\mid\!\sim, \mathbf{S}_{\mid\sim})$, where $\mathbf{S}_{\mid\sim}$ is a proof theory for $\mid\!\sim$. In other words, we are saying that it is not enough to know $\mid\!\sim$ to 'understand' the logic, but we must also know how it is presented (i.e. $\mathbf{S}_{\mid\sim}$).

The next shift in our concept of a logic is when we observed from application areas whose knowledge representation involves data and assumptions the need to add structure to the assumptions and the fact that the reasoning involved relies on and uses the structure. This view also includes non-monotonic systems. This led us to develop the topic of this book, namely the notion of *Labelled Deductive Systems* and adopt the view that this is the framework for presenting logics. Whether we accept these new systems as logics or not, we must have a general framework able to represent them.

The real departure from traditional logics (as opposed to just giving them more structure) comes with the notion of aggregating arguments. Real human reasoning does aggregate arguments (circumstantial evidence in favour of A as opposed to evidence for $\neg A$) and what is known as quantitative (fuzzy) reasoning systems make heavy use of that. Fortunately *LDS* can handle that easily. The section concludes with the view that a proper practical reasoning system has 'mechanisms' for updates, inputs, abduction, actions, etc. as well as databases (theories, assumptions) and that a proper logic is an integrated *LDS* system together with a specific choice of such mechanisms.[1]

[1]My personal view is that this is a logic, i.e. Logic = *LDS* system + several *mechanisms*. In AI circles this might

1.2 Logical systems as consequence relations

Traditionally, to present a logic **L**, we need to present first the set of well formed formulas of that logic. This is the *language* of the logic. We define the sets of atomic formulas, connectives, quantifiers and the set of arbitrary formulas. Secondly we mathematically define the notion of consequence, namely for a given set of formulas Δ and a given formula Q, we define the consequence relation $\Delta \mathrel{|\!\sim}_{\mathbf{L}} Q$, reading '$Q$ follows from Δ in the logic **L**'.

The consequence relation is required to satisfy the following intuitive properties: (Δ, Δ' abbreviates $\Delta \cup \Delta'$).

Reflexivity

$$\Delta \mathrel{|\!\sim} Q \text{ if } Q \in \Delta$$

Monotonicity

$$\Delta \mathrel{|\!\sim} Q \text{ implies } \Delta, \Delta' \mathrel{|\!\sim} Q$$

Transitivity (Cut)[2]:

$$\Delta \mathrel{|\!\sim} A;\ \ \Delta, A \mathrel{|\!\sim} Q \text{ imply } \Delta \mathrel{|\!\sim} Q$$

The consequence relation may be defined in various ways. Either through an algorithmic system $\mathbf{S}_{|\!\sim}$,or implicitly by postulates on the properties of $\mathrel{|\!\sim}$.

Thus a logic is obtained by specifying **L** and $\mathrel{|\!\sim}$. Two algorithmic systems $\mathbf{S}_1$ and $\mathbf{S}_2$ which give rise to the same $\mathrel{|\!\sim}$ are considered the same logic.

If you think of Δ as a database and Q as a query, then reflexivity means that the answer is yes to any Q which is officially listed in the database. Monotonicity reflects the accumulation of data, and transitivity is nothing but lemma generation, namely if $\Delta \mathrel{|\!\sim} A$, then A can be used as a lemma to obtain B from Δ.

The above properties seemed minimal and most natural for a logical system to have, given that the main applications of logic were in mathematics and philosophy.

be called *an agent*. Unfortunately, the traditional logic community are still very conservative in the sense that they have not even accepted non-monotonic reasoning systems as logics yet. They believe that all this excitement is transient, temporarily generated by computer science and that it will fizzle out sooner or later. They believe that we will soon be back to the old research problems, such as how many non-isomorphic models does a theory have in some inaccessible cardinal or what is the ordinal of yet another subsystem of analysis. I think this is fine for mathematical logic but not for the logic of human reasoning. There is no conflict here between the new and the old, just further evolution of the subject.

[2]There are several versions of the **Cut Rule** in the literature, they are all equivalent for the cases of classical and intuitionistic logic but are not equivalent in the context of this section. The version in the main text we call **Transitivity** (**Lemma Generation**). Another version is:

$$\frac{\Gamma \mathrel{|\!\sim} A; \quad \Delta, A \mathrel{|\!\sim} B}{\Delta, \Gamma \mathrel{|\!\sim} B}.$$

This version implies **monotonicity**, when added to **Reflexivity**.
Another version we call **Internal Cut**:

$$\frac{\Delta, A \mathrel{|\!\sim} \Gamma; \quad \Delta \mathrel{|\!\sim} A, \Gamma}{\Delta \mathrel{|\!\sim} \Gamma}.$$

A more restricted version of cut is **Unitary Cut**:

$$\frac{\Delta \mathrel{|\!\sim} A; A \mathrel{|\!\sim} Q}{\Delta \mathrel{|\!\sim} Q}$$

The above notion was essentially put forward by [Tarski, 1956] and is referenced to as Tarski consequence. [Scott, 1974], following [Gabbay, 1969], generalised the notion to allow Q to be a set of formulas Γ. The basic relation is then of the form $\Delta \mid\!\sim \Gamma$, satisfying:

Reflexivity

$$\Delta \mid\!\sim \Gamma \text{ if } \Delta \cap \Gamma \neq \varnothing$$

Monotonicity

$$\frac{\Delta \mid\!\sim \Gamma}{\Delta, \Delta' \mid\!\sim \Gamma}$$

Transitivity (cut)

$$\frac{\Delta, A \mid\!\sim \Gamma; \Delta' \mid\!\sim A, \Gamma'}{\Delta', \Delta \mid\!\sim \Gamma, \Gamma'}$$

Scott has shown that for any Tarski consequence relation there exists two Scott consequence relations (a maximal one and a minimal one) that agree with it (see my book [Gabbay, 1981]).

The above notions are monotonic. However, the increasing use of logic in artificial intelligence has given rise to logical systems which are not monotonic. The axiom of monotonicity is not satisfied in these systems. There are many such systems, satisfying a variety of conditions, presented in a variety of ways. Furthermore, some are proof theoretical and some are model theoretical. All these different presentations give rise to some notion of consequence $\Delta \mid\!\sim Q$, but they only seem to all agree on some form of restricted reflexivity ($A \mid\!\sim A$). The essential difference between these logics (commonly called *non-monotonic logics*) and the more traditional logics (now referred to as *monotonic logics*) is the fact that $\Delta \mid\!\sim A$ holds in the monotonic case because of some $\Delta_A \subseteq \Delta$, while in the non monotonic case the entire set Δ is used to derive A. Thus if Δ is increased to Δ', there is no change in the monotonic case, while there may be a change in the non monotonic case.

The above describes the situation current in the early 1980's. We have had a multitude of systems generally accepted as 'logics' without a unifying underlying theory and many had semantics without proof theory. Many had proof theory without semantics, though almost all of them were based on some sound intuitions of one form or another. Clearly there was the need for a general unifying framework. An early attempt at classifying non-monotonic systems was [Gabbay, 1985]. It was put forward that basic axioms for a consequence relation should be *reflexivity, transitivity (cut)* and *restricted monotonicity*, namely:

Restricted Monotonicity

$$\frac{\Delta \mid\!\sim A; \Delta \mid\!\sim B}{\Delta, A \mid\!\sim B}$$

A variety of systems seem to satisfy this axiom. Further results were obtained [Makinson, 1989, Makinson, 1993, Wójcicki, to appear, Wójcicki, 1989, Kraus *et al.*, 1990, Lehmann and Magidor, 1992] and the area was called 'axiomatic theory of the consequence relation' by Wojcicki.[3]

Although some classification was obtained and semantical results were proved, the approach does not seem to be strong enough. Many systems do not satisfy restricted monotonicity. Other systems such as relevance logic, do not satisfy even reflexivity. Others have richness of their own which is lost in a simple presentation as an axiomatic consequence relation. Obviously a different approach is needed, one which would be more sensitive to the variety of features of the systems in the field. Fortunately, developments in a neighbouring area, that of automated deduction, seem to give us a clue.

[3]In general, the exact formulations of transitivity and reflexivity can force some form of monotonicity.

1.3 Logical systems as algorithmic proof systems

The relative importance of automated deduction is on the increase, in view of its wide applicability. New automated deduction methods have been developed for non-classical logics, and resolution has been generalised and modified to be applicable to these logics. In general, because of the value of these logics in theoretical computer science and artificial intelligence, a greater awareness of the computational aspects of logical systems is developing and more attention being devoted to proof theoretical presentations. It became apparent to us that a key feature in the proof theoretic study of these logics is that a slight natural variation in an automated or proof theoretic system of one logic (say $\mathbf{L}_1$), can yield another logic (say $\mathbf{L}_2$).

Although $\mathbf{L}_1$ and $\mathbf{L}_2$ may be conceptually far apart (in their philosophical motivation, and mathematical definitions) when it comes to automated techniques and proof theoretical presentation, they turn out to be brother and sister. This kind of relationship is not isolated and seems to be widespread. Furthermore, non monotonic systems seem to be obtainable from monotonic ones through variations on some of their monotonic proof theoretical formulation.

This seems to give us some handle on classifying non-monotonic systems.

This phenomena has prompted us to put forward the view that a logical system $\mathbf{L}$ is not just the traditional consequence relation $\vdash$ (monotonic or non monotonic) but a pair $(\mid\!\sim, \mathbf{S}_{\mid\sim})$, where $\mid\!\sim$ is a mathematically defined consequence relation (i.e. the set of pairs (Δ, Γ) such that $\Delta \mid\!\sim \Gamma$) satisfying whatever minimal conditions on a consequence one happens to agree to, and $\mathbf{S}_{\mid\sim}$ is an algorithmic system for generating all those pairs [Gabbay, 1992a]. Thus according to this definition classical propositional logic $\mid\!\sim$ perceived as a set of tautologies together with a Gentzen system $\mathbf{S}_{\mid\sim}$ is not the same as classical logic together with the two valued truth table decision procedure $\mathbf{T}_{\mid\sim}$ for it. In our conceptual framework,($\mid\!\sim, \mathbf{S}_{\mid\sim})$ is *not the same logic* as $(\mid\!\sim, \mathbf{T}_{\mid\sim})$.

To illustrate and motivate our way of thinking, observe that it is very easy to move from $\mathbf{T}_{\mid\sim}$ for classical logic to a truth table system $\mathbf{T}^n_{\mid\sim}$ for Łukasiewicz n-valued logic. It is not so easy to move to an algorithmic system for intuitionistic logic. In comparison, for a Gentzen system presentation, exactly the opposite is true. Intuitionistic and classical logics are neighbours, while Łukasiewicz logics seem completely different. In fact for a Hilbert style or Gentzen style formulation, one can show proof theoretic similarities between Łukasiewicz's infinite valued logic and Girard's Linear Logic, which in turn is proof theoretically similar to intuitionistic logic.

This issue has a bearing on the notion of 'what is classical logic'. Given an algorithmic proof system $\mathbf{S}_{\mid\sim_c}$ for classical logic $\mid\!\sim_c$, then $(\mid\!\sim_c, \mathbf{S}_{\mid\sim_c})$ is certainly classical logic. Now suppose we change $\mathbf{S}_{\mid\sim_c}$ a bit by adding heuristics to obtain $\mathbf{S}'$. The heuristics and modifications are needed to support an application area. Can we still say that we are essentially in 'classical logic? I suppose we can because $\mathbf{S}'$ is just a slight modification of $\mathbf{S}_{\mid\sim_c}$. However, slight modifications of an algorithmic system may yield another well-known logic. In fact $\mathbf{S}'$ may be linear logic. So is linear logic essentially classical logic, slightly modified, or vice versae!

We give an example from goal directed implicational logic. Consider a language with implication only. It is easy to see that all wffs A have the form $A_1 \to (A_2 \to \ldots \to (A_n \to q)\ldots)$, q atomic, where A_i has the same form as A. We now describe a computation with database a multiset Δ of wffs of the above form and the goal a wff of the above form. We use the metapredicate $\Delta \vdash A$ to mean the computation succeeds; i.e. A follows from Δ. Here are the rules:

1. $\Delta, q \vdash q$, q atomic and Δ empty. (Note that we are not writing $A \vdash A$ for arbitrary A. We are not writing a Gentzen system).

2. $\Delta \vdash A_1 \to (A_2 \to \ldots \to (A_n \to q)\ldots)$ if $\Delta \cup \{A_1, \ldots, A_n\} \vdash q$. Remember we are dealing with multisets.

3. $\Delta' = \Delta \cup \{A_1 \to (A_2 \to \ldots (A_n \to q)\ldots)\} \vdash q$ if $\Delta = \Delta_1 \cup \Delta_2 \cup \ldots \cup \Delta_n$, $\Delta_i, i = 1, \ldots, n$ are pairwise disjoint and $\Delta_i \vdash A_i$.

The above computation characterises linear implication. If we relinquish the side condition in (3) and let $\Delta_i = \Delta'$ and the side condition (1) that Δ is empty, we get intuitionistic implication.

The difference in logics is serious. In terms of proof methodologies, the difference is minor. More examples in [Gabbay, 1992a, Olivetti and Gabbay, 1993].

1.4 Logical systems as algorithmic structured consequence relations

Further observation of field examples shows that in many cases the database is not just a set of formulas but a structured set of formulas. The most common is a list or multiset.[4] Such structures appear already in linear and concatenation logics and in many non-monotonic systems such as priority and inheritance systems. In many algorithmically presented systems much use is made (either explicitly or implicitly) of this additional structure.

A very common example is a Horn clause program. The list of clauses

$$\begin{array}{ll} (a_1) & q \\ (a_2) & q \to q \end{array}$$

does not behave in the same way as the list

$$\begin{array}{ll} (b_1) & q \to q \\ (b_2) & q \end{array}$$

The query $?q$ succeeds from one and loops from the other.

It is necessary to formulate axioms and notions of consequence relations for structures. This is studied in detail in a later chapter. Here are the main features:

- Databases (Assumptions) are structured. They are not just sets of formulas but have a more general structure such as multisets, lists, partially ordered sets, etc. To present a database formally, we need to describe the structures. Let $\mathcal{M}$ be a class of structures (e.g. all finite trees). Then a database Δ has the form $\Delta = (M, \mathbf{f})$, where $M \in \mathcal{M}$ and $\mathbf{f} : M \mapsto$ wffs, such that for each $t \in M, \mathbf{f}(t)$ is a formula. We assume the one point structure $\{t\}$ is always in $\mathcal{M}$. We also assume we know how to take any single point $t \in M$ out of M and obtain $(M', \mathbf{f}'), \mathbf{f}' = \mathbf{f} \upharpoonright M$. This we need for some versions of the cut rule and the deduction theorem.

- A structured-consequence relation $\mid\!\sim$ is a relation $\Delta \mid\!\sim A$ between structured databases Δ and formulas A. (We will not deal with structured consequence relations between two structured databases $\Delta \mid\!\sim \Gamma$ here. More details are in the chapter on structured consequence relation.

- $\mid\!\sim$ must satisfy the minimal conditions, namely

  Identity

$$\{A\} \mid\!\sim A$$

  Surgical Cut

$$\frac{\Delta \mid\!\sim A; \Gamma[A] \mid\!\sim B}{\Gamma[\Delta] \mid\!\sim B}$$

  where $\Gamma[A]$ means that A resides somewhere in the structure Γ and $\Gamma[\Delta]$ means that Δ replaces A in the structure. These concepts have to be defined precisely. If $\Delta = (M_1, \mathbf{f}_1)$ and $\Gamma = (M_2, \mathbf{f}_2)$ then $\Gamma[A]$ displays the fact that for some $t \in M_2$, $\mathbf{f}_2(t) = A$. We allow for the case that $M_2 = \mathbf{f}_2 = \varnothing$ (i.e. taking A out) We need a notion of substitution, which is a

[4]Classical logic cannot make these distinctions using conjunction only. It needs further annotation or use of predicates.

three place function $\mathbf{Sub}(\Gamma, \Delta, t)$, meaning that for $t \in M_2$ we substitute M_1 in place of t. This gives us a structure $(M_3, \mathbf{f}_3)$ according to the definition of **Sub**. $(M_3, \mathbf{f}_3)$ is displayed as $\Gamma[\Delta]$, and $\Gamma[\varnothing]$ displays the case of taking A out.
Many non monotonic systems satisfy a more restricted version of surgical cut:

$$\frac{\Gamma[\varnothing/A] \mid\!\sim A; \Gamma[A] \mid\!\sim B}{\Gamma[\Gamma[\varnothing/A]] \mid\!\sim B}$$

Another variant would be

Deletional Cut

$$\frac{\Gamma[\varnothing/A] \mid\!\sim A; \Gamma[A] \mid\!\sim B}{\Gamma[\varnothing/A] \mid\!\sim B}$$

.

- A logical system is a pair $(\mid\!\sim, \mathbf{S}_{\mid\sim})$, where $\mid\!\sim$ is a structured-consequence and $\mathbf{S}_{\mid\sim}$ is an algorithmic system for it.

Of course we continue to maintain our view that different algorithmic systems for the same structured consequence relation define different logics. Still although we now have a fairly general concept of a logic, we do not have a general framework. Monotonic and non-monotonic systems still seem conceptually different. There are many diverse examples among temporal logics, modal logics, defeasible logics and more. Obviously, there is a need for a more unifying framework. The question is, can we adopt a concept of a logic where the passage from one logic to another is natural, and along predefined acceptable modes of variation? Can we put forward a framework where the computational aspects of a logic also play a role? Is it possible to find a common home for a variety of seemingly different techniques introduced for different purposes in seemingly different intellectual logical traditions?

1.5 Logical systems as labelled deductive systems

To find an answer, let us ask ourselves what makes one logic different from another? How is a new logic presented and described and compared to another? The answer is obvious. These considerations are performed in the metalevel. Most logics are based on modus ponens anyway. The quantifier rules are formally the same anyway and the differences between them are metalevel considerations on the proof theory or semantics. If we can find a mode of presentation of logical systems where metalevel features can reside side by side with object level features then we can hope for a general framework. We must be careful here. In the logical community the notions of object-level vs metalevel are not so clear. Most people think of *naming* and *proof predicates* in this connection. This is not what we mean by metalevel here. We need a more refined understanding of the concept. There is a similar need in computer science.

We found that the best framework to put forward is that of a *Labelled Deductive System, LDS*, the topic of this book. Our notion of what is a logic is that of a pair $(\mid\!\sim, \mathbf{S}_{\mid\sim})$ where $\mid\!\sim$ is a structured (possibly non-monotonic) consequence relation on a language $\mathbf{L}$ and $\mathbf{S}_{\mid\sim}$ is an *LDS*, and where $\mid\!\sim$ is essentially required to satisfy no more than *Identity* (i.e. $\{A\} \mid\!\sim A$) and a version of *Cut*. This is a refinement of our concept of a logical system presented of the previous sections. We now not only say that a logical system is a pair $(\mid\!\sim, \mathbf{S}_{\mid\sim})$, but we are adding that $\mathbf{S}_{\mid\sim}$ itself has a special presentation, that of an *LDS*.

As a first approximation, we can say that an *LDS* proof system is a triple $(\mathcal{A}, \mathbf{L}, \mathbf{M})$, where $\mathbf{L}$ is a logical language (connectives and wffs) and $\mathcal{A}$ is an algebra (with some operations) of labels and $\mathbf{M}$ is a discipline of labelling formulas of the logic (from the algebra of labels $\mathcal{A}$), together with deduction rules and with agreed ways of propagating the labels via the application of the deduction rules. The way the rules are used is more or less uniform to all systems.

To present an *LDS* system we need first to define its set of formulas and its set of labels.

For example, we can take the language of classical logic as the formulas (with variables, constants and quantifiers) and take some set of function symbols on the same variables and constants as generating the labels. More precisely, we allow ordinary formulas of predicate logic with quantifiers to be our *LDS* formulas. Thus $\exists x A(x, y)$ is a formula with free variable y and bound variable x. To generate the labels, we start with a new set of function symbols $t_1(y), t_2(x, y), \ldots$ of various arities which can be applied to the *same* variables which appear in the formulas. Thus the labels and formulas can share variables, or even some constants and function symbols. In other words in some applications it might be useful to allow some labels to appear inside formulas A. We can form declarative units of the form $t_1(y) : \exists x A(x, y)$. When y is assigned a value $y = a$, so does the label and we get $t_1(a) : \exists x A(x, a)$. The labels should be viewed as more information about the formulas, which is not coded inside the formula, (hence dependence of the labels on variables x makes sense as the extra information may be different for different x). A formal definition of an algebraic *LDS* system will be given later, meanwhile, let us give an informal definition of an *LDS* system and some examples which help us understand what and why we would want labels.[5]

Definition 1.5.1 (Prototype Algebraic*LDS* System) *Let $\mathcal{A}$ be a first order language of the form $\mathcal{A} = (A, R_1, \ldots, R_k, f_1, \ldots, f_m)$ where A is the set of terms of the algebra (individual variables and constants) and R_i are predicate symbols (on A, possibly binary but not necessarily so) and $f_1, \ldots, f_m$ are function symbols (on A) of various arities. We think of the elements of A as atomic labels and of the functions as generating more labels and of the predicates as giving additional structure to the labels. A typical example would be (A, R, f_1, f_2) where R is binary and f_1, f_2 are unary.*

A diagram *of labels is a set D containing elements generated from A by the function symbols together with formulas of the form $\pm R(t_1, \ldots, t_k)$, where $t_i \in D$ and R is a predicate symbol of the algebra.*

Let $\mathbf{L}$ be a predicate language with connectives $\sharp_1, \ldots, \sharp_n$, of various arities, with quantifiers and with the same *set of atomic terms A as the algebra.*

We define the notions of a declarative unit, a database and a label as follows:

1. *An atomic label is any $t \in A$. A label is any term generated from the atomic labels by the symbols $f_1, \ldots, f_m$.*

2. *A formula is any formula of $\mathbf{L}$.*

3. *A declarative unit is a pair $t : A$, where t is a label and A is a formula.*

4. *A database is either a declarative unit or has the form $(D, \mathbf{f}, d, U)$, where D is a finite diagram of labels, $d \in D$ is the distinguished label, and $\mathbf{f}$ is a function associating with each label t in D either a database or a finite set of formulas and U is the set of all terms. (Note that*

[5]The idea of annotating formulas for various purposes is not new. A R Anderson and N. Belnap in their book on Entailment, label formulas and propagate lables during proofs to keep track of relevance of assumptions. Term annotations (Curry–Howard formula as type approach) are also known where the propagation rules are functional application. The Lambek Calculus and the categorial approach is also related to labelling. The extra arguments sometimes present in the Demo predicate of metalogic programming are also a form of labelling. What is new is that we are proposing that we use an arbitrary algebra for the labels and consider the labelling as part of the logic. We are creating a discipline of *LDS* and claiming that we have a unifying framework for logics and that almost any logic can be given an *LDS* formulation. We can give $\mid\!\sim$ an *LDS* formulation provided $\mid\!\sim$ is reflexive and transitive and each connective is either $\mid\!\sim$ monotonic or antimonotonic in each of its arguments. See the chapter on structured consequence relation. We are claiming that the notion of a logic is an *LDS*. This is not the same as the occasional use of labelling with some specific purpose in mind. We are translating and investigating systematically all the traditional logical concepts into the context of *LDS* and generalising them.

I am reminded of the story of the Yuppy who hired an interior decorator to redesign his sitting room. After much study, the decorator recommended that the Yuppy needed a feeling of space and so the best thing to do is to arrange the furniture against the wall, so that there will be a lot of space in the middle. The cleaning lady, when she first saw the new design was very pleased. She thought it was an excellent idea. 'Yes', said the Yuppy, 'and I paid £1000 for it'. 'That was stupid', exclaimed the cleaning lady, 'I could have told you for free! I arrange the furniture this way every time I clean the floor!'.

Of course she is right, but she used the idea of the new arrangement only as a side effect!

this is a recursive clause. We get simple databases if we allow **f** *to associate with each label t only single or finite sets of formulas. Simple databases are adequate for a large number of applications.)*

Definition 1.5.1 is simplified. To understand it intuitively, think of the atomic labels as atomic places and times (top of the hill, Jan 1st 1992, etc.) and the function symbols as generating more labels, namely more times and more places (*behind*(x), *day after*(t) etc.). We form declarative units by taking labels and attaching formulas to them. Complex structures $(D, \mathbf{f}, d, U)$ of these units are databases. This definition can be made more complex. Here the labels are terms generated by function symbols from atomic labels. We can complicate matters by using databases themselves as labels. This will give us recursively more complex, richer labels. We will not go into that now. The first simplification is therefore that we are not using databases as labels. The second simplification is that we assume constant domains. All times and places have the same elements (population) in them. If this were not the case we would need a function U_t giving for each $t \in D$ a set of terms, being the elements residing in t, and a database would have the form $(D, \mathbf{f}, d, U_t)$. Chapter 3 continues with the formal definition of *LDS*.

Example 1.5.2 Consider a language with the predicate *VS900*(x,t). This is a two sorted predicate, denoting Virgin airline flight London-Tokyo, where t is the flight date and x is a name of an individual. For example *VS*900 (Dov, 15.11.91) may be put in the database, denoting that Dov is booked on this flight scheduled to embark on 15.11.91.

If the airline practices overbooking and cancellation procedures (whatever that means), it might wish to annotate the entries by further useful information such as

- Time of booking;
- Individual/group travel booking;
- Type of ticket
- $\pm$ VIP.

This information may be of a different nature to that coded in the main predicate and it is therefore more convenient to keep it as annotation, or label. It may also be the case that the manipulation of the extra information is of a different nature to that of the predicate.

In general, there may be many uses for the label t in the declarative unit $t : A$. Here is a partial list

- Fuzzy reliability value:
  (a number $x, 0 \leq x \leq 1$.) Used mainly in expert systems.
- Origin of A:
  t indicates where the input A came from. Very useful in complex databases.
- Priority of A:
  t can be a date of entry of updates and a later date (label) means a higher priority.
- Time when A holds:
  (temporal logic)
- Possible world where A holds:
  (modal logic)
- t indicates the proof of A:
  (which assumptions were used in deriving A and the history of the proof). This is a useful labelling for Truth Maintenance Systems.
- t can be the situation and A the infon (of situation semantics).

Example 1.5.3 *Let us look at one particular example, connected with modal logic. Assume the algebra $\mathcal{A}$ has the form $(A, <)$, with a set of atomic labels A, no function symbols and a binary relation $<$. According to the previous definition, a diagram of labels, would contain a (finite) set $D \subseteq A$, together with a set of pairs of the form $\{t < s\}$, $t,\ s, \in D$. A database has the form $(D, \mathbf{f}, a)$, where D is a finite diagram and $\mathbf{f}$ is a function, say giving a formula $A_t = \mathbf{f}(t)$, for each $t \in D$.*

The perceptive reader may feel resistence to the idea of the label at this stage. First be assured that you are not asked to give up your favourite logic or proof theory nor is there any hint of a claim that your activity is now obsolete. In mathematics a good concept can rarely be seen or studied from one point of view only and it is a sign of strength to have several views connecting different concepts. So the traditional logical views are as valid as ever and add strength to the new point of view. In fact, manifestations of our *LDS* approach already exist in the literature in various forms, they were locally regarded as convenient tools and there was not the realisation that there is a general framework to be studied and developed. None of us is working in a vacuum and we build on each others work. Further, the existence of a general framework in which any particular case can be represented does not necessarily mean that the best way to treat that particular case is within the general framework. Thus if some modal logics can be formulated in *LDS*, this does not mean that in practice we should replace existing ways of treating the logics by their *LDS* formulation. The latter may not be the most efficient for those particular logics. It is sufficient to show how the *LDS* principles specialise and manifest themselves in the given known practical formulation of the logic.

The reader may further have doubts about the use of labels from the computational point of view. What do we mean by a unifying framework? Surely a Turing machine can simulate any logic. Is that a unifying framework? The use of labels is powerful, as we know from computer science. Are we using labels to play the role of a Turing machine? The answer to the question is twofold. First that we are not operating at the metalevel, but at the object level, (see point 4 below). Second, there are severe restrictions on the way we use *LDS*. Here is a preview:

1. The only rules of inference allowed are the traditional ones, modus ponens and some form of introduction rule (deduction theorem) for implication, for example.

2. Allowable modes of label propagation are fixed for all logics. They can be adjusted in agreed ways to obtain variations but in general the format is the same. For example, it has the following form for implications:
   Let $LDS_{\twoheadrightarrow}$ be a particular *LDS* system with labels $\mathcal{A}$, and with $\twoheadrightarrow$ a special implication characteristic to this particular *LDS* system. Then there exists a fixed set of labels Γ, which can characterise $\twoheadrightarrow$ as follows. For any theory Δ (of labelled wffs) we have:
   Δ proves $(A \twoheadrightarrow B)$ with label t iff $\forall x \in \Gamma$ [B can be proved from Δ and $x : A$ with label $t \otimes x$],
   where Γ is a set of labels characterising the implication in that particular logic. For example Γ may be restricted to atomic labels only, or to labels related to t, or some other restrictions. The freedom that different logics have is in the choice of Γ and the (possibly not only equational) properties of '$\otimes$'. For example we can restrict the use of modus ponens by a wise propagation of labels.

3. The quantifier rules are the same for all logics.

4. Metalevel features are implemented via the labelling mechanism, which is object language.

The reader who prefers to remain within the traditional point of view of:

assumptions (data) proving a conclusion

can view the labelled formulas as another form of data.

There are many occasions when it is most intuitive to present an item of data in the form $t : A$, where t is a label and A is a formula. The common underlying reason for the use of the label t is that t represents information which is needed to modify A or to supplement (the information in) A which is not of the same type or nature as (the information represented by) A itself. A is a logical formula representing information declaratively, and the additional information of t can certainly be added declaratively to A to form A', however, we may find it convenient to put forward the additional information through the label t as part of a pair $t : A$.

Take for example a source of information which is not reliable. A natural way of representing an item of information from that source is $t : A$, where A is a declarative presentation of the information itself and t is a number representing its reliability. Such expert systems exist (eg Mycin) with rules which manipulate both t and A as one unit, propagating the reliability values t through applications of modus ponens. We may also use a label naming the source of information and this would give us a qualitative idea of its reliability.

Another area where it is natural to use labels is in reasoning from data and rules. If we want to keep track, for reasons of maintaining consistency and/or integrity constraints, where and how a formula was deduced, we use a label t. In this case, the label in $t : A$ can be the part of the data which was used to get A. Formally in this case t is a formula, the conjunction of the data used. We thus get pairs of the form $\Delta_i : A_i$, where A_i are formulas and Δ_i are the parts of the database from which A_i was derived.

A third example where it is natural to use labels is time stamping of data. Where data is constantly revised and updated, it is important to time stamp the data items. Thus the data items would look like $t_i : A_i$, where t_i are time stamps. A_i itself may be a temporal formula. Thus there are two times involved, the logical time s_i in $A_i(s_i)$ and the time stamping t_i of A_i. For reasons of clarity, we may wish to regard t_i as a label rather than incorporate it into the logic (by writing for example $A^*(t_i, s_i)$).

To summarise then, we replace the traditional notion of consequence between formulas of the form $A_1, \ldots, A_n \vdash B$ by the notion of consequence between labelled formulas

$$t_1 : A_1; t_2 : A_2, \ldots; t_n : A_n \vdash s : B$$

Depending on the logical system involved, the intuitive meaning of the labels varies. In querying databases, we may be interested in labelling the assumptions so that when we get an answer to a query, we can record, via the label of the answer, from which part of the database the answer was obtained. Another area where labelling is used is temporal logic. We can time stamp assumptions as to when they are true and query, given those assumptions, whether a certain conclusion will be true at a certain time. Thus the consequence notion for labelled deduction is essentially the same as that of any logic: given assumptions does a conclusion follow.

Whereas in the traditional logical system the consequence is defined using proof rules on the formulas, in the *LDS* methodology the consequence is defined by using rules on both formulas and their labels. Formally we have formal rules for manipulating labels and this allows for more scope in decomposing the various features of the consequence relation. The meta features can be reflected in the algebra or logic of the labels and the object features can be reflected in the rules of the formulas. Recall, however, that there are severe restrictions on how we use *LDS* rules, as we discussed earlier.

The notion of a database or of a 'set of assumptions' also has to be changed. A database is a hierarchical configuration of labelled formulas. The configuration depends on the labelling discipline. For example, it can be a linearly ordered set $\{a_1 : A_1, \ldots, a_n : A_n\}, a_1 < a_2 < \ldots < a_n$. The proof discipline for the logic will specify how the assumptions are to be used. See for example the logic programming case study.

We summarise our current position on what is a logical system. A logical system is a pair $(\mid\!\sim, LDS_{\mid\sim})$, where $\mid\!\sim$ is a consequence relation between labelled databases Δ and declarative units $t : A$ and $LDS_{\mid\sim}$ is an algorithmic system for $\mid\!\sim$.

We need one more component to the notion of a logical system. In previous subsections, a logical system was presented as $(\mid\!\sim, \mathbf{S}_{\mid\sim})$, where $\mid\!\sim$ is a structured consequence relation satisfying *Identity* and *Surgical Cut* and $\mathbf{S}_{\mid\sim}$ is an algorithmic proof system for computing $\mid\!\sim$. We are now saying that we need to refine this notion and deal with *Labelled Deductive Systems*, where $\mid\!\sim$ is a consequence relation between labelled databases Δ and declarative units $t : A$ and that $\mathbf{S}_{\mid\sim}$ is replaced by some specific *LDS* discipline (algorithm) for computing the above. We need to be able to retrieve the old notion i.e. $(\mid\!\sim_1, \mathbf{S}_{\mid\sim_1})$ from the new notion $(\mid\!\sim_2, LDS_{\mid\sim_2})$. In other words, we must add into *LDS* the capability of proving and reasoning without labels. To achieve this we can first reason with labels and then strip the labels and give a conclusion without labels. The additional algorithm which we can use to strip the labels is called *flattening.* Thus a labelled theory Δ may prove $t_i : A$ and $s_i : \neg A$, with many different labels t_i and s_i, depending on various labelling considerations and proof paths. The *flattening* algorithm will allow us to decide whether we flatten the pair of sets $(\{t_i\}, \{s_i\})$ to $+$ or $-$ i.e. whether we say $\Delta \mid\!\sim A$ or $\Delta \mid\!\sim \neg A$. In fact, the flattening rules simply allow us to simplify the labels. They have the general form:

$$t_1 : A; t_2 : A; \ldots, t_m : A \vdash s : A$$

or if we have negation

$$t_i : A; t'_j : \neg A \mid\!\sim s : \pm A$$

See also Definition 3.2.8. See Example 2.3.1 for a clear cut case where only flattening is required.

Flattening can also be used during the deduction in an interlaced mode. For example, if the labels represent moments of time or priorities, we may say the value is $+$ if $\max\{t_i\} \geq \max\{s_j\}$, or we may *interlace* the flattening with the deduction itself, by flattening during the dedution before each proof step is taken. A careful inductive definition is neded. We need the notion $\Delta \vdash_n t : A$ maning $t : A$ can be proved from Δ in at most n steps. We cannow flatten the sets $\{t \mid \Delta \vdash_n t : A\}$ and $\{s \mid \Delta \vdash_n s : \neg A\}$ before we continue the proof. See Example 1.5.4 below.

Thus, given a structured theory Δ (without labels) and a candidate A, we can have the following procedure, using *LDS*, of deciding whether $\Delta \mid\!\sim ?A$.

- Label the elements of Δ with completely different atomic labels, representing existing structure in Δ.
- Use the *LDS* machinery to deduce all possibilities $\Delta \mid\!\sim t_i : A$ and $\Delta \mid\!\sim s_i : \neg A$.
- Flatten and get A (or interlace with flattening and get A).

Example 1.5.4 (Interlacing) *The database has*

$$\begin{array}{l} t_1 : A \\ t_2 : \neg A \\ t_3 : \neg A \to B \\ t_4 : A \to \neg B. \end{array}$$

Assume priority is $t_1 < t_2 < t_3 < t_4$, and assume a flattening process which gives higher priority rules superiority over low priority rules and similarly for facts but gives lexicographic superiority for rules over facts. Thus $t_4 t_1$ is stronger than $t_3 t_2$. If we deduce and then flatten, we get

$$\begin{array}{l} t_4 t_1 : \neg B \\ t_3 t_2 : B \end{array}$$

The flattening process would take $\neg B$.

If we pursue an interlace argument, we first flatten the premisses and take $\neg A$ and then perform the modus ponens and get B.

1.6 Aggregated systems

So far all our logical systems either proved or not proved a conclusion A. There was no possibility of aggregating arguments in favour of A as against arguments in favour of negation of A. The lack of aggregation is a basic characteristic which currently separates the symbolic qualitative school of reasoning from the numerical, quantitative one. We can view thenotion of flattening of the previous section as a special form of aggregation; a noncompromising, all or nothing, aggregation.

There are many systems around (many are recognised as probabilistic systems, expert systems, fuzzy systems) which attach numerical values to assumptions and rules, use various functions and numerical procedures to manipulate these values and formulas and reach conclusions.

In many cases we get systems which give answers which seem to make sense, which can be very successfully and profitably applied but which cannot be recognised or understood by traditional logic. The main feature common to all of these numerical systems (which is independent of how they calculate and propagate their values) is that their 'proofs' aggregate. They can add the numbers involved and thus aggregate arguments. The spirit is: *Five good rumours are better than one proof.*

To further illustrate, consider the following example

Example 1.6.1 (Aggregation and interlacing) *The assumptions are:*

$$\begin{array}{l} t_1 : A \to C \\ t_2 : B \to C \\ t_3 : A \\ t_4 : B \\ t_5 : D \to \neg C \\ t_6 : D \end{array}$$

Here we can conclude C in two different ways and conclude $\neg C$ in one way.

Non monotonic systems like defeasible logic will not allow us to draw any conclusion unless one rule defeats all others. If we had a numerical evaluation of the data, say t_i are numbers in $[0\ 1]$*, then we could aggregate our confidence in the conclusion. Thus we get:*

$$\begin{array}{c} (t_1 \cdot t_3 \uplus t_2 \cdot t_4) : C \\ t_5 \cdot t_6 : \neg C \end{array}$$

the two numbers can be compared and a conclusion reached.

If we operate in the context of *LDS*, we can use the labels to aggregate arguments. Any conclusion is proved with a label indicating its proof path. These can be formally (algebraically) added (aggregated) and an additional process (called *flattening*) can compare them. See Examples 4.1.1 and 2.3.12.

In consequence relation terms, the property of aggregation destroys the cut rule. The reason is as follows:

Assume $\Delta, A \mid\!\sim B$. This now means that the aggregated proofs in favour of B are stronger than the aggregated proofs in favour of $\neg B$. Similarly $\Gamma \mid\!\sim A$ would mean the balance from Γ is in favour of A.

If we perform the cut we get

$$\Delta, \Gamma \mid\!\sim ?B$$

Δ and Γ may interact, forming new additional proofs of $\neg B$, which outweigh the proofs for B.

Cut is a very basic rule in traditional logical systems and can be found in one form or another in each one of them. Thus it is clear that aggregation of arguments is a serious departure from traditional formal logic. Yet, it cannot be denied. In practical reasoning we do aggregate arguments and so logic, if it is to be successfully applied and be able to mirror human reasoning, must be able to cope with aggregation. Classical logic, if it is to be a universal language, must also be able to deal naturally with aggregation.

One form of cut is still valid. The unitary cut.

$$\frac{\Delta \mid\!\sim A; A \mid\!\sim B}{\Delta \mid\!\sim B}$$

This holds because there is nothing for Δ to interact with.

We thus require from our reasoning system that it satisfy only *Identity* ($A \mid\!\sim A$) and *Unitary Cut.*

To show how real and possibly destructive aggregation can be, consider the example of Prince Karlos and Princess Laura.

Example 1.6.2 *The prince and princess are separated. Both made it clear to the press that no third parties were involved and the separation was purely due to a personality clash. However, the editor Mr Angel of the* Daily Tabloid*, thought otherwise. First he observed that after her separation the princess moved to a house very near the Imperial Institute of Logic, Language and Computation. This in itself did not mean much, because both the Institute and the residence were in the centre of town. However, Mr Angel further found out that in the past two years, whenever the princess went on a European holiday, there was an Esprit project meeting in the same hotel, and surprisingly all projects involve a certain professor from the Institute. Again, this could be a coincidence, because it is a well known fact that Esprit project consortia find it most inspiring to be in the most expensive holiday resorts in Europe and it is equally well known that certain dynamic professors participate in many such projects.*

However the plot thickens when the princess, as part of her general social activity seems to actively support the new logics for computation. This could also be a coincidence because after all, this subject is going to transform the nature of our society. The various little arguments do seem to be aggregating, though not conclusively enough to risk an article in such a responsible paper as the Daily Tabloid*. The situation changed when it became known that the princess actively supports Labelled Deductive Systems and the Universality of classical logic. Under this aggregation of arguments an obvious conclusion could be drawn!*

1.7 Practical reasoning systems

Our discussion so far generalised the notion of a *deductive system*; namely, given a database Δ and a formula Q, we ask the basic question, does Δ prove Q? The various concepts we studied had to do with what form do Δ and Q take and what kinds of consequence relations $\mid\!\sim$ and algorithmic systems $\mathbf{S}_{\mid\sim}$ are involved.

In practical reasoning systems, the *deductive* question is but one of many which interest us. Other operations such as *updating, abduction, action, explanation* are also involved. If we rethink of Q as an *input*, we can partially list the kind of of operations which may be involved. These operations are performed using algorithms which accompany the deductive component. We refer to them as *mechanisms.*

- The input Q is a query from Δ. We are interested in whether $\Delta \mid\!\sim Q$ and possibly ask what proofs are available.

- The input Q is an update. We want to insert Q into Δ to obtain Δ'. We may possibly have to deal with inconsistency and restructuring of Δ.[6]

- The input Q is an abductive stimulus (goal). We are interested in Δ' such that $\Delta + \Delta' \mid\!\sim Q$. Where $+$ is a symbol (to be precisely defined) which 'adds' or 'joins' Δ and Δ' to 'combine' their declarative information.

  The $+$ operation may or may not be the same as update. The abductive question is to find (possibly the minimal) Δ' which helps prove the input.[7]

- The input Q may be a stimulus for action on the database outputting a new database or outputting an explanation or any other output of interest.

The new possibilities of a formula Q interacting with a database Δ, (via action or abduction or other mechanisms) allow for a new way of answering queries from Δ. To see this, consider the query $\Delta ? Q$. In the declarative aspect, we want an answer, namely we are asking whether $\Delta \mid\!\sim Q$. This can be checked via $\mathbf{S}_{\mid\sim}$,or semantically. Q does not act or change Δ in any way. In the interactive case, we trigger an action. Q acts on Δ to produce a new Δ'. Q is read imperatively.

[6]Such a view has been presented in Chapter 13 of Bob Kowalski's book [1979]. The systematic study of updates and theory change was initiated in [Alchourrón *et al.*, 1985, Gärdenfors, 1988].

[7]For a given system $(\mid\!\sim, \mathbf{S}_{\mid\sim})$, the abductive mechanism is usually dependent on $\mathbf{S}_{\mid\sim}$, the particular algorithmic proof system involved. Different applications might require different abductive procedures.

We can write $\Delta!Q$ to stress this fact. The result of the interaction is Δ'. Thus $\Delta!Q = \Delta'$. Thus given a Δ and a Q, we have two options. We can ask whether $\Delta \mid\!\sim Q$ holds, (written $\Delta?Q$, with $\mid\!\sim$ implicit) or we can let Q act on Δ, written $\Delta!Q$, where ! denotes the action. When an action ! is given, it is possible to derive a new consequence relation $\mid\!\sim$ dependent on the action ! (really we should write $\mid\!\sim_!$). The new $\mid\!\sim$ is defined by $\Delta \mid\!\sim Q$ iff $\Delta!Q = \Delta$. This view was particularly put forward by F. Veldman and pursued by J. van Benthem.

Let us clearly state our current view on the question of what is a logical system:

Definition 1.7.1 (Current tentative view of a logical system) *A logical system has the form* $(\mid\!\sim, \mathbf{S}_{\mid\sim}, \mathbf{S}_{\text{abduce}}, \mathbf{S}_{\text{update}}, \ldots)$ *where* $\mid\!\sim$ *is a labelled consequence relation,* $\mathbf{S}_{\mid\sim}$ *is a* Labelled Deductive System *with* Flattening *procedures and* $\mathbf{S}_{\text{abduce}}, \mathbf{S}_{\text{update}}$ *etc are* mechanisms *which are dependent on (make use of)* $\mathbf{S}_{\mid\sim}$.

The above definition of a logical system is really a definition of an *agent* (in the AI sense). Thus we are refining the notion of a logic to be a system more and more like a practical reasoner,or an agent. In fact, research in the applications of logic in AI has concentrated mainly on the study of essentially logical systems modelling various pure logical aspects of an agent in an application area. There is less emphasis on integrated combined systems, modelling how the agent uses these pure aspects in an interleaved and combined way. Yet, this integrated use is the dominant feature in any application area, the feature which gives the agent what we perceive as *intelligence*. Our view is to consider such systems as logics, and bring into the realm of logic the methodology of integrating various human reasoning mechanisms.

To focus our ideas, we start with an example, involving integrated practical reasoning.

Imagine a situation where a Head of Department goes to a member of staff trying to persuade him to teach a course which is not exactly in his area. Say the fellow is a logician and the course is operating systems. The Head of Department will explain the situation in the department, pointing out the shortage of staff, financial cutbacks, and argue that everyone must teach more etc. He may use emotional and other arguments, saying things like: think of the challenge! I cannot rely on someone else for this important course! You don't want to antagonise the dept! etc. The member of staff may put forward his general dissatisfaction with his promotion etc. At the end a deal may be struck. This is a typical reasoning interactive process involving many types of databases, beliefs, processes, views and logics. To understand human practical reasoning we must understand and model such interactions.

Our strategy is to build a complex formal model for describing human practical reasoning interactions, show the model is plausible and highlight the role of multilogic integration and combination in such a model. The emphasis is not on the actual practical modelling of human reasoning, (which is a major serious task) but on showing how systematic combinations of logical mechanisms can produce an intelligent agent. Thus, for example, we an try and construct an overall system capable of formalizing the above interaction. We shall describe below the basic components we expect such a system to have. Thus our new notion of a logical system must be capable of defining the following central concepts:

1. Declarative Unit

This concept corresponds to the notion of a formula in classical logic. It is supposed to be the basic information units of our system. It is a labelled formula in Gabbay's Labelled Deductive System ; a situation and an infon in Barwise and Perry Situation Theory; a node and a formula in an inheritance network; a world and a modal formula in a semantically presented modal logic; a formula and a fuzzy value in fuzzy logic and so forth.

Formally a declarative unit is a pair $\delta = (t : A)$, where A is a formula in some logic $\mathbf{L}$ and t a term in some algebra $\mathcal{A}$. The intended meaning is that t gives more information about A. There are various possibilities for what this information can be. It can be priority, it can indicate the source of A, or it can code the proof of A.

2. Databases

This concept corresponds to a theory (set of formulas) in classical logic. In general it is some structure (constellation, network) of declarative units. It is supposed to be, in each logic, the architecture of how several declarative units are put together to form a more complex data structure. In linear logic it is a multiset of formulas. In the Lambek calculus it is a list of formulas.

It is reasonable to require that a single declarative unit counts as a singleton database. We also allow $\varnothing$, the empty database.

3. Input

This notion is a function associating with each database Δ and a declarative unit δ, a new database, $\Delta' =$ *Input* (Δ, δ), the result of 'adding δ to Δ.

In classical logic there is only one input function, $\Delta = \Delta \cup \{\delta\}$. Same in linear logic, where the union is a multiset union. In other logics, where databases are more complex structures, there can be a variety of input functions. For example in the Lambak calculus, where a database is a list of formulas, there are two obvious input functions, appending δ either to the end or to the beginning of the list. It is reasonable to assume that the Input Function, is also defined for databases, i.e. *Input* (Δ_1, Δ_2) is meaningful and it is not necessarily commutative.

4. Integrity (Consistency)

Given a database Δ we assume we have a notion of Integrity. Certain databases are not wanted (they do not satisfy integrity). This notion is not exactly the same as consistency. It generalises consistency. In classical logic inconsistent databases can prove $\perp$ and so we can say that we do not want them. But we can equally say databases are not wanted if they for example list people's addresses without their telephone numbers. These are consistent but do not satisfy integrity (constraints).

5. Update

Updating a database is a function *Update* $(\Delta, \delta) = \Delta'$, inputting δ into Δ and obtaining a Δ' whose integrity is maintained. *Input* (Δ, δ) may not satisfy integrity. *Update* (Δ, δ), will yield a different theory which does satisfy integrity. Algorithmic versions of this function could conceivably contain a truth maintenance component which restores integrity.

There can be several possible update functions for a logic. We may or may not stipulate some minimal conditions on the update function.

6. Consequence

This is a binary relation between databases of the form $\Delta_1 \vdash \Delta_2$. Its meaning can vary from logic to logic and it must satisfy some minimal conditions. It may be only partially presented

- Hilbert consequence has the form $\varnothing \vdash \delta$
- Singleton Tarski consequence has the form $\delta_1 \vdash \delta_2$
- Structured (Tarski) consequence has the form $\Delta \vdash \delta$
- Structured (Scott) consequence has the general form $\Delta_1 \vdash \Delta_2$

In practice one version of the consequence may be presented in some (axiomatic or proof theoretic or semantical) manner, (e.g. Hilbert,) and other forms, (e.g. Tarski,) may or may not be definable from it.

For our purpose, we insist that the consequence relation be presented through an algorithmic proof procedure for determining, for arbitrary Δ and δ, whether $\Delta \vdash \delta$ holds or not. The procedure may succeed, fail or loop on each input. We refer to such a consequence relation as an *algorithmic consequence.*

7. Abudction

Abduction is an algorithm which is defined on top of a consequence relation $\vdash$. The definition of the abduction algorithm may possibly make use of the particular algorithmic proof procedure which presents $\vdash$. The abductive algorithm defines a function, *Abduce* $(\Delta, \delta) = \Delta'$, such that when Δ' is input into Δ, the goal δ succeeds. In symbols

- $\mathit{Abduce}(\Delta, \delta) = \varnothing$ when $\Delta \vdash \delta$.
- $\mathit{Abduce}(\Delta, \delta) = \Delta'$, where $\Delta'' = \mathit{input}(\Delta, \Delta')$ and $\Delta'' \vdash \delta$.

We can assume that the abduction algorithm we are dealing with is particularly convenient, for example, assume that it produced only labelled conjunctions of atoms.

Abduction is used in finding what is needed to have goals succeed. Since goals represnt what the agent 'wants', abduction procedures may be involved in causing actions.

8. Actions

We need notation for actions. This is comprised of atomic actions (a set Π_0 of atomic actions) and some algebraic operations giving new actions from old, generating the family Π of all actions. The theory of action is complex, involving post-conditions and pre-conditions. We assume the action 'a' is semantics, in the sense that it operates on models of the world. [Note, however, that we did not igve a notion of a model or semanitcs forthe structured databases proposed above.] a is controlled by a precondition α_a, and when applied to a model satisfying α_a, a post condition β_a is guaranteed to hold.

a can be understood as operating on theories as well as follows.

Let Δ be a theory. Let $\{\mathbf{m}_1, \mathbf{m}_2, \ldots\}$ be its models. For each $\mathbf{m}_i$, let $\mathbf{m}_i^a$ be $\mathbf{m}_i$ if α_a does not hold and let $\mathbf{m}_i^a$ be the result of applying a to $\mathbf{m}_i$ if α_a does hold. Let Δ_a be the theory of the models $\{\mathbf{m}_1^a, \mathbf{m}_2^a, \ldots\}$. Then Δ_a can be taken to be the result of applying a to Δ. When dealing with labelled formulas and databases, it is possible to have (in many cases) a $\delta_a = (t_a : E_a)$ such that $\Delta_a = \mathit{Update}\ (\Delta, \delta_a)$. i.e. there is a syntactical representation of the action a.

Thus the labels may allow us to present general actions as syntactical operations on databases. The action a can be represented by a labelled wff $t_a : E_a$ and the result of the action given a database Δ can be described by $\Delta_a = \mathit{Update}_\Pi(\Delta, t_a : E_a)$, where Update_Π is the input function associated with the action family Π.

We can assume for simplicity that E_a is a conjunction of atoms or their negations.

This approach allows us to represent the fact that in an interaction between two people, the same action 'a' may be understood differently by different persons and may have a different update value. If 'a' is the action then $\delta_a^i = (t_a^i : E_a^i)$ may be what person i 'understands' of it and $\mathit{Update}_\Pi^i(\Delta_i, \delta_a^i)$ is the resulting updated theory.

9. Goals, desires and wants

Each person or 'logic' has goals he wants to achieve or 'wants'. We represnt them as a sequence $(g_1, \ldots, g_n)$, each g_i is a declarative unit. The interpretation is that the person will take actions to create a situation (database) from which all goals succeed (i.e. follow as consequence).

Having now presented the basic components of a 'logic' or 'agent', we can describe how we view the 'interaction' between the two agents; the Head of Department and the member of staff. In the context of our model of the interaction, we will identify the role of the various non monotonic mechanisms described in this volume.

The following are some components involved in the interaction:

- We need two notions of time, external time (day, hour, minute, etc.) and cycle time, the counting of the moves by the agents. So, for example, agent 1 may ask a query at time t_1 which causes a reply at time $t_2 > t_1$. The external time is the t_1, t_2 axis while the cycle time is $\{1, 2, 3, \ldots\}$.

- Assume that at step n, time t_n, agent 1 makes a move. What he might do is to ask goal g from his database Δ_n^1 at step n. Assume g does not succeed from Δ_n^1. He can then ask the abduction mechanism to find what input δ is needed to make g succeed. Thus

$$\delta = \ \mathit{Abduce}\ (\Delta_n^1, g).$$

  δ cannot just be put into the database since the database is expected to be related to the real world. Agent 1 looks at his stock of actions and possibly finds an action 'a' whose precondition succeeds from his database Δ_n^1, and whose post condition is δ. Note that for simplicity we assumed that abductions and conditions are conjunctions of literals. Agent 1 executes this action. Agent 2 also updates his database according to what he sees of action a.

- Since the databases represent parts of the world, there are other things that may happen during the interaction. An external input may be asked or information exchanged.

The following lists the possible occurrrences at step n.

1. Agent 1 asks a query from agent 2.
2. Agent 1 takes an action to satisfy a goal.
3. An external input δ_i arrives for agent i.
4. Agent 1 gives information δ_1 to agent 2 who inputs it.

Obviously the input of data is a major activity for our agents. We view this process as syntactical model building. The following is a more detailed analysis:

- Agent 1 observes the real world and inputs data of the form $t : E$.
- Another agent proforms an action a and agent 1 interprets the action his own way and inputs some resulting data of the form $t^1_a : E^1_a$.
- Agent 1 receives a direct input $t^2 : E^2$ from agent 2.

The above are external inputs. They can be compared with internal updates such that arise from abduction.

How does non-monotonicity arise in such a set up? Suppose agent 1 gets input $t : E$. This input arises in an active environment. The agent understands the environment and can non-monotonically assume and add a further input $s : B$, which accompanies $t : E$. For example, a used car salesman seeing a man looking around his parking lot examining prices can assume that he has a potential customer. Of course it may be the local printing shop emloyee examining how the price stickers were printed. However, we do have the phenomena in general that more input is generated by such non-monotonic principles. I will call this phenomena *conduction* (from peoples *conduct*).

$$t : E \text{ yields } s : B \text{ by } \textit{conduction.}$$

This is not just default, although it may technically embrace it. It includes any principles which allow for more data to be put in in addition to any input.

Thus a conduction rule looks like this:

- If $\Delta \vdash \alpha : A$ and $t : E$ is received as input and $s : B$ is consistent (with Δ and $t : E$) then add $s : B$ to the input.

It would be instructive to construct a logical system in the sense of the above discussion. We now incrementally present one.

Example 1.7.2 (Minimal implication) *Our starting point is minimal propositional implicational logic with a constant for falsity. The langauge contains atomic propositions $\{p, q, r, \ldots\}$ and the implication connective $\{\to\}$ together with the falsity constant $\{\bot\}$. As a Hilbert system, minimal logic satisfies the following schemas (see [Gabbay, 1981]):*

- $A \to (B \to A)$
- $(A \to (B \to C) \to ((A \to B) \to (A \to C))$

and the rule of modus ponens

- $\dfrac{A; A \to B}{B}$.

The following theorems can be proved

- $\vdash A \to A$
- $\vdash (A \to (B \to C)) \to (B \to (A \to C))$.

A consequence relation $\mid\!\sim_m$ *can be defined by:*

- $A_1, \dots, A_n \mid\!\sim_m B$ *iff (def)* $\vdash A_1 \to (A_2 \to \dots \to (A_n \to B) \dots)$

Note that in minimal logic $\bot$ does not imply anything in particular, just itself. If we add the axiom schema $\bot \to A$, we get intuitionistic logic based on the connectives $\{\to, \bot\}$.

The following additional Hilbert axioms yield full minimal logic with $\wedge$ and $\vee$.

- $A \to (B \to A \wedge B)$
- $A \wedge B \to A$
- $A \wedge B \to B$
- $(A \to C) \to ((B \to C) \to (A \vee B \to C))$
- $A \to A \vee B$
- $B \to A \vee B$

We now have defined $\mid\!\sim_m$, the consequence relation of our logic. We proceed to define $\mathbf{S}_m$ (actually $\mathbf{S}_{\mid\sim_m}$), an algorithmic proof system for $\mid\!\sim_m$. There are many options to choose from, such as Gentzen systems, Tableaux, Term Translation into Classical logic, etc. We choose a goal directed formulation.

Definition 1.7.3 (Goal directed algorithm for minimal consequence) *1. First note that any formula B of the language (without $\vee$ and $\wedge$) has the form $B = (B_1 \to (B_2 \to \dots \to (B_n \to q) \dots)$, where q is atomic and B_i has the same form as B. q is called the* head *of B and $\{B_i\}$ is the* body. *If we allow $\wedge$ in the language, then every formula is equivalent to a set of wffs of the above form. This holds because of the following equivalences in minimal logic:*

$$A \wedge B \to C \text{ and } A \to (B \to C)$$
$$A \to (B \wedge C) \text{ and } (A \to B) \wedge (A \to C)$$

2. *A* theory Δ *is a list of wffs of the logic.*

3. *We define the following metapredicates:*

   - $\Delta ? A = 1$, *reading 'the goal A succeeds from the theory Δ'.*
   - $\Delta ? A = 0$ *reading 'the goal A finitely fails from Δ'.*

   The definition is as follows:

   (a) $\Delta ? q = 1$, *for q atomic if q is listed in Δ.*

   (b) $\Delta ? q = 0$, *for q atomic if q is not the head of any element in Δ.*

   (c) $\Delta ? A_1 \to (\dots (A_n \to q) \dots) = 1$ *(resp. 0) if* $\Delta * (A_1, \dots, A_n) ? q = 1$ *(resp. 0) where $*$ is concatenation.*

   (d) $\Delta ? q = 1$ *if for some $B = (B_1 \to \dots \to (B_n \to q) \dots)$ in Δ we have that $\Delta ? B_i = 1$, for $i = 1, \dots, n$.*

   (e) $\Delta ? q = 0$ *if for each B in Δ of the form $B_1 \to (B_2 \dots (B_n \to q) \dots)$ there exists an $1 \le i \le n$ such that $\Delta ? B_i = 0$.*

Theorem 1.7.4 (Completeness of algorithm) $A_1, \dots, A_n \mid\!\sim_m B$ *iff* $(A_1, \dots, A_n) ? B = 1$

Proof. See [Gabbay, 1992a]. ■

To obtain a proper algorithm for $\mathbf{S}_{\mid\sim_m}$, we need to specify exactly how we compute $\Delta?Q$. It is convenient for the purpose of ease of control to let the goal be a list of formulas Γ, as in some Prolog interpreters. Thus clause 3d of Definition 1.7.3 will now read:

$$\Delta?q * \Gamma = 1 \text{ if for some } B = (B_1 \to (\ldots \to (B_n \to q)\ldots) \text{ in } \Delta \text{ we have } \Delta?(B_1, \ldots, B_n) * \Gamma = 1.$$

We can agree to search the list Δ top down and agree where to continue the search when starting a new goal in the list of goals. The policy of some Prolog interpreters is always to start the search at the top of the database list Δ. This would yield a precise algorithm but may cause loops. For example $(q \to q)?q$ will loop. With a loop checker, however, we get decidability in P-space.

We now proceed with the incremental definition of our logic. We need next the notion of a database. This will contain integrity constraints and some clauses as data. The data is divided into two parts, permanent data and added hypothetical data. (e.g. to show $A \to B$, we hypothetically assume A and try to show B.) The next definition does the job.

Definition 1.7.5 (The basic logic) *1. A formula of the form $B_1 \to (\ldots \to (B_n \to \bot)\ldots)$ where $\bot$ does not appear in $B_i, i = 1, \ldots, n$, is called* an integrity constraint*.*

2. A formula in the pure $\to$ fragment (i.e. without $\bot$) is called a clause*.*

3. A simple database *is a concatenation of three (possibly empty) lists of formulas of the form $\Delta = \Delta_I * \Delta_P * \Delta_A$, where Δ_I is a list of integrity constraints, Δ_P is a list of clauses, called* protected *clauses (the significace of Δ_P will emerge later when we update), and Δ_A is another list of clauses called the* additional *clauses.*

4. A database Δ is inconsistent *if $\Delta?\bot = 1$. Note that Δ is also a* theory*, so $\Delta?\bot$ can be computed.*

Definition 1.7.6 (Updates) *1. Let $\Delta = \Delta_I * \Delta_P * \Delta_A$ be a consistent database with $\Delta_A = (C_1, \ldots, C_m)$ and let Q be a clause. We define the* update *of Δ by Q, denoted by $\Delta!Q$, to be the following database Δ'.*

- *$\Delta' = \Delta$ if $\Delta_I * \Delta_P * (Q)$ is not consistent.*
- *Otherwise, let Δ' be $\Delta_I * \Delta_P * (C_i, C_{i+1}, \ldots, C_m, Q)$ where i is the least number ≥ 1 such that the above theory is consistent.*

An update $\Delta!Q$ insists, if possible, on putting Q into Δ and maintains consistency by taking out from Δ those assumptions that are unprotected (i.e. in Δ_A) and old (i.e. earlier in the list Δ_A). If Q is inconsistent with $\Delta_I * \Delta_P$, then and only then do we reject the input.

In Definition 1.7.3, the algorithm $\mathbf{S}_{\mid\sim_m}$ for computing $\Delta?A \to B$ is based on the deduction theorem and the query $\Delta?A \to B$ is reduced to $\Delta * (A)?B$. In minimal logic, where Δ is a theory, $\Delta * (A)$ is always consistent, because we do not mind deriving $\bot$, and there is no notion of inconsistency. When we move to the notion of databases, integrity constraints and clauses, databases can be inconsistent and the old reduction of $\Delta?A \to B$ to $\Delta * (A)?B$ may need to face the fact that $\Delta * (A)$ is inconsistent. However, we do have in this case the notion of the update $\Delta!A$, and we could reduce the query $\Delta?A \to B$ to that of $\Delta!A?B$. This new reduction actually defines a new a conditional implication $A \Rightarrow B$, meaning B would be true if A were true i.e. update Δ by A and then (query) check B. The next definition gives the details. We are going to keep using the '$\to$' symbol for both kinds of implication and the context will decide what meaning we give to '$\to$'. See [Gabbay *et al.*, 1994].

Definition 1.7.7 (Computation for conditional $\to$) *We define a new computation $\Delta?!Q = 1$ and $\Delta?!Q = 0$ for clauses Q as follows:*

a. $\Delta?!q = 1$ if q is in Δ, for atomic q.

b. $\Delta?!q = 0$ for q atomic, if q is not the head of any clause in Δ.

c. $\Delta?!B_1 \to (B_2 \to \ldots \to (B_n \to q)\ldots) = 1$ *(resp. 0) iff* $(((\Delta!B_1)!B_2)\ldots!B_n)?!q = 1$ *(resp. 0).*

d. $\Delta?!q = 1$ *if for some* $B_1 \to (\ldots \to (B_n \to q)\ldots)$ *in* Δ *we have* $\Delta?!B_i = 1$*, for* $i = 1,\ldots,n$.

e. $\Delta?!q = 0$ *if for each* $B_1 \to (\ldots \to (B_n \to q)\ldots)$ *in* Δ *there exists an* i *such that* $\Delta?!B_i = 0$.

f. $\Delta?!\bot$ *is not defined. We only compute* $\Delta?\bot$.

We now have an update mechanism and a new consequence relation. Let us define some more mechanisms. First we deal with normal defaults of the form $\frac{A:B}{B}$, reading: if A is in Δ and it is consistent to add B then we do add B. Note that the default notion we are proposing here is straightforward and tailored for our case and is not a general theory of default. We are building a 'new logic' and we want to put some default aspects to it.

Definition 1.7.8 (Default mechanism) *1. A normal default* δ *is a pair* $\delta = (A, B)$ *where* A *and* B *are clauses.*

2. *A default database is a concatenation of several lists of clauses and constraints containing at least the following*
$$\Delta = \Delta_I * \Delta_P * \Delta_A * \Delta_D$$
where Δ_D *is a list added because of default.*

3. *We now define the default update* $\Delta!\delta$ *as follows*
$$\Delta!\delta = \Delta_I * \Delta_P * \Delta_A * \Delta_D * (B)$$
provided this theory is consistent and $\Delta?A = 1$.

So far we have built a logical system with a consequence relation, an algorithmic procedure for it and some mechanisms such as updates and default. We will add one more mechanism and then rest our case. This time we add *abduction.*

From the purely logical point of view, abduction is a syntactical action on a theory Δ and a goal Q, consistent with Δ, in a logic $(\mid\!\sim, \mathbf{S}_{\mid\sim})$, yielding some additional data Δ_B, consistent with Δ, (denoted by $\Delta_B =$ *Abduce* (Δ, Q)) such that $\Delta, \Delta_B \mid\!\sim Q$. i.e. we 'answer' the question of 'what do we need to consistently add to Δ to make it prove Q'?

Let us define *Abduce*(Δ, Q) for the logic $(\mid\!\sim_m, \mathbf{S}_{\mid\sim_m})$. The definition will be by induction on the computation steps of Q from Δ, as in Definition 1.7.3.

Definition 1.7.9 *Let* Δ *be a theory and* Q *be a goal in the minimal logic* $\mid\!\sim_m$ *of Example 1.7.2 (for the language with* $\to, \bot$ *and possibly* $\wedge$*), using the algorithm* $\mathbf{S}_{\mid\sim_m}$ *of Definition 1.7.3(3) which is complete by Theorem 1.7.4.*

We define a formula Abduce(Δ, Q) *in the full language of minimal logic, with* $\wedge$ *and* $\vee$ *such that:*

- $\Delta*$ Abduce $(\Delta, Q) \mid\!\sim_m Q$

1. Abduce$(\Delta, Q) = \top$ *if* $\Delta?Q = 1$.

2. Abduce$(\Delta, q) = q$*, for* q *atomic such that* q *is not the head of any clause in* Δ.

3. Abduce$(\Delta, A_1 \to (A_2 \to \ldots (A_n \to q)\ldots)) = A_1 \to \ldots (A_n \to$ Abduce $(\Delta*(A_1,\ldots,A_n), q)\ldots)$

4. *Let* q *be atomic and let* $B^j = B^j_1 \to \ldots \to (B^j_{n_j} \to q)\ldots)$*,* $j = 1,\ldots,m$ *be all clauses in* Δ *with head* q*. Then* Abduce$(\Delta, q) = \bigvee_{j=1}^m \bigwedge_{i=1}^{n_j}$ Abduce (Δ, B^j_i).

5. *In case we have conjunctions;*
Abduce$(\Delta, A \wedge B) =$ Abduce $(\Delta, A) \wedge$ Abduce (Δ, B).

Note that clause 4 of the above definition of *Abduce* may be simplified to be that any one of the disjuncts (say of the first B^j in the list Δ) is always chosen. However, when we take the disjunction we get a logically weaker abduced formula. Also note that clause 5 may give rise to inconsistency; $\mathit{Abduce}(\Delta, A)$ and $\mathit{Abduce}(\Delta, B)$ may be consistent but not necessarily their conjunction.

If we adopt the policy of taking disjunctions in clause 4, we increase the chances of finding a consistent abduced formula. In the $\to$ fragment, of course, we do not want disjunctions.

Examples 1.7.10 *1. Let Δ be $\{a\}$ and let the goal be q. The abduced fomula is* $\mathrm{Abduce}((a), q) = q$*. Note that if we were to take $\gamma = (a \to q)$ then certainly*

$$\Delta, \gamma \mid\!\sim_m q$$

so the abudced formula is not the logically weakest which can be added to Δ to prove the goal (since $\gamma \not\mid\!\sim_m q$ but $q \mid\!\sim_m \gamma$). However, in the presence of Δ it is the weakest.

2. Let Δ be

(a) $a \to (b \to q)$

(b) a

(c) $(c \to d) \to q$

and let the goal be q.

The abduced theory is $b \vee (c \to d)$

Lemma 1.7.11 Δ, Abduce $(\Delta, Q) \mid\!\sim_m Q$.

Proof. The proof is by induction on the definition of the abduced formula.

1. If the abduced formula is $\top$ then this means $\Delta \mid\!\sim_m Q$.

2. Assume that $Q = q$ is atomic and that it is not the head of any clause. Then the abduced formula is q and clearly $\Delta, q \mid\!\sim_m q$.

3. Assume Q has the form
$$Q = (A_1 \to \ldots \to (A_n \to q)\ldots).$$
Then the abduced formula is
$$A_1 \to \ldots \to (A_n \to \ \mathit{Abduce}\ (\Delta * (A_1, \ldots, A_n), q), \ldots).$$
We need to show
$$\Delta, A_1 \to (\ldots \to (A_n \to \ \mathit{Abduce}\ (\Delta * A_1, \ldots, A_n), q)\ldots) \mid\!\sim_m Q.$$
By the induction hypothesis
$$\Delta, A_1, \ldots, A_n, \ \mathit{Abduce}\ (\Delta * (A_1, \ldots, A_n), q) \mid\!\sim_m q,$$
hence
$$\Delta, A_1, \ldots, A_n, A_1 \to \ldots (A_n \to \ \mathit{Abduce}\ (\Delta * (A_1, \ldots, A_n), q)\ldots) \mid\!\sim_m q,$$
hence
$$\Delta, A_1 \to (\ldots A_n \to \ \mathit{Abduce}\ (\Delta * (A_1, \ldots, A_n), q)\ldots) \mid\!\sim_m Q.$$

4. Assume $Q = q$ is atomic and let $B^j = (B^j_1 \to \ldots \to (B^j_{n_j} \to q)\ldots), j = 1, \ldots, m$, be all the clauses in Δ with head q.

Then we need to show

$$\Delta,\ \mathit{Abduce}\ (\Delta, q) \mid\!\sim_m q$$

where

$$\mathit{Abduce}\ (\Delta, q) = \bigvee_{j=1}^{m} \bigwedge_{i=1}^{n_j} \mathit{Abduce}\ (\Delta, B_i^j).$$

By the induction hypothesis for j fixed, we have

$$\Delta,\ \mathit{Abduce}\ (\Delta, B_i^j) \mid\!\sim_m B_i^j$$

for $i = 1, \ldots, n_j$.

Hence for each j, since B^j is in Δ, we have:

$$\Delta, \bigwedge_{i=1}^{n_j} \mathit{Abduce}(\Delta, B_i^j) \mid\!\sim_m q.$$

Hence

$$\Delta, \bigvee_{j=1}^{m} \bigwedge_{i=1}^{n_j} \mathit{Abduce}(\Delta, B_i^j) \mid\!\sim_m q.$$

This completes the induction step and the lemma is proved. ■

Remark 1.7.12 (Abdtop) *The above defined abduction mechanism gives rise to abduced formulas which may be disjunctions, but not necessarily. If we are dealing with the $\to$ fragment only, we will not be able to add the abduced formula into Δ. We notice however, that conjunctions are no problem because in the $\to$ fragment, every formula with conjunctions is equivalent to a conjunction of pure $\to$ formulas. This conjunction can be added to Δ as an additional list. We have already observed that disjunctions arise from the fact that an atom q may have several clauses in the database with head q (item 4 of the inductive definition). Since all clauses are ordered, we can shoose as part of our abduction policy to use only an agreed one of them (say the top of the list). This will give us no disjunctions. Call such a modified abduction algorithm by* Abdtop(Δ, Q).

Example 1.7.13 *Consider the database Δ with one integrity constraint in it and one data item in it.*

1. $a \wedge s \wedge e \to \bot$
2. a

We are using conjunction but it can be eliminated. We can write item 1 as

$$a \to (s \to (e \to \bot)).$$

Consider the following two goals.

- $Q_1 = a \wedge e$
- $Q_2 = a \wedge e \to \bot$

Note that for conjunctions the obvious rule to use (at a risk that integrity constraints are violated) is:

- Abduce$(\Delta, A \wedge B)$ = Abduce $(\Delta, A) \wedge$ Abduce(Δ, B)

Therefore for our example:

$$\text{Abduce}(\Delta, Q_1) = \text{Abduce}(\Delta, a) \wedge \text{Abduce}(\Delta, e) = \text{Abduce}(\Delta, e)$$

Clearly Abduce$(\Delta, e) = e$*, as the only way to prove* e *from* Δ *is to abduce* e *itself (note we do not have* $\bot \mid\!\sim_m e$*).*

We now try to abduce the second goal, Q_2*.*

$$\text{Abduce}(\Delta, Q_2) = (a \wedge e \rightarrow \text{Abduce}(\Delta * (a) * (e), \bot) = a \wedge e \rightarrow s$$

Since

$$\text{Abduce}(\Delta * (a) * (e), \bot) = \text{Abduce}(\Delta * (a) * (e), a \wedge s \wedge e) = s$$

Remark 1.7.14 *There is a sense in which* Abduce(Δ, q) *is the logically minimal addition to* Δ *which can prove* Q*, namely*

If $\Delta, X \mid\!\sim_m Q$*, then* $\Delta', X \mid\!\sim_m$ Abduce(Δ, Q)*, where* Δ' *is some completion of* Δ*. It is not clear to me at this stage exactly what* Δ' *should be. Consider the following:*

$$\Delta = \{a \rightarrow b\}, Q = b.\ \text{Abduce}(\Delta, Q) = a.$$

Clearly, $\Delta, b \vdash Q$ *but* $\Delta, b \nvdash a$*. However, Clark's completion,* $\Delta' = \{a \leftrightarrow b\}$ *does the job;* $\Delta', b \vdash a$*.*

There is work for Horn clauses in this direction in [Console *et al.*, 1991], but our language here contains embedded implications.

Example 1.7.15 (An example of a logical system) *We can now define an example of a 'logical' system in our sense as follows:*

- *The language has* $\rightarrow$ *only.*
- *The notion of a theory and of a consequence relation is that of minimal logic* $\mid\!\sim_m$*.*
- *The algorithmic system is* $\mathbf{S}_{\mid\sim_m}$ *of Definition 1.7.3. Theorem 1.7.4 shows the algorithm is complete and sound.*
- *The abduction mechanism is* Abdtop *of Remark 1.7.12. The result is appended at the end of the* Δ_B *list.*
- *A database is comprised of several lists of clauses and integrity constraints of the form*

$$\Delta = \Delta_I * \Delta_P * \Delta_B * \Delta_A * \Delta_D$$

  where Δ_I *is the integrity constraint,* Δ_P *is the permanent data,* Δ_B *is the abduced data,* Δ_A *is the additional data and* Δ_D *is*

  the default data.
- *The update mechanism is as in Definitions 1.7.6 and 1.7.7, where for the purpose of performing an update the list* Δ_B *is considered 'protected' data while* $\Delta_A * \Delta_D$ *is considered 'additional data'. This gives defaults higher priority than hypotheticals.*
- *Input of hypotheticals is appended to the end of the* Δ_A *list.*
- *The default mechanism is as in Definition 1.7.8 and the result of default is appended at the end of the* Δ_D *list*

Note that some of our decisions in defining the logic of Example 1.7.15 are not the most reasonable. They need to be refined. We can be more careful how we update and more careful where to input the results of abduction by looking at what part of the database was used in the abduction. For example if we abduce q because of default rules, it makes more sense to put the result in the default database then in the abduced database.

However, for the purpose of illustrating what we mean by a logical system with mechanisms, the above is sufficient.

In the most general case, databases are *LDS* databases and not just lists. In this case the mechanisms and algorithms will be more complex.

1.8 Semantics

We cannot address the problem of what is a logical system without saying something about our view of semantics. The traditional view, for classical, intuitionisitc, or modal logic is to have some notion of a class of models and of an evaluation procedure of a formula in a model. Thus we may have a set $\mathcal{K}$ of models and a notion of validity in $\mathbf{m} \in \mathcal{K}$ of a formula A of the logic. We use the notation $\mathbf{m} \vDash A$. Given no details on the internal structure of $\mathbf{m}$ and on how $\mathbf{m} \vDash A$ is evaluated, all we can say about the model is that $\mathbf{m}$ is a {0,1} function on wffs. Completeness of $\mathcal{K}$ for $\mid\!\sim$ means that the following holds:

- $A \mid\!\sim B$ iff for all $\mathbf{m} \in \mathcal{K}$ (if $\mathbf{m} \vDash A$ then $\mathbf{m} \vDash B$).

We would like to present a different view of semantics.

We would like to remain totally within the world of logical systems (in our sense, i.e. *LDS* with mechanisms) and to the extent that semantics is needed, we bring it into the syntax. This can obviously and transparently be done in modal logic where the labels denote possible worlds and the proof rules closely reflect semantical evaluation rules. This in fact can also be done in general. So what then is the basic notion involved in a purely syntactical set up? What replaces the notions of a 'model', 'evaluation', and completeness? We give the following definition. It must be taken with care, as it is easy to give trivial interpretations.

Definition 1.8.1 (Syntactical semantics) *Let $\mid\!\sim$ be a consequence relation and let $\mathcal{K}$ be a class of consequence relations, not necessarily of the same language. for each $\mid\!\sim^* \in \mathcal{K}$, let $\mathbf{k}_{\mid\sim^*}$ be an interpretation of $\mid\!\sim$ into $\mid\!\sim^*$. This involves mapping of the language of $\mid\!\sim$ into the language of $\mid\!\sim^*$ and the following homomorphic commitment:*

- *$A \mid\!\sim B$ implies $A^* \mid\!\sim^* B^*$ (where A^* is $\mathbf{k}_{\mid\sim^*}(A)$ and resp. B^*).*

We say $\mid\!\sim$ is complete for $(\mathcal{K}, \mathbf{k})$, iff we have

- *$A \mid\!\sim B$ iff for all $\mid\!\sim^* \in \mathcal{K}, A^* \mid\!\sim^* B^*$.*

Example 1.8.2 *The following can be considered as semantical interpretations in our sense:*

1. *The Solovay–Boolos interpretation of modal logic* **G** *(with Löbs axiom) in Arithmetic, with $\Box$ meaning 'provable'.*
2. *The interpretation of intuitionistic propositional logic into various sequences of intermediate logics whose intersection is intuitionistic logic (e.g. the Jaskowski sequence).*
3. *The interpretation of modal logic into classical logic.*

Remark 1.8.3 *We gave a definition of interpretation for conseqeunce relations $\mid\!\sim$. In the general case, we have a general* LDS *proof system with algorithmic proof systems $\mathbf{S}_{\mid\sim}$ and various mechanisms. These should also be interpreted. Each algorithmic move in $\mathbf{S}_{\mid\sim}$ should be interpreted as a move package in $\mathbf{S}_{\mid\sim^*}$, and similarly for mechanisms.*

It is possible to justify and motivate our syntactical notion of semantics from the more traditional one. Let us take as our starting point the notion of Scott-semantics described in [Gabbay, 1976].

Definition 1.8.4 *Let* **L** *be a propositional language, for example the modal language with $\Box$ or intuitionistic language with $\to$.*

1. *A model for the language is a function s assigning a value in $\{0, 1\}$ to each wff of the language.*
2. *A* semantics *$\mathcal{S}$ is a class of models.*
3. *Let Δ be a set of wffs and A a wff. We say $\Delta \vDash_{\mathcal{S}} A$ iff for all $\mathbf{s} \in \mathcal{S}$ if $\mathbf{s}(B) = 1$ for all $B \in \Delta$ then $\mathbf{s}(A) = 1$.*

The above definition relies on the intuition that no matter what our basic concepts of a 'model' or interpretation is, sooner or later we have to say whether a formula A 'holds' in it or does not 'hold' in it. Thus the technical 'essence' of a model is a $\{0, 1\}$ function $\mathbf{s}$ (we ignore the possibility of no value).

It can be shown that this notion of semantics can characterise any monotonic (syntactical) consequence relation i.e. any relation $\mid\!\sim$ between sets Δ (including $\Delta = \varnothing$) of wffs and wffs A satisfying *reflexivity, monotonicity* and *cut.* Thus for any $\mid\!\sim$ there exists an $\mathcal{S}$ such that $\mid\!\sim$ equals $\vDash_{\mathcal{S}}$.

Given the above point of view, the notion of syntactical semantics subsumes that of Scott semantics. If, as in Definition 1.8.1, $\mid\!\sim$ is faithfully interpretable in a family of consequence relations $\mid\!\sim_i$, and these in turn have Scott semantics $\mathcal{S}_i$, then we can derive Scott sematics $\mathcal{S}$ for $\mid\!\sim$ through the interpretations. Conversely, given $\mid\!\sim$ and its Scott semantics $\mathcal{S}$, let $\mathcal{S}_i, i \in I$ be a large enough family of subsets of $\mathcal{S}$ and let $\mid\!\sim_i$ be $\vDash_{\mathcal{S}_i}$. Clearly the family $\{\mid\!\sim_i\}$ would be syntactical semantics for $\mid\!\sim$.

The semantics $\mathcal{S}$ can be given further structure, depending on the connectives of $\mathbf{L}$. The simplest is through the binary relation $\leq$, defined as follows:

- $\mathbf{t} \leq \mathbf{s}$ iff (definition) for all wffs $A, \mathbf{t}(A) \leq \mathbf{s}(A)$.

Other relations can be defined on $\mathcal{S}$. For example, if the original language is modal logic we can define:

- $\mathbf{t}R\mathbf{s}$ iff for all $\Box A$ of $\mathbf{L}$ if $\mathbf{t}(\Box A) = 1$ then $\mathbf{s}(A) = 1$.

One can then postulate connections between values such as:

- $\mathbf{t}(\Box A) = 1$ iff $\forall \mathbf{s}[\mathbf{t}R\mathbf{s} \Rightarrow \mathbf{s}(A) = 1]$

or for a language with $\to$:

- $\mathbf{t}(A \to B) = 1$ iff $\forall \mathbf{s}(\mathbf{t} \leq \mathbf{s}$ and $\mathbf{s}(A) = 1$ imply $\mathbf{s}(B) = 1)$.

In some logics and their semantics the above may hold. For example, the respective conditions above hold for the modal logic $\mathbf{K}$ and for intuitionistic logic. For other logics, further refinements are needed.

The nature of what is happening here can be best explained through a translation into classical logic. The language $\mathbf{L}$ can be considered as a Herbrand universe of terms (i.e. the free algebra based on the atomic propositions and the connectives acting as function symbols), and the models considered as another sort of terms, (i.e. the names of the models can be terms). The 'predicate' $\mathbf{t}(A) = 1$ can be considered as a two sorted predicate $\mathbf{Hold}(\mathbf{t}, A)$. Thus the reductions above become

- $\mathbf{Hold}(\mathbf{t}, \Box A)$ iff $\forall \mathbf{s}(\mathbf{t}R\mathbf{s} \Rightarrow \mathbf{Hold}(\mathbf{s}, A))$, where $\mathbf{t}R\mathbf{s}$ is $\forall B(\mathbf{Hold}(\mathbf{t}, \Box B) \Rightarrow \mathbf{Hold}(\mathbf{s}, B))$.

This condition reduces to

- $\forall \mathbf{s}[\forall X(\mathbf{Hold}(\mathbf{t}, \Box X) \Rightarrow \mathbf{Hold}(\mathbf{s}, X)) \Rightarrow \mathbf{Hold}(\mathbf{s}, A)] \Rightarrow \mathbf{Hold}(\mathbf{t}, \Box A)$

This is an internal reduction on **Hold**.

In general, we want to define $\mathbf{Hold}(\mathbf{t}, \sharp(A_1, \ldots, A_n))$ in terms of some relations $R_i(x_1, \ldots, x_{n_i})$ on sort $\mathbf{t}$ (first coordinate of **Hold**), and the predicates $\mathbf{Hold}(x, A_j)$ for subformulas of $\sharp(A_1, \ldots, A_n)$. $R_i(\mathbf{t}_1, \ldots, \mathbf{t}_{n_i})$ in turn, are expected to be defined using $\mathbf{Hold}(\mathbf{t}_i, X_j)$ for some formulas X_j.

Thus in predicate logic we have formulas φ_i and $\Psi_\sharp$ such that:

- $\mathbf{Hold}(\mathbf{t}, \sharp(A_i, \ldots, A_n)$ iff $\Psi_\sharp(\mathbf{t}, R_i,\ \mathbf{Hold}(x_i, A_j))$

- $R_i(\mathbf{t}_1, \ldots, \mathbf{t}_{n_i})$ iff (definition) $\varphi_i(\mathbf{t}_1, \ldots, \mathbf{t}_{n_i},\ \mathbf{Hold}(\mathbf{t}_j, X_k))$.

Together they imply a possible closure condition on the semantics

- $\mathbf{Hold}(\mathbf{t}, \sharp(A_1, \ldots, A_n)$ iff $\Psi_\sharp(t, \varphi_i(\ldots,\ \mathbf{Hold}(\mathbf{t}_j, X_k)),\ \mathbf{Hold}(x_i, A_k))$

which may or may not hold.

Remark 1.8.5 (Represntation of Algebras) *The above considerations can be viewed as a special case of a general set-representation problem for algebras. Let $\mathcal{A}$ be an algebra with some function symbols f_i satisfying some axioms. Take for example the language of lattices $\mathcal{A} = (A, \sqcap, \sqcup)$. we ask the following question: Can $\mathcal{A}$ be represented as an algebra of sets? In other words, is there a set S and a mapping $h(a) \subseteq S$, for $a \in A$ and a monadic first-order langauge $\mathbf{L}_1$ on S involving possibly some relation symbols $R_1, \ldots, R_k$ on S such that for all $s \in S$ and function symbol f of the algebra we have the following inductive reduction, for all $x_1, \ldots, x_n \in A$*

$$s \in h(f(x_1, \ldots, x_n) \textit{ iff } \vDash \Psi_f(s, h(x_1), \ldots, h(x_n))$$

where Ψ_f is a modadic wff of $\mathbf{L}_1$ involving $R_1, \ldots, R_k$ and the subsets $h(x_j)$.

If the relations $R(t_1, \ldots, t_m)$ on S can be defined using h by some formula φ_R of the algebra (involving the classical connectives and equality and the monadic predicates on the algebra $T_i(x)$ meaning $t_i \in h(x)$, then

$$\vDash R(t_1, \ldots, t_m) \textit{ iff } \mathcal{A} \vDash \varphi_R(T_1, \ldots, T_m, R).$$

Compare with section 8.5.

Remark 1.8.6 (Dependent semantics) *The above considerations are not the most general and do not reflect all that might happen. The considerations explain nicely semantics like that of modal $\mathbf{K}$, but we need refinements.*

Consider the logic $\mathbf{K}_1$ obtained by collecting all theorems of modal logic $\mathbf{K}$ together with the schema $\Box A \to A$ and the rule of modus ponens. Necessitation is dropped, so although $K_1 \vdash \Box A \to A$, we can still have $\mathbf{K}_1 \nvdash \Box(\Box A \to A)$. See Example 3.4.8. This logic is complete for the class of all Kripke structures of the form $\mathbf{m} = (S^{\mathbf{m}}, R^{\mathbf{m}}, a^{\mathbf{m}}, h^{\mathbf{m}})$, where $a^{\mathbf{m}} R a^{\mathbf{m}}$ holds. Completeness means

1. *$\mathbf{K}_1 \vdash A$ iff for every $\mathbf{m}$ as above $a^{\mathbf{m}} \vDash A$*

Let $\mathbf{a_m}$ be the function satisfying

2. *$\mathbf{a_m}(A) = 1$ iff $a_{\mathbf{m}} \vDash A$*

and let

3. *$\mathcal{S}_0 = \{\mathbf{a_m} \mid \mathbf{m}$ as above$\}$.*

Then we have here a semantics $\mathcal{S}_0 \subseteq \mathcal{S}$ (of the langauge $\mathbf{L}$ of modal logic) where $\mathbf{a_m}(\Box A)$ cannot be recuced to values of $\mathbf{s}(A)$ for $\mathbf{s} \in \mathcal{S}_0$, but can be reduced to values $\mathbf{s}(A)$, for $\mathbf{s} \in \mathcal{S}$. This is so because when we evaluate $a^{\mathbf{m}} \vDash A$, we evaluate at points $b \in S^{\mathbf{m}}$ such that $a^{\mathbf{m}} R^{\mathbf{m}} b$ and the Kripke structure but notnecessarily a $\mathbf{K}_1$ structure $(S^{\mathbf{m}}, R^{\mathbf{m}}, b, h^{\mathbf{m}})$ is a $\mathbf{K}$ structure as bRb need not hold. Let $\mathbf{b}^{\mathbf{m}}$ be the function defined by

4. *$\mathbf{b}^{\mathbf{m}}(A) = 1$ iff $b \vDash A$ in $\mathbf{m}$.*

   we get

5. *$\mathbf{a_m}(\Box A) = 1$ iff for all $\mathbf{s} \in \{\mathbf{b}^{\mathbf{m}} \mid a^{\mathbf{m}} Rb\}$, we have $\mathbf{s}(A) = 1$.*

   Let $\varphi(\mathbf{a}, \mathbf{b})$ mean as follows:

6. *$\varphi(\mathbf{a}, \mathbf{b})$ iff (definition) for some $\mathbf{m}, \mathbf{a} = \mathbf{a_m}$ and $b = \mathbf{b}^{\mathbf{m}}$ and $a^{\mathbf{m}} Rb$.*

   Then we have that $\mathbf{K}_1$ is characterised by a designated subset $\mathcal{S}_0$ of $\mathcal{S}$ and the truth definition:

7. *$\mathbf{s}(\Box A) = 1$ iff for all $\mathbf{s}' \varphi(\mathbf{s}, \mathbf{s}')$ and $\mathbf{s} R \mathbf{s}'$ imply $\mathbf{s}'(A) = 1$.*

8. $A \models B$ iff for all $\mathbf{s} \in \mathcal{S}_0, \mathbf{s}(A) = 1$ implies $\mathbf{s}(B) = 1$.

Compare with Example 4.1.3 for an essentially related meaning of φ.

We are now ready to say what it means to give technical semantics t a consequence relation $\mid\!\sim$.

Definition 1.8.7 (What is semantics for $\mid\!\sim$) *Let $\mid\!\sim$ be a consequence relation (Reflexive and trasitive) in a language with connectives. Then a semantics for $\mid\!\sim$ is any set theoretic representation (in the sense of Remark 1.8.1) of the free term algebra based on $\mid\!\sim$.*

The previous definition does not take account of remark 1.8.6. If we want a better concept of what is semantics, we need to talk about *fibred sematnics* and *label dependent connectives.* These topics will be addressed in chapter 15.

1.9 Conclusion

We have incrementally gone through several notions of 'what is a logical system' and have ended up with Definition 1.7.1 and Example 1.7.15 to illustrate it. This new concept of a logic is very far from the traditional concept. In artificial intelligence circles, what we call a 'logic' is perceived as an 'agent' or 'intelligent agent'. This is no accident. Whereas traditional logical systems (classical logic, intuitionistic logic, linear logic) model mathematical reasoning and mathematical proof, our new concept of logic attempts to model, and stay tuned to, human practical reasoning. What we tried to do is to observe what features and mechanisms are at play in human practical reasoning, and proceed to formalise them. The systems emerging from this formalisation we accept as the new 'logics'. It is therefore no surprise that in AI circles such systems are perceived as intelligent agents. However, compared with AI, our motives are different. We are looking for general logical principles of human reasoning and not necessarily seeking to build practical applied systems.

There is one more point to make before we can close this paper. The above 'logics' manipulate formulas, algebraic terms and in general syntactical symbols. We have maintained already in 1988 [Gabbay and Reyle, to appear] that deduction is a form of *stylised movement*, which can be carried out *directly* on natural objects from an application area. Thus 'logic' can be done not only on syntactical formulas, but on any set of structured objects, naturally residing in some application area.

To reason about gardening, for example, we can either represent the area in some language and manipulate the syntax in some logic, or we can directly manipulate and move the plants themselves and 'show' the conclusion. The style of movement is the 'logic'. This concept of *logic as movement* is clearly apparent in automated reasoning. Different kinds of 'shuffling' licenced by a theorem prover can lead to different 'logics', because then different sets of theorems become provable. Our insight was that similar movements can be applied directly on the objects of the application areas, and therefore reasoning can be achieved directly in the application area without formalisation. This philosophy has been carried out on Discourse Representation Structures in [Gabbay and Reyle, to appear].

We now address the question of how we propose to tackle the classification of non-monotonic systems. To explain our ideas, consider for the moment the theory of Boolean algebra. The class of boolean algebras can be characterised axiomatically by writing the axioms of boolean algebra. Any specific interesting Boolean algebra may or may not be characterisable using further axioms. This is the declarative way of classifying algebras. If a logical system is taken as our object of study (corresponding to a Boolean algebra) then the method of axiomatic presentation of consequence relation corresponds to identifying Boolean algebras by further axioms. In the Boolean algebra area, there is another way of identifying a certain algebra. We have a method of constructing and generating more algebras from given algebras by taking products and subalgebras. So we can start with the $\{0, 1\}$ Boolean algebra and generate any algebra in this manner. Thus by the Stone representation theorem, any boolean algebra can be represented as a subalgebra of a product of $\{0, 1\}$ algebras.

Suppose now we want to identify a certain class of algebras in a natural way. It may be that this class cannot naturally be identified by axioms nor can it be naturally constructed via a clearly identifiable sequence of products and subalgebras. It may be possible however to *combine* the two methods. Give some axioms A and give some ways of constructing algebras, say C, and the desired class is the intersection, we can denote that by (A, C). The analog in our case is clear. Axioms on the consequence relation gives us the $\vdash$ part and the *LDS* discipline gives us the **S** part. Together they can identify a logic $(\vdash, \mathbf{S})$. See the Section 'A case study: concatneation logic below.

Going back to our logics, what would be a mechanism for generating more logics? The answer is simple. Given two logics $\mathbf{L}_1$ and $\mathbf{L}_2$, $\mathbf{L}_1$ is defined as a labelled deductive system. Replace the labels of $\mathbf{L}_1$ by formulas of $\mathbf{L}_2$ and retain the $\mathbf{L}_1$ labelling discipline. Assuming the discipline can be meaningfully applied to formulas of $\mathbf{L}_2$ as labels, the result will be a new logic $\mathbf{L}_3$. In simple words, we use one logic on the labels, another logic on the rules to form a third logic. I agree that a lot of these "generated" logics will not be interesting. That is not what is of importance here. What is important is that we have a unifying mechanism *LDS* that when refined can generate known logics and possibly other interesting logics. In the theory and construction of boolean algebras we can also generate uninteresting systems, it is the good ones that we look at.

Chapter 2

Introducing Labelled Deductive Systems

2.1 Introduction

The previous chapter successively refined the notion of a logical system with a view of introducing *LDS* as the most general concept of a logic. This chapter motivates *LDS* through many examples from monotonic and nonmonotonic logics. Section 2.2 is mainly concerned with formulating known and familiar monotonic logics as *LDS*. This will persuade the reader that *LDS* is general enough. Section 2.3 presents many nonmonotonic systems and features as *LDS*. This should show that *LDS* is indeed natural and comprehensive framework. Within *LDS*, the distinction between monotonic and nonmonotonic systems is not important. Both kinds look essentially the same with some variations. The rest of the sections in the chapter present case studies in depth, showing how *LDS* presentations of known logics make us understand them much better and present them with technical simplicity and elegance. The chapter concludes with a discussion of the limitations of *LDS*.

2.2 Examples from implicational and monotonic logics

To motivate our approach we study several known examples in this section.

Example 2.2.1 below shows a standard deduction from Relevance Logic. The purpose of the example is to illustrate our point of view. There are many such examples in Anderson and Belnap's book. Example 2.2.3 below considers a derivation in modal logic. There we use labels to denote essentially possible worlds. The objective of the example is to show the formal similarities to the relevance logic case, Example 2.2.1. Example 2.2.4 can reap the benefits of the formal similarities of the first two examples and introduce, in the most natural way, a system of relevant modal logic. The objective of Example 2.2.4 is to show that the labels in Example 2.2.1 and Example 2.2.3 can be read as determining the metalanguage features of the logic and can therefore be combined "declaratively" to form the new system of Example 2.2.4. Example 2.2.5 considers strict implication. This example shows that for strict **S4** implication one can read the labels either as relevance labels or as possible world labels. Examples 2.2.6 and 2.2.7 show how labels can interact with quantifiers in modal logic.

Example 2.2.1 (Relevance and Linear Logic) *Consider a propositional language with implication "$\to$" only. The forward rule is modus ponens. From the theorem proving view, modus ponens is an object language consideration. Thus a proof of* $\vdash (B \to A) \to ((A \to B) \to (A \to B))$ *can proceed as follows:*

Assume $a_1 : B \to A$ *and show* $(A \to B) \to (A \to B)$*. Further assume* $a_2 : A \to B$ *and show* $A \to B$*. Further assume* $a_3 : A$ *and show* B*. We thus end up with the following problem:*

Assumptions

1. $a_1 : B \to A$
2. $a_2 : A \to B$
3. $a_3 : A$

Derivation

4. $a_2a_3 : B$ by modus ponens from lines (2) and (3).
5. $a_1a_2a_3 : A$ from (4) and (1).
6. $a_2a_1a_2a_3 : B$ from (5) and (2).
7. $a_2a_1a_2 : A \to B$ from (3) and (6).
8. $a_2a_1 : (A \to B) \to (A \to B)$ from (2) and (7).
9. $a_2 : (B \to A) \to ((A \to B) \to (A \to B))$ from (1) and (8).

The meta-aspect of this proof is the annotation of the assumptions and the keeping track of what was used in the deduction. A metalevel condition would determine the logic involved.

A formal definition of the labelling discipline for this class of logics will be given in a later chapter. For this example it is sufficient to note the following three conventions:

1. A deduction task has labelled assumptions and a labelled goal (later we will allow several goals). Each new assumption is labelled by a new atomic label.

   An ordering on the labels can be imposed, namely $a_1 < a_2 < a_3$. This is to reflect the fact that the assumptions arose from our attempt to prove $(B \to A) \to ((A \to B) \to (A \to B))$ and not for example from $(A \to B) \to ((B \to A) \to (A \to B))$ in which case the ordering would be $a_2 < a_1 < a_3$. The ordering can affect the proofs in certain logics.

2. If in the proof A is labelled by the multiset α and $A \to B$ is labelled by β then B can be derived with a label $\alpha \cup \beta$ where "$\cup$" denotes multiset union.

3. If B was derived using A as evidenced by the fact that the label α of A is a atomic and is in the label β of $B(\alpha \in \beta)$ then we can derive $A \to B$ with the label $\beta - \alpha$ ("-" is multiset subtraction).

The derivation can be represented in a more graphical way.

To show $(B \to A) \to ((A \to B) \to (A \to B))$, we use the metabox of Fig 2.1.

This is a way of representing the deduction. Note that in line 11, multiset subtraction was used and only one copy of the label a_2 was taken out. The other copy of a_2 remains and cannot be cancelled. Thus this formula is not a theorem of linear logic because the outer box does not exit with label $\varnothing$. In relevance logic, the discipline uses sets and not multisets. Thus the label of line 11 in this case would be a_1 and that of line 12 would be $\varnothing$. Strictly speaking we get R-mingle, a variation of relevance logic.

The above deduction can be made even more explicit as follows:

$(B \to A) \to ((A \to B) \to (A \to B))$ follows with a label from Box a_1.

Box a_1

$a_1 :$	$B \to A$ assumption
$a_2a_1 :$	$(A \to B) \to (A \to B)$ from Box a_2

Box a_2

Box a_1	1		show $(A \to B) \to (A \to B)$
	2	$a_1 : B \to A$	
Box a_2	3		show $A \to B$
	4	$a_2 : A \to B$	
Box a_3	5		show B
	6	$a_3 : A$	
	7	$a_2a_3 : B$	
	8	$a_1a_2a_3 : A$	
	9	$a_2a_1a_2a_3 : B$	
exit	10	$a_2a_1a_2 : A \to B$	
exit	11	$a_2a_1 : (A \to B) \to (A \to B)$	
exit	12	$a_2 : (B \to A) \to ((A \to B) \to (A \to B))$	

Figure 2.1:

a_2 :	$A \to B$ assumption
$a_2a_1a_2$:	$A \to B$ from Box a_3

Box a_3

$a_3 : A$	assumption
$a_2 : A \to B$	reiteration from box a_2
$a_2a_3 : B$	by modus ponens
$a_1 : B \to A$	reiteration from box a_1
$a_1a_2a_3 : A$	modus ponens from the two preceding lines
$a_2 : A \to B$	repetition of an earlier line
$a_2a_1a_2a_3 : B$	modus ponens from the two preceding lines

The following metarule was used:

We have a system of partially ordered metaboxes $a_1 < a_2 < a_3$. Any assumption in a box a can be reiterated in any box b provided $a < b$. In this example the box structure is not necessary but in general it might be useful.

Remark 2.2.2 a. The above presentation of the boxes makes them look more like possible worlds. The labels are the worlds and formulas can be exported from one world to another according to some rules. The next example describes modal logic in just this way.

b. Note that different metaconditions on labels and metaboxes correspond to different logics.

The following table gives intuitively some correspondence between metaconditions and logics. A precise definition can be found in Chapter 9.

Metacondition:	**Logic**
ignore the labels	intuitionistic logic
accept only the derivations which use all the assumptions	relevance logic
accept derivations which use all assumptions exactly once	linear logic

The metaconditions can be translated into object conditions in terms of axioms and rules. If we consider a Hilbert system with modus ponens and substitution then the additional (not necessarily independent) axioms involved are given below:

Linear Logic
$A \to A$
$(A \to (B \to C)) \to (B \to (A \to C))$
$(C \to A) \to ((B \to C) \to (B \to A))$
$(C \to A) \to ((A \to B) \to (C \to B))$

Relevance Logic
Add the schema below to linear logic:
$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$

R-Mingle
Add the schema below to relevance logic:
$A \to (A \to A)$

Intuitionistic Logic
Add the schema below to relevance logic:
$A \to (B \to A)$

The reader can note that the following axiom (Peirce Rule) yields classical logic. The corresponding metacondition is the Restart Rule, to be given in Chapter 3. We shall see that for example, we can define "Linear Classical Logic" by adding Peirce Rule to linear logic. A new logic is obtained.

Classical Logic
Add the schema below to intuitionistic logic:
$((A \to B) \to A) \to A$.

Example 2.2.3 (Modal logic) This example shows the metalevel-object level division in the case of modal logic. Modal logic has to do with possible worlds. We thus think of our basic database (or assumptions) as a finite set of information about possible worlds. This consists of two parts. The configuration part, the finite configuration of possible worlds for the database, and the assumptions part which tells us what formulas hold in each world. The following is an example of a database:

	Assumptions	**Configuration**
(1)	$t : \Box\Box B$	$t < s$
(2)	$s : \Diamond(B \to C)$	

The conclusion to show (or query) is:

$$t : \Diamond\Diamond C.$$

The derivation is as follows:

3. From (2) create a new point r with $s < r$ and get $r : B \to C$.

We thus have

Assumptions	**Configuration**
(1), (2), (3)	$t < s < r$

4. From (1), since $t < s$ get $s : \Box B$.
5. From (4) since $s < r$ get $r : B$.
6. From (5) and (3) we get $r : C$.
7. From (6) since $s < r$ get $s : \Diamond C$.
8. From (7) using $t < s$ we get $t : \Diamond\Diamond C$.

Discussion:
The object rules involved are:
$\Box E$ **Rule:**

$$\frac{t < s; t : \Box A}{s : A}$$

$\Diamond I$ **Rule:**

$$\frac{t < s, s : B}{t : \Diamond B}$$

$\Diamond E$ **Rule:**

$$\frac{t : \Diamond A}{\text{create a new point } s \text{ with } t < s \text{ and deduce } s : A}$$

Note that the above rules are not complete. We do not have rules for deriving, for example, $\Box A$. Also, the rules are all for intuitionistic modal logic. Modal and temporal logics will be studied in a special chapter.

The metalevel consideration may be properties of $<$, such as transitivity
$t < s$ and $s < r$ implies $t < r$
or linearity:
$t < s$ or $t = s$ or $s < t$ etc.

Example 2.2.4 (Modal relevance logic) *The reader can already see the benefit of separating the metalevel and the object level features. We can combine both the metalevel features of Examples 2.2.1 and 2.2.3 to create for example a modal relevance logic in a natural way. Each assumption has a relevance label as well as world label. Thus the proof of the previous example becomes the following:*

	Assumptions	**Configuration**
(1)	$(a_1, t) : \Box\Box B$	$t < s$
(2)	$(a_2, s) : \Diamond(B \to C)$	

We proceed to create a new label r using $\Diamond E$ rule. The relevance label is carried over. We have $t < s < r$.

3. $(a_2, r) : B \to C$

Using $\Box E$ rule with relevance label carried over, we have:

4. $(a_1, s) : \Box B$
5. $(a_1, r) : B$

Using modus ponens with relevance label updated

6. $(a_1, a_2, r) : C$

Using $\Diamond I$ rule:

7. $(a_1, a_2, s) : \Diamond C$

8. $(a_1, a_2, t) : \Diamond\Diamond C$

(8) means that we got $t : \Diamond\Diamond C$ using both assumptions a_1 and a_2.

There are two serious problems in modal and temporal theorem proving. One is that of Skolem functions for $\exists x \Diamond A(x)$ and $\Diamond \exists x A(x)$ are not logically the same. If we skolemise we get $\Diamond A(c)$. Unfortunately it is not clear where c exists, in the current world ($\exists x = c \Diamond A(x)$) or the possible world ($\Diamond \exists x = cA(x)$).

If we use labelled assumptions then, $t : \exists x \Diamond A(x)$ becomes $t : \Diamond A(c)$ and it is clear that c is introduced at t.

On the other hand, the assumption $t : \Diamond \exists x A(x)$ will be used by the $\Diamond E$ rule to introduce a new point $s, t < s$ and conclude $s : \exists x A(x)$. We can further skolemise at s and get $s : A(c)$, with c introduced at s. We thus need the mechanism of remembering or labelling constants as well, to indicate where they were first introduced.

Labelling systems for modal and temporal logics will be studied in a later Chapter.

Example 2.2.5 (S4 Strict implication) The following example describes the logic of modal **S4** strict implication. In this logic the labels can be read either as relevance labels or as possible worlds. **S4** strict implication $A \to B$ can be understood as a temporal connective, as follows:

"$A \to B$ is true at world t iff for all future worlds s to t and for t itself we have that if A is true at s then B is true at s". Thus $A \to B$ reads "From now on, if A then B".

Suppose we want to prove that $A \to B$ and $A \to (B \to C)$ imply $A \to C$. To show this we reason semantically and assume that at time t, the two assumptions are true. We want to show that $A \to C$ is also true at t. To prove that we take any future time s, assume that A is true at s and show that C is also true at s. We thus have the following situation:

1. $t : A \to B$

2. $t : A \to (B \to C)$

3. $t : A \to C$, from box

3.1	Assume $s : A$ Show $s : C$
	Since s is in the future of t, we get that at s,
	(1) and (2) are also true.
3.2	$s : A \to B$ from (1)
3.3	$s : A \to (B \to C)$ from (2)
	We now use modus ponens, because $X \to Y$ means
	"from now on, if X then Y"
3.4	$s : B$
3.5	$s : B \to C$ from (3.1) and (3.3)
3.6	$s : C$ modus ponens from (3.4) and (3.5)

exit $t : A \to C$

Notice that any $t : D$ can be brought into (reiterated) the box as $s : D$, provided it has an implicational form, $D = D_1 \to D_2$. We can thus regard the labels above as simply naming assumptions (not as possible worlds) and the logic has the reiteration rule which says that only implications can be reiterated.

Let us add a further note to sharpen our understanding. Suppose $\to$ is read as a **K4** implication (ie transitivity without reflexivity). Then the above proof should fail. Indeed the corresponding restriction on modus ponens is that we do perform $X, X \to Y \vdash Y$ in a box, provided $X \to Y$ is a reiteration into the box and was not itself derived in that same box. This will block line (3.6).

Example 2.2.6 (The Barcan formula) *Another example has to do with the Barcan formula*

	Assumption	**Configuration**
(1)	$t : \forall x \Box A(x)$	$t < s$

We show

$$s : \forall x A(x)$$

We proceed intuitively

1. $t : \Box A(x)$ *(stripping $\forall x$, remembering x is arbitrary).*
2. *Since the configuration contains $s, t < s$ we get*
$$s : A(x)$$
3. *Since x is arbitrary we get*
$$s : \forall x A(x)$$

The above intuitive proof can be restricted.
The rule

$$\frac{t : \Box A(x), t < s}{s : A(x)}$$

is allowed only if x is instantiated.
To allow the above rule for arbitrary x is equivalent to adopting the Barcan formula axiom:

$$\forall x \Box A(x) \to \Box \forall x A(x)$$

Example 2.2.7 (More on the Barcan formula) *To show $\forall x \Box A(x) \to \Box \forall x A(x)$ in the modal logic where it is indeed true.*

1. *Assume $t : \forall x \Box A(x)$*
We show $\Box \forall x A(x)$ by the use of the metabox:

	create α,	$t < \alpha$
(2)	$t : \Box A(x)$	*from (1)*
(3)	$\alpha : A(x)$	*from (2) using a rule*
		which allows this with x a variable.
(4)	$\alpha : \forall x A(x)$	*universal generalisation.*

(5) Exit: $t : \Box \forall x A(x)$.

This rule has the form:

Create α,	$t < \alpha$
Argue to get	$\alpha : B$
Exit with	$t : \Box B$

The above are just a few examples for the scope we get using labels. The exact details and correspondences will be worked out in the main body of the book.

A later chapter develops the formal mechanism we use in *LDS*, the mechanism of the **Metabox**. To explain intuitively what is involved, we look at a few more examples in this section and redo some of our earlier examples in the framework of the **Metabox**.

If we use modus ponens to go forward, we cannot prove theorems like

$$A \to (B \to C) \vdash B \to (A \to C).$$

To achieve that, we need to be able to use some metarule like the deduction theorem. We need to *assume* B and show that $B, A \to (B \to C) \vdash A \to C$ and then further assume A and show that $A, B, A \to (B \to C) \vdash C$.

Formally in the framework of labelled deduction we have to allow for a mechanism similar to the "assume" mechanism and we have to indicate what is the label of the assumption.

The general framework is the metabox rule.

Metabox:

(α) Assume the assumptions $\alpha_i : A_i$ (this is the input).

(β) Prove within the metabox and get $\beta_j : B_j$.

(γ) Exit the metabox with $\gamma_k : C_k$.

Example 2.2.8 Relevance Reasoning
The indices are $\alpha, \beta,$ and $\gamma = (\beta - \alpha)$. The reasoning structure is:
Assume $\alpha : A, \alpha$ atomic
Show $\beta : B$
If $\alpha \in \beta$ then exit with $(\beta - \alpha) : A \to B$.
To show $A \to (B \to C) \vdash B \to (A \to C)$

Assume

$$a_1 : A \to (B \to C)$$

we use the metabox to show $B \to (A \to C)$. See fig 2.2.

	1		Show $A \to C$
	2	$a_2 : B$	assumption
	3		Show C
	4	$a_3 : A$	assumption
	5	$a_1 a_3 : B \to C$	
	6	$a_1 a_3 a_2 : C$	
exit	7	$a_1 a_2 : A \to C$	
exit	8	$a_1 : B \to (A \to C)$	

Figure 2.2:

Example 2.2.9 (Łukasiewicz many valued logics) Consider Lukasiewicz infinite valued logic, where the values are all real numbers or rationals in [0,1]. We designate 0 as **truth** and the truth table for implication is

$$x \to y = \max(0, y - x)$$

Here the language contains atoms and implication only, assignments h give values to atoms in [0,1], $h(q) \in [0,1]$ and h is extended to arbitrary formulas via the table for $\to$ above. Define the relation

$$A_1, \ldots, A_n \vdash B$$

to mean that for all $h, h(A_1) + \ldots + h(A_n) \geqq h(B)$, where $+$ is a numerical addition.

This logic can be regarded as a labelled deductive system, where the labels are values $t \in [0,1]$. $t : A$ means that $h(A) = t$, for a given background h. The interesting part is that to show $t : A \to B$ (ie that $A \to B$ has value t) we assume $x : A$ (ie that A has value x) and then have to show that B has value $t + x$, ie show $t + x : B$.

This is according to the table of $\to$.

Thus in box form, Fig 2.3:

This has the *same structure* as the case of relevance logic, where + was understood as concatenation.

A full study of many valued logics is given in a subsequent chapter.

$$\boxed{\begin{array}{ll} x : A & \text{assumption} \\ \vdots & \\ \vdots & \\ t + x : B & \end{array}}$$

Figure 2.3:

Example 2.2.10 (Formulas as Types) Another instance of the natural use of labels is the Curry-Howard interpretation of formulas as types. This interpretation conforms exactly to our framework. In fact, our framework gives the incentive to extend the formulas as types interpretation in a natural way to other logics, such as linear and relevance logics and surprisingly, also many valued logics, modal logics, and intermediate logics. A formula is considered as a type and its label is a *definable* λ-term of the same type. Given a system for defining λ-terms, the theorems of the logic are all those types which can be shown non empty.

The basic propagation mechanism corresponding to modus ponens is:

$$\frac{\begin{array}{l} t^A : A \\ t^{A \to B} : A \to B \end{array}}{t^{A \to B}(t^A) : B}$$

It is satisfied by *application*.

Thus if we read the + in $t^{A \to B} + t^A$ as application, we get the exact parallel to the general schema of propagation. Compare with relevance logic where + was concatenation, and with many valued logics where + was numerical addition!

To show $t : A \to B$ we assume $x : A$, with x arbitrary, ie start with a term x of type A, use the proof rules to get B. As we saw, applications of modus ponens generate more terms which contain x in them via application. If we accept that proofs generate functionals, then we get B with a label $y = t(x)$. Thus $t = \lambda x t(x)$. This again conforms with our general schema for $\to$.

In a later chapter on the Curry-Howard interpretation we shall exploit this idea systematically. There are two mechanisms which allow us to restrict or expand our ability to define terms of any type. We can restrict λ-abstraction, (eg allow $\lambda x t(x)$ only if x actually occurs in t), this will give us logics weaker than intuitionistic logic, or we can increase our world of terms by requiring diagrams to be closed eg, for any φ of classical logic such that

$$\vdash (A \to B) \to [\varphi(A) \to \varphi(B)]$$

in classical logic, we want the following diagram to be complete, ie for any term t there must exist a term t' (see fig 1.4)

$$\begin{tikzcd}
\varphi(A) \arrow[r, "t'"'] & \varphi(B) \\
A \arrow[u] \arrow[r, "t"'] & B \arrow[u]
\end{tikzcd}$$

Figure 2.4:

Take for example the formula $A \to (B \to A)$ as type. We want to show a definable term of this type, we can try and use the standard proof (see Fig 2.5), however, with the restriction on

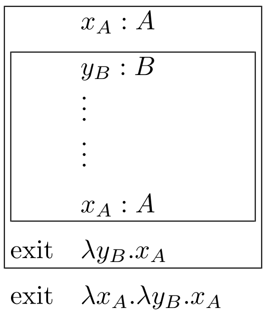


Figure 2.5:

λ-abstraction which requires the abstracted variable to actually occur in the formula, we cannot exit the inner box.

Example 2.2.11 (Realisability Interpretation) The well known realisability interpretation for intuitionistic implication is another example of a functional interpretation for $\to$ which has the same universal *LDS* form. A notation for a recursive function $\{e\}$ realises an implication $A \to B$, iff for any n which realises $A, \{e\}(n)$ realises B. Thus

$$e : A \to B \text{ iff } \forall n[n : A \Rightarrow \{e\}(n) : B]$$

It is an open problem to find an axiomatic description of the set of all wffs which are realisable.

2.3 Examples from non-monotonic logics

The examples in the previous section are from the area of monotonic reasoning. This section will give examples from non-monotonic reasoning. As we have already mentioned, we hope that the idea of *LDS* will unify these two areas.

The next example shows how labels can be used for information sources and priorities.

Example 2.3.1 (Mother-in-law) Let $B(a)$ be a literal meaning 'It is a sound investment for Dov to buy the house a for the price quoted'. The words in Figure 2.6 indicate source of information.

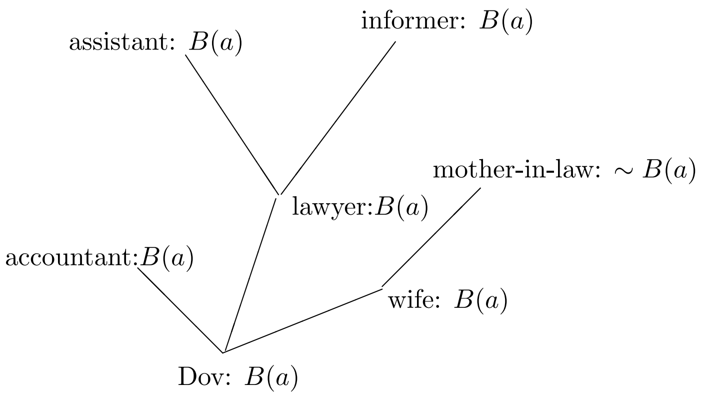


Figure 2.6:

The accountant and lawyer recommend that I buy the house. So do I and so does my wife. I like it, the accountant thinks I have the money and it is a good move. The lawyer checked with his assistant the legal aspects as well as with his informer in City Hall. The area development plan looks good. So practically everybody agrees that $B(a)$ is true except the mother-in-law. For her own (non-logical) reasons she says no. Figure 2.6 represents a labelled database. The tree structure gives the priorities. It is not a situation theory "situation" in spirit because it is not semantic. It is just an *LDS* representing the hierarchy of information where the labels are prioritised sources. In fact, the above database 'boils down' to Figure 2.7:

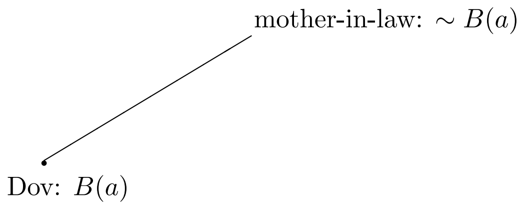


Figure 2.7:

We need a further mechanism (logical or not) to draw a conclusion from this database. We need to know the answer to the following question: is the final conclusion $B(a)$ or is it $\sim B(a)$. In a general setting, we need one mechanism (proof theory) for the relation $\Delta \vdash t : A$, for arbitrary Δ, t, A and another mechanism which we call *flattening* to 'abstract' from the sets $\{t \mid \Delta \vdash t : A\}, \{s \mid \Delta \vdash s :\sim A\}$ a final conclusion $\Delta \vdash A$ or $\Delta \vdash\sim A$ or neither, see Section 1.5.

Example 2.3.2 (Ordered Logic) An ordered logic database is a partially ordered set of local databases, each local database being a set of clauses. The following diagram (figure 2.8) describes an ordered logic database: The local databases are labelled t_1, t_2, t_3, s_1, s_2 and $\varnothing$ and are partially

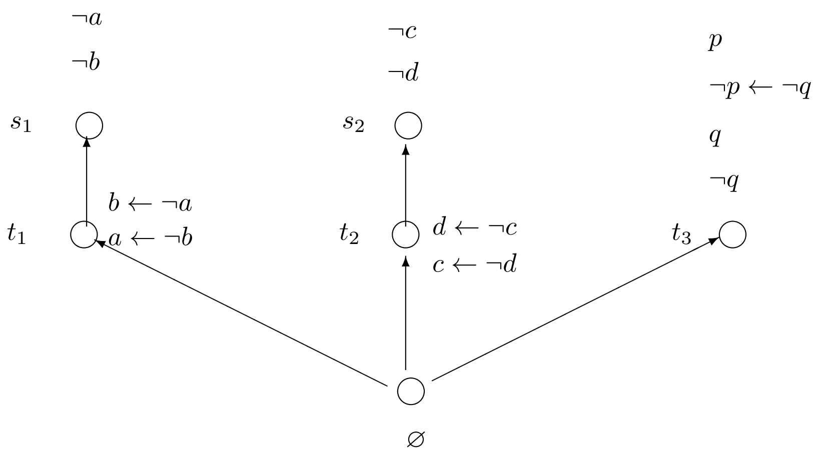


Figure 2.8:

ordered as in the figure.

To motivate such databases, consider an ordinary logic program $C_1 = \{p \leftarrow \neg q\}$. The computation of logic program assumes that, since q is not a head of any clause, $\neg q$ is part of the data, (this is the *closed world assumption*). Suppose we relinquish this principle and adopt the principle of asking an *advisor* what to do with $\neg q$. The advisor might say that $\neg q$ succeeds or might say that $\neg q$ fails. The advisor might have his own program to consult. If his program is C_2, he might run the goal q (or $\neg q$), look at what he gets and then advise. To make the situation symmetrical and general we must allow for Horn programs to have rules with both q and $\neg q$ (ie literals) in heads and bodies and have any number of negotiating advisors. Thus we can have $C_2 = \{\neg q\}, C_1 = \{q \leftarrow \neg q\}$ and C_1 depends on C_2. Ordered logic develops and studies various aspects of such an advisor system which is modelled as a partially ordered set of theories. Such a logic is useful, eg for multi-expert systems where we want to represent the knowledge of several experts in a single system. Experts may then be oredered according to an "advisory" or a relative preference relation.

A problem to consider is what happens when we have several advisors that are in conflict. For example, C_1 depends on C_2 and C_1 depends on C_3. The two advisers, C_2 and C_3, may be in conflict. One may advise $\neg q$, the other q. How to decide? There are several options:

1. We can accept q if all advisors say yes to q.

2. We can accept q if at least one advisor says yes to q.

3. We can apply some nonmonotonic or probabilistic mechanism to decide.

If we choose options (1) or (2) we are essentially in modal logic. To have a node t and to have $?q$ refer to advisors $t_1, \ldots, t_n$ with $t < t_i, i = 1, \ldots, n$ is like considering $?\Box q$ at t in modal logic with $t_1, \ldots, t_n$ possible worlds in option 1 and like considering $\Diamond q$ at t in option (2). Option (3) is more general, and an *LDS* approach is most useful. We see from this advisors examples an application area where the labels arise naturally and usefully. The area of ordered logic is surveyed in [Vermeir and Laenens, 1990].

Example 2.3.3 (Defeasible Logic) *This major approach to non-monotonic reasoning was introduced by [Nute, 1986]. The idea is that rules can prove either an atom q or its negation $\neg q$. If two rules are in conflict, one proving q and one proving $\neg q$, the deduction that is stronger is from a rule whose antecedent is logically more specific. Thus the database:*

$$\begin{array}{l} \mathit{Bird}\ (x) \rightarrow\ \mathit{Fly}\ (x) \\ \mathit{Big}\ (x) \wedge\ \mathit{Bird}\ (x) \rightarrow \neg\ \mathit{Fly}\ (x) \\ \mathit{Big}\ (a) \\ \mathit{Bird}\ (a) \end{array}$$

will entail $\neg$ Fly (a) because the second rule is more specific.

As an LDS *system the labelling of rules in a database Δ is very simple. We label a rule by its antecedent. The ordering of the labels is done by logical strength relative to some background theory Θ (which can be a subtheory of Δ of some form). Deduction pays attention to strength of labels.*

The previous examples motivated the need for the database to be labelled. The following examples motivate how a database in itself can be labelled and how it can in itself serve as a label. This will enhance the analogy between the label and the situation.

Example 2.3.4 (Jethrow's Career) Figure 2.9 is a database labelled S about Jethrow's performance.

S:

Student: survey	good teacher (J),
Letters:	good research (J),
Students:	fatherly figure (J)

Figure 2.9:

The label involved indicates the source supporting the truth of the predicate. The following database, figure 2.10, lists candidates for directorship of a new Max-Planck Institute in Germany.

mp:

(fairly strong, S): candidate (J),
(preferred, t): candidate (H),

Figure 2.10:

The database is labelled *mp*. It contains data about candidates labelled by their source, and some non-numeric evaluation.

Here we see that a whole database can be a label in another. There are two ways of looking at this phenomenon. One is to view the database as being a label. Another is to allow for links between databases via their labels.

Example 2.3.5 (Linked Databases) Consider two lists of data of the form Δ and Γ below.

$$\Delta = (t_1 : A_1, t_2 : A_2, t_3 : A_3, \ldots)$$
$$\Gamma = (s_1 : B_1, s_2 : B_2, s_3 : B_3, \ldots)$$

The two databases come with a link between t_2 and s_3. This can be formally described via a special operator

$$\mathit{Link}\ (\Delta(t_2), \Gamma(s_3), t_2, s_3).$$

The link can be understood in one of several ways:

1. t_i, s_i are situations and A_i, B_j are infons, and the link is a constraint. In this case more information about the link needs to be given.

2. Δ, Γ are two databases. t_i, s_j are the transaction times (in increasing temporal order) in which the items of data were put in the database.

   The links are the synchronisation between some items (ie $t_2 = s_3$).

3. The t_i, s_j are labels naming actions. The declarative formulas A_i, B_j are the respective post conditions of the actions. The link signifies some connection (to be additionally specified) between the actions. For example, the link can mean that one action should precede another.

Example 2.3.6 (Propositional Circumscription) Circumscription is defined semantically, via satisfaction in minimal models. Surprisingly, results of N Olivetti allow one to present an *LDS* discipline for (at least) propositional circumscription.

To explain the idea let $\vdash_m$ denote consequence in minimal models. For this consequence we have, for example, $p \vee q \vdash_m \neg p \vee \neg q$, which does not follow in classical logic. Suppose we try and find a semantic tableaux counter model for the above. In classical logic we try the tableaux construction and if all the top nodes are *closed* then there is no countermodel. For $\vdash_m$ we just change the notion of "*closed*". This can depend on labelling. A more precise study of this theme will be done later. See [Olivetti,].

Example 2.3.7 (Application to Planning) The notion of a structured database allows one to give a logical modelling to actions. To give an idea of how it is done, consider a blocks world example. Assume three blocks a, b and c and the initial state (model) Δ_1 to be {on (a, c), on ($table$, b), on ($table$, a)}. We move c from the top of a by our first action α and put it on the table and then move it again and put it on b by action β. The question is can we model the pair $(\Delta_1, \alpha * \beta)$? Clearly after executing α we get $\Delta_2 =$ { on ($table$, a), on ($table$, b), on ($table$, c)} and after executing β we get $\Delta_3 =$ {on ($table$, a) , on ($table$, b), on (b, c)}. If we want one database to model $(\Delta_1, \alpha * \beta)$, we can use the following structured database, where $0, \alpha, \beta$ are labels.

0. on (a, c)

0. on (*table*, *b*)

0. on (*table*, *a*)

α: on (*table*, *c*)

β: on (*b*, *c*)
$0 < \alpha < \beta$.

If we ask the query ?(on (x, c) we get three answers with three labels: 0: on (a, c); α: on (table, , c); β: on (b, c).

The situtation is no different from a previous Example 2.3.3 where we get Fly(a) and ¬ Fly (a) from different parts of the database. The relative strength of the data will determine which answer is accepted.

To sum up, we are saying we can model actions by adding their results to the data to form a structured database.

Example 2.3.8 (Probabilistic Nets) A typical example is a network of dependencies of the form in Figure 2.11

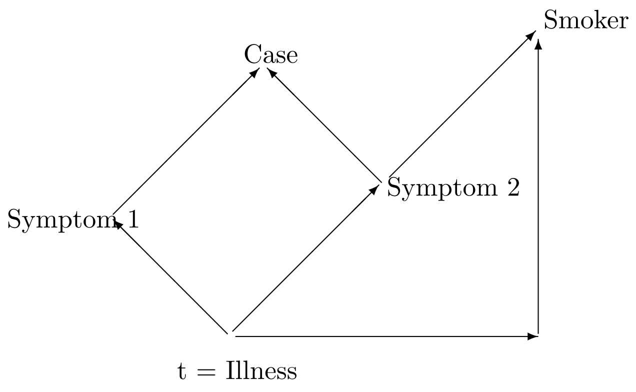


Figure 2.11:

There are various probabilistic dependencies in the network and one can put in numerical values at the nodes. If some of the values are known, the network can predict values at the remaining nodes. Thus
network ⊢ value x: node t
means that the value x is predicted at the node t.

Suppose we put in a value y at the node Smoker. On the basis of that and the other values we get a value x at node t. Suppose the value y: Smoker was obtained from another network, for example Figure 2.12
on the grounds that hippies smoke because they are hippies and yuppies smoke because they are stressed.

The Surgical Cut rule means that if we replace in the first network the node "Smoker" by the second diagram to obtain a new more complex network, then the complex network will give the same answer at t.

A notion of proper replacement (or Substitution) needs to be defined, which will include adjustments of the probability dependencies between nodes. Such adjustment is possible. There are working systems of this kind. One such a commercial system is HUGIN, based on [Lauritzen and Spiegelhalter, 8].

Example 2.3.9 (Databases as labels) *Let $\mathcal{M}$ be the theory of sequences of numbers and $\mathcal{A}$ algebra of labels and* **L** *classical language. Here is a database (figure 2.13).*

$$\begin{tikzcd}
\text{a: Hippie} & & \text{b: Yuppy} \\
& \text{?y: Smoker} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Figure 2.12:

$$s : \boxed{\begin{array}{l} \textit{1. } \ t_1 : A_1 \\ \\ \textit{2. } \ t_2 : \boxed{1.\ r_1 : B_1 \ \ 2.\ r_2 : B_2}. \\ \\ \textit{3. } \ \boxed{1.\ n_1 : C_1 \ \ 2.\ n_2 : C_2, \ \ 3.\ n_3 : C_3} : D \end{array}}$$

Figure 2.13:

The database is labelled s. It has three items in it, $t_1 : A_1, t_2 : A_2, t_3 : A_3$. The first, $t_1 : A_1$ is a pure declarative unit. The second is a labelled database (ie A_2 is a database) and the third is a declarative unit with formula D and a database as a label, t_3.

Example 2.3.10 (Situation semantics (Barwise and Perry)) Situation semantics, introduced in the early 1980's, is based on different basic notions. The primitives of the theory are *situations* (standing for events, properties and relations), and not models and possible worlds. These are partial states of the real worlds. Formally the situation when compared with the traditional notion of a model, should be considered as some limited parts of a model. But these situations can also be elements of situations, standing in relations to one another and to other things.

Types of situations, called *infons*, take the form of a collection of basic facts. The infons, σ, are to be considered as properties holding of situations s. This relation is denoted by $s \vDash \sigma$. These properties divide the situations into situation types. Declarative assertions are called *propositions*, they are comprised of a situation and a type of situation (infon) representing the assertion that the situation is of the indicated type. The notion of *constraints* is understood as a relation between types of situations. This relation is of the form $s_i \vDash \sigma_i \Rightarrow s_j \vDash \sigma_j$, namely: If in some situation s_i, σ_i holds then in some other situtation s_j, σ_j holds, (eg if there is smoke, there was fire). The traditional notion of a consequence relation is replaced by the notion of an information containment system (*ICS*). These are structures $(S, \Sigma, \vDash)$ which contain situations $s \in S$ and infons $\sigma \in \Sigma$ as above, together with a relation of the form $s \vDash \sigma$.

The formal connection of *LDS* with situation semantics is as follows:

- the label corresponds to the situation
- the formula corresponds to the infon.

This connection is not superficial or just mathematically formal. It is basic and conceptual. By connection, however, we do not necessarily mean agreement. By connection we mean that both disciplines focus on more or less the same phenomena and try to address them their own way. Consider the basic intuition of situation semantics, that the traditional models are not adequate and that the basic notion should be that of a limited situation, s, a portion of reality with various facts holding at that context. An infon σ is a bit of information holding in s. If we represent the information of the infon σ by a formula A, then we can annotate the formula by a label t, to indicate the kind of situation s in which the formula holds. Thus $s : \sigma$ becomes $t : A$. We can also represent this intuition by annotating formulas A by s yielding a pair $s : A$, indicating which "contexts" A is to be "applied". Conversely, assume we are given a labelled declarative unit $t : A$, if we want to give it appropriate semantical interpretation in a model, we need to interpret the label t in the model as well as the formula A. If we define a satisfaction relation of the form $M \vDash t : A$ then, by varying t we get various contexts t, within the model M, which can be viewed as the situations M_t with A as the infon, and we can let $M_t \vDash A$ mean $M \vDash t : A$.

The formal connection between the two theories can be given as follows:

1. An ICS is a system $(S, \Sigma, \vDash)$ where S are the situations $\sigma \in \Sigma$ are the infons and $s \vDash \sigma$ is the support relation. This can be turned into a formal *LDS* theory by letting the labels be S, the atomic formulas be σ and take the proof system to be identity (namely we let $A \vdash B$ iff $A = B$ and $\Delta \vdash A$ iff $A \in \Delta$).

2. Conversely, given an *LDS* theory Θ, an ICS can be derived by letting $s \in S$ be the labels, $\sigma \in \Sigma$ be the set of formulas and $\vDash_\Theta$ be defined by $s \vDash_\Theta \sigma$ iff $\Theta \vdash s : \sigma$.

Let us work out a detailed example of an *LDS* which is at the same time an Information Containment System.

1. **A time-location situation theory**

   (a) Let $\{l_1, l_2, \ldots\}$ be a set of indices for locations and $\{t_1, t_2, \ldots, \}$ be a set of indices for times. Let $\{R_1^{n_1}, R_2^{n_2}, \ldots\}$ be a set of preidcates of the indicated arity and $\{a_1, a_2, \ldots\}$ be a set of individuals.

   (b) A *situation* is a set of tuples of the form

   $$\alpha = (t, l, R_i^{n_i}, a_1, \ldots, a_{n_i}, \nu)$$

   where $\nu \in \{0, 1\}$. The meaning of α is that at time t and location l the relation R_i holds (for $\nu = 1$) or does not hold (for $\nu = 0$) between the individuals $a_1, \ldots, a_{n_i}$.

   (c) We write $s \leq s'$ iff for all $\alpha, \alpha \in s$ implies $\alpha \in s'$.

   (d) A set S of situations is called *realist* if S is directed under $\leq$. This notion is equivalent to there being a temporal-location model for S.

   (e) A situation is coherent if it does not contain both $(t, l, R, a_1, \ldots, a_n, 0)$ and $(t, l, R, a_1, \ldots, a_n, 1)$ for the same relation.

   (f) Let the language for infons be many sorted predicate logic with atoms of the form $\mathbf{R}_i^{2+n_i}(t, l, a_1, \ldots, a_{n_i})$ where the first sort is for time variables, the second sort for location and the rest is for individuals.

   (g) An infon is a disjunction of conjunctions of literals of the language. A literal is either an atom or its negation. We write literals $\mathbf{R}(t, l, a_1, \ldots, a_n)^\nu$, to mean $\mathbf{R}(t, l, a_1, \ldots, a_n)$ for $\nu = 1$ and $\neg\mathbf{R}(t, l, a_1, \ldots, a_n)$ for $\nu = 0$.

   (h) Given a situation s and an infon $\sigma = \vee_i \wedge_i \sigma_{ij}$, where σ_{ij} are literals we say $s \vDash \sigma$ iff for some i we have that for all j, $s \vDash \sigma_{ij}$.

   We say $s \vDash \mathbf{R}(t, l, a_1, \ldots, a_n)^\nu$ iff $(t, l, R, a_1, \ldots, a_n, \nu) \in s$.

   We now have an *ICS*, Information Containment System $(S, \vDash \Sigma, \leq)$.

2. **Translating in general an *ICS* into *LDS***

   Let us see how this would be represented as an *LDS*. We first do a general construction and later do an *LDS* system for this particular example.

   To turn any *ICS* system $(S, \vDash \Sigma, \leq)$ into an *LDS*, we take $(S, \leq)$ as our algebra of labels $\mathcal{A}$ and $\mathcal{M}$, and take Σ as our language $\mathbf{L}$. We choose a particular labelled theory Δ to be $\{s : \sigma \mid s \vDash \sigma\}$ and define an *LDS*-consequence from any Θ by $\Theta \vdash \alpha : A$ iff $\alpha : A \in \Theta$. This consequence $\vdash$ will give us back, in the case of Δ, our relation $\vDash$, namely $s \vDash \sigma$ iff $\Delta \vdash s : \sigma$.

The relation $\leq$ satisfies $\Delta \vdash s : \sigma$ and $s \leq s'$ implies $\Delta \vdash s' : \sigma$.

Conversely, given any *LDS* system and any theory Θ, we can define an information containment system (dependent on Θ) by taking:

S	$=$	set of labels
Σ	$=$	set of formulas

and define

$$s \vDash \sigma \quad \text{if} \quad \Theta \vdash s : \sigma.$$

The relation $\leq$ has to be defined externally as $s \leq s'$ iff for all $\sigma, s \vDash \sigma$ implies $s' \vDash \sigma$.

3. **An *LDS* formulation of the time location example**

   We now look at the example in (1) and see how we would naturaly turn it into an *LDS* theory. Since in *LDS* the emphasis is on syntax, we would take as our labels triples (s, t, l) of the situation, time and location. We take as our formulas literals of the form $\pm R(a_1, \ldots, a_n)$. The basic declarative unit is $(s, t, l) : \pm R(a_1, \ldots, a_n)$. The relation $\leq$ can be arbitrarily defined on the labels and can be written as $(s_1, t_1, l_1) \leq (s_2, t_2, l_2)$ meaning the situation s_1 at (t_1, l_1) is part of the situation s_2 at (t_1, l_2). (We may wish to require that $t_1 = t_2$ and $l_1 = l_2$ depending whether you accept that the same situation can arise at different times and places. It is up to us to formulate what we want).

   A theory is a constellation of declarative units. The following is an example of a theory

   $$\{\alpha : R_1(a, b); \beta : \neg R_2(c, d, a); \alpha \leq \beta.\}$$

   α, β are triples (labels).

   The proof theory is very simple:
   $$\frac{\alpha : A, \alpha \leq \beta}{\beta : A}.$$

   This reflects the fact that β contains α.

   Let Θ be any theory, e.g. the above theory. We get an *ICS* from it by letting

S	$=$	$\{a, \beta\}$
Σ	$=$	set of $\pm$ literals
$s \vDash_\Theta \sigma$		iff $\Theta \vdash s : \sigma$

   If we externally define $\alpha \leq' \beta$ iff for all $\sigma, \alpha \vDash \sigma$ implies $\beta \vDash \sigma$, we get that $\leq'$ is the same as $\leq$, since the proof rules ensure that.

   For example, let us show that for the above Θ, we have

   $$\beta \vDash_\Theta R_1(a, b).$$

   Here is the proof:

   (a) $\alpha : R_1(a, b)$ given in Θ

   (b) $\alpha \leq \beta$ given

   (c) $\beta : R_1(a, b)$ by rule

Note that so far, our language **L** does not allow for labels in the formulas. Thus formally situations cannot appear inside infons. This is only a formality because the label algebra $\mathcal{A}$ can be a sublanguage of **L**. In fact the *LDS* discipline allows for label dependent connectives.

Example 2.3.11 (Fallacies) *The reader should note that our point of view and the use of labels is genuinely more general and is capable of yielding more. We describe an unexpected application of our view. There is a serious, well-motivated and well-organised community; the informal logic and argumentation community, studying the nature of human reasoning and argumentation in general and attempting to foundationally explain the role of the fallacies in human arguments. Fallacies* *are argument structures which appear to be correct and convincing, but are actually wrong. Many of them can be effectively used in some situations, but not in others. Any account of real life human practical reasoning must give account of the fallacies. Well known among them is the fallacy Ad Hominem, the fallacy of attacking not the argument but the person presenting it. This kind of reasoning is sometimes acceptable and sometimes not. It is generally considered non-logical, although admittedly extensively used by the human practical reasoner. In our framework, this fallacy has a natural place.*

Consider the notion of a database Δ. This is a structure of declarative units of the form $t : A$, where t is the label and A the formula. The label t annotates A. Suppose the annotation indicates the priority of the formula A and that in an external ordering $<$ gives the relative strength of the priorities. Thus a priority database can be for example

$$\{t : A \to C, s : B, t < s\}$$

t and s can be numbers of algebraic terms and $t < s$ indicates that B has a higher priority than A. This priority can be used in derivation. For example, in the presence of $A \to \neg C, B \to C$ of equal priority, C will be derived.

The data items A and B are formulas of the logic $\mathbf{L}_1$, which is applied to some application area. In many areas it is quite reasonable to have the labels themselves be formulas α, β of another language and logic $\mathbf{L}_2$, describing the origin and nature of the data items, A, B. Some reasoning in $\mathbf{L}_2$ may be available to determine the priority (if any) of α and β. A formula $\Psi(\alpha, \beta)$ and a base theory Θ (possibly dependent on Δ) of $\mathbf{L}_2$ may be used for this purpose, i.e. we have:

$$\alpha \leq \beta \text{ iff } \Theta \vdash_2 \Psi(\alpha, \beta).$$

The simplest condition (in case $\mathbf{L}_2$ has some form of implication) is

$$\alpha \leq \beta \text{ iff } \Theta \vdash_2 \beta \to \alpha.$$

Note that our labels are wffs α of $\mathbf{L}_2$ labelling wffs A of $\mathbf{L}_1$ and the base theory Θ determines the priorities of labels. We now explain the logical force of the fallacy by an example. Suppose we are faced with the following deduction.

$$\begin{array}{l} \alpha : A \to \neg C \\ \beta : B \to C \\ \gamma : A \\ \gamma : B \\ \Theta \vdash_2 \beta \to \alpha \end{array}$$

We must conclude C, because β has higher priority than α. To counter this argument, we may either prove $\neg C$ from additional data or we may attack the source of information, i.e. add Θ_0 to Θ or try and show that $\Theta \cup \Theta_0 \not\vdash_2 \beta \to \alpha$?, (Note that $\mathbf{L}_2$ reasoning is also non-monotonic!). This move appears to us as attacking, not the argument, but its source. However, in the correct context (priority logic) it is a correct move. Other fallacies which are explainable in this framework are Ad Verecundiam, appeal to unsuitable authority, where the labelling is incorrect and fallacies of irrelevance. A systematic study of the fallacies in our context will (hopefully) be done elsewhere. [Gabbay, 1994]

Example 2.3.12 (Dempster–Shafer Rule) *Section 1.6 discussed the feature of aggregating arguments and evidence towards a conclusion. Examples 4.1.1 and 1.6.1 are sample cases. The present example presents a very well known rule of aggregation, the Dempster–Shafer rule. Our exposition relies on [Ng and Subrahmanian, 1994].*

The algebra $\mathcal{A}$ we are dealing with is the set of all subintervals of the unit interval [0,1]. The Dempster–Shafer addition on these intervals is defined by

$$[a, b] \oplus [c, d] = [\frac{a \cdot d + b \cdot c - a \cdot c}{1 - k}, \quad \frac{b \cdot d}{1 - k}]$$

where $k = a \cdot (1-d) + c \cdot (1-b)$, where '$\cdot$', '$+$', '$-$' are the usual arithmetical operations. The compatibility condition required on a, b, c, d is

$$\varphi([a,b],[c,d]) \equiv k \neq 1.$$

The operation $\oplus$ is commutative and associative. Let $\mathbf{e} = [0,1]$.
The following also holds:

- $[a,b] \oplus \mathbf{e} = [a,b]$
- *For $[a,b] \neq [1,1]$ we have $[a,b] \oplus [0,0] = [0,0]$*
- *For $[a,b] \neq [0,0]$ we have $[a,b] \oplus [1,1] = [1,1]$*
- *$[a,b] \oplus [c,d] = \varnothing$ iff either $[a,b] = [0,0]$ and $[c,d] = [1,1]$ or $[a,b] = [1,1]$ and $[c,d] = [0,0]$.*

In this algebra, we understand the declarative unit $[a,b] : A$ as saying that the probability of the event represented by A lies in the interval $[a,b]$. We have, of course

$$\frac{[a,b] : A \to B; [c,d] : A}{[a,b] \oplus [c,d] : B},$$

provided $\varphi([a,b],[c,d])$ holds.

It is also possible to move to a higher language and write clauses of the form

$$t : (t_1 : A_1) \to ((t_2 : A_2) \to (t_3 : A_3))$$

which is more like the way clauses are used in traditional Dempster–Shafer applications. Such languages are studied in Section 8.6.

2.4 Case study: Concatenation logic and linear logic

This section will illustrate the ideas of previous sections by applying them to one case study, that of concatenation logic **CL**. We give a Hilbert system formulation, an *LDS* formulation, and an consequence relation formulation for concatenation logic. We describe its data structures, deduction theorem and semantics. The full details will be developed later, in the chapters on resource logics.

A database in this logic is a sequence of formulas $(A_1, \ldots, A_n)$. To derive B from the database we must "use" the assumptions in the order shown. Further when we use modus ponens $A, A \to B \vdash B$, the ticket $A \to B$ must be supported by assumptions earlier in the data sequence than all those which support the minor premise A. Thus, for example

1. $A \to (B \to C)$
2. B
3. A

does not prove C because we have to start by using (1) and (3) "jumping over" (2). However,

1. $A \to (B \to C)$
2. A
3. B

does prove C because (1) and (2) give $(B \to C)$ then (3) is used and we get C.

Another example is:

1. $A \to (A \to B)$

2. A

This database does not prove B because (2) needs to be used twice. However, the database:

1. $A \to (A \to B)$
2. A
3. A

does prove B.

Another example is:

1. $A \to (A \to B)$
2. A
3. A
4. C

This database does not prove B because (4) is not used. Even if (4) were B (ie $C = B$) we still could not derive B because if we use (4) we are not using all the assumptions and we are not starting our proof at the first one.

Note that in principle there is a lot of scope here to define how we want to use the resource. For example we can say that we want to use the assumption in order but it does not matter where we start or finish, as long as order is preserved.

Thus the database:

1. C
2. $A \to B$
3. A
4. D

will prove B but the database:

1. $A \to B$
2. C
3. A
4. D

will not prove B.

The reader may ask why one should require such restrictions on resource? This particular restriction may or may not have any logical intuitive meaning (maybe it is useful for verification of stack handling in Forth?). We are just technically illustrating the options of resource use.

We have more options for restrictions like those in the following database:

1. $A \to (B \to C)$

2a. : A

2b. B

Our restriction is to use only *one* of any a, b pair. Thus we can deduce only $B \to C$ but not C.

This particular resource restriction arises naturally in applications. Assume we have the data

1. $A \to (B \to C)$

2. X

but we do not know whether X is A or X is B. For example, our research project will be funded and a letter has been sent but we do not know the starting date. It is either October 1 or April 1. H Williams studies such databases in [Kong and Williams, 1991].

The language contains implication $\rightarrow$ only.

Definition 2.4.1 (CL Hilbert System) *1. Consider the Hilbert system defined by the following axiom schemas and rules:*

Axioms

1	$A \rightarrow A$
2	$(A \rightarrow B) \rightarrow ((C \rightarrow A) \rightarrow (C \rightarrow B))$

Rules

3 $$\frac{A, A \rightarrow B}{B}$$

4 $$\frac{A \rightarrow B}{(B \rightarrow C) \rightarrow (A \rightarrow C).}$$

Define $\vdash A$ *as usual in Hilbert system theory.*

2. We now define a consequence relation for concatenation logic.
Our data structures are lists of wffs of the form $(A_1, \ldots, A_n)$*. We define*

$$(A_1, \ldots, A_n) \vdash B \text{ iff } \vdash A_1 \rightarrow (A_2 \rightarrow \ldots \rightarrow (A_n \rightarrow B) \ldots).$$

The above presented **CL** in Hilbert style. Notice that our idea of structured database is exemplified in the fact that our database is a list. We can now formulate the Cut rule.

Theorem 2.4.2 (Cut Rule for CL Hilbert Style) *(1) and (2) imply (3):*

1. $\Delta \vdash A$

2. $\Gamma[A] \vdash B$

3. $\Gamma[\Delta] \vdash B$

Where

$$\begin{array}{rcl} \Delta & = & (A_1, \ldots, A_n) \\ \Gamma[A] & = & (C_1, \ldots, C_m, A, D_1, \ldots, D_k) \\ \Gamma[\Delta] & = & (C_1, \ldots, C_m, A_1, \ldots, A_n, D_1, \ldots, D_k) \end{array}$$

Proof. In later chapters, Theorem 10.1.16. ■

Notice that Theorem 2.4.2 illustrates our notion of surgical cut. A appears in the data structure Γ and this is symbolised by writing $\Gamma[A]$. Δ can prove A and the cut rule says Δ can be substituted for A in the same position in Γ.

To give another example, if the data structure of a logic are diamonds and Γ is something like: and $\Delta \vdash A$ and Δ itself is a diamond configuration then the cut rule will substitute Δ for A in Fig 2.14. The substitution notion will be defined as part of the cut rule.

We now give a consequence relation presentation of **CL**. We denote the consequence relation by $\Vdash$, to compare it with $\Vdash$ of definition 2.4.1.

Definition 2.4.3 (Consequence presentation of CL) Consider data structures in the form of lists $\Delta = (A_1, \ldots, A_n)$. A structured-consequence relation on pairs (Δ, A), written as $\Delta \Vdash A$ is any relation satisfying the following two conditions.

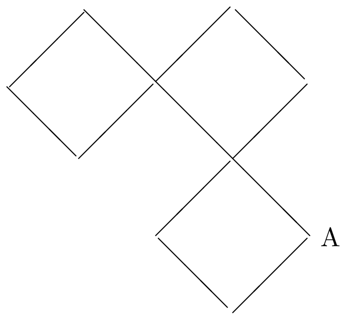


Figure 2.14:

1. **Identity (Restricted Reflexivity)**

$$(A) \Vdash A$$

2. **Surgical Cut (for the data structure)**

$$\frac{\begin{array}{c}(A_1, \ldots, A_n) \Vdash A \\ (C_1, \ldots, C_m, A, D_1, \ldots, D_k) \Vdash B\end{array}}{(C_1, \ldots, C_m, A_1, \ldots, A_n, D_1, \ldots, D_k) \Vdash B}$$

Let $\Vdash_{\mathbf{CL}}$ be the smallest consequence relation satisfying the right hand side for the data structure namely:
deduction theorem

$$(A_1, \ldots, A_n) \Vdash A \to B \text{ iff } (A_1, \ldots, A_n, A) \Vdash B$$

Such a smallest $\Vdash$ exists.

Remark 2.4.4 The previous definition illustrates two points.

1. How to define a logic by conditions on its consequence relation and its data structures.
2. The notion of *the* Cut rule and *a* deduction theorem depend on the data structures.

To further illustrate our ideas, note that the well known intuitionistic logic can be defined in a similar manner, as shown next.

Definition 2.4.5 1. *Intuitionistic Consequence*
Data structures are sets of wffs $\Delta \Vdash A$ is the smallest consequence relation on the data structure satisfying Reflexivity, Monotonicity, Cut and the Deduction Theorem.

2. *R-mingle Consequence*
Data structures are sets of wffs. $\Delta \Vdash A$ is the smallest Consequence Relation satisfying Restricted Reflexivity, Cut and the Deduction Theorem.

3. *Linear Consequence*
Data structures are multisets of formulas. $\Delta \Vdash A$ is the smallest Consequence Relation satisfying Restricted Reflexivity, Cut and the Deduction Theorem.

Notice that the difference between R-mingle and linear consequence is only in the data structures.

Remark 2.4.6 One needs to prove that indeed what is known as implicational linear logic, or implicational relevance logic or implicational intuitionistic logic are indeed what is defined in the previous definition. This is a non-trivial claim. See [Gabbay, 1992a, Gabbay *et al.*, 1994] for proofs. In particular, we will show that $\vdash_{\mathbf{CL}}$ of Definition 2.4.1 is the same as $\Vdash_{\mathbf{CL}}$ of 1.4.3.

We now present an *LDS* discipline for **CL**

Definition 2.4.7 (*LDS* Discipline for CL) *1. The labelling algebra is the free semigroup with associative binary operation $*$ and left identity* **e**. *We thus have*

- $\mathbf{e} * x = x$
- $(x * y) * z = x * (y * z)$.

2. The LDS *logic rules are $\to$ Introduction and $\to$ Elimination for labelled formulas*

$\to$ E rule

$$\frac{\begin{array}{l}\alpha : A\\ \beta : A \to B\end{array}}{\beta * \alpha : B}$$

$\to$ I Rule:
To introduce $\gamma : A \to B$, open a box. Assume $x : A$ with a new atomic (generator) label x and prove using the rules $\gamma * x : B$. Schematically as in figure 2.15:

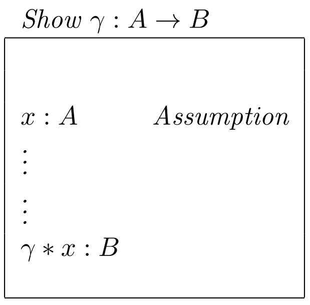


Figure 2.15:

Theorem 2.4.8 *$\vdash A$ in the Hilbert system if and only if* $\mathbf{e} : A$ *(A with label* **e** *) is provable in* LDS.

Proof. Later ■

Definition 2.4.9 (Concatenation (free semigroup frame) Semantics for CL) 1. A **CL** structure is a pair (X, h) where X is a set and h is a function assigning to each atomic proposition q and each finite sequence $\alpha \in X^*$ a truth value, ie $h(\alpha, q) \in \{0, 1\}$, where X^* is the set of all finite (or empty) sequences of elements of X.
The function h is called an assignment.
The function h can be extended to a function h^* on arbitrary formulas via the following recursive equation:

$$\begin{array}{ll}(*) & h^*(\alpha, A \to B) = 1 \text{ iff}\\ & \forall \beta \in X^*[h^*(\beta, A) = 1 \text{ and } (\alpha = \varnothing \vee \beta \neq \varnothing) \Rightarrow h^*(\alpha * \beta, B) = 1]\end{array}$$

where $\alpha * \beta$ is result of concatenating the sequence β to α from the right. * is assumed to be associative, ie

$$(a * b) * c = a * (b * c) = (a * b * c)$$

2. We say that a formula A is valid in the semantics if for all structures (X, h), we have $h^*(\varnothing, A) = 1$.

3. Note the requirement $\alpha = \varnothing \vee \beta \neq \varnothing$ in (*) above. If we relinquish this requirement we need the additional rule
$$\frac{\vdash A}{\vdash (A \to B) \to B}$$

Theorem 2.4.10 *For the Hilbert system* **CL** *we have* $\vdash_{\mathbf{CL}} A$ *iff* A *is valid in the* **CL** *semantics.*

Proof. The reader can verify soundness directly. To show completeness construct the canonical structure by letting X be the set of all wffs of the language and define $h((A_1, \ldots, A_n), q)$, for atomic q to be 1 iff $(A_1, \ldots, A_n) \Vdash q$. Show by induction that

$$h^*((A_1, \ldots, A_n), B) = 1 \text{ iff } (A_1, \ldots, A_n) \Vdash B$$

for arbitrary $A_1, \ldots, A_n$ and B. ■

Remark 2.4.11 *Note that the semantics for linear implication is obtained by taking X^* to be all finite multisets of elements of X, the semantics for relevant implication is obtained by taking X^* to be the set of all finite subsets of X and the semantics for intuitionistic implication is obtained by further requiring that $h(\alpha, q)$ be $\subseteq$ monotonic in α.*

We now give a goal directed computation procedure for the consequence relation $\Vdash_{\mathbf{CL}}$.

Definition 2.4.12 *Let data be lists of wffs and goals be wffs. Consider the following computation rules.*

1. *Identity*
   $(q)?(q)$ *success, for atomic* q.
2. *Deduction*
   $(A_1, \ldots, A_n)?A \to B$
   succeeds if
   $(A_1, \ldots, A_n, A)?B$
   succeeds.
3. *Atomic Rule*
   $(B_1 \to \ldots \to (B_k \to q) \ldots), A_1, \ldots, A_n)?q$
   succeeds if the list $(A_1, \ldots, A_n)$ *can be divided as a concatenation of* k *lists* $(A_1, \ldots, A_{m_1+1}, \ldots, A_{m_k})$ *where* $m_0 = 0 < m_1 < \ldots < m_{k-1} < m_k = n$, *such that all the following queries succeed for* $i = 0, \ldots, k-1$
   $$(A_{m_i+1}, \ldots, A_{m_{i+1}})?B_{i+1}$$

Notice that the ticket $(B_1 \to \ldots \to (B_k \to q) \ldots)$ *must appear* first *in the data structure.*

Theorem 2.4.13 $(A_1, \ldots, A_n) \Vdash B$ *iff* $(A_1, \ldots, A_n)?B$ *succeeds.*

Example 2.4.14 *1.*

$$\varnothing?(A \to B) \to ((C \to A) \to (C \to B))$$

succeeds if

$$(A \to B)?(C \to A) \to (C \to B)$$

succeeds if

$$(A \to B, C \to A)?C \to B$$

succeeds if

$$(A \to B, C \to A, C)?B$$

succeeds if

$$(C \to A, C)?A$$

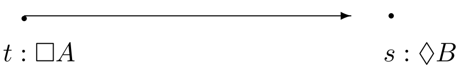


Figure 2.16:

succeeds if

$$(C)?C$$

succeeds if
success.

2.

$$(C \to A, C, C)?A$$

succeeds if

$$(C, C)?C$$

succeeds if
failure.

This fails because the rule of identity does not apply. The reader should compare this example with Definition 2.6.2.

2.5 Case study: Modal logic

Modal and temporal logic is a good motivating example for labels. In essence modal logic deals with information (i.e. formulas) related to different worlds or times and with patterns among these worlds. It is therefore very natural to name and explicitly refer to these worlds and the way they are related. We also have a distinguished world and time which is where we are. Thus an LDS approach, where we use labels to name worlds and a labelling language $\mathcal{A}$ to describe patterns of worlds comes very naturally indeed.

Example 2.5.1 (Some modal rules) *We begin with a simple configuration of Figure 2.16*

The modal axioms and the meaning of $\Box$ dictate to us that in the constellation displayed in Fig. 2.16, A must hold at s. Further, the meaning of $\Diamond$ tells us that there should exist a point r with $s < r$ such that $r : B$.

We can thus state two rules for manipulating modal databases.

$$(*1) \qquad \frac{t : \Box A; t < s}{s : A}$$

and

$$(*2) \qquad \frac{s : \Diamond B}{\textit{create } r, s < r \textit{ and } r : B}.$$

Using the first rule we manipulate the constellation displayed in Fig. 2.16 into the one of Fig. 2.17 and using the second rule we further manipulate it into that displayed in Fig. 2.18. In the predicate case r depends on the free variables of B. The second rule is good for modal logics like **K S4***, etc.*

The axiom of Löb:

$$\Box(\Box A \to A) \to \Box A$$

corresponds to the modification rule

$$(*3) \qquad \frac{s : \Diamond B}{\textit{create } r; s < r \textit{ and } r : B \wedge \Box \sim B}$$

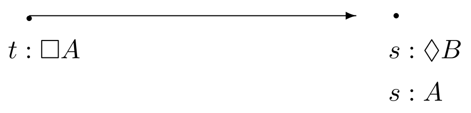


Figure 2.17:

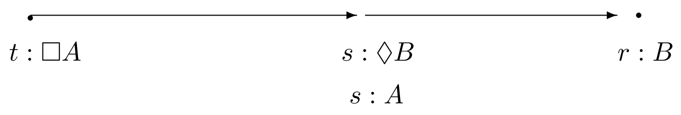


Figure 2.18:

thus in the logic with the Löb axiom we get from the configuration of Fig. 2.16 to the configuration in Fig. 2.19.

It is clear now how the rules work. They allow us to move from one configuration to another and the consequence relation is between configurations. For example, we have Fig. 2.16 ⊨ Fig. 2.18, in modal **K** *and with Löb's axiom we have Fig. 2.16 ⊨ Fig. 2.19.*

The above rules are elimination rules. We still need introduction rules

$$(*4) \qquad \frac{s : A; t < s}{t : \Diamond A}.$$

$$(*5) \qquad \frac{\begin{array}{c}\textit{create an arbitrary } s; t < s\\ \textit{and show } s : A\end{array}}{t : \Box A}.$$

Example for $\Box$ introduction:

Given $t : \Box(A \to B) \wedge \Box A$
Create $s, t > s$
Show $s : B$
Deduce $t : \Box B$.

The picture however is not as simple as it seems. In the usual formulations of modal logics, axioms correspond to conditions on the possible world relation.

In our presentation, axioms correspond to any one of a variety of features. Table 2.1 below offers a selection.

We see here how a second order axiom, i.e. the axiom of Löb, which corresponds to a second order semantical condition, can become a simple movement in LDS. *When* LDS *is translated into two sorted classical logic, the function symbols generating the labels may allow us to reduce the second order condition into first order, as is the case with McKinsey axiom $\Diamond\Box q \to \Box\Diamond q$, when added to* **K** *without transitivity.*

Remark 2.5.2 (Linear Modal Logic) *Suppose we deal with the modal logic for linear frames. Then the configuration in Fig. 2.20. can be expanded in three ways.*

*By rule (*2) we can create a point $u : A$, with $t < u$. In a non linear modal logic such as* **K, S4**, *etc. this would lead us to the configuration of Fig. 2.21.*

one more step would allow us to have $u : A \wedge B$ and hence by $\Diamond$ introduction we get $t : \Diamond(A \wedge B)$.

However in the case of linear modal logic, Fig. 2.21. is not allowed. We need to consider five possibilities.

$$\begin{tikzcd}
t : \Box A \arrow[r] & s : A \wedge \Diamond B \arrow[r] & r : B \wedge \Box \sim B
\end{tikzcd}$$

Figure 2.19:

Table 2.1:

Axioms	LDS **Features**
K axioms $\Box(A \to B) \to (\Box A \to \Box B)$ $\vdash A \Rightarrow \vdash \Box A$ $\Diamond A =$ def$\neg\Box\neg A$	The notion of basic constellation or a diagram, as in Definition 1.5.1 Note that in the modal case the relation R in the diagram is binary. The LDS formulation contains also some simple rules for $\Box$ and $\Diamond$ some of which were shown in the figure above, rules (*1), (*2)
$\Box A \to \Box\Box A$	Transitivity of R in the constellation
$\Box(\Box A \to A) \to \Box A$	In modal semantics the axiom has no first order condition. It corresponds to the finiteness of the frame. In LDS it corresponds to the modification rule (*3).
$\Diamond A \wedge \Diamond B \to$ $\Diamond(A \wedge B) \vee \Diamond(A \wedge \Diamond B)$ $\vee\Diamond(B \wedge \Diamond A)$	Corresponds to the linearity of the relation R. This affects the basic rule (*2) as explained in Remark 2.5.2

1. $t < u < r < s$

2. $t < u = r < s$

3. $t < r < u < s$

4. $t < r < u = s$

5. $t < r < s < u$

If, as a result of each *of these possibilities, we end up with* $t : \Diamond(A \wedge B)$ *then we can conclude* $t : \Diamond(A \wedge B)$.

We have to do that because our databases are linear and the above five configurations are all the minimal possible extensions in which u can be accommodated.

We thus have to modify all the rules with 'create' in them to mean:

$$\begin{tikzcd}
t : \Diamond A \arrow[r] & s : B \arrow[r] & r : B \\
t : \Box B & &
\end{tikzcd}$$

Figure 2.20:

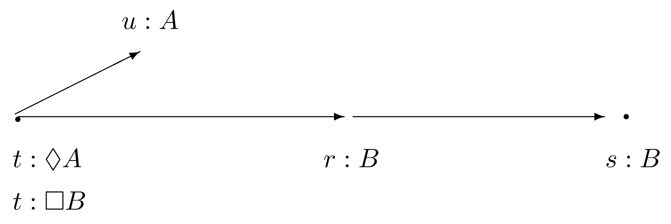


Figure 2.21:

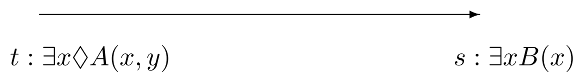


Figure 2.22:

$$\frac{\text{Given initial configuration}}{\begin{array}{c}\text{Split proof into } n \text{ branches according to all} \\ \text{minimal allowed extensions in which the created } u \\ \text{can be accommodated.}\end{array}}$$

(*3) becomes (**3)

$$(**3) \quad \frac{t : \Diamond A \text{ in a configuration}}{\begin{array}{l}\text{create } u\text{, consider all allowed minimal} \\ \text{extensions of } D \text{ with } u \text{ in them. Put} \\ u : A \text{ in and branch the proof. The ultimate} \\ \text{goal of the overall proof must succeed in} \\ \text{all branches.}\end{array}}$$

The above is computationally very expensive. In the example previously given, we need to go to five configurations in order to make the simple move

$$(*6) \qquad \frac{t : \Box A \wedge \Diamond B}{t : \Diamond(A \wedge B)}$$

However our *LDS* proof discipline does not stop us from adopting (*6) as a rule. Recall that the *LDS* discipline tries to enjoy both worlds—the classical world through the labels and the special non-classical world through the language of the formulas in the labels. For each application, desired balance can be sought.

We now come to quantifier rules. We have already assumed that different labels will have different sets of elements in them. To appreciate what this means, we take our clue from modal logic. Consider Fig. 2.22.

At the label t, an x exists such that $t : \Diamond A(x, y)$ holds. This x depends on t and on y. We therefore need a Skolem function $c^t(y)$. The index t is read to mean that c^t was created at t. We thus get $t : \Diamond A(c^t(y), y)$. Hence we can create a node $s : A(c^t(y), y)$. We also must indicate whether $c^t(y)$ 'exists' at the node s. If it does exist at s (probably because of some rules) then we write $s : c^t(y)$. The difference comes out in existential introduction. Suppose we have $s : E(c^t(y))$, can we infer $s : \exists x E(x)$? The answer depends whether $c^t(y)$ exists at s or not. Here are some rules:

$$(*7) \qquad \frac{s : c; s : E(c)}{s : \exists x E(x)}.$$

$$(*8) \qquad \frac{t : \exists x A(x, y_1, \ldots, y_n)}{t : A(c^t(y_1, \ldots, y_n), y_1, \ldots, y_n); t : c^t(y_1, \ldots, y_n)}.$$

$$(*9) \qquad \frac{t : \forall x A(x)}{t : A(u^t); t : u^t,}\ u^t \text{ is a universal constant.}$$

$$(*10) \qquad \frac{s : u^t; s : A(u^t); s : c^r}{s : A(c^r)}.$$

u^t is a new universal constant, r is arbitrary.

$$(*11) \qquad \frac{t : c^r; s : A(u^t)}{s : A(c^r)}\ u^t \text{ a universal constant.}$$

$$(*12) \qquad \frac{s : u^t; s : A(u^t)}{s : \forall x A(x)}\ u^t \text{ a universal constant.}$$

Rule (*9) is analogous to the classical logic rule which allows us to replace $\forall x A(x)$ by $A(u)$, where u is a universal constant, i.e. u is arbitrary. At any stage later in a classical logic proof, we can pass from $B(u)$ to $\forall u B(u)$ provided we discharged all additional assumptions. We can certainly pass from $B(u)$ to $B(c)$, c any constant. The same considerations apply to the labelled case except that we have to watch for the added complication that elements created in one label (world) (e.g. c^r, u^t) may not exist in another label (world). Imagine we have $t : \forall x A(x)$, this means A holds for all elements existing at t. We use rule (*9) and represent $t : \forall x A(x)$ by a universal constant u^t, i.e. we have now $t : u^t$ and $t : A(u^t)$. Suppose for some proof theoretical reason, $s : A(u^t)$ is obained. Thus we really have that A holds at s for an arbitrary element existing at t. Suppose now that we know that the element c^r created at r, exists at t. This is written as $t : c^r$. Then we can deduce $s : A(c^r)$. This is rule (*11).

We now explain rule (*10). Start with $t : \forall x A(x)$, this by rules (*9) and (*12) is equivalent to having $t : u^t$ and $t : A(u^t)$. Suppose that by some proof manipulation we end up with $s : u^t$ and $s : A(u^t)$. This means that the universal constant u^t is in label s and so is $s : A(u^t)$. We understand that as a proof of $s : \forall x A(x)$ from $t : \forall x A(x)$ and so we allow ourselves to deduce $s : \forall x A(x)$. Therefore for any c^r, which exists at s displayed as $s : c^r$, we get $A(c^r)$ at s, i.e. $s : A(c^r)$. This entire chain is summarized as rule (*10).

So far these rules assume that somehow an element c^t created at t ends up available at label s, i.e. $s : c^t$ holds. How do elements move around? We need special rules for that and they differ from system to system. In other words the logic must tell us how elements skolemized in one label can be transported to another label. These are called *visa rules*. Here are two sample rules corresponding to the Barcan and converse Barcan formulas:

$$(b1) \qquad \frac{t : x^r, t < s}{s : x^r}\ x \text{ either a constant } c \text{ or a universal constant } u.$$

$$(b2) \qquad \frac{t : x^r, s < t}{s : x^r}\ x \text{ either constant } c \text{ or a universal constant } u.$$

Example 2.5.3 (Barcan formula revisited) Use (b1) to show that

$$t : \forall x \Box A(x) \vdash t : \Box \forall x A(x)$$

1. Start $t : \forall x \Box A(x)$.
2. $\forall$-Elimination at t yields $t : \Box A(u^t), t : u^t$.
3. Create an arbitrary $s, t < s$.
4. $s : A(u^t), s : u^t$ by $\Box$ elimination rule and visa rule (b1).
5. $s : \forall x A(x)$, by (*12).
6. $t : \Box \forall x A(x)$, since s was arbitrary.

To present modal logic as an *LDS*, we take the usual language of modal logic with $\Box$ and $\Diamond$ as our **L** and take sets D with a binary relation $\leq$ as our labels. It is convenient to think of $(D, <)$ as a configuration.

Definition 2.5.4 *1. A declarative unit is a pair $t : A$ where t is a label and A is a formula of the modal language. A labelled term has the form $t : c_s$, where t, s are labels and c is a constant or variable of the modal language. The double indices for terms are needed because c can be created at label s and be used or be present at label t. We write $t : c_s$ to denote that.*

2. A configuration has the form $(D, \mathbf{f}, \leq, d)$, where $(D, \leq, d)$ is a finite partially ordered set, with $d \in D$ and $\mathbf{f}$ a function, $\mathbf{f} : D \mapsto$ wff and terms such that $\mathbf{f}(t) =$ a set of formulas and a set of labelled terms.

$\mathbf{f}(t)$ can be represented as
$$\{A_1, A_2, \ldots, c^{s_1}, c^{s_2}, \ldots, \}.$$
Note that the terms are labelled arbitrarily.

3. Queries have the form $s : A$.

Definition 2.5.5 A modal *LDS* system is determined by the following components (compare with Definition 3.2.8)

1. A class **K** of partially ordered sets $(D, \leq, d)$ to be used in the configurations of the system.
2. Inference rules. The inference rules manipulate configurations. These include creation and elimination of points $t \in D$ and the introduction and elimination of wffs. The rules are divided into the following categories.
   - 2.1. Introduction and elimination rules for connectives specialised for modal logic.
   - 2.2. Quantifier rules, including Skolemization, as intuitively defined in the discussion above.
   - 2.3. Individual elements *visa* permits (mobility of elements from one node to another). These have the general form
   $$\frac{(D, \mathbf{f}, \leq, d)}{(D, \mathbf{f}', \leq, d)}$$
   where $\mathbf{f}'$ is like $\mathbf{f}$ except that for some c^s and $t \in D$, $\mathbf{f}'(t) = \mathbf{f}(t) \cup \{c^s\}$.

We write
$$t : A; t : c^s; t < r$$
to represent the information that in the configuration $(D, \mathbf{f}, \leq, d)$ which we are dealing with (ie re-writing), we have $t, s, r \in D, t < r$, and $\mathbf{f}(t)$ contains A and c^s.

The notation '$\Delta(t,s); t < s; s : B; s : c^r$; s a new point' means that we are given a configuration $\Delta = (D, \mathbf{f}, \leq, d)$, with $t \in D$. We add a new point s to D to form $D' = D \cup \{s\}$ and extend $\leq$ by stipulating $t < s$ and let $\mathbf{f}'$ be like $\mathbf{f}$ on D and let $\mathbf{f}'(s) = \{B, c^r\}$. Then '$\Delta(t,s), t < s, s : B, s : c^r$' denotes $\Delta' = (D', \mathbf{f}', \leq, d)$.

A rule of the form

$$\frac{t : A}{\text{create } s, t < s; s : B; s : c^r}$$

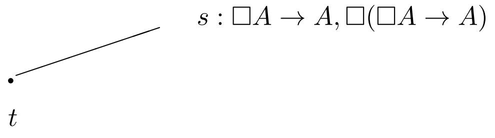
should be understood as

$$\frac{\Delta}{\text{`}\Delta(t,s); t < s; s : B; s : c^r\text{'}}$$

as explained above.

Example 2.5.6 *Show $\Box(\Box A \to A) \vdash \Box A$.*

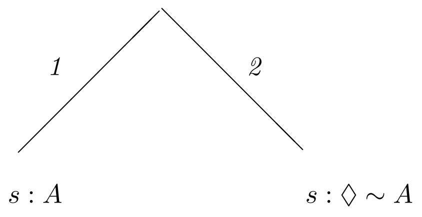
1. *Assume $t : \Box(\Box A \to A)$ and show $t : \Box A$.*
2. *Use an introduction rule. Create an $s, t < s$ and show $s : A$.*
   - 2.1. *Use $\Box$ elimination rule.*



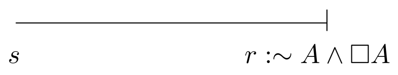
   - 2.2. *Use classical logic to rewrite the entry at s.*

$$s : \Diamond \sim A \vee A$$

   - 2.3. *Split into two cases*



Case 1 is a success because we needed to show $s : A$.
We proceed with case 2 and show it leads to a contradiction:
Create a new point r:



bring $\Box A \to A$ from s.

$$r : \Box A \to A$$

use classical logic and get:

$$r : A$$

a contradiction, because we also have $r :\sim A$.

3. *Since we showed (2) successfully, we conclude $t : \Box A$.*

Example 2.5.7 *Test whether $\Diamond\exists x A(x) \vdash \exists x \Diamond A x$*

1. *$t : \Diamond\exists x A(x)$*
2. *Create s, with $t < s; s : \exists x A(x)$*
3. *Skolemise and get $s : A(c^s)$.*
4. *$t : \Diamond A(c^s)$*
5. *$t : \exists x \Diamond A(x)$, to be derived only if there is a visa for c^s to be at t.*

Example 2.5.8 *Test whether $\exists x \Diamond Ax \vdash \Diamond\exists Ax$*

1. *$t : \exists x \Diamond A(x)$*
2. *$t : \Diamond A(c^t)$*
3. *Create s, with $t < s; s : A(c^t)$*
4. *$s : \exists x A(x)$, to be derived only if there is a visa for c^t to be at s.*
5. *$t : \Diamond\exists x A(x)$*

Example 2.5.9 *Our modal rules can be label dependent rules, for example $\Diamond$ can change meaning from world to world:*

$$\frac{t : \Diamond_t A}{\textit{create } t < s_1 < \ldots < s_{n(t)}, s_{n(t)} : A}$$

The semantic condition for this modality is:

$$\|\Diamond A\|_t = 1 \textit{ iff } \|A\|_{t+n(t)} = 1.$$

Different truth tables in different worlds.

Let us give a quick proof that

$$t : \Box A; t : \Diamond B \vdash t : \Diamond(A \wedge B)$$

1. Initial configuration
$$t : \Box A, \Diamond B$$

2. Create an $s, t < s$ with $s : B$ we get the configuration
$$t : \Box A, \Diamond B; s : B; t < s.$$

3. Move A to $s : A$, using the rule:
$$\frac{t : \Box A \quad t < s}{s : A}$$
we get the configuration
$$t : \Box A, \Diamond B; s : B, A; t < s.$$

4. Add $t : \Diamond(A \wedge B)$, using the rule
$$\frac{s : A, \quad t < s}{t : \Diamond A}$$
we get the configuration
$$t : \Box A, \Diamond B, \Diamond(A \wedge B); s : B, A; t < s.$$

We have thus proved that $\Box A, \Diamond B \vdash \Diamond(A \wedge B)$ because we started with the configuration in (1) with $t : \Box A, \Diamond B$ and we step by step manipulated it into the configuration in (4) which contained $t : \Diamond(A \wedge B)$.

Example 2.5.10 *To show that the logic depends on the class* **K** *of orders, assume we want* $(D, \leq, d)$ *to be linearly ordered. We now try and prove*

$$t : \Diamond A \wedge \Diamond B \vdash t : \Diamond(A \wedge B) \vee \Diamond(A \wedge \Diamond B) \vee \Diamond(B \wedge \Diamond A)$$

To show that assume

1. $t : \Diamond A, \Diamond B$
2. *Create* $s, t < s$ *with*
$$s : A$$
3. *Create* $r, t < r$ *with*
$$r : B$$

However, since only linear orders are allowed our options are

$$t < r < s,\ t < r = s,\ t < s < r.$$

In each of these options, the conclusion can be proved. The lesson to be learnt is that the class of allowed configurations can influence the proof theory.

Example 2.5.11 *We show that*

$$t : \Box A; t : \Diamond B \vdash t : \Diamond(A \wedge B)$$

1. *Initial configuration*
$$t : \Box A, \Diamond B$$
2. *Create an* $s, t < s$ *with* $s : B$ *we get the configuration*
$$t : \Box A, \Diamond B; s : B; t < s.$$
3. *Move* A *to* $s : A$, *using the rule:*
$$\frac{t : \Box A \quad t < s}{s : A}$$
we get the configuration
$$t : \Box A, \Diamond B; s : B, A; t < s.$$
4. *Add* $t : \Diamond(A \wedge B)$, *using the rule*
$$\frac{s : A, \quad t < s}{t : \Diamond A}$$
we get the configuration
$$t : \Box A, \Diamond B, \Diamond(A \wedge B); s : B, A; t < s.$$

We have thus proved that $\Box A, \Diamond B \vdash \Diamond(A \wedge B)$ *because we started with the configuration in (1) with* $t : \Box A, \Diamond B$ *and we step by step manipulated it into the configuration in (4) which contained* $t : \Diamond(A \wedge B)$.

Discussion
Advantages of the *LDS* proof discipline for modal logics:

- No conceptual problem in Skolemising and theorem proving, which is a major problem for modal logic.
- The principles involved are more general, good for any *LDS*.
- We can handle systems which have semantics which is higher order (Löb's system is higher order because it has to be characterised by a higher order condition on the Kripke frame, namely it being a finite irreflexive partial ordering).
- The modal logic can change from world to world, all we have to do is to make our rules label dependent (ie containing labels as parameters).

2.6 Case study: priority logic and PROLOG

Our next case study involving structure is propositional Horn clause computation without negation by failure. This case study is not semantically based. It is based on prioritized proof theory and is a typical challenge to translate to classical logic. Here *LDS* can help. If Δ is a set of Horn clauses and q an atom then the non-deterministic procedural interpretation of $\Delta?q$ should mean $\Delta \vdash_c q$, where $\vdash_c$ is provability in classical logic. Thus since the database $\{q, q \to q\}$ proves q, the computation of $?q$ from this database should succeed. In practice, i.e. in any PROLOG implementation, for this simple example, the database is represented as a list, i.e. either Δ_1

1. q
2. $q \to q$

or Δ_2

1. $q \to q$
2. q

and the computation is deterministic, searching the list either top down or bottom up, unifying with the head of a clause asking for the body. Thus if the implementation scans the clauses top down, we have that $\Delta_1?q$ succeeds while $\Delta_2?q$ loops.

Can we characterize the notion of $\Delta?q =$ success for the top down interpretation?

We can enumerate the database Δ_2 but obviously a translation into classical logic like the one of the modal logic case is not easily available.

It is no use translating Δ_2 into $\{q(1) \to q(1), q(2)\}$ and trying to figure out a truth table for $\to$ much in the same way we did for $\Diamond$ in the previous case study. I don't think we can find some natural translation. Of course one can always translate into classical logic using meta predicates like **database** ('Δ'), **succeed** ('Δ', 'A') etc. but this is not a direct translation. To make our case study more tractable let us change slightly the way the computation works. This will no longer yield computation but another familiar logic. The reason for doing so is simply that at this stage I do not know how to handle the PROLOG case. There is, however, a Gentzen system for it by J. van Benthem [1992]. The change we make is that we ask the pointer to continue moving in one direction only. Thus $?q$ from

1. $q \to q$
2. q

succeeds, because after clause 1 the pointer continues to clause 2.

On the other hand $?q$ from

1. $p \to q$
2. r
3. $r \to p$

fails, because the pointer, having passed clause 2, cannot go back to it.

I am using the word 'pointer' without due explanation. Use your own intuitions. In more complex PROLOG programs execution is much more complex using stacks and several pointers. This database and query has only one goal done in sequence. So it is easy to explain the 'pointer'.

Let Δ be a database in the form of a list of clauses $\Delta = (A_1, \ldots, A_n)$. The pointer is just a number $1 \leq k \leq n$. We query the pair (Δ, k) i.e. $(\Delta, k)?q$. To search for a solution we try and resolve with heads of the clauses $(A_k, A_{k+1}, \ldots)$. Once we resolve with clause j, we ask for the body of the clause. The computation must tell us for each k and j what is the new pointer. This we call the strategy of the pointer. For example if the pointer always goes back to $r = j - k$, we get that $((q \to q, q), 1)?q$ loops, while $((q \to q, q), 2)?q$ succeeds.

The above discussion presented an example where the database was a list. The list was needed for technical reasons, having to do with the implementation strategies of PROLOG interpreters. There is a rich family of systems which require priorities among the data. Among them are defeasible logics, cumulative defaults, Lambek Calculus, truth maintainance and belief revision. They all share the feature that they have no semantics but the data is organized according to priority or some network hierarchy and computation depends on this structure. The structure may come from the application area (defeasible logics, default) or be purely technical (maintaining consistency). This subsection deals with list structure. We would like our case study to be based on an application where the list structure in the database comes intuitively and naturally and has an obvious meaning. We find such a case in legal reasoning.

Consider the area of Esprit projects with which we are all familiar. There are rules for claiming expenses following an Esprit project meeting. These rules may vary slightly from country to country and from university to university. A typical rule could be of the form

$$C(x,d) \to S(x,d,50)$$

which reads:

x spent the day d at a conference $\to$
x gets subsistence of DM 50 for d.

This is a time dependent rule in the sense that it is valid at any time s *after* it was introduced. Thus if the rule was introduced at time t, we can put it at the database as

$$t : C(x,d) \to S(x,d,50).$$

If Dov has been to a conference at time s then the database will contain

$$s : C(\mathrm{Dov}, d).$$

To ask the query $?S(\mathrm{Dov}, d, 50)$ we resolve with the first clause and get $?C(\mathrm{Dov}, d)$. However, we can use the clause $s : C(\mathrm{Dov}, d)$ only if $t < s$.

The above control mechanism is not *temporal control* (as was the case in the previous case study) but rather *prioritized control*,[1] and although the priorities come from temporal considerations, the time does not enter into the computation, only the control strategy of the 'PROLOG pointer'. In fact there are cases where exceptions are made by the commission allowing to claim for conferences before the start time of the rule. In terms of labelled deduction, the rule is that to perform modus ponens on $t : A \to B$ with $s : A$, we must have $t < s$, i.e. the 'fact' must come after the 'rule', where 'after' is not necessarily real 'after' but virtual 'after' including special permission.

Turning back to the application at hand, obviously we are supposed to claim expenses only once for each conference we attend. Thus if there is another rule, say university rule, introduced at time t' which says:

$$t' : C(x,y) \to UP(x,y,150)$$

where $UP(x,y,150)$ means university participation of DM 150, we are not expected to use both rules and get a total of DM 200. Either we claim from Esprit or we claim from the university.

The system may be flexible enough to allow us to claim DM 50 from Esprit and get the rest (DM 50) from the university, but not all systems are like that. The important point is that the assumption $s : C(x,d)$ can be used *only once* in the computation.

Let us summarize the properties we have so far:

- The database is structured with labels which are ordered.

[1] Again, we are not being precise. What is a 'prioritized' system? As a first approximation, say that a labelling system is a *priority system* if the atomic labels form a partially ordered set $(T, \leq, d)$ and the label generating function is concatenation. i.e. the labels are finite sequences of elements of T. The ordering $\leq$ on T is used to define some priority ordering on the sequences. This priority ordering is then used in all proof theoretical rules.

- Modus ponens is allowed only when the ticket (i.e. the implication) is earlier than the minor premiss (i.e. the fact).
- Minor premisses (facts) can be used at most once.

Let us now consider another complication arising in the application area. The above rules are only valid for conferences held in European Community countries. For a conference in America, one needs permission from the project coordinator in Brussels. So we can write a more accurate rule:

$$t'' : AC(x,d) \wedge \mathrm{Per}(x,d) \to S(x,d,100).$$

x participates at an American conference at day d and x has permission then x can get DM 100.

The problem with the above is that Brussels insists that permission be asked *before* conference participation, and not *after* the event. Thus to represent the rule we cannot use ordinary conjunction, but we need the 'first A then B' connective, which we denote by $A \otimes B$. Our rule becomes

$$t'' : \mathrm{Per}(x,d) \otimes C(x,d) \to S(x,d,100).$$

Consider the query $?S(a,d,100)$. We resolve with the above rule and get the query $?\ \mathrm{Per}(a,d) \otimes C(a,d)$. To succeed with that query we need to look at facts after t, succeed with $\mathrm{Per}(a,d)$ first and then succeed with $C(a,d)$ with label bigger than that of $\mathrm{Per}(a,d)$.

We thus get the following additional property for our system:

- To show $A \otimes B$ we must succeed with A and with B, but must show A from an earlier part of the database than B.

So far, we made a distinction between 'facts' and 'rules'. In a full scale logic (not Horn clause) a rule can serve as a 'fact' for another rule. Take for example, the following:

t_2: If subsistence per day is only DM 50, we must appeal to the commission.

This has the form

t_2: $\forall x[C(x,d) \to S(x,d,50)] \to q$.

If $t < t_2$ then the rule can be used to derive q. At time r, how do we know whether the antecedent of t_2 holds? In our case it is explicitly stated as a rule, but in general it may be derivable from a group of rules. How do we check the implicational goal $r : \forall x[C(x,d) \to S(x,d,50)]$? We have to use hypothetical reasoning. We add $C(x_0,d_0)$ to the database and try and derive $S(x_0,d_0,50)$, with x_0 and d_0 a new Skolem constants. We thus get an additional principle.

There are two questions to be settled. First is when we add $C(x_0,d_0)$, i.e. with what priority (time) label? Intuitively the answer is at r, i.e. we add $r : C(x_0,d_0)$. This is not so simple. In general we may be checking $r : A \to B$ and so we want to add $r : A$, but A may have the form $A_1 \otimes A_2$. Do we add $r : A_1 \otimes A_2$ or do we add $r_1 : A_1, r_2 : A_2$, with r_1, r_2 new priority points with the restriction $r < r_1 < r_2$? It makes sense to choose the latter, in which case, if $r < s$ do we also require $r_2 < s$. (i.e. do we put r_2 immediately after r but before anything which comes after r?).

The second question is how do we compute with the additional $r : A$? Do we use the part of the database after $r : A$? or do we say that since our purpose was to determine $r : A \to B$ then 'future data' is not relevant. The validity of $r_1 : A \to B$ now means that data up to now together with r must yield B? These issues we leave for later.

Meanwhile note that rules can be used at any time after they are introduced, and as many times as necessary, while facts (in this particular Esprit example, though not necessarily in general) can be used only once. Thus the rule $A(x) \to B(x)$ can be used as often as needed, but the fact $A(d)$ can be used only once. This is a bit misleading because we can regard the rule as a family of rules for each instantiation of x. Thus we could write a family of rules.

$$\begin{array}{l} A(d) \to B(d) \\ A(e) \to B(e) \\ \vdots \end{array}$$

If $A(d)$ as a fact can be used only once and $A(d) \to B(d)$ can be fired only with $A(d)$ then the rule can be used only once anyway. We get

- Without a real loss of generality we can assume that propositional instances of rules can be used at most once.

We are now ready for an example.

Example 2.6.1 (Esprit travel expenses) Database

$$\begin{array}{l} t_1 : \forall x, d[\ Per(x,d) \otimes C(x,d) \to S(x,d,200)] \\ s_1 :\ Per(Dov,\ date) \\ s_2 : C(Dov,\ date) \\ t' : \forall x, d[C(x,d) \to\ \ UP(x,d,150)] \\ t_2 : \forall x, d[C(x,d) \to S(x,d,50)] \to q. \end{array}$$

Priority

$$t_1 < s_1 < s_2 < t' < t_2.$$

Computation pointer
At any time the pointer resides at some priority label r, i.e. the basic computation structure is $(\Delta, r)?t : G$. The label r is the pointer. The compuation rules tell us which rules we can use to resolve with G. In our particular example we can resolve with any rule with label s. The computation then continues with the new pointer label $max(r,s)$. Formally (we leave the pointer implicit):

- *$\Delta?t : q$ succeeds if $s : q \in \Delta, t = s$.*
- *$\Delta?t : q$ succeeds if for some $s : A \to q \in \Delta, s < t$ and $\Delta?t : A$ succeeds.*
- *$\Delta?t : A \to q$ succeeds if $\Delta \cup \{t : A\}|?t : q$ succeeds.*
- *$\Delta?t : A \otimes B$ if for some $s_2 > s_1 > t, \Delta?s_1 : A$ and $\Delta?s_2 : B$ succeeds.*

Let us ask the query $?s : q$, with $t_2 < s$. We can thus unify with rule t_2 and ask $?C(x_0,d_0) \to S(x_0,d_0,50)$. We add to the database $s : C(x_0,d_0)$ and ask $?s : S(x_0,d_0,50)$. This unifies with t_1 and we ask $?s :\ Per(x_0,d_0) \otimes C(x_0,d_0)$. The last query fails.

If at some future time a general permission is given, then of course the computation will succeed.

It is possible to develop the model further. One can add integrity constraints, to make the database more realistic. Once we have integrity constraints we can deal with conditions $?t : A \to B$ as a goal is computed by letting $t : A$ into the database, which may violate integrity constraints, so some truth maintenance has to be done. We will not go into the details of all our options. A. Martelli, L. Giordano, N. Olivetti and I have a paper on this [Gabbay *et al.*, 1994]. Our purpose here is just to illustrate a realistic use of labels which is purely syntactical. For this purpose our discussion so far is quite sufficient.

Let us give a simplified version of the computation so far in the form of propositional directional N-prolog. The version is a stylized, simplification of the above legal database problem, but it also happens to be a fragment of the Lambek calculus, exactly suited for linguistic application and implemented by Esther Köning [1994].

Definition 2.6.2 (directional N-PROLOG) *Let our language contain $\otimes$ and $\to$.*

1. *Clauses and goals*
   (a) *A is a clause if A is an atom.*
   (b) *A is a body if A is an atom.*
   (c) *If A_i are clauses and q an atom then $(A_1 \otimes \ldots \otimes A_n) \to q$ is a clause with body $A_1 \otimes \ldots \otimes A_n$ and head q.*
   Note that $\otimes$ need not be commutative.

(d) *A database is a list of clauses. We present a database Δ as $t_1 : A_1, \dots, t_n : A_n$, $t_1 < t_2 < \dots < t_n$. A_i are clauses. t_i are priorities and hence $<$ is strict. In case $t_1 = t_2$, we can simply write $t_1 : A_1 \wedge A_2$.*

2. *Let Δ be a database and B a goal. We recursively define the notion of $\Delta \vdash \alpha : B$ where α is a sequence of elements from $\{t_1, \dots, t_n\}$.*

   (a) *$\Delta \vdash \alpha : B$ if B is atomic and $\alpha : B \in \Delta$.*

   (b) *$\Delta \vdash \alpha : B_1 \otimes \dots \otimes B_k \to q$ iff the database*

   $$\Delta; x_1 : B_1, \dots x_k : B_k \vdash \alpha * \beta : q$$

   where $*$ is concatenation and where $t_1 < \dots t_n < x_1 < \dots < x_k$ and β is a subsequence of $(x_1, \dots, x_k)$.

   (c) *$\Delta \vdash \alpha : B$ iff B is an atomic q and for some*

   $$t_i : A_i \in \Delta, \text{ we have } A_i = C_1 \otimes \dots \otimes C_k \to q$$

   and $\alpha = (t_i) * \alpha'$ and

   $$\{t_{i+1} : A_{i+1}, \dots, t_n : A_n\} \vdash \alpha' : C_1 \otimes \dots \otimes C_k.$$

   (d) *$\Delta \vdash \alpha : C_1 \otimes \dots \otimes C_k$ iff Δ can be partitioned into k segments $\Delta = \Delta_1 * \dots * \Delta_k$ and $\Delta_j \vdash \alpha_j : C_j$ and $\alpha = \alpha_1 * \dots * \alpha_k$.*

Example 2.6.3

$$\begin{array}{l} t_1 : A \to B \\ t_2 : C \\ t_3 : A \\ t_1 < t_2 < t_3 \end{array}$$

proves $t_1 t_3 : B$.

Note that we require the pointer to continue to go in the same direction. Thus

$$\begin{array}{l} s_1 : A \\ s_2 : A \to B \\ s_1 < s_2 \end{array}$$

does not prove B.

We need not use all clauses.

Example 2.6.4 (Translation into classical logic) *It is possible to represent the above databases and reasoning structures in classical logic by introducing a new sort for the priority labels and describe the computation in classical logic. It might be rather unnatural. The translation is the following. We need predicate logic with two sorts of variables. The first sort, the $t_1, t_2, s_2, s_2, \dots,$ type of variables, range over an algebra $(A, *, <)$, where $*$ is concatenation, and $<$ is a irreflexive and transitive ordering. The other sort are $x, y, x_1, x_2, \dots,$ type of variables, which are the ordinary predicate calculus variables. Atomic predicates have the form $Q^*(t, x_1, \dots, x_n)$ where $n \geq 1, t$ is the algebra sort variable and x_i are ordinary predicate sort variables. We now translate into this two sorted predicate logic, the Directional N-prolog clauses as follows (τ is the translation):*

- *We associate with each atom $Q(x_1, \dots, x_n)$ of Directional N-prolog a predicate $Q^*(t, x_1, \dots, x_n)$ of the two sorted predicate logic.*
- *$\tau(\alpha : Q(x_1, \dots, x_n)) = Q^*(\alpha, x_1, \dots, x_n)$ where α is a label from the algebra.*
- *$\tau(\alpha : A \otimes B) = \exists t_1 t_2 (\alpha = t_1 * t_2 \wedge t_1 < t_2 \wedge \tau(t_1 : A) \wedge \tau(t_2 : B))$.*

- $\tau(\alpha : A \to q) = \forall t[\tau(t : A) \to \exists s$ *(s* subsequence *of* $t \wedge \tau(\alpha * s : q))$.

The notion of subsequence must be definable in the two sorted classical logic.

The following can be proved.

Lemma 2.6.5 *1. If* $t : A \in \Delta$ *then* $\Delta \vdash t : A$

2. If $\Delta \vdash t : A$ *and* $\Delta \subseteq \Delta'$ *then* $\Delta' \vdash t : A$

3.

$$\frac{\begin{array}{c} s < t \\ \Delta \vdash t : A \\ \Delta' \vdash s : A \to B \end{array}}{\Delta' + \Delta \vdash s * t : B}.$$

Lemma 2.6.6 *Let* **DH** *be the Hilbert system extension of* **CL** *of Definition 2.4.1 with the axiom*

$$A \to (B \to A)$$

Then $t_1 : A_1, t_2 : A_2, \ldots, t_n : A_n \vdash \alpha : B$ *for some* α *which is a subsequence of* $(t_1, \ldots, t_n)$ *iff* **DH** $\vdash A_1 \to (A_2 \to \ldots \to (A_n \to B) \ldots)$.

2.7 Case study: Modelling information flow

This section presents a model of information flow in *LDS*. We want to give a rigorous definition of a process, **Flow**, which begins with two *LDS* databases Δ and Γ and ends up in a new database $E =$ **Flow** (Δ, Γ), which is to be the result of inputting or 'flowing' the data in Γ into Δ. To use the notion of Barwise, we are giving a channel

$$\mathbf{C}(\Gamma) = \lambda x \ \mathbf{Flow} \ (x, \Gamma)$$

satisfying:

$$\Delta \overset{\mathbf{C}(\Gamma)}{\to} E = \ \mathbf{Flow} \ (\Delta, \Gamma).$$

To motivate our notion of information flow, we begin with two very familiar systems, namely strict **S4** implication $\to$ and intuitionistic implication $\to$ and scrutinise the difference between them. It is well known that the formula

$$A \to (B \to A)$$

is a theorem of intuitionistic logic, but is not a theorem of strict implication, unless A is of the form $A = (A_1 \to A_2)$.

Let us try to use natural deduction to prove this 'axiom'.

Our *Goal* is to show $A \to (B \to A)$. For this purpose we start a proof box, say Box a. We assume $t : A$ and try to prove $t : B \to A$, which is the goal of this box. Figure 2.23 illustrates what happens.

The proof will succeed if line 6 is allowed in Box b. Line 6 is justified by bringing data item $t : A$ from line 1 outside the box into it. Intuitionistic logic allows us to bring in any $t : A$ from outside, while strict implication **S4** allows us to bring in only $t : A$ where $A = A_1 \to A_2$, i.e. only formulas in implicational form.

We want to view the above as a dynamic construction process; a construction of a proof. We imagine we are in the middle of a proof in, say, line 5 of Box a. We have successfully derived previous lines and we want to derive line 5 which is to have the form

$$(5) \quad t : A \to B.$$

Box a	1		Show $t : B \to A$
	2	$t : A$	assumption
	3	$t : B \to A$	from box b
Box b	4		Show $s * t : A$
	5	$s : B$	assumption
	6	$t : A$	from 2
	7		

Figure 2.23:

Box a	1		Show $r : C$
	2	$t_1 : C_1$	
	3	$\vdots$	
	4	$\vdots$	
	5	$t : A \to B$	

Box b	1		Link to line 5 in Box a
	2		Show $r' : B$
	3	$s : A$	assumption

Figure 2.24:

The labelled natural deduction rules tell us that to show $t : A \to B$, we must go to a subcomputation recorded in Box b, in which we must assume A with a new label s and *prove* B with label $r'(s)$. We then *exit* Box b and continue to derive further steps in the proof, the immediate next step being recorded as line 6 of Box a. We have thus established line 5 by justifying it in the subproof in Box b. The following figure 2.24 is a better dynamic picture of our process.

The labels involved and the manipulation rules used all depend on the logic. Different logics will have different rules. However, we do see here a common pattern. Databases have goals, they are linked to others and data can be accessible from one database to another according to some logico-geometric rules. The process is dynamic in the sense that a pointer (indicating our current position in the proof process) is resident in some box (e.g. line 1, box b). We are trying to 'bring in' data and use the proof process to achieve the goal of the box it is in by successfully deriving more lines until we reach the goal. Once this goal is achieved, e.g. $r' : B$ is proved in Box b (say in line 7), then the pointer moves back to line 6 in the parent box (Box a), and continue trying to achieve the goal of Box a, namely prove $r : C$.[2]

Suppose we *freeze* the situation in the middle of a proof. What do we see? We see a family of linked databases where each database has a goal attached to it. We also see a *pointer* residing somewhere in one of the databases. We also see proof rules for proving the goals for allowing databass to access formulas and labels from another. The proof rules define the logic to which the databases are the data.

So far so good. How does this connect with the notion of information flow?

The answer is that it is now very easy to abstract from what we have just described and define

[2]Note that although in our example the links indicate subproofs, this need not be the case in general.

a notion of a database ready to receive infomation. Take the instant when the database is frozen in figure 2.24. We are in the middle of our proof, our pointer is at line 1 in Box b and there is data all around us. We have goals to fulfill, the most immediate one being the current goal.

Imagine that suddenly we get an input: $d : D$ from some external source. A labelled formula descends into the system. Where do we put it? Is it meant as an additional assumption for Box b? Is it meant for Box a? Do we start a new box? Do we move the pointer? Well, it is up to us to decide what to do and clearly whatever we decide is part of our *update (information flow) discipline.*

To be specific, let us agree that any input $d : D$ goes into the box where the pointer is, to help prove the goal of that box. Let us further agree that once the goal is proved the pointer goes back to achieve the previous goal, unless instructed otherwise.

Instruct by what? Up to now we were in absolute control of the proof process deciding what to prove next. How can we be instructed?

Obviously we can be instructed by the input! We assumed that we are receiving more data from an external source; we might also receive some instructions from that same source about what to prove next (call it 'hints' if you like). Thus in general the external source can input one of the following:

1. Data items with instructions of where to put them in the existing linked structure.

2. Instructions to start a new linked box. This can be understood as a 'hint' on proof strategy.

3. Instructions to change the existing structure of the existing database. This can be seen as 'information' input, if we understand information not only as raw data of the form $t : A$ but also as the structure and priorities of existing data.

We need to make the above mathematically precise. We need rules associating with each label $d : D$ of input, an algorithmic instruction of what to do with it in the current database. The instructions should be as general and 'functional' as possible. They should depend on the 'proof theory' of existing data and the geometry of the database, as opposed to using local side effects such as line numbers.

Here are some examples of instructions:

- Put $t : D$ among the data to be used in the current box (to help achieve the current goal of the pointer), at the place where the pointer is.

- Replace the current goal by $t : D$.

- Start a new linked box with goal $t : D$ (and do your best to achieve it). The pointer will therefore move to the new box which will serve as a new current box for future inputs.

- t: *truth* can be an instruction to modify (add, change) existing labels by (using) t.

Example 2.7.1 (Linguistic Database) *To be specific, let us describe a particular* LDS *system and a particular information flow function* **Flow** *. Let* $\mathbf{L}$ *be a propositional language with implication* $\to$ *only and atoms. Let* $\mathcal{M}$ *be the theory of finite chains represented as models* $\{1, 2, \ldots, n\}$. *Let* $\mathcal{A}$ *be a set of atomic labels and let* $*$ *be associative concatenation. Compare with Examples 2.2.1 and 2.3.5.*

To get a database ready for input, we need (1) to give each database proof rules and a goal (to prove), (2) we need to locate the pointer and (3) we need to define the notion of accessiblity of data from box to box. To simplify the definitions, and to avoid the need of getting into a new definition of a recursive construction of more complex databases, let us fix the above as follows:

- *let* q *be a fixed atom. Stipulate that the goal of any database is to prove* $\alpha : q$, *where the label* α *is a concatenation of all labels of assumptions which are in the database or are accessible to it. (like in linear logic , we use all our assumptions in achieving the goal).*

- *the proof rule is modus ponens:*

$$\frac{\begin{array}{c} t : A \to B \\ s : A \end{array}}{s * t : B}$$

  notice the 's' comes before the 't'.

- *The geometric accessibility of data is as follows:*

  Let database

$$\Delta = (t_1 : A_1, \dots, t_n : A_n)$$

  be linked to the database

$$\Gamma = (s_1 : B_1, \dots, s_m : B_m)$$

  through linking s_i to t_j (in this direction) then $t_j : A_j$ is immediately accessible to Γ. The accessibility relation is the transitive closure of the above.

- *The position of the pointer can either be indicated explicitly or be geometrically defined.*

- *The input rules are to put data at the left of the location of the pointer until the goal is achieved. Then, unless the next input instructs otherwise, the pointer moves to the next (parent) database through the links. If no such database exists (no links) the pointer starts a new empty database. Note that we are using one look ahead here!*

- *A special label W_o is understood, when coming as an input with* truth, *to instruct the pointer to start a new database and link it to its current position.*

Example 2.7.2 (Linguistic Parsing) *This is just a hint of what we have in [Gabbay and Kempson, 1991]. Consider*
John who sat down smiled.

We understand the above as logical input into the empty database according to agreed lexical meaning (ignore tense):
Lexicon:
John: e
sit: e → t
smile e → t
who: ∅

John, sit, smile *are data,* who *is an instruction to start a new database. When encountering* who *as input the pointer is instructed to start a new database and to put a variable label $u : e$ in and link it to John (the first label of a formula e to the left of the pointer). The goal is always to prove t.*

After the input John who sat *we get figure 2.25*

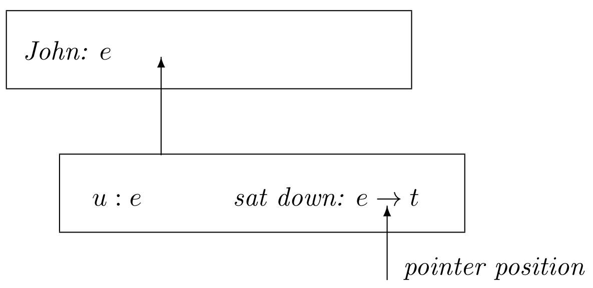


Figure 2.25:

The pointer can prove t with label John sat *in the second database by accessibility. It is now free to move back to the top database unless the next input instructs it to start a new one. (We need one look ahead here). In our case, the pointer, when seeing smiles: e → t, will put it in the top database. Figure 2.26 shows the final position:*

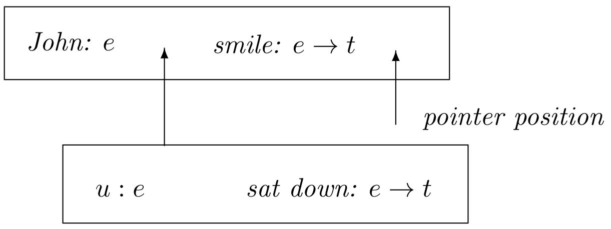


Figure 2.26:

We will not go into more details of what can be done with this linguistic model. See our paper [Gabbay and Kempson, 1991] for details.

Let us summarise what we have so far. We presented a well motivated notion of linked databases and a notion how such a database can receive input. The model seems to be compatible with our proof theoretic intuitions as well as with some lingusitic ones (assuming the Gabbay–Kempson [Gabbay and Kempson, 1991] paper is convincing!). From *LDS* point of view we are simply defining a computational notion of updates and flow of information. From situation theory point of view we have provided a concrete reasonable realisation for sites and information flow.

We have so far defined **Flow** (Δ, Γ) for Γ a sequence of declarative units. We have not defined **Flow** (Δ, Γ) for arbitrary Γ. This we can do if we have a procedure to reduce Γ into linear input. Assume we have some agreed way of reducing Γ into $\Gamma =$ **Flow** $(\varnothing, \mathbf{l}(\Gamma))$ where $\mathbf{l}(\Gamma)$ is a $\mathbf{l}$ sequence of instruction constructing Γ from the empty database.

Given such function $\mathbf{l} = \lambda x \mathbf{l}(x)$ we can now define:

- **Flow** $(\Delta, \Gamma) =$ def **Flow** $(\Delta, \mathbf{l}(\Gamma))$

I'll not go into details of how define the function $\mathbf{l}$. It is not a simple matter.

2.8 Fibred semantics for LDS

This section will motivate the type of semantics required for *LDS*. We motivate our semantics by giving several case studies where this semantics arises naturally and then present the general definition of a fibred structure. Our starting point is the preferential semantics of Kraus-Lehmann-Magidor, given for an arbitrary consequence relation (based on the classical connectives) satisfying reflexivity, transitivity, and restricted monotonicity.

The structures have the form $\mathbf{m} = (W, <, h)$, where W is a set of states, $<$ is a relation on W, h a function assigning to each state $w \in W$ a family $h(w)$ of classical structures.

We write $w \vDash A$ iff A holds in every structure of $h(w)$.

We assume that h is such that the set $h(A) = \{w \mid w \vDash A\}$ satisfies

$$\forall x \in h(A) \exists y \in h(A) \forall z \in h(A)[y < x \text{ and } z < y \text{ imply } z = y].$$

We define $A \mid\!\sim_{\mathbf{m}} B$ iff for every minimal $w \in h(A)$, $w \vDash B$.

If we require of $\mid\!\sim$ to satisfy

$$A \vee B \mid\!\sim C \text{ iff } A \mid\!\sim C \text{ and } B \mid\!\sim C$$

then the function h can be assumed to give only one classical model $h(w)$ for each w. In such a case h can be directly presented as an assignment to the atoms, $h(w, q) \in \{0, 1\}$.

The immediate question we can ask ourselves is why should we restrict ourselves to have a function $h(w)$ giving classical models? We can allow $h(w)$ to yield a class of other models e.g. Kripke models or in fact, other preferential models.

Thus a *Fibred Preferential Structure* is a structure $\mathbf{m} = (W, <, \mathbf{h})$ such that for each $w \in W$ $h(w)$ is a set of fibred preferential structures. This is a recursive definition. Let us not worry at the moment on how to give a non recursive (looping) definition of the concept and ask ourselves under what circumstances we might wish to consider such structures? One such example is the following. Suppose we want to systematically *reflect* a consequence relation $\mid\!\sim$ in the language itself. Thus we want a *new* connective '$\Rightarrow$' with the single defining axiom

$$A \mid\!\sim B \text{ iff } \varnothing \mid\!\sim A \Rightarrow B.$$

This looks like a deduction theorem. In a sense it is, but it does the job of reflecting the consequence relation $A \mid\!\sim B$ (which is metalevel) in the object level through the connective $\Rightarrow$ and $\mid\!\sim A \Rightarrow B$. The question is what is the semantics of $\Rightarrow$, given that we know the semantics of $\mid\!\sim$. Let us try to find out what we need. Assume a preferential semantics for $\mid\!\sim$. Consider $C \mid\!\sim A \Rightarrow B$. Since $A \Rightarrow B$ is a legitimate wff, we should be able to semantically evaluate the above in a model $\mathbf{m} = (W, <, \mathbf{h})$. Thus $C \mid\!\sim A \Rightarrow B$ holds in $\mathbf{m}$ iff for all minimal $w \vDash C$ we have $w \vDash A \Rightarrow B$.

We do not know how to evaluate $w \vDash A \Rightarrow B$ because $A \Rightarrow B$ is supposed to mean $A \mid\!\sim B$ and to evaluate that we need a model $\mathbf{m}'$ not a point (state) w. In the fibred preferential semantics, $\mathbf{h}(w)$ is a set of models $\{\mathbf{m}_j\}$ and we can use the following formal definition:

$$w \vDash A \Rightarrow B \text{ iff } A \mid\!\sim B \text{ holds in all models } \mathbf{m}_j \in \mathbf{h}(w).$$

In fact what we have just been describing is not only an intuitive motivation for fibred semantics, but also a general methodology for reflecting any consequence relation $\mid\!\sim$ into the object language, by adding a *new* connective $\Rightarrow$ (with the axiom $A \mid\!\sim B$ iff $\mid\!\sim A \Rightarrow B$) and by methodologically 'fibering' a semantics for it.

Another example where fibering is needed is when we want to *combine two logics*. Assume that we are given two logics, $\mathbf{L}_1$ and $\mathbf{L}_2$, say $\mathbf{L}_1$ is intuitionistic $\rightarrow_i$ and $\mathbf{L}_2$ is strict implication $\rightarrow_m$. We would like to form a unified logic $\mathbf{L}$ in the language with both connectives $\rightarrow_i$ and $\rightarrow_m$. We want to fibre the semantics of each to obtain a combined semantics.

How do we do it?

Consider $p \rightarrow_i (q \rightarrow_m r)$. The outer connective is intuitionistic. Thus if we regard $x = (q \rightarrow_m r)$ as atomic, we can consider an intuitionistic Kripke model for $p \rightarrow_i x$. This has the form $(S, \leq h)$ where h is the assignment to the atoms (p, q, r) but not to x. We have for all $s \in S$

$$s \vDash p \rightarrow x \text{ iff } \forall s' \geq s (\text{if } s' \vDash p \text{ then } s' \vDash x).$$

We do not know how to evaluate $s' \vDash x$ because $x = q \rightarrow_m r$ and $\rightarrow_m$ is a modal connective.

Suppose now we associate with each $t \in S$ a modal model $\mathbf{m}_t = (S_t, R_t, h_t)$. Then we can say that

$$s' \vDash q \rightarrow_m r \text{ iff } \mathbf{m}_{s'} \vDash q \rightarrow_m r.$$

Of course we know how to evaluate $\rightarrow_m$ in modal models.

$\mathbf{m}_t \vDash q \rightarrow_m r$ iff (say for all $t' \in S_t, t' \vDash q \rightarrow_m r$) iff (for all $t', t'' \in S_t$, if $t' R_t t''$ and $t'' \vDash q$ then $t'' \vDash r$).

Of course these models $\mathbf{m}_t$ must satisfy the conditions of the intuitionistic semantics i.e. $t \leq s$ and $\mathbf{m}_t \vDash q$ imply $\mathbf{m}_s \vDash q$ for all atoms q. So we cannot arbitrarily associate models m_t to each t of the intuitionistic model. They must satisfy the coherence conditions of these models.

In the general case, we need the following notions:

1. Each semantics has a clear notion of satisfaction of a formula in the model, i.e. model $\vDash$ formula.

2. Each semantics and model in the semantics can identify the notion of a *basic semantic unit* t (possible world, state) in which each atom q is evaluated $(t \vDash q)$.

Given (1) and (2) we can *fibre* the semantics together by adding functions $\mathbf{F}^{\mathbf{m}}(t) = \mathbf{n}$ giving for each model $\mathbf{m}$ of each semantics and each semantic unit t of $\mathbf{m}$ a model $\mathbf{n}$ of the *other* semantics.

The above discussion motivates the prototype definition of fibred semantics below. For more details on semantics and on combination of logics see the corresponding chapters of the book.

Definition 2.8.1 (Fibred Semantics) *We define a prototype semantics. A full study of semantics is given in a following chapter.*

1. *Let $\mathbf{W}$ be a model of an algebraic language of the form $\mathbf{W} = (W, <_i^w, f_j^w)$ where W is a set, $<_i^w$ are relations on W and f_j^w are functions on W.*
2. *Let* LDS *be a labelled deductive system with algebra $\mathcal{A}$ and language $\mathbf{L}$ with connectives $\sharp_i$ with n_i places, respectively, as defined in Definition 1.5.1*
3. *We say a structure of the form $\mathbf{m} = (\mathbf{W}, S, \mathbf{a}, R_\sharp, g, D, h, \pi, \mathbf{F})$ is a* fibred point structure *for the* LDS *system $(\mathcal{A}, \mathbf{L})$ if the following holds.*

3.1. *$\mathbf{W}$ is a model for the language $\mathcal{A}$, ie each relation $<_i$ and fucntion symbol f_i of $\mathcal{A}$ have their interpretations $<_i^W$ and f_i^W in $\mathbf{W}$.*

3.2. *$S \subseteq W^2$, giving for each $w \in W$ a subset $S^w \subseteq W$, and $\mathbf{a}$ is a function such that $\mathbf{a}(w) \in S^w$.*

3.3. *$R_\sharp$, for each connective $\sharp(A_1, \ldots, A_n)$ of $\mathbf{L}$, is a function giving for each $w \in W$, an $n+1$ neighbourhood relation $R_\sharp(w) \subseteq S^w \times (2^{S^w})^n$ i.e. it is a relation of the form $R_\sharp^w(y, Q_1, \ldots, Q_n), y \in S^w, Q_i \subseteq S^w$.*

3.4. *g is a function giving for each ground term t of $\mathcal{A}$ and each $w \in W$ an element $g^w(t) \in S^w$. We require that $g^w(f(t_1, \ldots, t_k)) = f^w(g^w(t_1), \ldots, g^w(t_k))$.*

3.5. *D is a non empty domain.*

3.6. *h is an assignment function, assigning to each variable or constant x and each w a value $h(w, x) \in D$.*

3.7. *π is a predicate assignment assigning for each $w \in W$ and each $y \in S^w$ and each n place predicate or function symbol P from $\mathbf{L}$ a corresponding n place relation or function $\pi^w(y, P)$ on D.*

3.8. *$\mathbf{F}$ is a binary function on W, assigning to each w and y from W another $z = \mathbf{F}^w(y) \in W$, such that $y \in S^z$.*

Example 2.8.2 (Modal logic semantics) *Consider propositional Kripke structures for modal logic. These have the form (S, R, a, h) where $R \subseteq S^2, a \in S$ and h is the assignment. Consider the modal language with $\Box$ and the algebra of labels with atomic labels T and relation $<$. We turn the Kripke model into a fibred model as follows.*

1. *Let $W = S$*
2. *Let $y \in S^w$ iff for some natural number $n \geq 0$, $wR^n y$ (R^0 is equality). Let $\mathbf{a}(w) = w$.*
3. *Let $R_\Box^w = R \upharpoonright S^w, h^w = h \upharpoonright S^w$.*
4. *g has to be defined on the elements of T.*
5. *Let $\mathbf{F}^w(y) = y$.*

Thus (S, R, a, h) is transformed into $(S, S^w, a, R, g, h, \mathbf{F})$.

Essentially we are regarding the Kripke model $\mathbf{m}_y = (S^y, R \upharpoonright S^y, y, h \upharpoonright S^y)$ as the Kripke model associated with $y \in S^w$. i.e. $\mathbf{F}^w((y) = y)$.

Definition 2.8.3 (Satisfaction in fibred structures) *Let Δ be an* LDS *database. Let* $\mathbf{m}$ *be a fibred structure for the language. We define the notion $y \vDash_w \Delta$, for $w \in W, y \in S^w$, by induction as follows:*

1. *Δ is a declarative unit of the form $t(y_1, \ldots, y_m) : A(x_1, \ldots, x_n)$*

1.1. *A is atomic:*

$$y \vDash_w \Delta \textit{ iff } y = t^w(h^w(y_1), \ldots, h^w(y_m)), \textit{ and } (h^w(x_1), \ldots, h^w(x_n)) \in \pi^w(y, A)$$

1.2. *A has the form $\sharp(B_1, \ldots, B_k)$:*
In this case we have

$$y \vDash_w A \textit{ iff } R^w_\sharp(g^w(y), Q_1, \ldots, Q_k) \textit{ holds}$$

where $Q_i = \{y \in S^w \mid y \vDash_w B_i\}$

2. *$\Delta = (D, \mathbf{f}, a)$ where D is a finite diagram of labels and for each $t \in D, \mathbf{f}(t) = \Delta_t$ is a database.*

   $y \vDash_w \Delta$ iff $g^w(a) = y$ and the diagram D is satisfied in the model $\mathbf{W}$ through the function g^w. (i.e. whenever $t <_i s \in D$ then $g^w(t) <^w_i g^w(s)$ holds in the model $\mathbf{W}$) and for each $t \in D$,

$$\mathbf{a}(\mathbf{F}^w(g^w(t))) \vDash_{\mathbf{F}^w(g^w(t))} \Delta_t$$

3. *$\Delta = (D_1, \mathbf{f}_1, a_1) : (D_2, \mathbf{f}_2, a_2)$.*

   This is the case where we have a whole database serving as a label.

   $y \vDash_w \Delta$ iff $g^w(a_1) = y$ and the diagram D_1 is satisfied in the model $\mathbf{W}$ through the function g^w and for each $t \in D_1$ the following holds:

   If $g^w(t) \vDash_w \mathbf{f}_1(t)$ then $\mathbf{a}(f^w(g^w(t))) \vDash_{\mathbf{F}^w(g^w(t))} (D_2, \mathbf{f}_2, a_2)$.

   To understand the intuition behind this case consider Case 2 for a database $\Delta = (D_1, \mathbf{f}_1, a_1)$, where $\mathbf{f}_1(t) = (D_2, \mathbf{f}_2, a_2)$ for all t. In this case we require $y \vDash_w \Delta$ iff $g^w(a_1) = y$ and $\mathbf{W}$ is a model of D_1 and for all $t \in D_1$ $\mathbf{a}(\mathbf{F}^w(g^w(t))) \vDash_{\mathbf{F}^w(g^w(t))} \mathbf{f}_2(t)$. The only difference between this definition and our case is that we require the further qualification $y \vDash_w \mathbf{f}_2(t)$. Thus the set of points at which we evaluate $\mathbf{f}$ is not only required to be in D_2 but also to support $\mathbf{f}_1$.

4. *We say $\Delta \vDash_{\mathbf{W}} \Delta'$ iff for all $w, y \in W$, if $y \vDash_w \Delta$ then $y \vDash_w \Delta'$.*

Example 2.8.4 *Consider an* LDS *system with no atomic labels and a language* $\mathbf{L}$ *with no connectives, just with atomic propositions $\{p, q, r, \ldots\}$. The definitions of labels and databases (definition 1.5.1) reduces to the following:*

1. *A database is a formula*

2. *If Δ, Γ are databases, so is $\Delta : \Gamma$.*

3. *A label is a database*

4. *A declarative unit has the form $\Delta : A$, where Δ is a database and A a formula.*

Note that if we read $\Delta : \Gamma$ as $\Delta \Rightarrow \Gamma$ then the databases (labels) will syntactically be identical with all formulas in the language with implication. This suggests that we understand a formula of concatenation logic $A \rightarrow B$ as $A : B$; A is the label of B. This view has support from applications. The statements 'if A then B' or 'when A then B' do have the meaning of A qualifying the extension of the truth set of B, and so A can be viewed as labelling B.

The semantics for this case also simplifies. We have no need for the algebraic language for labels. We need no $(R_\sharp, g, D)$. All we need is $(\mathbf{W}, S, \mathbf{a}, h, \mathbf{F})$. The clauses for the semantics become:

1. *For each $w \in W, S^w \subseteq W$.*

2. *h^w is an assignment, giving for each atom q and $y \in S^w$ a truth value $h^w(y, q)$.*

3. *$\mathbf{F}$ is a binary function as before, and $\mathbf{a}(w) \in S^w$ selects the actual world of the model indexed by w.*

4. *Satisfaction reduces to the following:*

$$y \vDash_w A : B \text{ iff } y \vDash_w A \text{ implies } \mathbf{a}(\mathbf{F}^w(y)) \vDash_{\mathbf{F}^w(y)} .$$

Assume now that $h^w(y)$ is independent of w. (i.e. $h^w(y) = h(y)$ for all w) and that $W = S^w$ for all w. Let $(W, +, 0)$ be a semigroup with addition and left unit 0. Let $\mathbf{F}^w(y) = w + y$ and let the function $\mathbf{a}$ be identity, $\mathbf{a}(w) = w$.

Then by definition of satisfaction

$$\begin{aligned} \vDash A : B & \text{ iff } \forall w, y[y \vDash_w A : B] \\ & \text{ iff } \forall w, y[y \vDash_w A \text{ implies } \vDash_{w+y} B] \\ & \text{ iff } \forall w, y(y \vDash_w A \text{ implies } w + y \vDash B). \end{aligned}$$

The above condition is exactly the truth condition for $w \vDash A \to B$ of concatenation logic.

2.9 IPR - Interactive practical reasoning systems

We call this section Interactive Practical Reasoning systems or logics because deduction is only one aspect of practical reasoning activity. The task of this section is to clarify these concepts and show the place of *LDS* within this framework, namely within *IPR*. Needless to say, I present here my current views, which will undoubtedly be modified, as the book progresses and as I receive (I hope) the criticism of my colleagues and readers.

We begin by distinguishing two orthogonal aspects of *IPR*, Interactive Practical Reasoning systems. These are:

1. A single data structure vs a network of data structures.

2. Declarative + procedural activity vs imperative interactive activity.

The first dimension, single vs several databases is very simple to explain. It is the difference between dealing with distributed and non-distributed knowledge. As we shall see later, classical logic will be a deductive system for a single data structure, while modal logic will be a deductive system for several structures.

The declarative vs imperative distinction can be summed up in a nutshell as the difference between *querying* a database and *interacting* with it, (for example updating it). Any deductive system or a theorem prover or even a logic programming system is a declarative procedural system, while a system of action logic or a truth maintenance system or even a logically based metalevel planning system are imperative interactive activities.

The following matrix (fig 2.27) summarises our features with some examples. A full discussion follows.

The Declarative-Procedural Aspect

We begin with a single database. Here the basic situation is that of a structured database Δ and a query Q from it, i.e. $\Delta?Q$. This situation was described in Chapter 1, Section 3. Our view in Section 1 was that a logical system (i.e. a single database, declarative procedural system in the terminology of this section) is a pair $(\vdash, \mathbf{S}_\vdash)$, where $\vdash$ is a consequence relation (satisfying some minimal conditions) and $\mathbf{S}_\vdash$ is an algorithmic system for it, as described in Section 1.

The class of a single database declarative system includes a variety of both monotonic and non-monotonic logics. A crude classification, just to illustrate our point is into at least three categories:

Interactive Practical Reasoning

	Single data structure	Network of data structures
Declarative	• Classical Logic • Linear Logic • Defeasible Logic • Circumscription	• Modal Logic • Temporal Logic • Belief Systems
Imperative	• Truth Maintenance Systems • Question Answering • Abduction	• Executable Temporal Logic • Action Logics • Machine Learning • Dialogue Logics

Figure 2.27:

1. Systems which read the connectives in different ways, eg classical, intuitionistic and Łukasiewicz logics. These systems interpret the implication differently. The varieties of consequence relations stem from the different readings.

2. Resource logics; or systems which differ in the way they handle the assumptions. Such examples are relevance and linear logics.

3. Systems relying on structured database, where the reasoning and data is structured, and the deduction process is driven by the structure. These include inheritance systems, ordered logics (of section 2.3) and priority logics.

Turning now to the network aspect, we observe that each one of the above single database logics can be developed and extended into a multiple, network database logic. If we take several databases and form a network, we need two logical components. The local logic, the logic of each database characterises how we get queries answered locally. We also need a global logic, the network logic, because we want to talk about connections between the different local databases. For this purpose, we need connectives or operators in the language which express the connection. The local logic or logics must have these additional connectives. To illustrate, consider a very simple network of two databases Δ_1 and Δ_2 with consequence relations $\vdash_1$ and $\vdash_2$. The language is that of classical logic with an additional connective N. $\vdash_1, \vdash_2$ can be taken as classical. From the point of view of a single database, NA is considered atomic. The single database, when queried $\Delta?NA$, treats it as atomic. Recognising N as a network operator, Δ_1 "sends" A to the appropriate place for querying, in this case to Δ_2. Thus

$$\Delta_1?NA = 1 \text{ iff } \Delta_2?A = 1$$
$$\Delta_2?NA = 1 \text{ iff } \Delta_1?A = 1$$

Written in this way, Δ_1 and Δ_2 need not even be answering queries using the same logic. The network logic is the discipline of where to send A when queries $?NA$, and is independent of the "local" logics, $\vdash_1$ and $\vdash_2$.

Thus, for example, modal logic is a network logic with a discipline for $\Box$. In modal logic, if we take each node Δ_i with classical logic as its logic $\vdash_i$ (ie $\Delta_i?A$ is computed in classical logic), we get *classical modal logic.* It is therefore wrong for example to refer to modal logic (based on classical logic) as a *non-classical logic.* It is classical logic extended with additional network connectives. We can equally have intuitionistic modal logic or a non-monotonic modal logic. An example of a modal relevance logic was given in section 2.2.

A single database logic can be structured. For example the database can have two assumptions with the proviso that one can use exactly one of them. Here the structure is used to control deduction. Of course a network of databases can be amalgamated into a single, structured database, and the notion of a network can be subsumed in the notion of a single database. This is mathematically possible to do but not always profitable to do. Different approaches have their advantages. This book will clarify (I hope) some of the problems involved. We certainly have to know all the translation and reduction options and of their conceptual and logical standing. Chapter 7 of the current draft does a bit of that.

The network of databases can give rise to a very intricate and fascinating systems. Consider a single node t in the network. Its database is Δ_t. Its logic is $\vdash_t$, with $\mathbf{S}_{\vdash_t}$ as its algorithmic system. The local node t regards each network connective N as foreign and when queried it, ie $\Delta_t?NA$, it refers to the global network logic for advice. The network logic, if it is modal for example, may send A to be queried in nodes $s_1, \ldots, s_n$ accessible to t. Thus we reduce $\Delta_t?NA$ to the queries $\Delta_{s_i}?A$ and if N is necessity, we expect all $\Delta_{s_i}?A$ to succeed. This is a requirement of the network logic. It is independent of the local logics of node s_i (ie $\vdash_{s_i}$). It is characterised by the first order global condition

$$\mathbf{Hold}(t, NA) \text{ iff } \forall s[s \textbf{ accessible} \text{ to } t \to \mathbf{Hold}(s, A)]$$

There is no reason why we should have a classical global condition. One can have a non-monotonic probabilistic condition, for example:

NA **holds** at t if A **holds** in most worlds **accessible** to t.

The above illustrates that we truly have two types of logics here. The local and the global.

In fact in the most general case the global logic may be locally dependent. For example NA holds at t iff A holds in *Most*(t) worlds accessible to t, where *Most*(t) depends on t.

More on this in the chapter on label dependent connectives.

We conclude with an example from the world of word processors. Take for example MacWrite. It allows you, when typing a letter, to open a window t_1, and take part of the text and work on it. You can open a secondary window t_2 and take some text from t_1 and work on it. Continue in this manner and get a sequence of windows $t_1, \ldots, t_n$. Suppose we now want to take a piece of text from window t_i and place it in window t_j. Can we do it? The way MacWrite manages this is an example of a network logic. In fact, MacWrite does allow for arbitrary movements from one window to another and therefore its network logic is the modal logic **S5**. Thus MacWrite is an **S5** word processor.

The Interactive and imperative reasoning Aspect

When we query a single database, symbolised as $\Delta?Q$, we want an answer. We may need a procedure to compute the answer. We may reduce the problem of the database and query to other databases and queries.

$$\Delta?Q = 1 \text{ if } \Delta_i?Q_i = x_i i = 1, \ldots, n, x_i = 0, 1$$

But we are not changing the database nor are we interacting with it. There is the aspect of interacting with our data when encountering a query. This aspect of logic is new and exciting. Dialogue logic, and in computing, human computer interaction, human assisted deduction, etc, are extreme examples of a logic arising out of two interacting agents. The pure logician who may be reading this section may object to my use of the word "logic" here. His patience and goodwill may be already stretched to the limit by my treating non-monotonic logics as "logics". However, please bear with me a little longer.

Consider the query $\Delta?Q$. In the declarative aspect, we want an answer. In the interactive case, we trigger an action. Q is read imperatively. We can write $\Delta!Q$ to stress this fact.

The result of the interaction is Δ'. Thus $\Delta!Q = \Delta'$. As an example of action assume that we have integrity constraints on databases Δ. We want to add Q to Δ and applying the integrity constrains checker to yield the resulting Δ'. Thus $\Delta!Q = \Delta'$. We are assuming here a procedure for maintaining integrity constraints.

Another example is Abduction. We describe how it arises in many linguistics contexts. We have a database Δ and a deductive system $\mathbf{S}_\vdash$. When we query $\Delta?Q$, we apply $\mathbf{S}_\vdash$ to find out whether $\Delta \vdash Q$. Suppose we want to know what is the minimal Δ_0 to add to Δ so that to ensure that $\Delta, \Delta_0 \vdash Q$. We assume that there are restrictions in the system as to what we can add,

which can be either proof theoretic or contextual (structured) etc. Otherwise we simply add Q, ie $\Delta_0 = \{Q\}$ Δ_0 can be the *abductive* set to be added. It can also arise in query answering of the form
Question: Does $\Delta \vdash Q$?
Answer: Yes, if Δ_0 holds.

Again we have an imperative aspect here

$$\Delta!Q = \Delta \cup \Delta_0$$

Notice that there are three systems involved here.

- The logic of the Δ's
- The logic of the imperative procedures (integrity constraints, or abductive procedures etc)
- A metalogic that relates the two previous logics

2.10 The role of LDS

How does *LDS* fit into the *IPR* picture? *LDS* can manipulate labelled formulas, by manipulating both formulas and their labels in some prescribed ways. The use of labels allows us to include extra information via properties of labels. This information can be any of the following:

- Structural and/or resource information for a single database.
- Network information for several databases.
- Metalevel information for the imperative aspect.

Thus *LDS* can serve as the underline logical tool for describing and presenting the different logics in fig 2.2.7.

To show how this is done is the object of this book.

1. The need for labelled deductive systems

- The first step was the realization that practical applications which lend themselves to possible logical analysis contain several independent and related structures. These structures must be recognized by any logic which we use to describe and reason about the application.
- The second step is that we have an option, either to use traditional logics, such as classical logic to represent these structures or to use specialized logics. Careful analysis of two case studies shows conceputally we might wish to use specialized logics, but computationally we are better off with classical logic.
- The third step is to look for a good framework which could give us the kind of logics we want for applications. The proposed framework was labelled deductive systems, where the declarative units are of the form Labels: Formulas. The important intuitive point is that we agree to manipulate the existing natural structures of the application together, side by side, without reduction or translation.

Our case studies showed that additional structure can come from semantics (bringing semantics into the logic, as in the case of temporal logic) or from resource or proof theoretic considerations (bringing priorities as labels). In either case the formalism (algebraic *LDS*) is very similar. In fact, some successful widespread formalisms such as the A-Box reasoning of KL-one is already a example for an *LDS* in our sense. Therefore we propose to use this formalism (*LDS*) in application areas where logic is needed. The practical usefulness and applicability of such a program can be demonstrated. We are currently trying to do exactly that. The reader can reflect upon his own personal experience to see how useful the labels can be. The relevance of *LDS* to the question of the universality of classical logic is that any *LDS*, either syntactically presented or semantically presented, can be translated into classical logic.

The perceptive and critical reader may have already asked himself why we need to present and use *LDS* as an intermediate step in the translation of a logic **L** into classical logic. Semantical translations are known, and given a semantics for the logic, we can translate the semantics directly into classical logic and thus obtain a translation. This can certainly be done for many modal and temporal systems. We need not formulate these systems as *LDS* and then translate the *LDS* into classical logic. However, *LDS* is necessary for the translation of logics which have no known semantics, for example, any logic formulated proof theoretically through elimination and introduction rules. We need to take the following steps:
Step 1: Formulate the logic proof theoretically as an algebraic *LDS* of the kind described in this section (especially note that the introduction rules depend definitionally on the elimination rules, which may not be the case in the original logic).
Step 2: Translate the *LDS* formulation into classical logic.

The perceptive reader might continue to ask that once we manage to produce the *LDS* in Step 1 above, do we not now have a semantics, i.e. a term semantics arising from the rules? The answer is that it is not necessarily the case. In a proper semantics the semantical value of '$\sharp(A_1, \ldots, A_n)$' is reduced to the values of 'A_i'. We allow elimination rules which reduce a formula to other formulas, not necessarily subformulas of it, as long as some general measure of complexity is reduced and a very long path involving all connectives may be involved in the reduction. It is not at all clear that any semantics can be extracted out of this reduction. More on these points will be given in later chapter.

2. Advantages of LDS
The advantage of *LDS* is that it is a natural and adaptable way of doing logic. We consider a logical system as basically a discipline for databases and labels, giving structures and mechanisms for deduction. The mechanisms could be either proof theory or other mechanisms, such as abduction or circumscription, or explanation, etc. This scenario is very natural and can be adapted to a variety of application areas. To use *LDS* in an application area we need to

1. Recognize underlying structures in the application. These include
   (a) the declarative information to be manipulated and reasoned with;
   (b) the units (objects) to be manipulated and their relative structure. (e.g. semantic objects such as worlds or individuals, syntactic objects such as priorities or probabilities);
   (c) any compatibility or conflict in natural manipulative movement between (a) and (b).
2. Having recognized in (1) the natural components of the system, we devise an *LDS* to represent and reason about them. Note that *LDS* is not a single logic, but a family of logics.

The above process needs to be applied anyway for any use of traditional logic. What I am saying to the reader is that he should *not* approach the problem with a pre-determined pre-fabricated logic (such as the logic of his youth) and therefore be compelled to force all representation into it.

In fact, what many researchers often do in practice is to use labels as side effects (implementation tricks) to *retain* distinctions that their (favourite) logic forces them to abandon.[3]

Once we adopt the discipline of *LDS*, we see that there are other advantages.

- The labels can be formally used to bring into the object level meta-level features of the logic. A simple example would be to use the labels to trace the proof of the current goal. Conditions on the label can restrict the next move. These conditions are basically meta-level, but such meta-level features can be incorporated into an object level algebraic labelled proof rule.

[3]My favourite analogy is that of a person who has a wife (husband) and a mistress (lover). Obviously they are there for a reason and may have considerable influence (good or bad)! There may be a need for them much in the same way that there is a need for the label. Rather than supress them and treat them as a side effect, I am proposing that we bring them forward into the open, and recognize their influence. We should openly declare that a declarative unit is a formula and a label (or several labels) and analogously in some societies it may be better to admit (the unfortunate fact) that a 'family' unit is a husband/wife and a mistress/lover (read Balzac!). We should recognize the structure involved so that we can handle it better.

- *LDS* allows for a uniform method for bringing the semantics into the syntax. Recall Example 2.3.10 as a striking example of that, with a connection with situation theory.
- *LDS* is a unified framework for monotonic and non-monotonic logics.
- *LDS* is more computationally transparent (through the labels) and hence more receptive to meaningful optimisations.
- The proof theory of *LDS* can be more flexible. In classical logic, there is proof theory and there are tableaux systems which can be considered as model building systems. One is syntax based and the other is semantically based. In *LDS*, a tableaux procedure is just another *LDS*. The basic notion of consequence in *LDS* is that of a database proving a labelled formula for example $t_1 : A_1, t_2 : A_2 \ldots \vdash s : B$. A tableaux refutation for an *LDS* consequence relation will start with
$$\text{True } [t_i : A_i], \text{ False } [s : B]$$
and manipulate that. But such a system is just another *LDS*!
Thus we have: $\Delta_1 \vdash_{LDS_1} \alpha : A$ iff in the tableaux system $\Delta_2 \vdash_{LDS_2}$ '*closed*' where LDS_2 is an *LDS* system dealing with *signed* labelled formulas.
- The devices of labelled Skolemization and visas across worlds are very powerful tools of *LDS*.
- Different worlds can have different reasoning systems (time dependent reasoning).
- We can reduce higher order properties to lower order.

3. Limitations of LDS
The main limitation is that an *LDS* logic needs to commit itself. Take for example **S4**. This logic has one modality which is reflexive and transitive. Presented as a Hilbert system, we write a certain number of axioms:
$$\begin{array}{l} \Box A \to A \\ \Box A \to \Box\Box A \\ \Box(A \wedge B) \leftrightarrow (\Box A \wedge \Box B) \\ \vdash A \Rightarrow \vdash \Box A. \end{array}$$

The above system is not committed to any interpretation. $\Box A$ means any of the following:

- A holds in all accessible worlds.
- A is a set in the Euclidean plane and $\Box A$ is its topological interior.
- A is provable
- $\Box$ represents the progressive of English (e.g. if A is 'John walks', then $\Box A$ is 'John is walking').
- $\Box$ can mean some algebraic operation.

In any *LDS* interpretation, we need to identify the labels, in order to state what the declarative units are. The nature of t, the algebra of t, will commit us to some—if not exactly one—of the possible interpretations. We thus lose generality. We gain power, but we have to sacrifice our semantic options. When we are applying *LDS*, we may not mind that because the application area already dictates the interpretation. Thus we must be careful to choose the right level of labelling. Not too detailed to be able to remain within the realm of logic, avoiding turning the system into an implementation.

This argument can be used against the universality of classical logic. To translate a specific logic into classical logic, such as the **S4** necessity $\Box$, we need to commit its interpretation. Some application areas, such as legal reasoning, may not allow us to commit the interpretation. Furthermore, such an application area may not be particularly interested in the computational advantages offered by the translation and may be more sensitive to the naturalness of the representation.

By the way, this limitation is equally valid when we translate into or use directly classical logic. $\Box$ has to be translated into classical logic and the translation requires commitment (unless we use the *term translation*).

Part II

Formal Theory of Labelled Deductive Systems

Chapter 3

Introducing Algebraic Labelled Deductive Systems

3.1 Introduction and overview

The previous chapter, Introducing *LDS*, gave many examples of how labelling is used, both in monotonic and nonmonotonic reasoning systems. We need now to give formal definitions of the concepts involved.

The basic mechanism for introducing an *LDS* is to choose an algebra $\mathcal{A}$ and to consider declarative units $t : A$ and databases $\Delta = (D, \mathbf{f}, d, U)$ as defined in Definition 1.5.1. and in Definition 3.2.4

The rationale behind this definition as well as several case studies motivating it were amply demonstrated in Chapter 2, 'Introducing LDS'. From the technical point of view the basic idea is to add annotations (labelling) to formulas of a given logic $\mathbf{L}$ giving further information about the formulas through the labels. The labels come from another language, which we call the labelling language, or algebra. Thus for an algebra $\mathcal{A}$ and a logic $\mathbf{L}$ we can be used to construct a new system $(\mathcal{A}, \mathbf{L})$, an *LDS*. In this *LDS* we create new databases of the form $\Delta = (D, \mathbf{f}, d, U)$. These new databases are themselves declarative 'units'. We are perfectly within our framework if we regard them as a sort of 'formula' and introduce a new algebra $\mathcal{B}$ and start labelling these new 'formulas'. Thus we can have new 'atomic' declarative units of the form $b : \Delta$ and get new databases of the form $(D, \mathbf{f}, b, U), b \in D$, where for any term $t \in D, \mathbf{f}(t) = \Delta_t$ is a databse of the *LDS* $(\mathcal{A}, \mathbf{L})$. Symbolically this means that we are constructing the new *LDS*, $(\mathcal{B}, (\mathcal{A}, \mathbf{L}))$, see example 3.2.7 for useful additional resource labelling. Furthermore, symmetrically, we can use the databases Δ of the *LDS* $(\mathcal{A}, \mathbf{L})$ also as labels. Given another logic $\mathbf{L}_1$ and a wff A of $\mathbf{L}_1, \Delta : A$ can mean that Δ is a database 'justifying' A, or describing the context in which A was introduced, or any other information about A. We can thus use the *LDS* $(\mathcal{A}, \mathbf{L})$ as a labelling system for another logic $\mathbf{L}_1$ and thus form the new *LDS* $((\mathcal{A}, \mathbf{L}), \mathbf{L}_1)$.

If we start with $(\mathcal{A}, \mathbf{L})$ and iterate the above constructions we get a complex (higher level) *LDS* based on $\mathcal{A}$ and $\mathbf{L}$. These matters will be studied in a later chapter.

The purpose of the present chapter is to present a notion of *LDS* most suitable for applications. This notion is that of algebraic *LDS*. Algebraic *LDS* play another role. They are a kind of *LDS* which arise in turning an arbitrary formal logical system into an *LDS* system. In fact, there are some specific kind of algebras involved. These algebras are good for theoretical studies (of how to turn any logic into an *LDS*) but not necessarily good for general applications. Thus for all the applied consumers of logic, more general algebras are needed. Also the consumer of logic will feel more comfortable with a more sophisticated *LDS* proof theory, of a kind which may be mathematically reducible to simpler notions.

We are, therefore, introducing general algebraic *LDS* in the present chapter, outlining features of interest for applications. The next chapter, chapter 4, will introduce and study some specific algebraic *LDS*, with some specific algebras. These specific algebras are the kind of algebras that arise in turning an arbitrary logic into an *LDS*.

We present in section 3.2 a simplified version of the basic definitions and then proceed to deal with the proof theory. Since we have labelling at our disposal, the proof theory can be simplified. For one thing, we need present only elimination rules to define the system and the introduction rules can be defined 'inversely' and automatically from the elimination rules. This is unusual and is possible only because the labels can be used for metalevel control. This is done in section 3.3.

On the predicate level, Skolem constants become label dependent and allow us to present the beautiful theory of quantification, run time skolemization and visa rule, as hinted in the introductory chapter, 'Introducing LDS', and as will be presented in a later chapter. Section 3.4 presents the metabox system, which is a way of formulating special introduction rules for presenting special logics. For the applied logician, the metabox system is most useful. For the pure logician, it is redundant, as it can be reduced to a more complex labelling. The mathematical study is postponed to chapter 4.

3.2 Algebraic LDS

Definition 3.2.1 (Labelling langauges) *(a) A labelling langauge is a higher order language with the following symbols:*

- *(a) Function symbols $\{f_1, f_2, \ldots\}$ on individuals with arities $r_1, r_2, \ldots$, respectively.*
- *(b) Relation symbols $\{R_1, R_2, \ldots\}$ on individuals with arities $s_1, s_2, \ldots$ respectively.*
- *(c) The classical connectives, equality $=$, and possibly the set theoretic operations of union $\cup$ and intersection $\cap$, and possibly other higher order functions on predicates.*
- *(d) Element variables $x_1, x_2, \ldots$ and individual constantx $c_1, c_2, \ldots$.*
- *(e) Relation and function variables of all arities.*
- *(f) Higher order quantifier symbols $\forall, \exists$ for individual relation and function variables of all arities.*

(b) By a syntactical labelling theory we mean a theory τ (first order or higher order) in the langauge $\mathcal{A}$ above.

(c) By a (model theoretic) algebra $\mathbf{m}$ for the language $\mathcal{A}$ we mean a classical (higher-order) structure with domain S in which the symbols (functions and relations) of $\mathcal{A}$ are interpreted.

We are calling $\mathbf{m}$ an 'algebra' rather than an '$\mathcal{A}$-model' for conceptual clarity. In many of our applications more emphasis will be put on the function symbols of $\mathcal{A}$ than on its relation symbols.

(d) Let $\mathbf{m}$ be an algebra, τ be a syntactical labelling theory and φ a formula of $\mathcal{A}$, and h an assignment to its free variables. The notion of satisfaction, $\mathbf{m} \vDash_h \varphi$, meaning that φ holds in $\mathbf{m}$ under h, is defined classically, bearing in mind that $\mathbf{m}$ is taken as a full *higher-order model, namely the higher-order quantifiers in φ range over* all *relations and functions. We with $\mathbf{m} \vDash \varphi$ if $\mathbf{m} \vDash_h \varphi$ for all h. Let $\mathbf{M}$ be a class of models. We write $\mathbf{M} \vDash \varphi$ if $\mathbf{m} \vDash \varphi$ for all $\mathbf{m} \in \mathbf{M}$. Let $\mathbf{M}_\tau$ be the class of all structure $\mathbf{m}$ such that $\mathbf{m} \vDash \tau$.*

(e) τ is said to be first-order if its formulas contain only first order quantifiers.

(f) Let $\mathcal{A}$ be a labelling language. The Herbrand domain of $\mathcal{A}, H_\mathcal{A}$ is the set of all first order terms definable in $\mathcal{A}$. Namely $H_\mathcal{A}$ contains exactly all individual variables and constants of $\mathcal{A}$ and is closed under the application of the first order function symbols of $\mathcal{A}$

(g) Given $t_1, t_2 \in H_\mathcal{A}$ and a class $\mathbf{M}$ of models, we write $t_1 =_\mathbf{M} t_2$ iff in every model $\mathbf{m} \in \mathbf{M}$, we have $\mathbf{m} \vDash t_1 = t_2$. We similarly define $t_1 =_\tau t_2$ iff for every model $\mathbf{m} \vDash \tau$, we have $\mathbf{m} \vDash t_1 = t_2$.

(h) *Let $\mathcal{A}$ be a labelling langauge. By a syntactical diagram D, we mean a pair (D_1, D_2), where $D_1 \subseteq H_{\mathcal{A}}$ and D_2 is a set of formulas. Each formula $\varphi \in D_2$ is either atomic or the negation of an atomic formula.*

(i) *Since D_1 and D_2 are disjoint, we can assume $D = D_1 \cup D_2$. I.e. the diagram is a set of terms and formulas.*

Let D be an $\mathcal{A}$ diagram and let $\mathbf{M}$ be a class of models. D is said to be $\mathbf{M}$ satisfiable iff for some model $\mathbf{m} \in \mathbf{M}$ and an assignment $h, \mathbf{m} \vDash_h D_2$. D is said to be τ satisfiable iff for some model $\mathbf{m}$ and assignment $h, \mathbf{m} \vDash_h \tau \cup D_2$. We write $D \vDash_{\mathbf{M}} \varphi$ iff $D \cup \{\sim \varphi\}$ is not $\mathbf{M}$ satisfiable.

(j) *Since D is a diagram of Herbrand terms and formulas, it is possible to define an algorithmic consequence relation for a query φ from D. For example, we can assume D is a logic program (let us have no negative literals for convenience) and that τ is a Horn clause theory. If φ is a suitable query, we can write $D \models\!\!\!\equiv_\tau \varphi$ to mean that φ succeeds as a query from $D \cup \tau$. In many cases it is more convenient to deal with our algebras syntactically in this manner.*

Example 3.2.2 (Free concatenation algebra) *Let S be a nonempty set. Let S^* be the set of all finite sequences of elements from S. Let $\varnothing$ be the empty sequence. We identify $S \subseteq S^*$ by the map $s \mapsto (s)$. Let $*$ be concatenation on S^*. Let, for $x = (x_1 * x_2) \in S^*$ the functions $\mathbf{r}(x)$ and $\mathbf{l}(x)$ be*

$$\begin{aligned}
&\mathbf{r}(x) = (x_2)\\
&\mathbf{l}(x) = (x_1)\\
&\mathbf{r}(\varnothing) = \mathbf{s}(\varnothing) = \varnothing.
\end{aligned}$$

We need the following additional functions on S^*

- *seq(x) is defined by*
  seq$(a) = (a)$ for a atomic.
  seq$(x * y) = \mathrm{seq}(x) * \mathrm{seq}(y)$, where $*$ is concatenation of sequences.

- *multiset(x) is defined by*
  multiset$(a) = \{a\}$, the singleton multiset
  multiset$(x * y)$ = multiset$(x) \cup$ multiset (y)

- *set(x) is defined similarly.*

Then $(S^*, *, \mathbf{r}, \mathbf{l},$ seq, multiset, set$, \cup, \cap)$ is an algebra, we denote it by $\mathcal{F}$. It is convenient to assume that S contains all natural numbers and some additional letters as atoms. This assumption becomes useful when we use this algebra for resource management.

Example 3.2.3 (Possible world algebra) *Let S be a non empty set and let R be an n-place relation on S ($R \subseteq S^n$). Then (S, R) is an algebra. Note that we have no function symbols, but we still call it an algebra. In case R is binary, we get a possible world model where S is the set of worlds and $R \subseteq S^2$ is the accessibility relation.*

Definition 3.2.4 (Object level algebraic LDS) *1. Let $\mathcal{A}$ be a labelling langauge. A logical language $\mathbf{L}$ with Skolem constants parameterised from $\mathcal{A}$ is a predicate language with variables and predicates, a set of connectives $\sharp_1, \ldots, \sharp_n$ with appropriate arities and the quantifiers $\forall$ and $\exists$. The notion of a wff with free variables is defined in the traditional manner. The language may also contain function symbols $e_1, e_2, \ldots$ of various arities, and constants. The language shares the variables $x_1, x_2, \ldots$ with the labelling language $\mathcal{A}$. The constants of the langauge $\mathbf{L}$ are not atomic constants but are generated from special unary function symbols in the following manner. We assume we always have a sequence $c_1(x), c_2(x), \ldots$ of terms made of function symbols with one variable (for elements for the domains associated with labels). If t, s are labels $s : c_1(t)$ means that $c_1(t)$ is an element created (Skolemized) at label t but now residing at s. Thus the terms of $\mathbf{L}$ are generated by first generating the Herbrand universe*

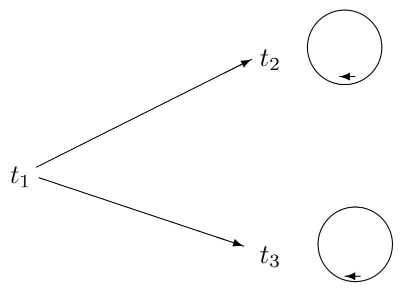


Figure 3.1:

$H_{\mathcal{A}}$ from the atomic labels and from the labelling algebra function symbols $f_1, \ldots, f_k$. Then we apply $c_1, c_2, \ldots$ to generate the constants of $\mathbf{L}$ and then we apply the function symbols of $\mathbf{L}, e_1, e_2, \ldots$ to generate the full range of terms of $\mathbf{L}$. Such terms we denote by $\alpha_1, \alpha_2, \ldots$.

2. *The language of an algebraic labelled deductive system is a pair $(\mathcal{A}, \mathbf{L})$ where $\mathcal{A}$ is a labelling language and $\mathbf{L}$ is a logical language parameterised by $\mathcal{A}$. $\mathcal{A}$ and $\mathbf{L}$ share the set of free variables and may share some of the constants and function symbols.*

3. *A declarative unit is a pair $t : A$, where t is a term of $\mathcal{A}$ and A is a wff of the logic. t and A may share free variables and constants. A labelled term has the form $s : \alpha$, where s is a label and α is a term of $\mathbf{L}$.*

4. *A database is a tuple $\Delta = (D, \mathbf{f}, d, U)$ where D is a diagram of the labelling language $\mathcal{A}$, and $\mathbf{f}$ and U are two functions, associating with each term in D a wff $A_t = \mathbf{f}(t)$ and a set of terms U_t. U_t is the set of $\mathbf{L}$ terms residing at label t. The functions $\mathbf{f}$ and U can also be displayed by writing $\{t : A, t : c_i(x)\}, t \in D$. Note that D may contain also some relations $\pm R(t_1, \ldots, t_n)$. $d \in D$ is the 'actual' world of D.*

Example 3.2.5 (Modal logic with temporal labels) *A possible world* LDS *language can be as follows. The language is the first order language of modal logic with connectives $\Box$ and $\Diamond$. The labelling language is the first order language of a binary relation R. We have no function symbols in the algebra but we do have an infinite stock of constants. We can think of the constants as moments of time and of R as the earlier-later relation. The modal language contains the functions $c_1(t), c_2(t), \ldots, t$ ranges over labels, generating the terms of the modal language. We can think of $c(t)$ as an individual born at time t. A diagram D has the form $\{t_1, \ldots, t_n, \pm R(s_i, s_j)\}$ where s_i, s_j are from among $t_1, \ldots, t_n$ which are terms in D. This can be graphically displayed as a proper geometric diagram such as*

where arrows display the relation R. Thus D in this case is

$$\{t_1, t_2, t_3, R(t_1, t_2), R(t_1, t_3), R(t_3, t_3), R(t_2, t_2)\}.$$

A database is a tuple $(D, \mathbf{f}, d, U)$, where $d \in D$ and $\mathbf{f}$ associates with each term $t \in D$ a wff $\mathbf{f}(t) = A_t$ of predicate modal logic, and U_t is a set of terms of the modal language of the form $\{c_i(s_j)\}$, for $t \in D$. We can think of the elements of U_t as the individuals living at time t. Thus $c(s) \in U_t$ means that the individual born at time s is alive at time t. We can also display this fact by writing $t : c(s)$. Thus, according to this notation, $t : c(t)$ always holds. The formula $\mathbf{f}(t) = A_t$ is a modal language formula which is supposed to hold at time t. Thus $t : \Diamond A(c(s))$ means that at time t it is possible that A holds for the element $c(s)$, who was born at time s. We are not saying that $c(s)$ is alive at t. Figure 3.2 displays a database based on the diagram of Fig. 3.1.

Definition 3.2.6 (Semantics for algebraic LDS) *Let $(\mathcal{A}, \mathbf{L})$ be an* LDS *language. Let $\sharp_1, \ldots, \sharp_n$ be the connectives of the language, with arities $r(i), i = 1, \ldots, n$ respectively. A possible world semantical interpretation for $(\mathcal{A}, \mathbf{L})$ has the form $\mathbf{I} = (\mathbf{M}, \psi_1, \ldots, \psi_n, \psi)$ where $\mathbf{M}$ is a class of models*

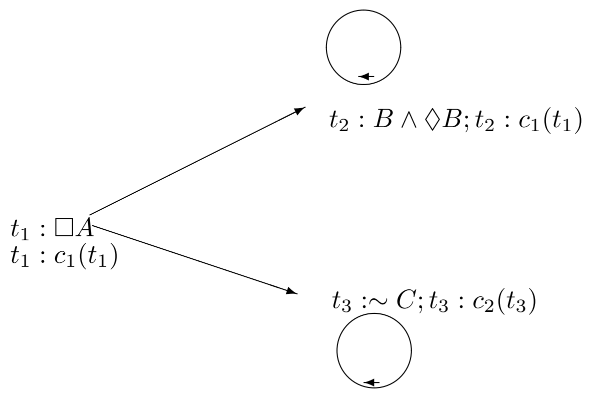

$t_2 : B \wedge \Diamond B; t_2 : c_1(t_1)$
$t_1 : \Box A$
$t_1 : c_1(t_1)$
$t_3 :\sim C; t_3 : c_2(t_3)$

Figure 3.2:

for the language $\mathcal{A}$ and each ψ_i has the form $\psi_i(x, Q_1, \ldots, Q_{r(i)})$, is a wff of the monadic extension of the language of $\mathcal{A}$, (which may be $\mathcal{A}$ itself if $\mathcal{A}$ is already higher order), containing $r(i)$ new monadic predicates Q_j and one free variable x, and $\psi = \psi(a, Q)$ is a closed wff of the language with one monadic predicate and the constant a. Note that in many applications $\mathcal{A}$ is based in first order logic.

A structure of the semantics has the form $(\mathbf{m}, a, V, h)$ where $\mathbf{m} \in \mathbf{M}$ is a model of $\mathcal{A}$, $V_m, m \in M$ is a non-empty domain and h is an assignment associating with each n-place atomic P of $\mathbf{L}$ and each $m \in M$ a subset $h(m, P) \subseteq V_m^n, a \in M$ and where M is the domain of $\mathbf{m}$. h can be extended to an h' containing h, by assigning subsets of M to the monadic predicates of ψ_i and ψ.

Satisfaction $\vDash_h$ can be defined for arbitrary wffs of $\mathbf{L}$ via the inductive clauses and the usual 'abuse' of notation, ($V = \bigcup_{m \in M} V_m$).

0. *$m \vDash P(b_1 \ldots, b_n)$ iff $(b_1, \ldots, b_n) \in h(m, P), b_i \in V$.*

1. *$m \vDash \neg A$ iff $m \nvDash A$.*

2. *$m \vDash A \wedge B$ iff $m \vDash A$ and $m \vDash B$.*

3. *$m \vDash \sharp_i(A_1, \ldots, A_k)$ iff $\mathbf{m} \vDash \psi_i(m, Q_1, \ldots, Q_k)$, with $Q_i = \{n \mid n \vDash A_i\}$, extending the assignment h.*

4. *$m \vDash \exists y A(y)$ iff for some $y \in V_m, m \vDash A(y)$.*

5. *We say that the structure $(\mathbf{m}, a, V, h)$ satisfies a formula A of $\mathbf{L}$ iff $\mathbf{m} \vDash \psi(Q_1, \ldots, Q_k)$, with $Q_i = \{m \mid m \vDash A\}$.*

6. *A database $\Delta = (D, \mathbf{f}, d, U)$ is said to hold at a structure $(\mathbf{m}, a, V, h)$ iff there exist functions $g_1 : D \to M$ and $g_2 : U \to V$ such that g_1 validates the diagram D in $M, g_1(d) = a$, and for every $t : A(x_1, \ldots, x_n)$ in D we have $g_1(t) \vDash A(g_2(x_1), \ldots, g_2(x_n))$.*

7. *We say $\Delta_1 \vDash_\tau \Delta_2$ iff for every structure $\mathbf{m} \in \mathbf{M}_\tau$ and every g_1^1, g_2^1 which validate Δ_1 in the structure, there exists extensions g_1^2 and g_2^2 of g_1^1 and g_2^1 respectively, which validate Δ_2.*

So far our definitions and examples have concentrated on the notions of algebras, declarative units and databases. This is parallel to defining the notion of well formed formula in classical logic. We have not provided the reader yet with proof theory nor with semantics. Our next task is to define the general form of proof theory for object level algebraic *LDS*. This means that we want to give proof rules which allow us to define the notion of $\Delta \vdash \Gamma$, where Δ and Γ are databases. In particular we want to define the notion of $\Delta \vdash t : A$, for declarative units $t : A$.

The reader should note that more refined and formal definitions and theorems on the proof theory of algebraic *LDS* are given in a later chapter. The purpose of this section is just to introduce the various aspects of algebraic *LDS*.

To explain our approach we give in a later chapter a detailed case study in the pure propositional implicational language with $\{\to\}$ alone. The examples are not trivial, since in principle they cover all substructural logics.

This chapter will give fairly general definitions.

Example 3.2.7 (LDS with resource labelling using free concatenation algebras) *We use the algebra $\mathcal{F}$ in the example 3.2.2 for a language with $\to$ only. $(t_1, \ldots, t_n) : A$ can mean that A was derived (using modus ponens) from assumptions named $t_1, \ldots, t_n$ and the derivation used the assumptions in that order. The modus ponens propagation rule is:*

$$\frac{\alpha : A; \beta : A \to B}{\beta * \alpha : B}$$

A more general example is to use the algebra $\mathcal{F}$ to add a resource labelling to an arbitrary LDS *of the form $(\mathcal{A}, \mathbf{L})$. We consider the cross product $\mathcal{F} \times \mathcal{A}$ of the two algebras and use this as labels. Thus a declarative unit of the new* LDS *has the form $(a, \alpha) : A$, where a is from $\mathcal{F}$ and α from $\mathcal{A}$. For $t = (a, \alpha)$ and a formula φ of either language (of $\mathcal{F}$ or of $\mathcal{A}$) we let $\varphi(t)$ hold iff $\varphi(a)$ resp. $\varphi(\alpha)$ holds.*

The label propagation rules for the new LDS *$(\mathcal{F} \times \mathcal{A}, \mathbf{L})$ are defined pointwise as follows.*

Let $\sharp(A_1, \ldots, A_n)$ be an arbitrary connective of $\mathbf{L}$. Consider one of its elimination rules in $(\mathcal{A}, \mathbf{L})$, say, of the following form:

$$\frac{\varphi_{\mathcal{A}}; \alpha_1 : B_1; \ldots, \alpha_k : B_k; \alpha : \sharp(A_1, \ldots, A_n)}{\psi_{\mathcal{A}}; r_1 : C_1; \ldots; r_m C_m}.$$

We are writing $\varphi_{\mathcal{A}}, \psi_{\mathcal{A}}$ with subscript $\mathcal{A}$ to emphasise that they are in the $\mathcal{A}$ language.

We would like to use our resource algebra $\mathcal{F}$ to record exactly how each $r_i : C_i$ is derived. Let us use an atomic letter b to name uniquely this particular rule. In the LDS *$(\mathcal{F} \times \mathcal{A}, \mathbf{L})$ this rule becomes*

$$\frac{\varphi_{\mathcal{F}}; \varphi_{\mathcal{A}}; (a_1, \alpha_1) : B_1; \ldots; (a_k, \alpha_k) : B_k; (a, \alpha) : \sharp(A_1, \ldots, A_n)}{\psi_{\mathcal{F}}, \psi_{\mathcal{A}}; (b * a * a_1, \ldots, *a_k) * (1), r_1) : C_1; \ldots; ((b * a * a_1 * \ldots, *a_k) * (m), r_m) : C_m}$$

where $\varphi_{\mathcal{A}}, \psi_{\mathcal{A}}$ apply to the $\mathcal{A}$ components of the labels and $\varphi_{\mathcal{F}}, \psi_{\mathcal{F}}$ are in the language of $\mathcal{F}$ and apply to the $\mathcal{F}$ components of the labels. For example $\varphi_{\mathcal{F}}(a_1, \ldots, a_n, a)$ might say $a_1 = a_2 = \ldots = a_n = a$, i.e. we can eliminate only if all components were proved from the same assumptions. This is a sort of resource management. Thus any derived labeled formula $\alpha : A$ of $(\mathcal{A}, \mathbf{L})$ can be represented in $(\mathcal{F} \times \mathcal{A}, \mathbf{L})$ with a label as $(a, \alpha) : A$, where a records the derivation exactly.

Definition 3.2.8 (Proof theory for algebraic *LDS*) *Let $(\mathcal{A}, \mathbf{L})$ be an algebraic LDS. The following is a stock of typical proof rules. The elimination rule for connectives and the quantifier elimination, visa and flattening rules are called* data rules. *The introduction rule for connectives and the quantifier introduction rules are called* goal rules. *All the various conditions φ, ψ on the labels appearing in the rules are called the* enabling conditions.

1a. Elimination rules for connectives

An elimination rule for a connective $\sharp(A_1, \ldots, A_n)$ has the form

$$\frac{\varphi; t_1 : B_1; \ldots, t_k : B_k; s : \sharp(A_1, \ldots, A_n)}{\psi; r_1 : C_1, \ldots; r_m : C_m}.$$

The terms $r_1, \ldots, r_m$ are new (created) function symbols dependent on the free variables (universal constants) of A_i, B_j.

Where $\varphi(t_1, \ldots, t_k, s)$ is a formula of $\mathcal{A}$ called the pre-condition for the (firing of the) rule and $\psi(t_1, \ldots, t_k, s, r_1, \ldots, r_m)$ is a formula of A called the post condition. ψ may contain equality. The rule is to be understood as saying:

> *If we have proved the wffs above the line with labels satisfying φ then we can create new labels and deduce the formulas below the lines and the new and old labels satisfy ψ.*

ψ may contain equality just in case we want to say the new labels are equal to some old ones. To understand the roles of φ and ψ consider example 2.5.1. In rule (∗1) of this example, φ is $t < s$ and the application of the rule transfoms figure 2.16 into figure 2.17.

In rule (∗2) of this example ψ is $s < r$ and the application of the rule transforms figure 2.17 into figure 2.18. Note that ψ gives the 'position' of the newly created r_i relative to the old points, thus allowing us to transform the old database into the new one.

ψ can also be used to cancel data from the database. Consider Example 3.2.10. Here ψ is used to throw assumptions out. In this context, option (j) of Definition 3.2.1 is more convenient. According to this option, a rule of the form

$$\frac{\varphi;\ \textit{Old Data}}{\psi;\ \textit{Additional Data}}$$

can be understood as:

If the current database Δ contains the labelled formulas above the line (Old Data) and the diagram D of Δ proves φ in the sense $D \models_\tau \varphi$, then put the Additional Data into Δ and put ψ in as well.

1b. Introduction rules for connectives
Introduction rules in LDS *are defined in terms of elimination rules and hence need not be introduced separately. Their use will be properly defined when we give the notion of a proof. We view them as the respective 'inverses' of the elimination rules. Intuitively, when we are in the middle of a proof and we want to introduce a connective $t : \sharp(A_1, \ldots, A_n)$ with label t, we are really saying 'we already have this connective'. If this is indeed true, then for an aritrary elimination rule of this connective (e.g. like in (1a) above), if we assume the antecedent, without the connective, we must be able to prove the consequent of the rule. If we can demonstrate this capability for each elimination rule, then we can introduce the connective. Take for example*

$$\frac{A; A \to B}{B}.$$

We show we already have $A \to B$ (i.e. introduce $A \to B$) by showing we can assume A and get B. Another example is $\frac{A\wedge B}{A}$ $\frac{A\wedge B}{B}$. To introduce $A \wedge B$ we must show we can get the conclusions of each rule, without the connective, namely assume $\varnothing$ and get A and assume $\varnothing$ and get B.

2. Quantifier elimination and introduction rules
These include the usual classical quantifier rules and visa rules

(a.)

$$\frac{t : \forall x A(x)}{t : A(u^t); t : u^t, u^t \textit{ a new universal constant}}$$

(b.)

$$\frac{t : \forall x A(x); t : c^s}{t : A(c^s)}$$

(c.)

$$\frac{t : c^s; t : A(c^s)}{t : \exists x A(x)}$$

(d.)

$$\frac{t : \exists x A(x, y)}{t : A(c^t(y), y); t : c^t(y)}.$$

$t : c^t(y)$ is optional in (d). It may not be adopted in modal logic but may be adopted in numerical or priority logics.

(e.) To introduce $t : \forall x A(x)$ we prove $t : A(u_0^t)$, for any arbitrary new constant u_0^t.

Some important side conditions may be involved in (e.).

3. Visa rules
These have the form

$$\frac{t_i : A_i; t_i : c^{s_i}; r_j : B_j; \psi(t_i, s_i, r_j)}{r_j : d^{s'_j}} \; j = 1 \ldots k \text{ and } i = 1, \ldots, n$$

where $d^{s'_j} \in \{c^{s_i} \mid i = 1, \ldots n\}$, for $j = 1, \ldots, k$.
The meaning of the visa rule is that if c^{s_i} exist at t_i where A_i holds and if ψ holds and B_j holds at r_j then $d^{s'_j}$ exist at r_j.

- *Sample visa rule from modal logic:*

$$\frac{t : c^s; t < r, r : \Box\bot}{r : c^s}.$$

The rule says that if c^s exists at t then it exists at any endpoint above t. It happens to correspond to the following refinement of the Barcan formula.

$$\forall x \Box A(x) \to \Box(\Box\bot \to \forall x A(x)).$$

4. Flattening Rules
These have the form (note only one formula A is involved):

$$\frac{t_i : A; \psi(t_i, r_j)}{r_j : A}$$

- *Sample flattening rules from substructural logics (e.g. R-mingle):*

$$\frac{t : A; s : A}{t * s : A}$$

$$\frac{t : A; t \leq s}{s : A}$$

5. A proof π_0 of level 0 from a database $(D, \mathbf{f}, d, U)$ is a sequence of labelled databases $(D_n, \mathbf{f}_n, d, U_n)$ and justification function $\mathbf{J}^0$ satisfying the following:

5.1 $(D_0, \mathbf{f}_0, d, U_0) = (D, \mathbf{f}, d, U)$ and $\mathbf{J}^0(0)$ = 'assumption'.

5.2 $(D_{n+1}, \mathbf{f}_{n+1}, d, U_{n+1})$ is obtained from $(D_n, \mathbf{f}_n, d, U_n)$ by applying a data rule and adding the consequent of the data rule to $(D_n, \mathbf{f}_n, d, U_n)$ to obtain $(D_{n+1}, \mathbf{f}_{n+1}, d, U_{n+1})$. The data rule is applicable iff $D_n \vdash \varphi$, φ is the enabling precondition in which case $\mathbf{J}^0(n+1)$ = name of the data rule. Note that the post condition ψ of the data rule (when needed) will tell us how to 'add' the consequent to the old database to obtain the new one. Compare the discussion under (1a) of the present definition.

We write $(D, \mathbf{f}, d, U) \vdash_0 (D', \mathbf{f}', d, U')$ iff there exists a proof π_0 of level 0 leading from one to the other.

6. *A proof of level $\leq n$ has the form of linked sequences of databases with a main sequence π, and justification function $\mathbf{J}$. The first element of the main sequence π is $(D, \mathbf{f}, d, U)$. Each element of the sequence is obtained from a previous one either according to one of the cases of a proof of level 0 or according to the following case.*

   Introduction case
   For some connective $\sharp(A_1, \ldots, A_n)$ let

$$\frac{\varphi_j; t_1^j : B_1^j, \ldots, t_{k(j)}^j : B_{k(j)}^j; s_j : \sharp(A_1, \ldots, A_n)}{\psi_j; r_1^j : C_1^j, \ldots; r_{m(j)}^j : C_{m(j)}^j}$$

   $j = 1 \ldots k$, be all the elimination rules involving $\sharp$. We want to introduce $s : \sharp(A_1, \ldots, A_n)$. Suppose we also have that for each j there is a proof π_j of level $\leq n-1$ of each $(D_n, \mathbf{f}_n, d, U_n) + \{\psi_j(s_j/s), r_i^j : C_i^j\}$ from $(D_n, \mathbf{f}_n, d, U_n) + \{\varphi_j(s_j/s), t_i^j : B_i^j\}$ (i.e. we can prove the consequent of each rule from the antecedent of the rule without the use of $s : \sharp(A_1, \ldots, A_n)$, as if we already have it) then the introduction step allows us to move onto $(D_{n+1}, \mathbf{f}_{n+1}, d, U_{n+1}) = (D_n, \mathbf{f}_n, d, U_n) + \{s : \sharp(A_1, \ldots, A_n)\}$.

   *The meaning of '+' is that we add into the old database the labelled wffs indicated, together with the conditions φ, ψ as shown. Going back to Example 2.5.1, rule (*1) of this example is an elimination rule:*

$$\frac{t : \Box A; t < s}{s : A}$$

   So to show $t : \Box A$, we take an arbitrary s, such that $t < s$, i.e. add $\{s; t < s\}$ to the old database and prove that we can add $s : A$ to it.

   We link the proofs π_j into the proof π at line n, via the justification function, $\mathbf{J}(n+1) = \{$proofs $(\pi_j, \mathbf{J}_j)\}$.

   We write $(D, \mathbf{f}, d, U) \vdash_n (D', \mathbf{f}', d, U')$ if there is a proof of level $\leq n$ of the consequent from the antecedent. We write $(D, \mathbf{f}, d, U) \vdash (D', \mathbf{f}', d, U')$ if there is a proof of any level of the consequent from the antecedent.

7. *For the sake of tractability we can also assume that in the elimination rule the complexity of $B_1, \ldots, B_n$ and $C_1, \ldots, C_m$ is strictly less than the complexity of $\sharp(A_1, \ldots, A_n)$. We should assume some suitable complexity function. So for example the $\Rightarrow$ elimination rule, modus ponens*

$$\frac{t : A; s : A \Rightarrow B}{f(t, s) : B}$$

   is acceptable because A, B are subformulas of $A \Rightarrow B$. However, the classical disjunction elimination rule, written in traditional form

$$\frac{\begin{array}{c} A \vee B \\ A \ldots C \\ B \ldots C \end{array}}{C}$$

   where $X \ldots Y$ means there exists a (inductively simpler!) proof from X to Y, is not in acceptable form. A proper treatment of $\vee$ elimination is done in the next chapter in terms of flattening rules. We can rely, however, on the deduction theorem for $\Rightarrow$, when we do have it and write the rule as follows:

$$\frac{t_1 : A \Rightarrow C; t_2 : B \Rightarrow C; s : A \vee B}{f(t_1, t_2, s) : C}$$

   where C is a wff with less nested disjunctions than $A \vee B$. See also remark 4.2.2.

It is easier to understand the proof rules and the visa rules if we translate them into two sorted classical logic. The next definition accomplishes that.

Definition 3.2.9 (Proof theoretical translation of algebraic LDS **into classical logic)** *Let $(\mathcal{A}, \mathbf{L})$ be a proof theoretic* LDS *as in definition 3.2.8. We define a translation $*$ into two sorted predicate logic, relative to the data rules of the* LDS *as follows: With each atomic $Q(x_1, \ldots, x_n)$ we associate a two sorted classical predicate $Q^*(t, x_1, \ldots, x_n)$. We also use the predicates $C(t, y)$ and $E(t, y)$, reading y was created at t, and y exists at t respectively. The first coordinate accepts terms of the sort of the algebra $\mathcal{A}$. The inductive definition of the translation $[t : A]^*$, for an arbitrary wff A and label t is as follows:*

1. $[t : Q(x_1, \ldots, x_n)]^* = Q^*(t, x_1, \ldots, x_n)$, *$Q$ atomic.*
2. $[t : \neg A]^* = \neg[t : A]^*$.
3. $[t : A \wedge B]^* = [t : A]^* \wedge [t : B]^*$.
4. $[t : \exists y A(y)]^* = \exists y(E(t, y) \wedge [t : A(y)]^*)$, *$y$ not in t.*
   $[t : \forall y A(y)]^* = \forall y(E(t, y) \rightarrow [t : A(y)]^*)$, *$y$ not in t.*
5. *Let $\sharp$ be any connective of $\mathbf{L}$ whose elimination rules are*

$$\frac{\varphi_j; t_1^j : B_1^j; \ldots; t_{k(j)}^j : B_{k(j)}^j; s_j : \sharp(A_1, \ldots, A_n)}{\psi_j; r_1^j : C_1^j; \ldots; r_{m(j)}^j : C_{m(j)}^j}.$$

   Then

$$[t : \sharp(A_1, \ldots, A_n)]^* = \textit{(definition)}$$
$$\textstyle\bigwedge_j [\forall t_1^j, \ldots, t_{k(j)}^j \{\bigwedge_{i=1}^{n(j)} [t_i^j : B_i^j]^* \wedge \varphi_j \rightarrow$$
$$\exists r_1^j, \ldots, r_{m(j)}^j (\psi_j(s_j/t) \wedge \textstyle\bigwedge_{i=1}^{m(j)} [r_i^j : C_i^j]^*)\}]$$

   where (s_j/t) means substitute t for s_j.

   The translation correctly represents the algebraic LDS provided the following holds in classical logic.

6. $\forall y \exists! t C(t, y)$.
7. $\forall y \forall t(C(t, y) \rightarrow E(t, y))$
8. *For every visa rule of the form:*

$$\frac{t_i : A_i; t_i : c^{s_i}; r_j : B_j; \psi(t_i, s_i, r_j)}{r_j : d^{s'_j}}$$

   $j = 1, \ldots, k$ and $i = 1, \ldots, n$, where $d^{s'_j} \in \{c^{s_i}, \ldots, c^{s_n}\}$ we have the axiom

$$(\forall c_1, \ldots, c_n)(\forall t_1, \ldots, t_n)(\forall s_1, \ldots, s_n)(\forall r_1, \ldots, r_k)(\psi(t_i, s_i, r_j) \wedge$$
$$\textstyle\bigwedge_{i=1}^n [t_i : A_i]^* \wedge E(t_i, c_i) \wedge C(s_i, c_i) \wedge \bigwedge_{j=1}^k [r_j : B_j]^* \rightarrow$$
$$\textstyle\bigwedge_{j=1}^k E(r_j, d^{s'_j}))$$

9. *For every flattening rule of the form*

$$\frac{t_i : A; \psi(t_i, r_j)}{r_j : A}$$

   we have the axiom

$$(\forall t_1 \ldots)(\forall r_1, \ldots)(\bigwedge_i [t_i : A]^* \wedge \psi(t_i, r_j) \rightarrow \bigwedge_j [r_j : A]^*)$$

Note that we need to assume in the introduction rules that some complexity goes down. Unlike the semantic translations, B_i^j and C_i^j are not necessarily subformulas of $\sharp(A_1, \ldots, A_n)$ and so if some complexity goes down we end up with a finite number of steps with several formulas of the form $s : Q$, Q atomic.

Example 3.2.10 (**(φ, ψ) MP rule for $\rightarrow$**) *The usual rule of modus ponens can be labelled as follows*

$$\frac{\alpha : A; \beta : A \rightarrow B; \varphi(\beta, \alpha)}{\gamma : B; \psi(\alpha, \beta, \gamma)}.$$

1. *For example, if our labelling algebra is the possible world algebra of Example 3.2.3 and $\rightarrow$ is taken to mean strict implication, then the MP rule becomes*

$$\frac{\alpha : A; \beta : A \rightarrow B; \beta < \alpha}{\alpha : B}.$$

2. *If on the other hand, the algebra is the free concatenation algebra of example 3.2.7 then our MP becomes*

$$\frac{\alpha : A; \beta : A \rightarrow B}{\beta * \alpha : B}$$

   One can take any semigroup $(S, *)$ with operation $*$ and define the above Elimination rule. We have seen such examples in the Chapter 'Introducing LDS'.

3. *The formula ψ can be used for deletion as well as multiple conclusions. Consider lienar implication, where each assumption can be used at most once. Thus $A; A \rightarrow B \mid\!\sim B$ but having invoked modus ponens, these particular assumptions cannot be reused. We can achieve this aimin several ways:*

   (a) *Use multisets of atomic labels and write modus ponens as*

$$\frac{\alpha : A; \beta : A \rightarrow B; \alpha \cup \beta = \varnothing}{\alpha \cup \beta : B}$$

   The condition $\varphi(\beta, \alpha) = \alpha \cap \beta = \varnothing$ will block the reuse ofany assumption.

   (b) *Use a predicate $C)\alpha)$ to mean 'α is cancelled'. Write modus ponens as*

$$\frac{\neg C(\alpha); \neg C(\beta); \alpha : A; \beta : A \rightarrow B}{\alpha \cup \beta; C(\alpha); C(\beta)}$$

   We can read teh rule as:
   *If $C(\alpha)$ and $C(\beta)$ finitely fail from the Diagram of the current database, and $\alpha : A$ and $\beta : A \rightarrow B$ are in the current database then add to it $\alpha \cup \beta : B$ and $C(\alpha)$ and $C(\beta)$. Now $\alpha : A$ and $\beta : A \rightarrow B$ can no longer participate in modus ponens because $C(\alpha)$ and $C(\beta)$ will succeed. For this we assume that we use syntactical consequence (option *j) of Definition 3.2.1). If we choose model theoretic consequence, adding ψ would be a proper update (chance $\neg C(x)$ to $C(x)$).*

A more detailed study of Algebraic *LDS* is continued in Chapter 4.

3.3 Introduction rules: A discussion

The definition of the proof theory for algebraic *LDS*, Definition 3.2.8 is a general definition, in terms of an arbitrary labelling language $\mathcal{A}$, and elimination rules only. The introduction rules can be defined using the elimination rules. Although mathematically such a formalism is sufficient to define a large class of logics, in practice some logics are formulated by elimination rules and introduction rules with special restrictions on them which makes them no longer the 'inverse' of the elimination rules. Such presnetations of logics make it easier to work with them. These restrictions are, of course, eliminable in terms of a more elaborate labelling system, but may be preferable in practice. The aim of this section is to identify these convenient restrictions and to give a general definition of an *LDS* using them. We first illustrate the implicational fragment of modal logic. Consider the modal logic of possible world models of the form $(S, R, a, h), a \in S, R \subseteq S^2, h$ the assignment, where a is the actual world and where we require aRa to hold. We understand the consequence relation $A_1, \ldots, A_n \vdash B$ to mean that in every model (S, R, a, h) if $a \vDash_h A_i, i = 1, \ldots, n$ then $a \vDash_h B$. See example 3.4.8 below.

The above definition means that $A, A \to B \vdash B$ holds but $C \to A, C \to (A \to B) \vdash ?C \to B$ does not hold, since reflexivity is required to hold only in the actual world. Thus if aRs holds and $a \vDash C \to A$ and $a \vDash C \to (A \to B)$ and $s \vDash C$ hold then we are assured that $s \vDash A$ and $s \vDash A \to B$, but unless sRs we may not have $s \vDash B$. Thus $a \vDash ?C \to B$ may not hold.

Let us examine the proof theory of this implication from the point of view of modus ponens.

Let the labelling language be that of example 3.2.7 and we will check what kind of (φ, ψ) modus ponens is needed. Consider the following database.

1. $t_1 : D \to (C \to (A \to B))$
2. $t_2 : D \to (C \to A)$
3. $t_3 : D$
4. $t_4 : D$

The database is perceived as holding in the actual world a of some model (S, R, a, h) and hence modus ponens is allowed. We can thus deduce

5. $t_1 t_3 : C \to (A \to B)$
6. $t_2 t_4 : C \to A$

We want to show $C \to B$ with some label s. We know that the semantic meaning of $C \to B$ that C implies B in any possible world s is such that aRs holds. So we can assume C and try and show B with the appropriate labels. Since $C \to (A \to B)$ and $C \to A$ hold in the actual world we can get $A \to B$ and A in the world s, but we cannot continue the modus ponens in s to get B because sRs may not hold.

Let us continue formally

7. To show $C \to B$

x	$:C$	assumption
$t_1 t_3 x$	$: A \to B$	allowed modus ponens
$t_2 t_4 x$	$: A$	allowed modus ponens
$(t_1 t_3 x)(t_2 t_4 x)$	$: B$	not allowed

We observe that the use of modus ponens should be different in a nested box, and we need to formulate the right rules to caputure the right semantical meaning. In fact, the natural formulation will probably tinker with the introduction rule for $\to$, since it will give restrictions on modus ponens inside a box. This is not compatible with our policy of letting the introduction rule for $\to$ be the 'inverse' of (automatically and uniformly be defined from) the elimination rule for $\to$.

We can solve the problem by offering a new labelling system and new rules. Let the labels be pairs (m, t). m is the level of nesting and t the former formal label. Thus the data will be

1. $(0, t_1) \quad D \to (C \to (A \to B))$
2. $(0, t_2) \quad D \to (C \to A)$

3. $(0, t_3)$ D

4. $(0, t_4)$ D

The modus ponens rule is:

$$\frac{(m,\alpha) : A; (n,\beta) : A \to B; m = n = 0 \vee n < m}{(m, \beta\alpha) : B}$$

We thus get:

5. $(0, t_1 t_3) : C \to (A \to B)$

6. $(0, t_2 t_4) : C \to A$

7. To show $(n, \alpha) : C \to B$. For some n and α we open a box and assume $(n, x) : C$. For x arbitrary and try and show $(n, \alpha(x)) : B$.

   So let us do this:

(7.1)	$(1, x) : C$	assumption
(7.2)	$(1, t_1 t_3 x) : A \to B$	from 7.1 and 5
(7.3)	$(1, t_2 t_4 x) : A$	from 7.1 and 6
(7.4)	We cannot apply modus ponens on 7.2 and 7.3 because the rule does not apply	

   If the modal logic was reflexive at all points, the modus ponens rule would have been

$$\frac{(m,\alpha) : A; (n,\beta) : A \to B; n \leq m}{(m, \beta\alpha) : B}$$

   and we could have continued

   (7.3) $(1, (t_1 t_3 x)(t_2 t_4 x)) : B$.

   Note that if we were to label our assumptions as in modal logic, say as in example 3.2.5, then the appropriate elination rule would be

$$\frac{t : A \to B; s : A; tRs}{s : B}$$

   and the introduction rule for $\to$ is the expected 'inverse' of the elimination rule i.e. if $\Delta; t < y; y : A$ can prove $\Delta; t < y; y : B$, where y is arbitrary, then we can deduce $\Delta; t : A \to B$. In this labelling language the labels are worlds, the labelling language contains the accessibility relation R and the actual world a and aRa is required to hold. The database becomes the multiset

$$\{a : D \to (C \to (A \to B)); a : D \to (C \to A); a : D; a : D; aRa\}$$

   Deduction will allow to use modus ponens to get $a : C \to (A \to B)$ and $a : C \to A$ because aRa is unavailable. To show $a : C \to A$, we need to assume $\{y : C; aRy\}$ for an arbitary y and show $y : A$, which is not possible.

In conclusion, note that we did manage to adhere to the principle that the $\to$ introduction rule is automatically and uniformly dependent on the elimination rule by changing the labelling.

This can always be done as we shall prove later. Nevertheless, it is more convenient to present logics where the introduction rule is modified and different from the expected inverse of the elimination rule and where the labelling is not changed. We thus need to define the new notions of metabox reasoning which allows us to do that.

It will be convenient to allow for composite labelling of the form $(a_1, \ldots, a_k, \alpha) : A$ where each a_i component performs essentially a metalevel task while α is the 'main' label.

Another way of looking at the notion of algebraic labelling, is that we consider our labelling algebra as a cross product of several subalgebras, each perforing a specific metalevel task.

The above example and short discussion shows that we generally need, for each connective, introduction and elimination rules, which include the propagation of labels and include proof rules for subcomputations.

It is convenient for us to enclose the subcomputation specified geometrically in a box which we call a **Metabox**. The introduction rules we call **Goal Rules**, because the goal is to introduce the new connectives. The elimination rules we call **Data Rules**, because they increase our data directly, not via metabox subcomputation. The reason for using these new names is because not all data or goal rules are elimination or introduction rules. Some may be rewrite rules, Skolemisation rules etc. The important distinction will then be whether we are working directly on the data (assumptions) or modifying the goal. The distinctions will become more clear as we progress through the subject matter. The speical rules for subcomputation we call **Metabox Rules**. These include rules for passing a formula with the connective from one box to another, as well as for algorithmic proof within a box.

We shall later prove that in the presence of labels only data rules (elimination rules) are needed. We will prove that for any *LDS*, $(\mathcal{A}, \mathbf{L}, \mathbf{M})$ with proof rules $\mathbf{M}$ there exists another *LDS*, $(\mathcal{A}', \mathbf{L}', \mathbf{M}')$, which 'proves' the 'same' labelled formulas, but which use only elimination rules, the introduction rules being defined as the 'inverses' of the elimination rules. The above notions are first explained through a series of examples, and then the general definitions are given. The examples will also illustrate the practical way we annotate our proofs:

Example 3.3.1 *We take a traditional proof of* $((c \to a) \to c) \to ((c \to a) \to a)$*:*

1. $(c \to a) \to c$ *assumption*
2. $c \to a$ *assumption*
3. c *modus ponens (1), (2)*
4. a *modus ponens (2), (3).*

The following is how we look at it in our metabox system:

Box 1

1.	$((c \to a) \to c) \to ((c \to a) \to a)$	from Box 2

Since Box 2 is referred to in Box 1 to justify a line.
We write Box 1 < Box 2.

Box 2

1.	$(c \to a) \to c$	assumption
2.	$(c \to a) \to a$	from Box 3

exit Box 2: $((c \to a) \to c) \to ((c \to a) \to a)$

We also have Box 2 < Box 3.

Box 3

1.	$c \to a$	assumption
2.	$(c \to a) \to c$	imported from Box 2
3.	c	modus ponens (1), (2)
4.	a	modus ponens (1), (3)

exit box 3 $(c \to a) \to a$

We summarise the rules we used:

- The goal rule for $A \to B$ is the opening of a box with assumption A and the showing of B. The exit is the introduction rule.
- The data rule is the fact that we can use modus ponens. This is the elimination rule.

- The meatabox rule is the fact that assumptions can be propagated (reiterated) into inner boxes (ie Box$i <$ Box$j$).

We can imagine different metabox rules, which give rise to new logics. Here is an alternative propagation rule:

$\star$ If A is an assumption in Box a of the form $A_1 \to A_2$ and $a < b$ and b is an immediate $<$ successor box of a then A can be propagated into b.

Example 3.3.2 *To show the power of this notation, let us do a modal logic deduction, with $\Box$ and $\to$:*

Box 0

$\boxed{\Box A \to (\Box(A \to B) \to \Box B) \textit{ from Box 1}}$

Box 1

1.	$\Box A$ *assumption*
2.	$\Box(A \to B) \to \Box B$ *from box 2*

exit: $\Box A \to (\Box(A \to B) \to \Box B)$

Box 2

1.	$\Box(A \to B)$ *assumption*
2.	$\Box B$ *from box 3*

*Note that if we adopt the alternative $\star$ propagation rule for $\to$ mentioned above, we get strict modal-**K** implication.*

One more point before we go on. We saw that given a system of linked boxes $a_1 < a_2 < a_3$ we can propagate assumptions, for example for a_1 into a_3. To make a box more independent, we can list all reiterations at the beginning of a box. Thus box 3 of 3.3.1 becomes:

Box 3

0.	$(c \to a) \to c$ reiteration
1.	$c \to a$ assumption
2.	Same as (0)
3.	c modus ponens
4.	a modus ponens

exit: $(c \to a) \to a$

Box 3 is now independent of its place and relation to other boxes.

Furthermore, to keep trace exactly what assumptions were used, we can label the assumptions and propagate the labels.

All of this is done in the definitions below:

The rest of this Chapter defines metabox algorithmic proof rules for a general *LDS* and illustrate it in a language with $\to$. The definitions for modal logics are given in a separate chapter.

We consider various restrictions on the two rules

$$\to I : \quad \frac{\begin{array}{l} A \\ \dots \\ \dots \\ B \end{array}}{A \to B}$$

and

$$\to E : \quad \frac{A, A \to B}{B}$$

The various restrictions will yield classical implication, intuitionistic implication, relevant implication, linear implication, entailment implication and strict implication. A later Chapter shows how the restrictions and the logics are connected.

metabox

$$\boxed{\begin{array}{ll} x : A & \text{assumption} \\ \vdots & \\ \vdots & \\ y : B & \text{conclusion} \end{array}}$$

Figure 3.3:

3.4 The metabox system

We begin with a mathematical definition of proof boxes. The definition is logically correct but can be simplified and made more efficient for implementation as an automatic system. We give the less efficient definitions first for explanatory considerations.

The previous section described three components for a labelled deduction system, the introduction rules, the elimination rules and the metabox rules. The first two are familiar; we now have to introduce and define the metabox.

Imagine we are going forward using our proof rules. We proceed line by line, let us say from line 1 (denoted l_1), to line 99 (denoted l_{99}). This proof looks like the usual proofs we are familiar with, except that instead of the usual formulas at each line, we have labelled formulas

$$l_n = t_n : A_n$$

We now want to use an introduction rule at line l_{100}, to show that we can prove the labelled formula of the form, for example, $t : A \to B$. Our metabox rules tell us we have to open a box as in the figure below 3.3, assuming $x : A$ as an assumption and proving $y : B$ as a conclusion. What kind of proof rules can we have for this metabox subcomputation?

We have several components here:

1. What are the rules of proof within the box? These include both formula rules and label rules.

2. Which of lines $l_1, \ldots, l_{99}$ can we use within the box? Earlier lines are units already proved. Are they available inisde the box?

3. We also have a notational problem. What notation do we use to indicate the fact that line 100, namely $l_{100} = t : A \to B$ is justified by the metabox?

4. How do we present and deal with nestings of metaboxes? Are there going to be logical connections, and if yes, how do we represent them?

5. What kind of lables do we use within the box? What is the label x going to be? A new atomic label? How does it relate to existing labels?

Our strategy is to give a definition of a deduction within a box which is completely context free. We *import* into the box all information we need for the algorithmic proof computation within the box and then there is no need to know how the box relates to the rest of the proof.

We now survey our options in choosing the metabox rules. We can have different rules depending on how deeply the boxes are nested. The outer box, the metabox of complexity 0 is the outer derivation using elimination rules only. A metabox of complexity 1 is the derivation with a box justifying an introduction of a connective. Similarly complexity $n + 1$ indicates a box justifying an introduction of a connective within a box of complexity n.

Let us now survey our options.

Options for Metabox of Complexity 0

Let us examine more closely the basic box of complexity 0 for forward proof: (We illustrate using the language with $\to$). Its structure looks as follows:

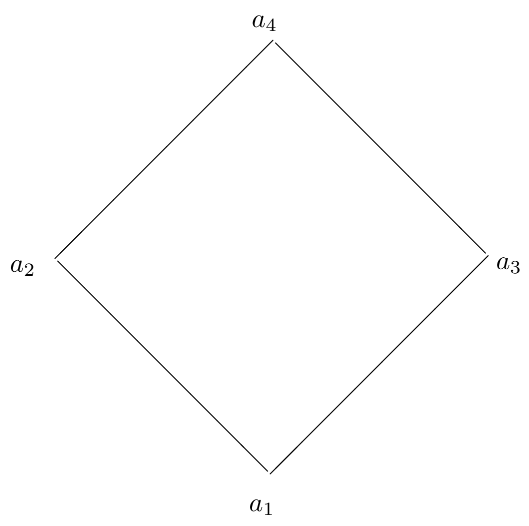


Figure 3.4:

Assumptions $A_1, \ldots, A_n$.

$$\begin{array}{c} t : A \\ \vdots \\ s : A \to B \\ \vdots \\ \varphi(t, s) \\ \vdots \\ r : B; \psi(t, s, r) \\ \vdots \end{array}$$

We use the elimination rules, which for $\to$ is modus ponens, as the only rules for going forward. The following are the parameters (distinctions) involved in this concept:

3.4.0.1 Labelling of the Assumptions

We use a labelling language $\mathcal{A}_1$ whose elements have the form $t = (a, \alpha)$, where α is from a labelling language $\mathcal{A}_0$, referred to as 'main labels', and a is from a free concatenation algebra (see example 3.2.7 which indicates resource. We need an agreement (considered as part of the logic) of how to label new assumptions. For example several copies of the same assumption may have the same main label α but never the same resource labels, as shown.

$$\begin{array}{c} (a_1, \alpha) : A \\ (a_2, \alpha) : A \\ (a_3, \beta) : B \end{array}$$

or $\{(a_1, \alpha, A), (a_2, \alpha, A), (a_3, \alpha, B), \ldots\}$

- The second component may indicate relative strength among the data $\{(a_i, \alpha, A_i)\}$ for example See figure 3.4.

3.4.0.2 Forward Proofs Level 0

We use several simple forward proof rules. For implication, $\rightarrow$, we use MP and distinguish the following resource considerations which are done by the first resource component of the label.

1. We note which formulas (assumptions) are used to yield a conclusion;
2. We count how many times each formula was used.
3. We ask whether the assumption was used as a minor premise or as a ticket? Eg:

minor	A
ticket	$A \rightarrow B$
conclusion	B

4. We check the length of steps and the sequencing of the steps of the proof.
5. We have to decide how the main labels are propagated during proof (i.e. the second component). We can use a propagation formula ψ.
6. We give a formal definition of when do we consider the proof successful.

Options for Metabox of Complexity $n + 1$
The motivation is to introduce a $t : \sharp(A_1, \dots, A_n)$ for example in the case of implication, to show $t : A \rightarrow B$ (a complex wff).
One way to show it is by direct elimination rule, as for $\rightarrow$ we use modus ponens:

$$\begin{array}{c} t_1 : X \rightarrow (A \rightarrow B) \\ \dots \\ t_2 : X \\ t = t_1 t_2 : A \rightarrow B \end{array}$$

A second way is to use the introduction rule procedure to show $t : \sharp(A_1, \dots, A_n)$, for example $t : A \rightarrow B$. In this case we use a subproof (box) ie use proof steps as follows.

Imagine we are at line k of the proof:

line k: show $t : A \rightarrow B$ at level $n + 1$, if the metabox below is successful:

Assume A with some labelling
… use proof steps of level $\leq n$ to show B
with some other appropriate label

The following are the parameters (distinctions) involved in this construction:

1. What are the labels of the assumption for a nested box and how these labels relate to existing ones.
2. In the proof at level $n+1$, line k requires us to intoruce a $t : \sharp(A_1, \dots A_n)$ e.g. for implication to show $A \rightarrow B$. We are doing a metabox sub-computation. What resources (assumptions, lemmas and goals, etc) of the level $n + 1$ of the proof (up to line k) can we use in the box and in what form? This is $\mathbf{R}^w$ of 3.4.2 below.

   For example we may say we can reiterate only implications of the form $C \rightarrow D$ and that these can be used only as tickets. Another example is that we allow to reiterate only from the immediately enveloping box only. The way to use a reiteration may depend on its label, which carries information about where it was first introduced. We may also decide to change the label as we reiterate the formula into the box.

3. What distinctions and what proof rules we adopt for the box ie, in the sense of 2 above (inductively on n). These distinctions may use other logics which are already known. In other words the logic of the inner box may be different from the logic of the outer box.

4. What parameters to give to line k, with $(A \to B)$ if the box succeeds? By parameters we mean here a label to be used in the continuing proof in level $n+1$.

   Note that the metabox has the structure:

Entry conditions (import)
Box proof conditions
Exit conditions (export)

Example 3.4.1 *Consider a proof of* $(A \to (A \to B)) \vdash (A \to B)$*. We have:*

$$\begin{array}{lll} l_1 = & a_1 : A \to (A \to B) & \textit{assumption} \\ l_2 = & a_1 : A \to B & \textit{from metabox in fig 3.5} \end{array}$$

metabox

		show B
$l_{2.1}$	$a_2 : A$	assumption
$l_{2.2}$	$a_1 a_2 : A \to B$	from l_1 and $l_{2.1}$
$l_{2.3}$	$a_1 a_2 a_2 : B$	from $l_{2.1}$ and $l_{2.3}$

Figure 3.5:

The above metabox is context dependent. It relies on the fact that it justifies line l_2 and can use in the internal proof line l_1. To make it context independent, we can put line l_1 in the metabox itself. Thus it becomes, fig 3.6.

		show B
l_1	$a_1 : A \to (A \to B)$	reiteration
$l_{2.1}$	$a_2 : A$	assumption
$l_{2.2}$	$a_1 a_2 : A \to B$	from $l_1, l_{2.1}$
$l_{2.3}$	$a_1 a_2 a_2 : B$	from $l_{2.1}, l_{2.2}$

Figure 3.6:

Once we have all the reiterations in the box, the box becomes self contained. Furthermore, there is no need for the decimal numbering system, since the box is self contained. the goal to show in the above box is the formula B. When we have many nested boxes, the goals to show in the outer boxes also play a part in the algorithmic proof procedures of the inner box. These, therefore, also have to be reiterated. (The rule which uses previous goals to show is called the Restart Rule.)

We therefore end up with the following box, see fig 3.7 below, presented after cleaning up and re-organising

The justification of a line within a box may refer to another box, as in the following:

$r_n = t : A \to B$ $\qquad \pi_n =$ *from box m.*

Reiterations	**Previous Goals**
$R_1 = a_1 : A \to (A \to B)$	*none*
Assumption	**Current Goal**
$A_1 = a_2 : A$	$a_1 a_2 a_2 : B$
Proof Lines	**Justification**
$r_1 = a_2 : A$	$\pi_1 =$*assumption*
$r_2 = a_1 a_2 : A \to B$	$\pi_2 =$ *from* R_1 *and* r_1 *by modus ponens*
$r_3 = a_1 a_2 a_2 : B$	$\pi_3 =$ *from* r_1 *and* r_2 *by modus ponens*

Figure 3.7:

We are now ready to give a general definition of the metabox. This general definition also includes the rules for the label propagation. As we have seen, the 'main' labels could be numbers, algebraic constants, or even formulas of a logic. Generally, when we are given the labels, we must also be given their algebraic or logical structure. This structure we call the Labelling Logic (or Algebra of Labels) $\mathcal{A}$. It is convenient to adopt an algebra of elements which are pairs $t = (a, \alpha)$. The first component is from the free concatenation algebra and the second component is from the main arbitrary algebra. The first component is there to trace the proofs while the second component is the main labelling. So, for example, if the second component is reliability numbers, we can have modus ponens as

$$\frac{(a_1, 0.7) : A; (a_2, 0.9) : A \to B}{(a_2 a_1, 0.63) : B}$$

Definition 3.4.2 (Metabox computation) *We start with a labelling logic $\mathcal{A}$ on a set of well formed formulas $\mathbf{L}$. We form the logic which we call $\mathrm{LDS}_1 = (\mathcal{A}_1, \mathbf{L})$. This LDS_1 has elimination rules, and its introduction rules are 'inverse' to the elimination rules. Our purpose in this definition is to show how a metabox discipline can change the introduction rules in a way as to get a new logic LDS_2. To achieve this, we form a new algebra $\mathcal{A}_2$ whose elements are of the form $t = (a, \alpha)$, where a is an element of a free concatenation algebra $\mathcal{F}$ of example 3.2.7, and $\alpha \in \mathcal{A}_1$.*

The manipulation of the labels for the elimination rules $(\mathcal{A}_2, \mathbf{L})$ are as in example 3.2.7. $\mathcal{A}_2$ gives us complete resource control over derivations of $(\mathcal{A}_1, \mathbf{L})$ and is used in the formulation of the metabox discipline defined below. We assume a set of conditions $\mathbf{C}$ on labels t_i. Among them are formulas $\varphi_{MP}(t, r), \psi_{MP}(t, r, s)$, to be used in propagating resource $\mathcal{F}$ labels through modus ponens, and $\psi_E(t, r, s)$ to be used on $\mathcal{F}$ labels for exiting from a box.

A *Let $\mathbf{A}$, $\mathbf{B}$, $\mathbf{R}$, $\mathbf{G}$, denote finite sequences of sets of labelled formulas. If $\mathbf{H}$ is such a sequence, its elements have the form $(\mathbf{H}(1), \mathbf{H}(2), \ldots, \mathbf{H}(n))$, where n is its length. Each $\mathbf{H}(i)$ is a set of labelled formulas of the form $\mathbf{H}(i) = \{(t_i, H_i)\}$, where t_i is the label and H_i is the formula. In the definition below, $\mathbf{A}$ sequences have labels of the form $t = (a, \alpha)$, a atomic from $\mathcal{F}$, and it is assumed that for $\mathbf{A}$ sequences $i \neq j \to a_i \neq a_j$. Other sequences have as their labels as propagated by the rules of the labelling algebra. For example if the labelling algebra is the set of all finite sequeces of atomic labels with concatenation, then the propagated labels will be as in example 3.2.7.*

For the case of $\mathbf{B}$, $\mathbf{R}$, $\mathbf{G}$ sequences, we allow $a_i = a_j$ for $i \neq j$.

B *We define a proof box (in short box) of complexity 0 as a tuple of the form $\mathrm{Box} = (\mathbf{R}, \mathbf{A}, \mathbf{B}, \Pi, \mathbf{G}, Q)$, which satisfies certain conditions, to be described below. In this tuple, $\mathbf{R}$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{G}$ are finite sequences of sets of labelled formulas satisfying the agreed notational convention of (A) above. Q is a set of formulas and Π is a function giving for each $i \leq$ length of $\mathbf{B}$ a value. The intended meaning of the above sequences is as follows:*

* $\mathbf{R}$ *is a sequence of reiterations (ie essentially formulas already proved before and which can be used in the current proof).*

* $\mathbf{A}$ *is the sequence of the assumptions.*

Reiterations	*Past Goals*
$l_1 : R_1$	$s_1 : G_1$
Assumptions	*Current Goal*
$t_1 : A_1$	$t : Q$
Proof	*Justification*
$r_1 : B_1$	Π_1
$\vdots$	$\vdots$
$r_n : B_n$	Π_n

Figure 3.8:

* Π *is the justification function, giving a value for each line of the proof, indicating why it is valid.*

* $\mathbf{G}$ *is the sequence of past queries and* (q, Q) *is the current query.*

The labels are annotations of the form (a, α) *a is a sequence of names of assumptions showing resource, while* α *is a general label from a general algebra showing any other information in the general case. For example the label* α *can be a possible world in the case of a metabox system for modal logic.*

The box can be presented graphically as in Figure 3.8.

For boxes of complexity 0, the following conditions must be satisfied

0 All assumptions $\mathbf{A}(i) = (t_i, A_i)$ *are labelled as* $t_i = (a_i, \alpha_i)$*, where* a_i *are all different and atomic. This is already a convention in (A) above. Note that we may adopt a different convention. We may want* $t_i = \{A_i\}$*, that is, we label a formula by itself, and have an interesting interaction between the metabox logic and the labelling logic.*

1 If $\mathbf{B}(n) = \{(t_n^k, B_n^k)\}$ *is the last element in the proof sequence* $\mathbf{B}$ *then for some* k $B_n^k = Q, t_n^k = q$.

2 The justification function Π *satisfies one of the following for each* $\mathbf{B}(j)$.

2.1 $\Pi(\mathbf{B}(j)) = \mathbf{A}(i)$*, ie,* $\mathbf{B}(j)$ *is an assumption.*

2.2 $\Pi(\mathbf{B}(j)) = \mathbf{R}(i)$*, ie* $\mathbf{B}(j)$ *is a reiteration.*

2.3 $\Pi(\mathbf{B}(j)) = \mathbf{B}(i), i < j$*, ie* $\mathbf{B}(j)$ *is a repetition.*

2.4 With each connective $\sharp(A_1, \ldots, A_n)$ *and with each elimination rule for that connective of the form*

$$\frac{\varphi; t_1 : B_1; \ldots; t_k : B_k; s : \sharp(A_1, \ldots, A_n)}{\psi; r_1 : C_1; \ldots; r_m : C_m}$$

We associate a justification clause as follows:

$$\Pi(\mathbf{B}(j)) = (\mathbf{B}(i_0), \mathbf{B}(i_1), \ldots, \mathbf{B}(i_k))$$

with $i_0 < j, \ldots, i_k < j$ *and*

$$(s, \sharp(A_1, \ldots, A_n)) \in \mathbf{B}(i_0)$$

and

$$(t_1, B_1) \in \mathbf{B}(i_1), \ldots, (t_k, B_k) \in \mathbf{B}(i_k)$$

and $\varphi(t_1, \ldots t_k, s)$ *hold as and* $\mathbf{B}(j) = \{(r_i, C_i)\}$ *and* $\psi(t_1, \ldots, t_k, s, r_1, \ldots, r_m)$ *hold.*

For example for the case of implication, the clauses will be as follows:
$\Pi(\mathbf{B}(j)) = (\mathbf{B}(i), \mathbf{B}(k))$, *for* $i, k < j$ *and* $\mathbf{B}(i) = (t_i, B_i)$ *and* $\mathbf{B}(k) = (t_k, B_i \to B_j)$ *and* $\varphi_{MP}(t_i, t_k)$ *and* $\psi_{MP}(t_i, t_k, t_j)$ *hold where* φ, ψ *are conditions in* $\mathbf{C}$ *having to do with modus ponens. For example* φ *can be* $t_i \cap t_k = \varnothing$ *and* ψ *can be* $t_j = t_i \cup t_k$ *(in case the labels are finite multisets).*

C Let us assume that the notion of a proof box of complexity $< m$ *has been defined. We define the notion of a proof box of complexity* $\leq m$ *to be a tuple Box* $= (\mathbf{R}, \mathbf{A}, \mathbf{B}, \Pi, \mathbf{G}, Q)$ *satisfying the following:*

0 as before

1 as before

2 The justification function Π *satisfies, for each* $\mathbf{B}(j)$ *one of 2.1-2.4 as before, or as in 2.5 as below:*

2.5 We are justifying the introduction of a connective $\mathbf{B}(j) = \{t : \sharp(A_1, \ldots, A_n)\}$, *with a label* $t = (a, \alpha)$. *We thus have the elimination rules for this connective do work.*

Let
$$\frac{\varphi_w; t_1^w : B_1^w; \ldots; t_{k(w)}^w : B_{k(w)}^w; t_w : \sharp(A_1, \ldots, A_n)}{\psi_w; r_1^w : C_1^w; \ldots r_{m(w)}^w : C_{m(w)}^w}$$
be the wth eliminition rule. Let θ *be a general substitution instantiating* t_w *to* t *i.e.* $t_w\theta = t$. *Then if we claim we have* $t : \sharp(A_1, \ldots, A_n)$ *already, we must be able to show each* $r_i^w\theta : C_i^w\theta$ *and* $\psi_w\theta$ *from each respective*
$$\varphi_w\theta; t_1^w\theta : B_1^w\theta; \ldots; t_{n(w)}^w\theta : B_{n(w)}^w\theta.$$

We therefore require that the justification of $\mathbf{B}(j), \Pi(\mathbf{B}(j))$ *to be a set of boxes each of complexity less than m. The wth box has the form.*

2.5.1 Box (w) $= (\mathbf{R}^w, \mathbf{A}^w, \mathbf{B}^w, \Pi^w, \mathbf{G}^w, Q^w)$
where
$\mathbf{A}^w$ *is a sequence of length 1 with* $\mathbf{A}^w(1) = \{t_1^w\theta : B_1^w\theta \ldots, t_{k(w)}^w\theta : B_{k(w)}^w\theta\}$.

2.5.2 $\mathbf{R}^w$ *is the sequence of reiterations. For most metabox disciplines (though not always) it is obtained by concatenating*
$\mathbf{R} * \mathbf{A} * (\mathbf{B}(1), \ldots, \mathbf{B}(j-1))$
(denotes concatenating).*

(a)

2.5.3 $\mathbf{B}^w$ *is the new proof.*

2.5.4 $\mathbf{G}^w$ *is obtained by concatenating* $\mathbf{G} * ((q, Q))$ *where* Q *is the current goal and* q *is its label obtained from the last element* $\mathbf{B}(n)$ *of* $\mathbf{B}$ *(see condition 1).*

2.5.5 Q^w *is the new query.*

2.5.6 If $\mathbf{B}^w(k) = (t_k, Q^w)$ *is the last element of the proof* $\mathbf{B}^w$ *then the appropriate* ψ^w, *acting now as a Box exit condition, is satisfied.*

2.5.7 We say in this case that Box(w) is subordinate *to Box. The concept means that Box(w) is used to justify a line* $(\mathbf{B}(j))$ *in the proof of Box. Denote this relation by Box(w)* $<$ *Box and let* $\leq$ *be the transitive and reflexive closure of* $<$. $\leq$ *is a tree relation.*
The above simply says that the justification of $t : \sharp(A_1, \ldots, A_n)$ *set of boxes proving the appropriate elimination rules.*

Definition 3.4.3 *We say* $t_1 : A_1; \ldots t_n : A_n \vdash t : B$ *iff (by definition) there exists a box of some complexity with* $\mathbf{R} = \varnothing, \mathbf{A} = ((t_1, A_1) \ldots (t_n, A_n)), \mathbf{G} = \varnothing$ *and* (t, B) *is the last element of* $\mathbf{B}$, *ie* $Q = B$.

Box 0	1	Reiterations : $\varnothing$	Previous Goals : $\varnothing$
	2		Show $B \to (A \to B)$
	3	Assumption : $\varnothing$	
	4	$B \to (A \to B)$	from Box 1 :
Box 1	5	Reiterations : $\varnothing$	Previous Goals : $B \to (A \to B)$
	6		Show $A \to B$
	7	Assumption : B	
	8	$A \to B$	from Box 2 :
Box 2	9	Reiteration : B	Previous goals : $B \to (A \to B), A \to B$
	10		Show B
	11	Assumption : A	
	12	B	$\Pi(1) = 0$
	13		
	14		
	15	We have Box 2 $<$ Box 1 $<$ Box 0	

Figure 3.9:

Example 3.4.4 *1. Here is a box for $B \to (A \to B)$.*
See figure 3.9

We now give a series of examples to further illustrate the general definitions.

Example 3.4.5 (Intuitionistic Logic) 1. Assumptions*: sets of formulas (no labels), the goal is a single formula.*

2. Level 0*: modus ponens forward.*
Success is reaching the goal.

3. *The conditions for proof rules for an inner box introducing $A \to B$: at Line k: at level $n+1$.*

   Take in the box assumption A and add to the assumptions any previous line and then reach B by the rules of level n.

Example 3.4.6 (Classical Logic) *Modify level $n+1$ in intuitionistic logic by allowing previous goals in the box and success of box means reaching in the box the goal B or any previous goals.*
The following illustrates the new condition (called Restart) (see fig 3.10):
The box on Level 2 is successful because line 1.2.3 proves a which is in the show corner. We have $Box2 < Box1 < Box0$.

Example 3.4.7 (A labelled Proof) *This is a straightforward box proof of*
$\varnothing : (B \to A) \to ((A \to B) \to (A \to B))$.
The justification is Box a_1 below (see fig 3.11):

Level 0 Box 0 :	1		To show $((a \to b) \to a) \to a$
	2	$\varnothing$	assumptions
	3	$((a \to b) \to a) \to a,$	follows from box 1
Level 1 Box 1 :	4		To show a
	5	$(a \to b) \to a),$	assumptions
	6	$a \to b,$	from box 2
Level 2 Box 2 :	7		Show either a (previous goal) or b (current goal)
	8	$a,$	local assumption
	9	$(a \to b) \to a,$	reiteration
	10	$a,$	from 8
	11	$a,$	from 5 and 6
	12		

Figure 3.10:

Box a_1	1		Show $(A \to B) \to (A \to B)$
	2	$a_1 : B \to A$	assumption
	3	$a_1 : (A \to B) \to (A \to B),$	from Box a_2
Box a_2	4		Show $A \to B$
	5	$a_2 : A \to B$	assumption
	6	$a_2a_1 : A \to B,$	from Box a_3
Box a_3	7		Show B
	8	$a_3 : A$	assumption
	9	$a_2 : A \to B$	
	10	$a_1 : B \to A$	
	11	$a_3a_2 : B$	
	12	$a_2a_3a_1 : A$	
	13	$a_2a_3a_1 : B$	
exit	14	$a_2a_1 : A \to B$	
exit	15	$a_1 : (A \to B) \to (A \to B)$	
exit	16	$\varnothing : (B \to A) \to ((A \to B) \to (A \to B))$	

Figure 3.11:

Example 3.4.8 (Modal logic $\mathbf{K}_1$) *This example shows how the logic of an inner box can be different for the logic of an outer box. The example we chose is from modal logic. It is well known that the modal logic* $\mathbf{K}$ *with modality* $\Box$ *is axiomatised by the three schemas.*

1. $\Box(A \wedge B) \leftrightarrow \Box A \wedge \Box B$
2. $\dfrac{\vdash A}{\vdash \Box A}$
3. *All substitutions of truth functional tautologies.*

It is complete for the class of all Kripke structures of the form (SR, a) *where* $a \in S$ *is the actual world and* $R \subseteq S \times S$ *is the accessibility relation.*

Consider the semantical condition aRa. *The class of all Kripke models in which* aRa *holds, define a modal logic* $\mathbf{K}_1$. *We do have in this logic* $\vdash_{\mathbf{K}_1} \Box A \to A$ *but* $\not\vdash_{\mathbf{K}_1} \Box(\Box A \to A)$.

The LDS *of 2.2.3 will handle this easily and most naturally, as* aRa *will be part of the configuration. No other additions to the rules of 2.2.3 is necessary.*

A Hilbert type axiomatisation of $\mathbf{K}_1$ *is not easily formulated. We would need to replace axiom (2) above by the two axioms*

2a. $\Box A \to A$

2b. $\dfrac{\vdash_{\mathbf{K}} A}{\vdash_{\mathbf{K}_1} \Box A}$

In other words, we have to use $\vdash_{\mathbf{K}}$ *provability to define* $\vdash_{\mathbf{K}_1}$ *provability.*

If we write $A \to B$ for $\mathbf{K}_1$-strict impliction, then the metabox rules for $\to$ are as follows:

1. A formula D can be reiterated from Box a into Box b only if the following two conditions hold:
   1.1. D has the form $D_1 \to D_2$.
   1.2. Box b is immediately subordinate to Box a (in the sense of 2.5.7 of definition 3.4.2).
2. Modus ponens
   $$\frac{X, X \to Y}{Y}$$
   can be performed in box b provided one of the following conditions is satisfied.
   2.1. $X \to Y$ is a reiteration
   2.2. Box a is the outermost box.

Note that relaxing condition (1.2) corresponds to the transitivity of the accessibility relation and that relaxing condition (2.1) corresponds to the semantical condition of reflexivity of the accessibility relation.

Thus modus ponens can behave differently in different boxes.

The previous definitions described the metabox discipline for the case where the geometry was based on nested metaboxes. It is more conveient in many cases to order the boxes and rely on external ordering to achieve the effect of nesting. In fact, ordering is more general than nesting and can address a wider variety of logics. Here are the definitions:

Definition 3.4.9 (Ordered metabox computation) *Let* $\sharp(A_1, \ldots A_n)$ *be a new n-place connective. A* Metabox system *for this connective has the following components:*

1. *A language* $\mathcal{A}$ *of labels.*

2. **Goal Rules**
The goal rules for $\sharp$ have the form:
To show $\alpha : \sharp(A_1, \dots A_n)$ open one or more metaboxes b_j with $\alpha_{i,j} : C_{i,j}$ as assumptions and $\gamma_{k,j} : D_{k,j}$ as goals to show. The $\alpha_{i,j} : C_{i,j}$ and $\gamma_{k,j} : D_{k,j}$ are the other formulas involved in an elimination rule of $\alpha : \sharp(A_1, \dots, A_n)$ and it is assumed that according to some measure of complexity, the complexity of b_j is less than the complexity of the original problem. The labelling langauge $\mathcal{A}$ is involved in the relationship between $\alpha, \alpha_{i,j}$ and $\gamma_{k,j}$.

   For example, to show $\alpha : A \to B$, open a box, assume $a : A$ and show $\alpha a : B$, where a is a new atomic label.

3. **Data Rules (Elimination rules)**
These have the form:

$$\frac{t_1 : A_1; \dots; t_n : A_n, \text{ with restrictions } \varphi(t_1, \dots, t_n) \text{ on } t_i}{s : B; \text{ with } t_i \text{ and } s \text{ satisfying some formula } \psi(t_i, s) \text{ of the logic } \mathcal{A}}$$

   where A_i involve the connective $\sharp$ in some complexity and B involves $\sharp$ to a less complexity. This is an elimination rule.

   The simplest form is that A_i involve the connective $\sharp$ and B does not. Examples are:

   Modus Ponens: $$\frac{\alpha : A; \beta : A \to B; \alpha \cap \beta = \varnothing}{\alpha \cup \beta : B}$$

   Ackerman Rule: $$\frac{\alpha : A; \beta : B \to (A \to C)}{\alpha \cup \beta : B \to C}$$

   Possible Worlds Rule: $$\frac{t : \Diamond A}{s : A}$$

   s being a new label with tRs, where R is the external relation on labels.

4. **Metabox Rules**
These rules say which assumptions are reiterated into the new box in (1) and how they are to be used.

   For example we may allow only implications $A \to B$ to be reiterated and require that they be used as tickets. Or we may allow $\Box A(u)$ to be reiterated even though u is free in A. (This corresponds to the Barcan formula in modal logic.)

5. **Ordered metaboxes** *A metabox system is comprised of the following components.*

   (a) A finite partially ordered set of metaboxes names $\{a_i, <\}$. The relation $a < b$ is the subordination relation, meaning that the metabox b is used to justify a line in the proof in metabox a. Let $a <_1 b$ read that b is an immediate successor of a.

   (b) With each metabox name a a sequence $\mathbf{B}(a)$ of triples is associated. The triples have the form (α, A, Π), where α is a label, A is a formula and Π is the justification. The following conditions are satisfied:

   (5.2.1) Each element in the sequence is either an assumption, a reiteration or is obtained from previous elements of the sequence via a data rule.

   (5.2.2) If (α, A, Π) is an assumption then Π = "assumption".

   (5.2.3) If (α, A, Π) is a reiteration then form some box names $a_1, \dots, a_n = a$ we have Π = "reiteration (α, A) from Box a_1" and $a_1 <_1 a_2 <_1 \dots <_1 a_n = a$ and in the box $\mathbf{B}(a_1)$ the line (α, A, Π') appears earlier than a line (β, B, Π'') where the justification Π'' = "from Box a_2" appears. We also require that the reiteration be allowed by the metabox rules.

(5.2.4) (α, A, Π) *is justified by a box b. In this case* $\Pi=$ *"from box b using the appropriate exit rule".*

(5.2.5) (α, A, Π) *is obtained through a data rule from earlier elements in the sequence. In this case there is a rule R:*

$$\frac{t_i : A_i}{s : B}$$

and instances of this rule appear earlier in the sequence and $\alpha : A$ *is an instance of* $s : B$ *and* Π = *"from earlier instances in this box of the rule R".*

(c) Each box b is introduced uniquely once in one other box $a < b$ *to justify a line* (α, A, Π) *in the sequence* $\mathbf{B}(a)$*. The assumption of b are first in the sequence* $\mathbf{B}(b)$ *and agree with the Goal Rules of the connectives of A. The last element in the sequence* $\mathbf{B}(b)$ *is* $\beta : B$*. We further require that* $\alpha : A$ *and* $\beta : B$ *and the assumptions of b satisfy to correct exit rule quoted by* Π*.*

Chapter 4

Proof Theory and Semantics for Algebraic LDS

4.1 Introduction

This chapter develops proof theory and semantics for algebraic *LDS* in detail, with certain specific algebras, needed in representing an arbitrary logic consequnce relation as an *LDS*. We deal with propositional languages first, and illustrate our methodological principles. A later chapter will study the options available in *LDS* for quantifiers. The algebras studied in this chapter are more specific than those given in definition 3.2.4. That definition requires an algebra (labelling language) $\mathcal{A}$ and a language $\mathbf{L}$. In our case we assume that $\mathcal{A}$ is a cross product of the form $\mathcal{A} = \mathcal{F} \times \mathcal{A}_0$, where $\mathcal{F}$ is the free concatenation algebra of definition 3.2.2 and $\mathcal{A}_0$ contains some specific function symbols and an ordering which will be used later in formulating a proof theory for our systems. This ordering shows up again in a later chapter on the algebraic study of consequence relations.

This specific presentation of the algebra $\mathcal{A}$ does not restrict its generality. The algebra $\mathcal{F}$ is used to keep track of proofs and does not interfere with the main labelling algebra $\mathcal{A}_0$, which can be arbitrary. The ordering in $\mathcal{A}_0$ is a natural one, and can always be formally defined as equality for any algebra which does not already have a natural ordering.

Example 4.1.1 (Typical example) *Suppose we are given data of the form:*

$$\begin{aligned} &d_1 : A \\ &d_2 : A \\ &d_3 : A \to B \\ &d_4 : A \to B \end{aligned}$$

d_1, d_2, d_3 indicate different sources for the data items and also serve as letters to name them. The source d_1 is more reliable than d_2, and we associate a number with A, indicating how reliable A is. The source d_2 also puts for ward A,but it is less reliable. Say d_1 puts forward A with reliability value $\alpha_1 = 0.7$. Similarly, $\alpha_2 = 0.5$ and $\alpha_3 = \alpha_4 = 0.2$. Suppose we agree that when we perform modus ponens we multiply the numbers. We can now derive B in several ways:

$$\begin{array}{lll} d_3 * d_1 : B & \textit{reliability} & \alpha_3 \cdot \alpha_1 = 0.14 \\ d_3 * d_2 : B & \textit{reliability} & \alpha_3 \cdot \alpha_2 = 0.10 \\ d_4 * d_2 : B & \textit{reliability} & \alpha_4 \cdot \alpha_2 = 0.10 \end{array}$$

The d sequence actually records how B is derived (notice we put d_3 first in the sequence because it names the ticket) and the α multiplication sequence gives the numerical reliability. We might wish to forbid the application of modus ponens between data of incompatible sources. Let us say that sources with odd indices are not compatible with sources with even indices. Thus the derivation *of $d_4 * d_1 : B$ is blocked. Thus we may write $\varphi(x, y)$ to indicate that x and y are compatible. Then the modus ponens rule becomes*

$$\frac{\varphi(a, b); (a, \alpha) : A; (b, \beta) : A \to B}{(b * a, \alpha \cdot \beta) : B}$$

Suppose we accept as proven formulas obtained with reliability $\geqq 0.5$. There may not be a single proof with such reliability, however, if there are several derivations of the same formula from different sources, we can aggregate the reliabilities.

Let us examine what kind of algebra we need for this system.

- *The operation $\otimes$ for modus ponens.*

  $$(a, \alpha) \otimes (b, \beta) = (b * a, \alpha \cdot \beta)$$

  where '' is conatenation and '$\cdot$' is number multiplication.*

- *The relation $\varphi(x, y)$ of compatibility of labels (to keep modus ponens within data of the same source).*

- *The relation $\leq$ of strength among labels, where*

  $$(a, \alpha) \leq (b, \beta) \text{ iff definition } a \subseteq b \text{ and } \alpha \leqq \beta$$

  $a \subseteq b$ means the sequence b contains the sequence a and $\alpha \leq \beta$ is numerical comparison. $\leq$ is the relation of higher priority and relying on more data.

- *The operation $\uplus$ on labels indicating aggregation.*

  $$(a, \alpha) \uplus (b, \beta) = \text{definition } (a \cup b, \alpha + \beta)$$

  where $+$ is numerical addition and $\cup$ is disjoint union of sequences.

Thus in our example B can be proved with label

$$(d_3, \alpha_3) \otimes (d_1, \alpha_1) \uplus (d_4, \alpha_4) \otimes (d_2, \alpha_2) = (d_3 * d_1 \cup d_4 * d_2, 0.24)$$

See also Examples 1.6.1 and 2.3.12.

- *The set $\mathbf{E}$ of distinguished labels $\mathbf{e} \in \mathbf{E}$, which indicate acceptability. For example $\mathbf{E} = \{(x, \alpha) \mid 0.5 \leq \alpha\}$. $\mathbf{E}$ must satisfy:*

  $$x \in \mathbf{E} \text{ and } x \leq y \text{ imply } y \in \mathbf{E}.$$

  Thus we accept any B which can be derived with a label $\mathbf{e} : B, \mathbf{e} \in \mathbf{E}$.

The above considerations inidcate that the kind of semantics we need for general implicational logics has the following form.

Definition 4.1.2 (General semantics for implicational logics) *Let $\mid\!\sim$ be a propositional consequence relation for a language with $\to$ satisfying identity ($A \mid\!\sim A$) and transitivity ($A \mid\!\sim B$ and $B \mid\!\sim C$ imply $A \mid\!\sim C$). Assume that $A \to B$ is antimonotonic in A and monotonic in B. Then the following is a general semantics for $\mid\!\sim$.*

1. *Models have the form $\mathbf{m} = (S, \leq, \uplus, \varphi, \mathbf{e}, f, h)$, where S is a nonempty set of possible worlds. $\mathbf{e} \in S$ is the actual world. $\varphi \subseteq S \times S$ is the compatibiltiy relation, $\uplus$ is the associative and commutative aggregate binary function and h is the assignment. For each atomic q, $h(q) \subseteq S$. $\leq$ is a reflexive and transitive relation on S. f is a binary 'accessibility' function on S. h satisfies the condition $t \in h(q)$ and $t \leq s$ imply $s \in h(q)$.*

2. *The notion of satisfaction $t \vDash_h A$, under h, is defined by induction as follows:*
   - *$t \vDash_h q$ iff $t \in h(q)$ for atomic q.*
   - *$t \vDash_h A \to B$ iff $\forall s(\varphi(t,s)$ and $s \vDash_h A$ imply $f(t,s) \vDash_h B)$.*
   - *We say $\mathbf{m} \vDash A$ iff $\mathbf{e} \vDash_h A$.*
3. *Let $\mathcal{K}$ be a class of models.*
   - *We say $\mathcal{K} \vDash A$ iff $\mathbf{m} \vDash A$ for all $\mathbf{m} \in \mathcal{K}$.*
   - *A model $\mathbf{m}$ is* normal *iff $\varphi(t,s)$ is not dependent on t.*
   - *A class of models $\mathcal{K}$ is normal iff all $\mathbf{m} \in \mathcal{K}$ are normal.*

Example 4.1.3 (Modal logic in the general semantics) *1. To explain the previous definition, let us see what modal logics look like in such an environment. Consider the modal logics **K** or **S4**. These are complete for suitable classes of Kripke structures of the form (S,R,a,h), where $R \subseteq S^2, a \in S$ and h is the assignment. It is possible to obtain completeness for the class of Kripke models where the possible worlds are trees. Thus, for a set S_0 of atoms we have $S \subseteq S_0^*$, where S_0^* is the set of all finite sequences of elements of S_0 and S is closed under initial segments. The actual world is $\varnothing$, the empty sequence.*

*The accessibility relation for modal **K** can be defined by*

$$xRy \text{ iff for some atom } s \in S_0 \text{ we have } y = x * (s)$$

where $*$ is concatenation.

*The accessibility relation for **S4** can be defined by:*

$$xRy \text{ iff for some } z \text{ (which may also be empty) we have } y = x * z.$$

Satisfaction for modality $\Box B$ (also written $\top \to B$, using strict implication $\to$ and truth *$\top$) is defined in the traditional manner in a Kripke model.*

$x \vDash \Box B$ iff for all y such that xRy we have $y \vDash B$.

*for modal **K** we observed that*

$$xRy \text{ iff } \exists z \in S_0(x = y * z)$$

thus we get

$$\begin{aligned} & x \vDash \Box B \text{ iff } \forall z(z \in S_0 \text{ implies } y * z \vDash B) \\ & \text{iff } (\forall z(\varphi(x,z) \text{ and } z \vDash \top \text{ imply } f(x,z) \vDash B) \end{aligned}$$

where $\varphi(x,z) = z \in S_0$ and $f(x,z) = x * z$.

The last clause is identical with the truth condition of $\top \to B$ of the previous definition.

*If we choose $\varphi(x,z)$ to mean $z \in S_0^*$ we get **S4** semantics. If we choose $\varphi(x,z)$ to mean $x = \varnothing \vee z \neq \varnothing$, we get the semantics where R is transitive everywhere but reflexive only at the actual world.*

2. There is another uniform way of presenting the semantics of strict implication $A \to B$. The table for strict implication is

$$t \vDash A \to B \text{ iff } \forall s(tRs \text{ and } s \vDash A \text{ imply } s \vDash B).$$

We define a function $f(x,y)$ on trees as follows. Let u be the maximal sequence such that for some $x_0, y_0, x = u * x_0, y = u * y_0$. Denote u by $u = x \cap y$. u may be equal to $\varnothing$. Denote x_0 by $x - x \cap y$ and similarly y_0. See figure 4.1

Define $f(x,y)$ to be $x * (y - x \cap y) = x * y_0$. In particular we get

$$f(x,u) = x * \varnothing = x \text{ and } f(u,x) = u * x_0 = x.$$

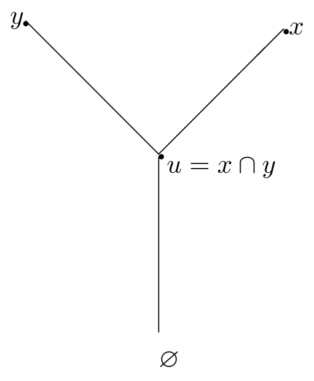


Figure 4.1:

Let $\varphi(x, y)$ be $x \cap y = x$.

Thus we can write

$$x \vDash A \to B \text{ iff for all } y(x \cap y = x \text{ and } y \vDash A \text{ imply } f(x, y) \vDash B).$$

Since $x \cap y = x$ implies xRy and implies $f(x, y) = y$, we get that $\to$ is **S4** *strict implication. To get* **K** *semantics for $\to$ we let $\varphi(x, y)$ be the conjunction of the two conditions $x \cap y = x$ and $(y - x \cap y) \in S_0$, i.e. y is only one atom longer than x.*

This definition of the function f is good for all substructural and many valued implications.

Example 4.1.4 (Intuitionistic logic in the general semantics) *Kripke models for intuitionistic logic can also be taken as trees. They have the form $(T, \leq, \varnothing)$ where $x \leq y$ iff $\exists z(y = x * z)$. This indicates that in the general semantics we should require $z \leq f(x, z)$. For the choice of $\varphi(x, y) = x \cap y = x$ and $f(x, y) = x * (y - x \cap y)$ we get that the table:*

$$x \vDash A \to B \text{ iff } \forall y(\varphi(x, y) \text{ and } y \vDash A \text{ imply } f(x, y) \vDash B)$$

gives intuitionistic implication.

Example 4.1.5 (Translation through the general semantics) *It is possible to translate any wff A into classical logic with binary $\varphi(x, y)$ and function symbol $f(x, y)$ through the semantics. We translate the pair $[t : A]$ (meaning $t \vDash A$ or t labels A) and the translation is $*$. With atomic q associate a classical unary predicate $Q(x)$.*

$$\begin{array}{rcl} [t : q]^* & = & Q(t) \\ [t : A \to B]^* & = & \forall y(\varphi(t, y) \wedge [y : A]^* \Rightarrow [f(t, y) : B]^*) \end{array}$$

Example 4.1.6 (Linear database) 1. *Consider a language with $\to$ and algebra $\mathcal{A}$. We define the notion of* linear database $t : \Delta$ *for this language. A linear database with base t has the form*

$$t : \{s_1 : A_1; \ldots ; s_n : A_n, \varphi(t, s_1), \varphi(t_1, s_2), \ldots , \varphi(t_{n-1}, s_n)\}$$

where $t_0 = t$ and $t_i = f(t_{i-1}, s_i)$.

Such databases arise in evaluating wffs.

Consider $B = A_1 \to (\ldots \to (A_n \to q) \ldots)$. Then

$t \vDash B$ *iff* $\forall s_1(s_1 \vDash A_1$ *and* $\varphi(t, s_1)$ *imply* $f(t, s_1) \vDash A_2 \to (\ldots (A_n \to q) \ldots)$.

$t_1 = f(t, s_1) \vDash A_2 \ldots (A_n \to q) \ldots)$ *iff* $\forall s_2(s_2 \vDash A_2$ *and* $\varphi(t_1, s_2)$ *imply* $f(t_1, s_2) \vDash A_3 \to \ldots \to (A_n \to A) \ldots)$.

The 'database' is the sequence

$$t : \{s_1 : A_1; \ldots ; s_n : A_n, \ldots , \varphi(t_i, s_{i+1}), \ldots\}$$

2. *Consider an unlabelled traditional substructural logic $\mathbf{L}_{\twoheadrightarrow}$ for an implication $\twoheadrightarrow$. Assume the theorems of $\mathbf{L}_{\twoheadrightarrow}$ are axiomatised in some way, say as a Hilbert system or a Gentzen system, etc. Assume the language also contains unary predicates $Q(x)$, one binary predicate $\varphi(x, y)$ and the connective $\wedge$, and the universal quantifier $\forall$. Let $\mathcal{A}$ be an algebra with φ and f (we are abusing notation here and using the same letter φ) and, assume that the variables of $\mathbf{L}_{\twoheadrightarrow}$ range over $\mathcal{A}$ and that all elements of $\mathcal{A}$ are constants of $\mathbf{L}_{\twoheadrightarrow}$. We can thus legitimately write $Q(t)$, for t in $\mathcal{A}$.*

   We can translate any $t : A$ of $(\mathcal{A}, \mathbf{L})$ into a formula $[t : A]^\tau$ of $\mathbf{L}_{\twoheadrightarrow}$, ($\tau$ denotes the translation) as follows:

   - $[y_0 : q]^\tau = Q(y_0), q$ *atomic*
   - $[y_0 : A_1 \to (A_2 \to \ldots (A_n \to q) \ldots)]^\tau$
     $= \forall x_1, \ldots , x_n([\bigwedge_{i=1}^n [x_i : A_i]^\tau \wedge \bigwedge_{i=1}^n \varphi(y_{n-1}, x_n) \twoheadrightarrow Q(y_n))$ *where $y_1, \ldots , y_n$ are abbreviations for terms defined by the recursion equation*

     $$y_i = f(y_{i-1}, x_i)$$

     Thus y_0 is the only free variable of the above translation.

   Note that the translated formulas reside in a fragment of the $\twoheadrightarrow$ logic, defined as follows:

   Universal $\twoheadrightarrow$ fragment

   - *An atomic A is in the fragment*
   - *If A_k is in the fragment, and B is atomic, then*

     $$\bigwedge_k A_k \twoheadrightarrow B$$

     is in the fragment.
   - *If $A(x)$ is in the gragment so is $\forall x A(x)$.*

   We shall see later that for some algebras $\mathcal{A}$ and some axioms on $\twoheadrightarrow$ we may have:

   (*) $\varnothing \vdash_{(\mathcal{A},\mathbf{L})} \mathbf{e} : A$ *iff* $\vdash_{\mathbf{L}_{\twoheadrightarrow}} [\mathbf{e} : A]^\tau$.

   Note that if $\mathbf{L}_{\twoheadrightarrow}$ has some nice semantics, then (*) will endow a semantics on $(\mathcal{A}, \mathbf{L})$.

   It is useful to think of $\twoheadrightarrow$ as intuitionistic implication.

This chapter will concentrate on the following topics:

- Proof theory for algebraic *LDS*, natural deduction formulation.
- Conversion methods from natural deduction formulation to a Hilbert formulation and back.
- A theorem showing that one can adopt the view that introduction rules are always the 'inverse' of elimination rules.
- A methodology for combining any two algebric *LDS* systems
- Tableaux methods for any algebraic *LDS*
- Semantics and general completeness theorems.
- Examples and applications from well known logics.

4.2 Basic proof theory

This section develops the basic proof theoretic notions for resource propositional *LDS*. We need the exact formulations of the language, the labelling, the consequence relation and various notions of theoremhood.

Definition 4.2.1 (Resource propositional LDS) *1. Let* $\mathbf{L}$ *be a fixed propositional language with atomic propositional variables* $\{q_1, q_2, \ldots\}$ *and connectives* $\{\natural_1, \ldots, \natural_n\}$ *of arities* $r_1, \ldots, r_n$ *respectively. Let* $\mid\!\sim$ *be a consequence relation on* $\mathbf{L}$ *satisfying identity* $(A\mid\!\sim A)$ *and transitivity* $(A\mid\!\sim B$ *and* $B\mid\!\sim C$ *imply* $A\mid\!\sim C)$*. A connective* $\natural(\ldots, x, \ldots)$ *is said to be monotonic (anit-monotonic) in place* x *if the following holds for all* A, B*.*

$A\mid\!\sim B$ *implies* $\natural(\ldots, A, \ldots)\mid\!\sim\natural(\ldots, B, \ldots)$ *(resp.* $\natural(\ldots, B, \ldots)\mid\!\sim\natural(\ldots, A, \ldots)$*).*

We can assume the connectives of $\mathbf{L}$ *are either monotonic or antimonotonic or both in each argument and that the arities can be written as* $r_i = r_i^+ + r_i^-$*. The meaning of the* r_i^+ *and* r_i^- *partition is that the connective is monotonic up in* r_i *places and monotonic down in* r_i^- *places. We shall see later that such logics and connectives are very general indeed. For example,* $A \to B$ *is monotonic up in* B *and down in* A*. Thus* $\to$ *has arity* $r = 2$ *with* $r^+ = r^- = 1$*.*

2. *A resource algebraic labelling language* $\mathcal{L}$ *for* $\mathbf{L}$ *and* $\mid\!\sim$ *has the form*

$$\mathcal{A} = (\delta, \tau, S, \leq, \uplus, \varphi_i, \mathbf{e}, f_{i,k,j}(y, x_1, \ldots, x_{r_i^-}), j = 1, \ldots, r_i^+, i = 1, \ldots, n, k = 1, \ldots, k_i,)$$

where S *is the set of atomic labels,* $\mathbf{e} \in S$ *is a distinguished label.* $f_{i,k,j}$ *are Skolem functions which generate from* S *the set* S^* *of all algebraic labels.* $\leq$ *is a reflexive and transitive relation on* S^**.* τ *is a syntactical theory in the (possibly higher order) language of the algebra, which* $(S^*, \leq, \mathbf{e}, \uplus, \varphi_i, f_{i,k,j})$ *satisfies and* $\delta(x)$ *is a formula (possibly higher-order) of the algebra with one free variable. The* φ_i *are* $r_i^- + 1$ *relations on* S^* *and* $\uplus$ *is an associative binary, commutative aggregation function on* S^**.*

$\uplus$ *satisfies the following, for each term* $t(\alpha_1, \ldots, \alpha_{r_i^-}, x)$ *of the algebra with a free variable* x *and* $\alpha_1, \ldots, \alpha_{r_i^-}$ *and each function symbol* $f_{i,k,j}(\alpha, \alpha_1, \ldots, \alpha_{r_i^-})$*;* $t(\alpha_1, \ldots, \alpha_{r_i^-}, f_{i,k,j}(\alpha, \alpha_1, \ldots, \alpha_{r_i^-}))\uplus \ldots \uplus t(\alpha_1, \ldots, \alpha_{r_i^-}, f_i, k, r_i^+(\alpha, \alpha_1, \ldots, \alpha_{r_i^-})) \leq t(\alpha_1, \ldots, \alpha_{r_i^-}, \alpha)$*.*

A database Δ *has the form* $\Delta = (D, \mathbf{f}, d)$*, where* D *is a finite diagram of the algebra* $\mathcal{A}$ *(i.e. a set of labels and relations on them),* d *is a label and* $\mathbf{f}$ *is a function associating with label* t *of* D *a wff* $\mathbf{f}(t)$*.*

Let $\delta(x)$ *be a (possibly higher-order) formula of the algebra* $\mathcal{A}$*. We write* $(D, d) \mid\!\equiv \delta(t)$*, for a label* t*, iff* (D, d) *satisfies* $\delta(t)$ *as a Herbrand model, where quantifiers range over elements of* D *only. This can be achieved syntactically if* D *is a Horn clause database and* $D \mid\!\equiv \delta$ *means that* δ *succeeds as a query from* $D \cup \tau$*. Compare with item (j) of Definition 3.2.1.*

Usually a logical system comes with a pair (δ', δ) *of formulas.* δ' *is a formula restricting the diagram* (D, d)*, for example saying all labels in* D *are atomic (from* S*), while* $\delta(t)$ *says what kind of labels can be proved from the database, for example, saying that the label* t *contains all labels in* D*.*

In many cases we have

$$(D, d) \mid\!\equiv \delta(t) \text{ iff } \vDash \delta_D \Rightarrow \delta(t)$$

where δ_D *is the conjunction of all the formulas of* D*. The right hand side is validity in any classical model while the left-hand side is* $\mid\!\equiv \delta_D \Rightarrow \delta(t)$*, i.e. validity in all Herbrand universes. They are the same for certain syntactical forms.*

3. *For each connective* $\natural_i$ *consider the following* φ_i*-elimination rules* $e_{i,k,j}, j = 1, \ldots, r_i^+, k = 1, \ldots, k_i$*.*

$$\frac{\varphi_i(\alpha,\alpha_1,\ldots,\alpha_{r_i^-});\alpha_1:A_1;\ldots,\alpha_{r_i^-};\alpha:\sharp_i(A_1,\ldots,A_{r_i^-},B_1,\ldots,B_{r_i^+})}{f_{i,k,j}(\alpha,\alpha_1,\ldots,\alpha_{r_i^-}):B_j}$$

In view of the above rule we can write $\sharp(A_1,\ldots,A_{r_i^-},B_1\ldots,B_{r_i^+})$ *as* $(A_1,\ldots,A_{r_i^-})\to_i(B_1,\ldots,B_{r_i^+})$.

4. *There are some natural conditions to require on the algebra and the* φ *elimination rules. These conditions arise in the* LDS *formulation of current well known existing logics. Among them are*

   - $\forall x\varphi(\mathbf{e},x)$
   - *Residuation:*
     for every $n+1$ *place* $y=f(x,x_1,\ldots,x_n)$ *and every* $b,a_1,\ldots,a_n$ *there exists a unique* a *such that* $b=f(a,x_1,\ldots,x_n)$.
   - $\mathbf{e}$ *is a unit:*
     For every $n+1$ *place* f *and* a *we have* $f(a,\mathbf{e},\ldots,\mathbf{e})=a$.
   - *For a binary* $f(x,y)$, $\forall y[f(a,y)\leq f(b,y)]\Rightarrow a\leq b$

   We can require the algebra to be archimedian, in the following sense. Let $a_1,\ldots,a_n\in\mathcal{A}$ *be a finite number of elements and let* $b\in\mathcal{A}$. *Let* $\{\{a_1,\ldots,a_n\}\}$ *be the set generated from* $\{a_1,\ldots,a_n\}$ *by repeatedly applying the function symbols of* $\mathcal{A}$ *to these elements.*

   Let b *be any element and let* $I_b=\{a\in\mathcal{A}\mid a\leq b\}$. *Then* $\{\{a_1,\ldots,a_n\}\}\cap I_b$ *is finite. Furthermore, for all* $a_1,\ldots,a_n,b$ *there exists a natural number* k *that any term obtained from more than* k *repeated application of function symbols to* a_i *is* not $\leq b$.

5. *Let* $\mathcal{A}_1,\mathcal{A}_2$ *be two algebras for the same language* $\mathbf{L}$, *i.e. both have the form of (3) above:*

   $$\mathcal{A}_m=(\tau^m,S^m,\uplus^m,\varphi_i^m,\mathbf{e}^m,\leq^m,f_{i,k,j}^m).$$

   We define the algebra $\mathcal{A}_1\times\mathcal{A}_2$. *Let* $S=S^1\times S^2$. *The wffs and functions of* $\mathcal{A}$ *are pairs of wffs and functions* $\varphi=(\varphi^1,\varphi^2),f=(f^1,f^2)$, *where* φ^m,f^m *are in the language of* $\mathcal{A}_m$. *Let* $\varphi((s_1,s_2)),s_1\in(S^1)^*,s_2\in(s^2)^*$ *hold iff both* $\varphi^1(s_1)$ *and* $\varphi^2(s_2)$ *hold. Let* $f((s_1,s_2))$ *be* $(f^1(s_1),f^2(s_2))$. *In short the functions and predicates are evaluated coordinatewise. Let* $\tau=(\tau^1,\tau^2)$ *and let* $\leq=(\leq^1,\leq^2),\mathbf{e}=(\mathbf{e}^1,\mathbf{e}^2)$, $\uplus=(\uplus^1,\uplus^2)$, *and let the new function symbols be*

   $$f_{(i,k,j)(i',k',j')}=(f^1_{(i,k,j)},f^2_{(i',k',j')})$$

   Let e_1,e_2 *be two elimination rules from the respective algebras of the form*

   $$e_m\quad\frac{\varphi_m;\alpha_1^m:A_1,\ldots,\alpha_n^m:A_n;\beta_m:B}{f^m(\beta_m,\alpha_1,\ldots,\alpha_n):C}$$

   Then the following elimination rule $e_{1,2}$ *is available in* $\mathcal{A}$.

   $$e_{1,2}\quad\frac{(\varphi_1,\varphi_2);(\alpha_1^1,\alpha_1^2):A_1;\ldots;(\alpha_n^1,\alpha_n^2):A_n;(\beta^1,\beta^2):B}{(f^1,f^2)((\beta^1,\beta^2),(\alpha_1^1,\alpha_1^2),\ldots,(\alpha_n^1,\alpha_n^2)):C}$$

6. *Conversely, let* $\mathcal{A}=\mathcal{A}_1\times\mathcal{A}_2$ *be an algebra which is a cross product of* $\mathcal{A}_1$ *and* $\mathcal{A}_2$. *Assume the elimination rules of* $\mathcal{A}$ *all have the form:*

   $$\frac{\varphi_1(\beta^1,\alpha_1^1,\ldots,\alpha_1^1);\varphi_2(\beta^2,\alpha_1^2,\ldots,\alpha_n^2);(\alpha_i^1,\alpha_i^2):A_i,\ldots;(\beta^1,\beta^2):B}{(f^1(\beta^1,\alpha_i^1),f^2(\beta^2,\alpha_i^2):C}$$

Then one can consider only the first $\mathcal{A}_1$ component of the algebra and derive an LDS, $(\mathcal{A}_1, \mathbf{L})$ *from the first component of the rules. So the first component of the rule above is:*

$$\frac{\varphi_1; \alpha_i^1 : A_i; \beta^1 : B}{f^1(\beta^1, \alpha_i^1) : C}$$

Let e_1 be a rule of $\mathcal{A}_1$ and e_2 be a rule of $\mathcal{A}_2$. e_2 is said to be a companion to e_1 if $e^{1,2} = (e_1, e_2)$ is a rule of $\mathcal{A}$.

7. *A semialgebraic resource labelling language has the form $\mathcal{A} = \mathcal{F} \times \mathcal{L}$, where $\mathcal{F}$ is the free algebra of definition 3.2.2. Let $(a, \alpha) \leq (b, \beta)$ be defined as $a = b$ and $\alpha \leq \beta$. The (i, k, j)th rule $e_{i,k,j}$ now becomes the (ρ_i, φ_i) resource rule (labelled $e_{i,k,j}$):*

$$\frac{\rho_i(a, a_1, \ldots, a_{r_i^-}); \varphi_i(\alpha, \alpha_1, \ldots \alpha_{r_i^-}); (a_1, \alpha_1) : A_1; \ldots; (a_{r_i^-}, \alpha_{r_i^-}) : A_{r_i^-}; (a, \alpha) : \sharp(A_1, \ldots, A_{r_i^-}, B_1, \ldots, B_{r_i^+})}{(((e_{i,k,j}, a); (a_1, \ldots, a_{r_i^-})), f_{i,k,j}(\alpha, \alpha_1, \ldots, \alpha_{r_i^-})) : B}$$

Remark 4.2.2 (Notes on resource elimination rules) 1. *Note that the assumptions in the elimination rule e are ordered. Thus, for a three place connective*

$$(A_1, A_2) \rightarrow_3 B$$

the rule becomes

$$\frac{\rho(a, a_1, a_2); \varphi(\alpha, \alpha_1, \alpha_2); (a_1, \alpha_1) : A_1 (a_2, \alpha_2) : A_2; (a, \alpha) : (A_1, A_2) \rightarrow_3 B}{(((e, a), (a_1, a_2)), f(\alpha, \alpha_1, \alpha_2)) : B}$$

A_1 goes in the first slot of $\rightarrow_3$ and A_2 in the second.

For example the two disjunction rules become

$$e_1 \quad \frac{(a, \alpha) : A \vee B}{((e_1, a), f_1(\alpha)) : A}$$

$$e_2 \quad \frac{(a, \alpha) : A \vee B}{((e_2, a), f_2(\alpha)) : B}$$

It is possible to recognise the left and right disjunct from the rule applied.

Of course, the modus ponens condition φ plays an important role in restricting the (mis)use of the disjunction rules. We can restrict the modus ponens of $\{\alpha : A; \beta : A \rightarrow B\}$ by requiring that for no label t do we have $\{f_1(t), f_2(t)\} \subseteq$ set $(\alpha) \cup$ set (β).

The elimination rules for conjunction can be taken as

$$\frac{t : A \wedge B}{t : A} \text{ and } \frac{t : A \wedge B}{t : B}$$

There is a difference between conjunction and disjunction. Disjunction has one rule e which has two componenets, e_1 and e_2. Conjunction has two rules. Each rule has one component. In general a connective, say $A \twoheadrightarrow (B_1, B_2)$, may have several rules each having two components. The difference between a new rule and a new component comes in the corresponding introduction rule. Our approach to introduction rules is that they can be obtained uniformly and automatically from the elimination rules.

Thus if e_1 and e_2 are two elimination rules for $\twoheadrightarrow$ above, each with two components we get the form

$$e_{i,j} \frac{\varphi(\beta, \alpha); \alpha : A; \beta : A \twoheadrightarrow (B_1, B_2)}{f_{i,j}(\beta, \alpha) : B_j}.$$

To be able to introduce $\beta : A \twoheadrightarrow (B_1, B_2)$ at some stage in the proof we must show that essentially our proof machinery already has it at that stage. This means that if we assume *the antecedent $\alpha : A$ for a suitable α (i.e. such that $\varphi(\beta,\alpha)$ holds), we can show in a subproof at that same stage that the consequent for each rule e_1 and e_2 of $\twoheadrightarrow$. Showing the consequent means showing at least one of the B_j.*

Thus we need to show that for an arbitrary α such that $\varphi(\beta,\alpha)$ and arbitrary rule e_i there exists at least one j such that

$$\varphi(\beta,\alpha); \alpha : A \vdash f_{ij}(\beta,\alpha) : B_j$$

So to show conjunction $\beta : A \wedge B$ we must show the result of each rule i.e. we must show $\beta : A$ and show $\beta : B$, since the rules are:

$$\frac{\beta : A \wedge B}{\beta : A} \quad \textit{and} \quad \frac{\beta : A \wedge B}{\beta : B}.$$

To introduce $\beta : A \vee B$ we must show one *of the two consequences of the single rule i.e. either $f_1(\beta) : A$ or $f_2(\beta) : B$.*

By way of illustration, let us intuitively prove $A \to A \vee B$.

Assume $\beta : A$
Show $\beta : A \vee B$

If we claim we 'have' $\beta : A \vee B$ we must be able to 'perform' the elimination rule. We choose the $f_1(\beta) : A$ option. Can we derive $f_1(\beta) : A$ from $\beta : A$? Yes, if in the algebra we have $\beta \leq f_1(\beta)$.

Let us check how to prove

(a) $\beta : A \vee B$

(b) $\alpha : A \to C$

(c) $\alpha : B \to C$

(d) $\beta * \alpha : C$

We reason as follows:

(e) $f_1(\beta) : A$ *from (a)*

(f) $\alpha * f_1(\beta : C$ *from (b) and (c)*

(g) $f_2(\beta) : B$ *from (a)*

(h) $\alpha * f_2(\beta) : C$ *from (c) and (g).*

Consider now $\lambda x(\alpha * x) = t(x)$. We have

$$t(x/f_1(\beta)) : C$$
$$t(x/f_2(\beta)) : C$$

and we want to deduce

$$t(x/\beta) : C.$$

We need this as a flattening rule of the algebra.

We can also write it as:

$$t(f_1(\beta)) \uplus t(f_2(\beta) \leq t(\beta).$$

Note that C must be derived from A and from B in teh same way, as exhibited by the term $\lambda x t(x)$.

2. *Note that the* LDS *arising out of the labelling $\mathcal{A}_1 \times \mathcal{A}_2$ is not the intersection in any sense of the logics with $\mathcal{A}_1$ and $\mathcal{A}_2$. It is more like the sharing of the proof paths and propagating the labels of each algebra side by side. To illustrate, consider the database:*

$$\begin{array}{l} t_1 : A \\ t_2 : A \to (A \to B) \\ t_3 : A \\ t_1 < t_2 < t_3 \end{array}$$

The logic $\mathcal{A}_1$ is labelled in such a way that it starts modus ponens by taking A from above the $A \to B$ and then alternates to below and so on. While $\mathcal{A}_2$ starts the opposite way. Thus both logics individually can prove B but the cross product cannot prove B. So the cross product is not the intersection of what can be proved in both.

3. *Further note that the definition of the Elimination Rules for resource* LDS *can be refined as follows:*

   - *A φ-rule of the form:*

   $$\frac{\varphi(t, t_1, \ldots, t_k); t_1 : A_1; \ldots; t_k : A_k; t : B}{f(t, t_1, t_2, \ldots, t_k) : C}$$

   can be written as a (φ, ψ) rule eliminating $t : B$

   $$\frac{\varphi(t, t_i); t_i : A_i; t : B}{\psi(s, t, t_i); s : C}$$

   where $\psi(s, t, t_1, \ldots, t_k)$ is $f(t, t_1, \ldots, t_k) \leq s$.

   We write '$\leq s$' rather than '$= s$' because of the inference rule

   $$\Delta \vdash s_1 : A \textit{ and } s_1 \leq s_2 \textit{ imply } \Delta \vdash s_2 : A.$$

4. *Furthermore, the corresponding 'inverse' introduction rule becomes the following:*
   $\Delta \vdash t : B$ if for some $t_1, \ldots, t_k$, and t such that $\varphi(t, t_1, \ldots, t_k)$ holds, we have that:
   $\Delta \cup \{t_1 : A_1; \ldots; t_k : A_k\} \vdash s : C$, where $t_1, \ldots, t_k$ are arbitrary *in the set $\langle x_1, \ldots, x_k\rangle \varphi(t, x_1, \ldots, x_k) = \{(t_1, \ldots, t_k) \mid \varphi(t, t_1, \ldots, t_k\}$, and t is the y such that $\psi(s, y, t_1, \ldots, t_k)$ holds (if such a y exists).*

   We can use the notation

   $$t = \sigma y \psi(s, y, t_1, \ldots, t_k),$$

   or

   $$t = s/f(t_1, \ldots, t_k)$$

   to denote the y such that $\psi(s, y, t_1, \ldots, t_k)$ holds.

Definition 4.2.3 (Proof theory for resource LDS) *Let $(\mathcal{A}, \mathbf{L})$ be a resource* LDS *with $\mathcal{A} = \mathcal{F} \times \mathcal{A}_0$. We allow for $\mathcal{F}$ or for $\mathcal{A}_0$ not to appear, in which case $\mathcal{A} = \mathcal{F}$ or $\mathcal{A} = \mathcal{A}_0$.*

1. *A database $\Delta = (D, \mathbf{f}, d)$ is a set of labelled formulas of the form $t : A$, where $t = (a, \alpha)$, a from $\mathcal{F}$ and α from $\mathcal{A}_0$, as formally defined in Definition 4.2.1.*

2. *A labelled consequence relation is a relation between databases and declarative units of the form $\Delta \vdash_{\mathcal{A}} t : A$; (written as $\Delta \vdash t : A$ for convenience) as follows:*

   We use two parameters m and n. m for the depth of the use of introduction rules and n for the length of the proof using elimination rules. Thus $\Delta \vdash_{m,n} t : A$ means $t : A$ can be proved in at most n lines. Each line is either obtained from previous lines by elimination rules or is justified by an Introduction rule whose subproof uses at most $m - 1$ nested subproofs.

(a) $\Delta \vdash_{0,0} t : A$ *if* $t : A \in \Delta$ *and* $(D, d) \models \delta(t)$*, (or in many cases* $\vDash \delta_D \Rightarrow \delta(t)$*) as in Definition 4.2.1.*

(b) $\Delta \vdash_{0,n+1} t : A$ *if for some elimination rule of the form*

$$\frac{\rho; \varphi; t_i : C_i, i = 1, \ldots, k}{t : A}$$

we have that ρ *and* φ *hold in the algebra and that* $\Delta \vdash_{0,n_i} t_i : C_i, i = 1, \ldots, k$*, for some* $n_i \leq n$*.*

(c) $\Delta \vdash_{m+1,n+1} t : A$ *if for some elimination rule of the form*

$$\frac{\rho; \varphi; t_i : C_i, i = 1, \ldots, k}{t : A}$$

we have that ρ *and* φ *hold and that* $\Delta \vdash_{m_i,n_i} t_i : C_i, i = 1, \ldots, k$*, for some* $m_i \leq m$ *and* $n_i \leq n$*.*

(d) $\Delta \vdash_{m+1,0} (a, \alpha) : (A_1, \ldots, A_{r_i^-}) \rightarrow_i (B_1, \ldots, B_{r_i^+})$ *if for arbitrary new atomic* $a_1, \ldots, a_{r_i^-}$ *satisfying* $\rho_i(a, a_1, \ldots, a_{r_i^-})$ *and arbitrary* $\alpha_1, \ldots, \alpha_{r_i^-}$ *satisfying* $\varphi_i(\alpha, \alpha_1, \ldots, \alpha_{r_i^-})$ *we have that for each* k *there exists a* j *such that for some* m', n*, with* $m' \leq m$ *we have for each* j

$$\Delta \cup \{(a_1, \alpha_1) : A_1 \ldots, (a_{r_i^-}, \alpha_{r_i^-}) : A_{r_i^-}\} \vdash_{m',m} ((e_{i,k,j}, a), (a_1, \ldots, a_{r_i^-})), f_{i,k,j}(\alpha, \alpha_1, \ldots, \alpha_{r_i^-})) : B_j$$

(e) $\Delta \vdash_{m,n} t : A$ *if* $\Delta \vdash_{m',n'} t' : A$ *for some* $m' \leq m$ *and some* $n' \leq n$*, and some* $t' \leq t$*.*

(f) We let $\Delta \vdash t : A$ *be defined as* $\Delta \vdash_{m,n} t : A$*, for some* m, n*.*

(g) Let $\vdash A$ *mean* $\varnothing \vdash \mathbf{e} : A$

(h) Note, in view of remark 4.2.2, that $\Delta \vdash_{\mathcal{A}_1} B$ *and* $\Delta \vdash_{\mathcal{A}_2} B$ *does not necessarily imply* $\Delta \vdash_{\mathcal{A}_1 \times \mathcal{A}_2} B$*.*

Example 4.2.4 (Resource for implication) *Consider the language with binary implication* $A \rightarrow B$*. Let* $|\!\sim$ *be a consequence relation for* $\rightarrow$ *satisfying identity and transitivity and assume that* $A \rightarrow B$ *is antimonotonic in* A *and monotonic in* B*. Let* $\mathcal{A}_0$ *be an algebra with binary* $\otimes$ *and a constant* $\mathbf{e}$*. Let the* (ρ, φ) *modus ponens rule be:*

$$\frac{(a, \alpha) : A; (b, \beta) : A \rightarrow B; \rho(b, a); \varphi(\beta, \alpha)}{(b * a, \beta \otimes \alpha) : B}$$

In many known substructural logics $\mathbf{e}$ *is a left unit satisfying* $\mathbf{e} \otimes \alpha = \alpha$ *for all* α *and that* $\rho(\varnothing, a)$ *and* $\varphi(\mathbf{e}, \alpha)$ *hold for all* a *and* α*. We shall see in (a) below that this property corresponds to adopting the axiom* $\vdash A \rightarrow A$*. Also the database formula* $\delta(x)$ *expresses in many substructural logics the notion of relevance and/or resource use. For example* $\delta(t)$ *could be the formula* $\forall x(x \in \mathit{Set}\ (t))$*. This formula holds in any Herbrand universe iff for all constants in the universe appear in* t*.*

For the above algebras and elimination rule, let us check what properties of $\mathcal{A}$ *are needed to get some theorems of the form* $\varnothing \vdash (\varnothing, \mathbf{e}) : A$*, for various wffs* A*.*

The reader should note that we do not claim completeness. A completeness theorem should be proved for each combination of axioms. To illustrate, consider the Hilbert system with the single axiom $\vdash A \rightarrow A$ *and the rule of Modus ponens*

$$\vdash A \textit{ and } \vdash A \rightarrow B \textit{ imply } \vdash B.$$

Let $\mathcal{A}$ *be an algebra with* $\mathbf{e}, \otimes, \leq$ *and* φ*. From case (a) below and from Example 4.2.8 we get that the corresponding conditions are:*

- ●● $\varphi(\mathbf{e}, \alpha) \Rightarrow \alpha \leq \mathbf{e} \otimes \alpha$
- ●● $\varphi(\mathbf{e}, \alpha) \Rightarrow \delta(\mathbf{e} \otimes \alpha)$
- ●● $\varphi(\mathbf{e}, \mathbf{e})$

•• $\mathbf{e} \otimes \mathbf{e} \leq \mathbf{e}$

If the algebra $\mathcal{A}$ satisfies these conditions, then the corresponding algebraic LDS *satisfies*

$$\bullet \varnothing \vdash_{\mathcal{A}} \mathbf{e} : A \rightarrow A$$

and

$$\varnothing \vdash_{\mathcal{A}} \mathbf{e} : A$$

and

$$\varnothing \vdash_{\mathcal{A}} \mathbf{e} : A \rightarrow B$$

imply

$$\varnothing \vdash_{\mathcal{A}} \mathbf{e} : B.$$

In other words we get soundness. What we still need to prove is completeness, namely if for some formula $D, \varnothing \not\vdash \mathbf{e} : D$ then for some algebra $\mathcal{A}_D$, satisfying hte above condition, we have that

$$\varnothing \not\vdash_{\mathcal{A}_D} D.$$

We can define a translation τ_δ, similar to the one given in definition 4.1.6(b), where the atoms $t : q$ is translated as $Q(t) \wedge \delta(t)$ and $\twoheadrightarrow$ is intuitionisitc implication. We believe (conjecture) that we can show

$$\varnothing \vdash \mathbf{e} : A \text{ iff } [\mathbf{e} : A]^{\tau_\delta}$$

is a theorem of intuitionisitc logic, at least for wffs A which are temselves intuitionistic theorems.

(a) *We check $\varnothing \vdash ?(\varnothing, \mathbf{e}) : A \rightarrow A$. The above holds if for arbitrary a in $\langle x \rangle \rho(\varnothing, x)$ and arbitrary α in $\langle x \rangle \varphi(\mathbf{e}, x)$ we have*

$$\{\rho(\varnothing, a); \varphi(\mathbf{e}, \alpha); (a, \alpha) : A\} \vdash (\varnothing * a, \mathbf{e} \otimes \alpha) : A$$

This will follow if $a \leq \varnothing * a$ and $\alpha \leq \mathbf{e} \otimes \alpha$ and $\{\rho(\varnothing, a), \varphi(\mathbf{e}, \alpha); (a, \alpha)\} \models \delta((\varnothing * a, \mathbf{e} \otimes \alpha)$.

We do have $a = \varnothing * a$, but we do need to require $\alpha \leq \mathbf{e} \otimes \alpha$, if we want $A \rightarrow A$ to be a theorem. If we assume also that $\mathbf{e} \otimes \alpha = \alpha$ that will also yield the axiom.

The minimal condition for $A \rightarrow A$ is

$$\bullet \bullet \forall \alpha(\varphi(\mathbf{e}, \alpha) \Rightarrow \alpha \leq \mathbf{e} \otimes \alpha)$$

$$\bullet \bullet \forall a(\rho(\varnothing, a) \Rightarrow a \leq \varnothing * a).$$

$$\bullet \bullet \{\rho(\varnothing, a); \varphi(\mathbf{e}, \alpha); (a, \alpha)\} \models \delta((\varnothing * a, \mathbf{e} \otimes \alpha)).$$

(b) *Consider the rule*

$$\frac{\vdash A \rightarrow B}{\vdash (B \rightarrow C) \rightarrow (A \rightarrow C)}.$$

To check this assume

$$\varnothing \vdash (\varnothing, \mathbf{e}) : A \rightarrow B$$

This means for arbitrary a and α in $\langle x \rangle \rho(\varnothing, x)$ and $\langle x \rangle \varphi(\mathbf{e}, x)$ respectively we have

$$\{\rho(\varnothing, a); \varphi(\mathbf{e}, \alpha); (a, \alpha) : A\} \vdash (\varnothing * a, \mathbf{e} \otimes \alpha) : B$$

while we need to show for arbitrary a, α as above that

$$\{\rho(\varnothing, a); \varphi(\mathbf{e}, \alpha); (a, \alpha) : B \rightarrow C\} \vdash (\varnothing * a, \mathbf{e} \otimes \alpha) : A \rightarrow C$$

For the latter we need for arbitrary b and β in $\langle x \rangle \rho(\varnothing * a, x)$ and in $\langle x \rangle \varnothing(\mathbf{e} \otimes \alpha, x)$ respectively to have

$$\{\varphi(\mathbf{e} \otimes \alpha, \beta); \rho(\varnothing * a, b); \rho(\varnothing, a); \varphi(\mathbf{e}, \alpha); (a, \alpha) : B \rightarrow C; (b, \beta) : A\} \vdash ((\varnothing * a) * b, (\mathbf{e} \otimes \alpha) \otimes \beta) : C.$$

To show that we need $\rho(\varnothing, b)$ and $\varphi(\mathbf{e}, \beta)$ to hold, so that we get

$$(\varnothing * b, \mathbf{e} \otimes \beta) : B.$$

Further, we need $\rho(a, \varnothing * b)$ and $\varphi(\alpha, \mathbf{e} \otimes \beta)$ to hold and $a * (\varnothing * b) \leq (\varnothing * a) * b$ and $\alpha \otimes (\mathbf{e} \otimes \beta) \leq (\mathbf{e} \otimes \alpha) \otimes \beta$.

Thus the condition is

- $\bullet\rho(\varnothing * a, b) \Rightarrow \rho(\varnothing, b) \wedge \rho(a, \varnothing * b) \wedge (a * (\varnothing * b) \leq (\varnothing * a) * b)$.
- $\bullet\varphi(\mathbf{e} \otimes \alpha, \beta) \Rightarrow \varphi(\mathbf{e}, \beta) \wedge \varphi(\alpha, \mathbf{e} \otimes \beta) \wedge (\alpha \otimes (\mathbf{e} \otimes \beta) \leq (\mathbf{e} \otimes \alpha) \otimes \beta)$.

From now on we assume the algebra satisfies for all a, α:

$$\varnothing * a = a; \mathbf{e} \otimes \alpha = \alpha; \varphi(\mathbf{e}, \alpha); \rho(\varnothing, a)$$

we also assume that $\delta(t)$ always holds. This will simplify the examples.

(c) *Let us check when*

$$\varnothing \vdash (\varnothing, \mathbf{e}) : B \rightarrow (A \rightarrow B).$$

The above holds if

$$\{(a, \alpha) : B\} \vdash ?(a, \mathbf{e} \otimes \alpha) : A \rightarrow B$$

where a is arbitrary for $\langle x \rangle \rho(\varnothing, x)$ and α is arbitrary for $\langle y \rangle \varphi(\mathbf{e}, y)$.

Let b be arbitrary for $\langle x \rangle \rho(a, x)$ and β be arbitrary for $\langle y \rangle \varphi(\alpha, y)$. We must check whether we have $\{\rho(a, b); \varphi(\alpha, \beta); (a, \alpha) : B; (b, \beta) : A\} \vdash ?(a * b, \alpha \otimes \beta) : B$.

Clearly if our algebra allows for $a \leq a * b$ and $\alpha \leq \alpha \otimes \beta$ then we can indeed succeed.

The above means that the condition we need on the algebra is

$$\bullet\bullet\, \rho(a, x) \wedge \varphi(\alpha, y) \Rightarrow a \leq a * x \wedge \alpha \leq \alpha \otimes y.$$

(d) *Let us check the axiom*

$$\varnothing \vdash ?(\varnothing, \mathbf{e}) : A \rightarrow ((A \rightarrow B) \rightarrow B).$$

Again this reduces to checking

$$\{(a, \alpha) : A\} \vdash ?(a, \alpha) : (A \rightarrow B) \rightarrow B.$$

which reduces to showing for any (b, β) such that $\rho(a, b)$ and $\varphi(\alpha, \beta)$ hold that:

$$\{\rho(a, b); \varphi(\alpha, \beta); (a, \alpha) : A; (b, \beta) : A \rightarrow B\} \vdash ?(a * b, \alpha \otimes \beta) : B$$

where further b is arbitrary in $\langle x \rangle \rho(a, x)$ (i.e. $b \neq a$) and β is arbitraray in $\langle y \rangle \varphi(\alpha, y)$. The only way to get the conclusion is to perform modus ponens on the assumptions, in which case we get $(b * a, \beta \otimes \alpha) : B$ but we need $\varphi(\beta, \alpha)$ to hold.

We thus need the algebra to satisfy the following conditions:

$$\bullet\bullet\, \rho(a, x) \Rightarrow \rho(x, a) \wedge (x * a \leq a * x)$$

$$\bullet\bullet\, \varphi(\alpha, y) \Rightarrow \varphi(y, \alpha) \wedge (y \otimes \alpha \leq \alpha \otimes y)$$

(e) *Let us check the axiom*

$$\varnothing \vdash ?(\varnothing, \mathbf{e}) : (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C))$$

Again we get that we need

$$\{(a, \alpha) : A \rightarrow (B \rightarrow C)\} \vdash ?(a, \alpha) : B \rightarrow (A \rightarrow C)$$

and we need to show that for any (b,β) such that $\rho(a,b)$ and $\varphi(\alpha,\beta)$ hold and such that b is arbitrary in $\langle x\rangle\rho(a,x)$ and β is arbitrary in $\langle y\rangle\varphi(\alpha,y)$ we have

$$\{\rho(a,b);\varphi(\alpha,\beta);(a,\alpha):A\to(B\to C);(b,\beta):B\}f\vdash ?(a*b,\alpha\otimes\beta):A\to C$$

We must show for any (c,γ) such that $\rho(a*b,c)$ and $\varphi(\alpha\otimes\beta,\gamma)$, which are arbitrary respectively, we have

$$\begin{array}{l}\{\rho(a,b);\varphi(\alpha,\beta);\rho(a*b,c);\varphi(\alpha\otimes\beta,\gamma);(a,\alpha):A\to(B\to C);(b,\beta):B;(c,\gamma):A\}\\ \vdash((a*b)*c,(\alpha\otimes\beta)\otimes\gamma):C\end{array}$$

using modus ponens we can get: $((a*c)*b,(\alpha\otimes\beta)\otimes\gamma):C$ provided $\varphi(\alpha,\gamma)$, $\rho(a,c),\varphi(\alpha\otimes\gamma,\beta)$ and $\rho(a*c,b)$ all hold.

We thus need the following to hold

$$\bullet\bullet\ \rho(a,x_1)\wedge\rho(a*x_1,x_2)\Rightarrow\rho(a,x_2)\wedge\rho((a*x_2,x_1)\wedge((a*x_2)*x_1\leq(a*x_1)*x_2)$$

$$\bullet\bullet\ \varphi(\alpha,y_1)\wedge\varphi(\alpha\otimes y_1,y_2)\Rightarrow\varphi(\alpha,y_2)\wedge\varphi((\alpha\otimes y_2,y_1)\wedge((\alpha\otimes y_2)\otimes y_1\leq(\alpha\otimes y_1)\otimes y_2)$$

(f) *Note that if we let $a=\varnothing$ and $\alpha=\mathbf{e}$ in the conditions of (d) above, then if we assume $\rho(\varnothing,x)$ and $\varphi(\mathbf{e},y)$ hold for any x and y we get the following:*

$$\rho(x_1,x_2)\Rightarrow(x_2*x_1\leq x_1*x_2)\wedge\rho(x_2,x_1)$$

$$\varphi(y_1,y_2)\Rightarrow(y_2\otimes y_1\leq y_1\otimes y_2)\wedge\varphi(y_2,y_1)$$

which are exactly the conditions of (c) above. Indeed axioms (a) and (d) imply axiom (c) because from

$$(A\to B)\to(A\to B)$$

we can get by commutativity

$$A\to((A\to B)\to B).$$

(g) *We check the axiom*

$$\varnothing\vdash ?(\varnothing,\mathbf{e}):(A\to(B\to C))\to((A\to B)\to(A\to C))$$

We have to check whether

$$\{(a,\alpha):A\to(B\to C)\}\vdash ?(a,\alpha):(A\to B)\to(A\to C).$$

For this purpose we need to show that for any (b,β) such that $\rho(a,b)$ and $\varphi(\alpha,\beta)$ hold and such that b is arbitrary in $\langle x\rangle\rho(a,x)$ and β is arbitrary in $\langle y\rangle\varphi(\alpha,y)$ we have

$$\{\rho(a,b);\varphi(\alpha,\beta);(a,\alpha):A\to(B\to C);(b,\beta):A\to B\}\vdash ?(a*b,\alpha\otimes\beta):A\to C.$$

We must therefore show that for any (c,γ) such that $\rho(a*b,c),\varphi(\alpha\otimes\beta,\gamma),c$ is arbitraray in $\langle x\rangle\rho(a*b,x),\gamma$ arbitrary in in $\langle y\rangle\varphi(\alpha\otimes\beta,y)$, we have

$$\begin{array}{l}\{\rho(a,b);\varphi(\alpha,\beta);\rho(a*b,c);\varphi(\alpha\otimes\beta,\gamma);(a,\alpha):A\to(B\to C);\\(b,\beta):A\to B;(c,\gamma):A\}\vdash ?((a*b)*c,(\alpha\otimes\beta)\otimes\gamma):C\end{array}$$

To do modus ponens we need the following properties to hold

- $\rho(a,c)\wedge\varphi(\alpha,\gamma)$
- $\rho(b,c)\wedge\varphi(\beta,\gamma)$

These two will respectively give us

$$(a * c, \alpha \otimes \gamma) : B \to C$$

and

$$(b * c, \beta \otimes \gamma) : B$$

We further need

$$\rho(a * c, b * c) \wedge \varphi(\alpha \otimes \gamma, \beta \otimes \gamma)$$

which will allow us to get by modus ponens

$$((a * c) * (b * c), (\alpha \otimes \gamma) \otimes (\beta \otimes \gamma)) : C$$

we therefore need to have that

- $(a * c) * (b * c) \leq (a * b) * c$
- $(\alpha \otimes \gamma) \otimes (\beta \otimes \gamma) \leq (\alpha \otimes \beta) \otimes \gamma$

The Conditions on the algebra we require are therefore the following for all a, b, c and α, β, γ.

- $\bullet \rho(a, b) \wedge \rho(a * b, c) \Rightarrow \rho(a, c) \wedge \rho(b, c) \wedge \rho(a * c, b * c) \wedge ((a * c) * (b * c) \leq (a * b) * c)$.
- $\bullet \varphi(\alpha, \beta) \wedge \varphi(\alpha \otimes \beta, \gamma) \Rightarrow \varphi(\alpha, \gamma) \wedge \varphi(\beta, \gamma) \wedge \varphi(\alpha \otimes \gamma, \beta \otimes \gamma) \wedge ((\alpha \otimes \gamma) \otimes \beta \otimes \gamma \leq (\alpha \otimes \beta) \otimes \gamma)$.

(h) *We check the axiom*

$$\varnothing \vdash ?(\varnothing, \mathbf{e}) : (A \to B) \to (A \to (A \to B))$$

As before, we need to show

$$\{(a, \alpha) : A \to B\} \vdash ?(a, \alpha) : A \to (A \to B)$$

For this purpose we need to show, as before, that for any (b, β) such that $\rho(a, b)$ and $\varphi(\alpha, \beta)$ hold and b is arbitrary in $\langle x \rangle \rho(a, x)$ and β is arbitrary in $\langle y \rangle \varphi(\alpha, y)$ we have

$$\{\rho(a, b); \varphi(\alpha, \beta); (a, \alpha) : A \to B; (b, \beta) : A\} \vdash ?(a * b, \alpha \otimes \beta) : A \to B$$

This can be shown if

- $a \leq a * b$ *and* $\alpha \leq \alpha \otimes \beta$.

We an alternatively try and show the consequent using the elimination rule, in which case we need to show that for an arbitrary (c, γ) such that $\rho(a * b, c) \wedge \varphi(\alpha \otimes \beta, \gamma)$ holds and c is arbitrary in $\langle x \rangle \rho(a * b, x)$ and γ is arbitrary in $\langle y \rangle \varphi(\alpha \otimes \beta, y)$ we have

$$\{\rho(a, b); \varphi(\alpha, \beta); \rho(a{*}b, c); \varphi(\alpha{\otimes}\beta, \gamma); (a, \alpha) : A \to B; (b, \beta) : A; (c, \gamma) : A\} \vdash ?((a{*}b){*}c, (\alpha{\otimes}\beta){\otimes}\gamma) : B$$

there are two ways we can proceed:
since the algebra allows for

- $\rho(a, b) \wedge \varphi(\alpha, \beta)$

then we can perform the modus ponens and get

$$(a * b, \alpha \otimes \beta) : B$$

and we need further that

- $a * b \leq (a * b) * c$ *and* $\alpha \otimes \beta) \leq (\alpha \otimes \beta) \otimes \gamma$

On the other hand, if the algebra allows for

- $\rho(a, c) \wedge \varphi(\alpha, \gamma)$

then we can perform modus ponens and get

- $(a * c, \alpha \otimes \gamma) : B$

We further need that

- $a * c \leq (a * b) * c$ *and* $\alpha \otimes \gamma \leq (\alpha \otimes \beta) \otimes \gamma$

Thus the overall conclusion is that the above axiom can be proved from the empty database $\varnothing$ *if the algebra itself satisfies the following condition for all* a, b, c *and* α, β, γ.

$$\begin{array}{ll} \bullet\bullet & \{\rho(a,b) \wedge \varphi(\alpha,\beta) \Rightarrow (a \leq a * b) \wedge (\alpha \leq \alpha \otimes \beta)\} \\ & \vee\{[\rho(a,b) \wedge \rho(a * b, c) \Rightarrow (a * b \leq (a * b) * c]\wedge \\ & \wedge[\varphi(\alpha,\beta) \wedge \varphi(\alpha \otimes \beta,\gamma) \Rightarrow (\alpha \otimes \beta \leq (\alpha \otimes \beta) \otimes \gamma)]\} \\ & \vee\{[\rho(a,b) \wedge \rho(a * b, c) \Rightarrow \rho(a,c) \wedge (a * c \leq (a * b) * c)] \\ & \wedge[\varphi(\alpha,\beta) \wedge \varphi(\alpha \otimes \beta,\gamma) \Rightarrow \varphi(\alpha,\gamma) \wedge (\alpha \otimes \gamma \leq (\alpha \otimes \beta) \otimes \gamma)]\} \end{array}$$

(i) *Let us check*

$$\varnothing \vdash ?(\varnothing, \mathbf{e}) : (A \to B) \to ((C \to A) \to (C \to B))$$

As before, we need an arbitrary a *in* $\langle x \rangle \rho(\varnothing, x)$ *and an arbitrary* α *in* $\langle x \rangle \varphi(\mathbf{e}, x)$ *for which we have*

$$\{(a,\alpha) : A \to B\} \vdash ?(a,\alpha) : (C \to A) \to (C \to B).$$

This reduces to allowing any (b,β) *such that* b *is arbitrary in* $\langle x \rangle \rho(a,x)$ *and* β *is arbitrary in* $\langle x \rangle \varphi(\alpha,x)$ *to have*

$$\{\rho(a,b); \varphi(\alpha,\beta); (a,\alpha) : A \to B; (b,\beta) : C \to A\} \vdash ?(a * b, \alpha \otimes \beta) : C \to B.$$

We must now show that for any (c,γ) *such that* c *is arbitrary in* $\langle x \rangle \rho(a * b, x)$ *and* γ *is arbitrary in* $\langle x \rangle \varphi(\alpha \otimes \beta, x)$ *we have*

$$\{\rho(a*b,c); \varphi(\alpha\otimes\beta,\gamma); \rho(a,b); \varphi(\alpha,\beta); (a,\alpha) : A \to B; (b,\beta) : C \to A; (c,\gamma) : C\} \vdash ?((a*b)*c, (\alpha\otimes\beta)\otimes\gamma) : B.$$

To succeed in this we need $\rho(c,b) \wedge \varphi(\gamma,\beta)$ *to hold and* $\rho(b * c, a) \wedge \varphi(\beta \otimes \gamma, \alpha)$ *to hold. we further need*

$$a * (b * c) \leq (a * b) * c$$

and

$$\alpha \otimes (\beta \otimes \gamma) \leq (\alpha \otimes \beta) \otimes \gamma.$$

Thus the two conditions we need are:

- $\bullet\rho(a,b) \wedge \rho(a * b, c) \Rightarrow \rho(c,b) \wedge \rho(b * c, a) \wedge (a * (b * c) \leq (a * b) * c)$
- $\bullet\varphi(a,\beta) \wedge \varphi(\alpha \otimes \beta, \gamma) \Rightarrow \varphi(\gamma,\beta) \wedge \varphi(\beta \otimes \gamma, \alpha) \wedge (\alpha \otimes (\beta \otimes \gamma) \leq (\alpha \otimes \beta) \otimes \gamma.$

Example 4.2.5 (Semilinear, many valued algebra) *Consider the albegras* $\mathcal{F}$ *and* $\mathcal{A}_0$ *defined as follows:*

1. $\rho(a,b) =$ *def (set* $(a) \cap$ *set* $(b) = \varnothing$*), where set* $(a) \cap$ *set* $(b) = \varnothing$ *means that the expressions* a *and* b *share no atoms in common.*

2. *Let* $a \leq b$ *be multiset*$(a) \subseteq$ *multiset*(b).

3. *Let the algebra* $\mathcal{A}_0$ *be all real numbers in the interval [0,1]. Let* $\mathbf{e}$ *be 0 and let* $\beta \otimes \alpha =$ *min*$(1, \alpha + \beta)$. *Clearly* $\otimes$ *is commutative and* $\mathbf{e} \otimes \alpha = \alpha$.

Let $\alpha \leq \beta$ mean the same as $\leq$ of the real numbers.

Let $\varphi(\alpha, \beta)$ be $\top$.

to show in $\mathcal{A}_0$, $\beta : A \to B$, we need to asusme $\alpha : A$ for an arbitrary α and show $\beta \otimes \alpha : B$.

Thus we have the truth table TruthValue$(A \to B) =$ *max* $(0, \text{TruthValue } (B) - \text{TruthValue } (A))$ *with 0 being* Truth.

Let us check whether $A \to (B \to A)$ is a theorem of this logic. The condition to check is

$$\bullet \bullet\, \rho(a, x) \wedge \varphi(\alpha, y) \to a \leq a * x \wedge \alpha \leq \alpha \otimes y.$$

which becomes

$$\text{set } (a) \cap \text{ set } (b) = \varnothing \Rightarrow \text{ multiset } (a) \subseteq \text{ multiset } (a * x) \wedge \alpha \leq \textit{ min } (1, \alpha + y)$$

This condition is satisfied.

Example 4.2.6 (An algebraic condition for an axiom) *The next definition is motivated by Example 4.2.4 and is intended to associate a formula τ_A of the algebra $\mathcal{A}$ with every wff A of a logic* **L** *such that in the* LDS $(\mathcal{A}, \mathbf{L})$ *we* hope to have *that (under suitable conditions on A):*

$$\varnothing \vdash \mathbf{e} : A \textit{ iff } \mathcal{A} \vDash \tau_A.$$

We explain the basic idea by an example. Consider the algebra $\mathcal{A} = \mathcal{F} \times \mathcal{A}_0$ and the $\mathcal{A}_0$ database Δ_0.

$$\begin{array}{l} \alpha_1 : B \\ \alpha_2 : A \to B \\ \alpha_3 : A \end{array}$$

and we want to prove in $\mathcal{A}_0$, $\beta : A \to (A \to B)$ from this database. We assume the algebra $\mathcal{A}_0$ has the modus ponens rule:

$$\frac{\varphi(\alpha, \beta); \alpha : A; \beta : A \to B}{f(\beta, \alpha) : B}$$

We proceed with a proof in $\mathcal{A}_0$ and assume $x_1 : A$ with an arbitrary label x_1 in $\langle x \rangle \varphi(\beta, x)$ and try and prove $f(\beta, x_1) : A \to B$.

This can be done in several ways. To record the different ways we ue resource labelling by working in the algebra $\mathcal{A} = \mathcal{F} \times \mathcal{A}_0$. So let us display our database as an $\mathcal{A}$ database Δ_1 below:

$$\begin{array}{l} (a_1, \alpha_1) : B \\ (a_2, \alpha_2) : A \to B \\ (a_3, \alpha_3) : A \\ (c_1, x_1) : A \end{array}$$

where a_1, a_2, a_3, c_1 are atomic resource labels from $\mathcal{F}$. In the pure resource logic with $\mathcal{F}$ labels, ignoring the algebraic $\mathcal{A}_0$ labels, we can prove $A \to B$ in two ways.

1. *Using the Elimination Rules only. In which case we get $a_2 : A \to B$*
2. *Use an introduction rule for $A \to B$.*

Option 1 will also succeed in the $\mathcal{A}_0$ algebra if $\alpha_2 \leq f(\beta, x_1)$ in $\mathcal{A}_0$.

Let us write $t = a_2$ for the resource label with which $A \to B$ can be proved and let

$$\Pi_1 = [\varphi(\beta, x_1) \Rightarrow (\alpha_1 \leq f(\beta, x_1))]$$

for the associated formula, which if it holds in $\mathcal{A}_0$ the corresponding $\mathcal{A}_0$ proof will succeed.

Thus we look at t of $\mathcal{F}$ such that $\Delta_1 \vdash_{\mathcal{F}} t : A \to B$ and check whether $\mathcal{A}_0 \vDash \Pi_t$ and this ensures that $\Delta_1 \vdash_{\mathcal{A}_0} \beta : B$.

Turning to the second option in $\mathcal{A}_0$, we assume $x_2 : A$ with a label x_2 arbitrary in $\langle x\rangle\varphi(f(\beta, x_1), x)$ and get the $\mathcal{A}$ database Δ_2 below:

$$\begin{array}{l}(a_1, \alpha_1) : B \\ (a_2, \alpha_2) : A \to B \\ (a_3, \alpha_3) : A \\ (c_1, x_1) : A \\ (c_2, x_2) : A\end{array}$$

and we want to prove in $\mathcal{A}_0$ $\gamma = f(f(\beta, x_1), x_2) : B$. On pure resource considerations we can prove b in $\mathcal{F}$ in several different ways:

$$\begin{array}{l}a_1 : B \\ a_2a_3 : B \\ a_2c_1 : B \\ a_2c_2 : B\end{array}$$

The corresponding $\mathcal{A}_0$ labels:

$$\begin{array}{l}\alpha_1 : B \\ f(\alpha_2, \alpha_3) : B \\ f(\alpha_2, x_1) : B \\ f(\alpha_3, x_2) : B\end{array}$$

Therefore we can deduce in $\mathcal{A}_0$ the required $\gamma = f(f(\beta, x_1), x_2) : B$ provided the following holds:

$$\Pi_2 = \varphi(f(\beta, x_1), x_2) \Rightarrow \alpha_1 \leq \gamma \vee (f(\alpha_2, \alpha_3) \leq \gamma) \vee (f(\alpha_2, x_1) \leq \gamma) \vee (f(\alpha_3, x_2) \leq \gamma).$$

We therefore conclude that in $\mathcal{A}_0$ the database Δ_0 can prove $\beta : B$ if the algebra satisfies

$$\forall x_1, x_2(\Pi_1 \vee \Pi_2)$$

The general situation, however, is not as simple as it seems from the above discussion. Let us look at the formula $(((A \to A) \to B) \to B$. To find the condition needed we try

$$(a, \alpha) : (A \to A) \to B \vdash ?(a, \alpha) : B$$

We cannot get B by direct modus ponens (ignoring the labels). We need to observe that we should try and prove $A \to A$. So we try to show $x : A \to A$ with a label x such that $f(\alpha, x) \leq \alpha$ and $\varphi(\alpha, x)$ holds. Clearly, to show $x : A \to A$, we assume $y : A$, and try and show $f(x, y) : A$. The simplest is to assume $y \leq f(x, y)$. We are thus looking for an x such that $\forall y(\varphi(x, y) \Rightarrow y \leq f(x, y))$ and such that $\varphi(\alpha, x)$ holds.

The choice $x = \mathbf{e}$ can do that, except that $\varphi(\alpha, \mathbf{e})$ might not hold. In fact the condition for $((A \to A) \to B) \to B$ to be a theorem is indeed that $\varphi(\alpha, \mathbf{e})$ holds.

To summarize, for $\mathbf{e} : ((A \to A) \to B) \to B$ to hold, we need:

$$\bullet\bullet\ \forall\alpha[\varphi(\alpha, \mathbf{e}) \wedge f(\alpha, \mathbf{e}) \leq \alpha]$$

The lesson we learn from this example si that there may be ways to prove the goals which do not arise from direct modus ponens pattern matching. We may have: $t_1 : A \to B, ; t_2, : C \vdash^? t : B$, and we may need to try and show $x : A$ with an appropriate label x. Thus to find all possible proofs algorithmically we need a good proof strategy. Note that the definition of the consequence relation $\Delta \vdash t : A$ (Definition 4.2.3) is correct but it is inductive. For example, we can get

$$\alpha : (A \to A) \to B \vdash \alpha : B$$

because $\alpha : (A \to A) \to B \vdash_{1,0} \mathbf{e} : A \to A$ and hence it proves $\vdash_{1,1} f(\alpha, \mathbf{e}) : B$ and provided $f(\alpha, \mathbf{e}) \leq \alpha$, we get the desired result.

We thus need a good proof strategy. Let us adopt a goal directed strategy. The strategy of N-Prolog, which is as follows:

1. *To show $?q, q$ atomic, find a clause $A_1 \rightarrow \ldots \rightarrow (A_n \rightarrow q) \ldots)$ in the data and show $?A_i, i = 1, \ldots, n$.*
2. *To show $A \rightarrow B$, add A to the data and show B.*
3. *Success for q atomic if q is in the data.*

This strategy is an improvement. It will certainly deal with the previous examples, but still, there are cases where it fails. Consider

$$((A \rightarrow B) \rightarrow B) \rightarrow ((B \rightarrow A) \rightarrow A$$

To find the condition for this we try

$$(a, \alpha) : (A \rightarrow B) \rightarrow B \vdash ?(a, \alpha) : (B \rightarrow A) \rightarrow A$$

which reduces to

$$(a, \alpha) : (A \rightarrow B) \rightarrow B; (b, \beta) : B \rightarrow A \vdash ?(a * b, f(\alpha, \beta)) : A$$

and $\varphi(\alpha, \beta)$ must hold.

Now we can again see the improvement of the goal directed approach. Forward pattern matching for modus ponens will give us nothing. The goal directed approach will indicate that we need to prove B with a label $x : B$ such that $\varphi(\beta, x)$ holds and

$$f(\beta, x) \leq f(\alpha, \beta).$$

Again, in a goal directed procedure we get that we need to prove $y : A \rightarrow B$ with a label y such that $\varphi(\alpha, y)$ holds and $f(\alpha, y) \leq x$. To show $y : A \rightarrow B$, we assume $z : A$ for arbitrary z in $\langle z \rangle \varphi(y, z)$ and show $f(y, z) : B$. Here we loop because we have to go through the clause $(A \rightarrow B) \rightarrow B$ again.

Thus we get no condition for this case unless we improve our proof procedures.

The N-Prolog goal directed proof procedure is complete for intuitionistic implication. Additional rules can be given to caputre classical logic and some intermediate logics. The above axiom is not an intuitionistic theorem. Perhaps then as a beginning step we restrict our conditions finding algorithms to axioms which are intuitionistic theorems, and hope to obtain completeness proofs for such cases.

Let us agree that by a substructural Hilbert system for $\rightarrow$ *we mean a Hilbert system whose theorems are all intuitionistically valid. It is for this kind of system that we give the next definition.*

Definition 4.2.7 (Algebraic conditions for substructural axioms) *Let $(\mathcal{A}_0, \mathbf{L})$ be an* LDS. *Consider the resource algebra $\mathcal{A} = \mathcal{F} \times \mathcal{A}_0$. Assume $(\mathcal{A}, \mathbf{L})$ is an algebra with rules of the sort described in Definition 4.2.1. Let Δ be a database in $\mathcal{A}$. Assume that if $(a_1, \alpha_1) : A; (a_2, \alpha_2) : A \in \Delta$ and $\mathcal{A}_0 \nvdash \alpha_1 = \alpha_2$ then $a_1 \neq a_2$. Let $\Delta_{\mathcal{F}}$ be the $\mathcal{F}$ database $\{a : A \mid$ for some $\alpha, (a, \alpha) : A \in \Delta\}$. Similarly $\Delta_{\mathcal{A}_0} = \{\alpha : A \mid$ for some $a, (a, \alpha) : A \in \Delta\}$.*

The notions $\Delta_{\mathcal{F}} \vdash a : A$ and $\Delta_{\mathcal{A}_0} \vdash \alpha : A$ are well defined. We shall abuse notation and write $\Delta \vdash_{\mathcal{F}} a : A$ and $\Delta \vdash_{\mathcal{A}_0} \alpha : A$ instead. We now define the formula $\tau(\Delta, \beta, B)$ of the language $\mathcal{A}_0$, which is the condition which $\mathcal{A}_0$ has to satisfy so that $\Delta \vdash_{\mathcal{A}_0} \beta : B$. We begin by defining a formula $\tau(\Delta, t, \beta, B)$, for any t of $\mathcal{F}$ for which $\Delta \vdash_{\mathcal{F}} t : B$. The definition is by induction on the proof of $t : B$ as defined in Definition 4.2.3.

Case $m = n = 0$
In this case for some $\alpha, (t, \alpha) : B$ is in Δ. We can deduce $\Delta \vdash \beta : B$ provided $\alpha \leq \beta$ holds. So in this case let $\tau(\delta, t, \beta, B)$ be $(\delta_D \Rightarrow \delta(t)) \wedge \alpha \leq \beta$. If $\delta(t)$ is not used, i.e. is always true then the formula is $\alpha \leq \beta$.

Cases $m, n + 1$
In this case $t : B$ is obtained from previously proved formulas by an elimination rule, as in Definition 4.2.3. Say the rule is rule e below:

$$\frac{\rho(a, a_1, \ldots, a - n); a_1 : A_1; \ldots, a_n : A_n; a : A}{t = ((e, a), (a_, \ldots, a_n) : B}$$

Let $r_1, \ldots, r_k$ be companion rules for e above, of the form:

$$r_i \quad \frac{\varphi_i(\alpha^i, \alpha_1^i, \ldots, \alpha_n^i); \alpha_1^i : A_1; \ldots ; \alpha_n^i : A_n; \alpha^i : A}{f^i(\alpha^i, \alpha_1^i, \ldots, \alpha_n^i) : B}$$

By the induction hypothesis there are formulas $\tau(\Delta, a, \alpha^i, A)$ and $\tau(\Delta, a_j, \alpha_j^i, A_j)$ such that if the algebra $\mathcal{A}_0$ satisfies them then $\Delta \vdash_{\mathcal{A}_0} \alpha^i : A, \Delta \vdash_{\mathcal{A}_0} \alpha_j^i : A_j$ respectively hold. Therefore the formula $\tau(\Delta, t, \beta, B)$ is :

$$\bigvee_i [\tau(\Delta, a, \alpha^i, A) \wedge \bigwedge_j \tau(\Delta, a_j^i, \alpha_j^i, A_i) \Rightarrow (f^i(\alpha^i, \alpha_1^i, \ldots, \alpha_n^i) \leq \beta)]$$

Case $m + 1, 0$
In this case $t : B$ is obtained by an introduction rule (denoted by e). Thus B has the form $B = (A_1, \ldots, A_{r^-}) \to (B_1, \ldots, B_{r^+})$.

We have for arbitrary new $\mathcal{F}$ labels $a_1, \ldots, a_{r^-}$ satisfying $\rho(t, a_1, \ldots, a_{r^-})$, that for each $j = 1, \ldots, r^+$

$$\Delta'_{\mathcal{F}} = \Delta_{\mathcal{F}} \cup \{a_1 : A_1; \ldots, a_{r^-} : A_{r^-}\} \vdash ((e_j, t), (a_1, \ldots, a_{r^-}) : B_j$$

Let $r_1, \ldots, r_k$ be all elimination rules for $\to$ in $\mathcal{A}_0$. These have the form:

$$r_i \quad \frac{\varphi_i(\alpha^i, \alpha_1^i, \ldots, \alpha_{r^-}^i); \alpha_1^i : A_1; \ldots ; \alpha_{r^-}^i : A_{r^-}; t : B}{f_j^i(\alpha^i, \alpha_1^i, \ldots, \alpha_{r^-}^i) : B_j}$$

By the induction hypothesis, there exist formulas $\tau_i = \tau(\Delta_i, ((e_j, t), (a_1, \ldots, a_{r^-})), f_j^i(\alpha^j, \alpha_1^i, \ldots, \alpha_{r^-}^i), B_j)$ giving the conditions under which

$$\Delta_i = \Delta \cup \{\alpha_1^i : A_1; \ldots ; \alpha_{r^-}^i : A_{r^-}\} \vdash_{\mathcal{A}_0} f_j^i(\alpha^i, \alpha_1^i, \ldots, \alpha_{r^-}^i) : B_j.$$

Thus $\tau(\Delta, t, \beta, B)$ is the following:

$$\bigwedge_i [\forall \alpha^i \forall \alpha_1^i, \ldots, \alpha_{r^-}^i (\varphi_i(\alpha^i, \alpha_1^i, \ldots, \alpha_{i-1}^i) \wedge \tau_i \Rightarrow (\alpha_i \leq \beta)]$$

Case of deduction from $s : A; s \leq t$ **deduce** $t : A$
In this case, let $\tau(\Delta, t, \beta, B)$ be $\tau(\Delta, s, \beta, B)$.

The above inductive definition defined formula $\tau = \tau(\Delta, t, \beta, B)$ for every t such that $\Delta \vdash_{\mathcal{F}} t : B$. The formula says that if $\mathcal{A}_0 \vDash \tau$ then $\Delta \vdash \beta : B$. In other words the fact that $\Delta \vdash_{\mathcal{F}} t : B$ holds means that there is a resource proof of B from the database Δ and t is the term encoding the proof steps. Whether a corresponding $\mathcal{A}_0$ label propagation is possible depnds on the algebra $\mathcal{A}_0$. If $\mathcal{A}_0 \vDash \tau$ then this is possible. To find whether $\Delta \vdash_{\mathcal{A}_0} ?\beta : B$ can succeed we must find all possible resource proof routes t such that $\Delta \vdash_{\mathcal{F}} t : B$ and for each consider $\tau(\Delta, t, \beta, B)$. The disjunction of all possible such τ's will give us the minimal condition on $\mathcal{A}_0$ to enable $\Delta \vdash_{\mathcal{A}_0} \beta : B$.

If the number of such t's is finite then we are finished. What if the number of t's is infinite? For example $\{a_1 : A; a_2 : A \to A; a_3 : A \to B\}$ can prove B with an infinite number of labels of the form $t_m = a_2^m a_1 a_3$. For this reason we assumed that the algebra $\mathcal{A}_0$ is archimedian. The 'longer' the term t, the more $\mathcal{A}_0$ function symbols are applied to the initial labels of Δ to get the final label of B. Thus after a large number of applications of $\mathcal{A}_0$ function symbols, the resulting label α will not be less than $\beta(\leq \beta)$. So the number of ts to consider is finite.

Example 4.2.8 (Labels for intuitionistic implication) *Let us check intuitionistic implication. The axioms are*

$$\vdash A \to (B \to A)$$
$$\vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C))$$

and the rule

$$\frac{\vdash A; \vdash A \to B}{\vdash B}$$

The conditions on labels corresponding to the first two axioms were examined in example 4.2.4. These are respectively

$$\begin{array}{l} \forall \alpha, y(\varphi(\alpha, y) \Rightarrow \alpha \leq \alpha \otimes y) \\ \forall \alpha\beta\gamma(\varphi(\alpha, \beta) \wedge \varphi(\alpha \otimes \beta, \gamma) \Rightarrow \varphi(\alpha, \gamma) \wedge \varphi(\beta, \gamma) \wedge \\ \quad \wedge \varphi(\alpha \otimes \gamma, \beta \otimes \gamma) \wedge ((\alpha \otimes \gamma) \otimes (\beta \otimes \gamma) \leq (\alpha \otimes \beta) \otimes \gamma)) \end{array}$$

The condition for modus ponens is

Assume $\vdash A, \vdash A \to B$. *This means that* $\varnothing \vdash \mathbf{e} \vdash \mathbf{e} : A$ *and* $\forall \alpha$ *arbitrary in* $\langle x \rangle \varphi(\mathbf{e}, x)(\alpha : A \vdash \mathbf{e} \otimes \alpha : B)$ *hold. The above should imply* $\varnothing \vdash \mathbf{e} : B$. *If* $\mathbf{e}$ *is in* $\langle x \rangle \varphi(\mathbf{e}, x)$ *then we can substitute for* α *and get* $\vdash \mathbf{e} \otimes \mathbf{e} : B$. *Thus the condition is*

$$\bullet\bullet\ \varphi(\mathbf{e}, \mathbf{e}) \wedge \mathbf{e} \otimes \mathbf{e} \leq \mathbf{e}.$$

Example 4.2.9 (Labels for concatenation logic) *This system has the axioms*

1. $A \to A$
2. $(A \to B) \to ((C \to A) \to (C \to B)$
   and the rules
3. $\dfrac{\vdash A; \vdash A \to B}{\vdash B}$
   and
4. $\dfrac{\vdash A \to B}{\vdash (B \to C) \to (A \to C)}$

From Examples 4.2.4 and 4.2.8 we get the following conditions:

1. $\varphi(\mathbf{e}, \alpha) \Rightarrow (\alpha \leq \mathbf{e} \otimes \alpha)$
2. $\varphi(\mathbf{e}, \alpha) \wedge \varphi(\mathbf{e} \otimes \alpha, \beta) \wedge \varphi((\mathbf{e} \otimes \alpha) \otimes \beta, \gamma) \Rightarrow \varphi(\gamma, \beta) \wedge \varphi(\beta \otimes \gamma, \alpha) \wedge (\alpha \otimes (\beta \otimes \gamma \leq (\alpha \otimes \beta) \otimes \gamma)$
3. $\varphi(\mathbf{e}, \mathbf{e}) \wedge \mathbf{e} \otimes \mathbf{e} \leq \mathbf{e}$
4. $\varphi(\mathbf{e} \otimes \alpha, \beta) \Rightarrow \varphi(\mathbf{e}, \beta) \wedge \varphi(\alpha, \mathbf{e} \otimes \beta) \wedge (\alpha \otimes (\mathbf{e} \otimes \beta) \leq (\mathbf{e} \otimes \alpha) \otimes \beta)$.

Definition 4.2.10 (Labelled Hilbert systems)

1. *Let* $(\mathcal{A}, \mathbf{L})$ *be a resource algebraic labelled deductive system as defined in 4.2.1 and let* $\vdash$ *be its labelled consequence relation. Let* $\mathbf{H} = \mathbf{H}(\mathcal{A}, \mathbf{L})$ *be the set of theorems of* $(\mathcal{A}, \mathbf{L})$, *i.e.* $\mathbf{H} = \{A \mid \varnothing \vdash \mathbf{e} : A\}$.
2. *Let* $(\mathcal{A}, \mathbf{L})$ *be a resource algebraic* LDS, *as before, and let* $\mathbf{H}_0$ *be a set of wffs of* $\mathbf{L}$. *Define a new consequence relation* $\Delta \vdash_{\mathbf{H}_0} t : A$ *by letting*

   $$\Delta \vdash_{\mathbf{H}_0} t : A \textit{ iff def } \Delta \cup \{\mathbf{e} : A \mid A \in \mathbf{H}_0\} \vdash t : A$$

   i.e. we prove $t : A$ *by allowing us to use* $\mathbf{e} : A, A \in \mathbf{H}_0$ *at any time.*
3. *We write* $\Delta \vdash^E t : A$ *if there is a proof of* $t : A$ *from* Δ *using elimination rules only. In terms of definition 4.2.3 this means that* $\Delta \vdash_{0,n} t : A$ *for some* n.
4. *A set* $\mathbf{H}_0$ *of wff is said to be a* Hilbert formulation *of* $(\mathcal{A}, \mathbf{L})$ *if the following holds for all* Δ *and* $t : A$:

   $$\Delta \vdash t : A \textit{ iff } \Delta \vdash^E_{\mathbf{H}_0} t : A.$$

   In other words, any theorem $\Delta \vdash t : A$ *can be proved using axioms from* $\mathbf{H}_0$ *without introduction rules.*

5. *Of special interest is the case of* $\Delta = \varnothing$ *and* $t = \mathbf{e}$. *We have:*

$$\varnothing \vdash \mathbf{e} : A \text{ iff } \varnothing \vdash^{E}_{\mathbf{H}_0} \mathbf{e} : A$$

The left hand side uses introduction rules (i.e. is a labelled natural deduction system) while the right hand side does not, it uses $\mathbf{H}_0$ *instead, and can therefore be seen as a Hilbert axiomatization of the theorems of* $(\mathcal{A}, \mathbf{L})$.

Example 4.2.11 (Hilbert axioms for labelling conditions) *We would like to know under what conditions on the algebra of labels can we find a Hilbert type axiomatization for* $(\mathcal{A}, \mathbf{L})$. *We examine the case of a language with a binary* $\to$ *only, where the algebra is* $(\tau, S, \leq, \mathbf{e}, \otimes)$ *and where the elimination rule has the form*

$$\frac{\varphi(\beta,\alpha); \alpha : A; \beta : A \to B}{\beta \otimes \alpha : B}$$

It is convenient to assume that the algebra satisfies the following additional conditions

- $\forall x \varphi(\mathbf{e}, x)$
- $\forall a, b, \alpha([\varphi(a,\alpha) \wedge \varphi(b,\alpha) \Rightarrow a \otimes \alpha \leq b \otimes \alpha] \Rightarrow a \leq b)$
- $\forall x(\mathbf{e} \otimes x = x \otimes \mathbf{e} = x)$

and residuation*:*

- $\forall a, b \exists! y(a = y \otimes b)$

Our strategy is as follows:

Imagine we are in the middle of a proof from some assumptions. At a ceratin point, say line 99 of the proof, we introduce $\beta : A \to B$ *using an introduction rule. This means that we start a subproof whose first line is the assumption* $\alpha : A$, *where* α *is arbitrary in* $\langle x \rangle \varphi(\beta, x)$ *and its last line is* $\gamma : B$ *where* $\gamma \leq \beta \otimes \alpha$. *Figure 4.1 illustrates the situation:*

98 $t : C$
99 *show* $\beta : A \to B$ *from box*
99.0 $\alpha : A$ *assumption* α *arbitrary in* $\langle x \rangle \varphi(\beta, x)$
$\vdots$
99.k $\gamma : B, \gamma \leq \beta \otimes \alpha$
exit $\beta : A \to B$

Figure 4.2:

Our intention is to dispense with the box justifying line 99 by bringing in several Hilbert axioms and deducing $\beta : A \to B$ *by modus ponens only. We thus want the situation in figure 4.2*

98 $t : C$
98.1 $\mathbf{e} : A_1$ *axiom*
$\vdots$
98.n $\mathbf{e} : A_n$ *axiom*
$\vdots$
99 $\beta : A \to B$ *from axioms and previous lines by modus ponens only.*

Figure 4.3:

The proof that a replacement is possible is by induction on the length and case analysis of the derivation in the box, i.e. on the number k. We start with the innermost boxes (i.e. assume that no *introduction rules ar eused inside the box of fig 4.1) and eliminate these boxes and continue to work our way outwards until all boxes are eliminated. we will be left with anew proof using elimination rules only and axioms.*

The following is a complete case analysis, for the proof in the box:

Case $k = 1$
The box proof has the following structure, fig 4.3.

99.0	$\alpha : A$, *assumption* α
99.1	$\gamma : B, \gamma \leq \beta \otimes \alpha$

Figure 4.4:

We distingush several subcases:

Subcase (1, a)
$\gamma : B$ *is derived from* $\alpha : A$.

Subcase (1, b)
$\gamma : B$ *is obtained from an earlier line outside the box, say line 37 is* $\gamma' : B$.

Subcase (1, c)
$\gamma : B$ *is obtained from* $\alpha : A; \delta : A \to B; \varphi(\delta, \alpha)$ *by elimination.* $\delta : A \to B$ *is outside the box, say line 37 is* $\delta : A \to B$.

Subcase (1, d)
A *has the form* $Z \to B$ *and we have outside the box, say line 37 is* $\varepsilon : Z$ *and* $\gamma : B$ *is obtained by elimination from*

$$\alpha : Z \to B; \varepsilon : Z; \varphi(\alpha, \varepsilon)$$

the above completes the case analysis for $k = 1$. *We now list the cases for* $k > 1$.

Case $k > 1$
In this case the proof structure looks like the one in figure 4.4.

$$\begin{array}{l} \alpha : A \\ \vdots \\ \varepsilon : X \\ \vdots \\ \delta : X \to B \\ \vdots \\ \gamma : B \end{array}$$

Figure 4.5:

We have the following subcases.
Subcase (k, a)

$\gamma : B$ *is obtained from* $\gamma' : B$ *in the box.*
Subcase (k, b)

$\gamma : B$ *is obtained from* $\gamma' : B$ *outside the box.*
Subcase (k, c)

$\gamma : B$ *is obtained by elimination from* $\varepsilon : X$ *and* $\delta : X \to B$, *where* $\varphi(\delta, \varepsilon)$ *holds and both* $\varepsilon : X$ *and* $\delta : X \to B$ *are outside the box.*

Subcase (k, d)

Like subcase (k, c) but with $\varepsilon : X$ outside the box and $\delta : X \to B$ inside.
Subcase (k, e)

Like (k, c) but with $\varepsilon : X$ inside the box and $\delta : X \to B$ outside the box.
Subcase (k, f)

Like (k, c) but with $\varepsilon : X$ and $\delta : X \to B$ both inside the box.
This ends our case analysis. We now examine by induction on k how the box can be eliminated. We go case by case.

Subcase $(1, a)$
In this case we must have $B = A$ and $\alpha \le \beta \le \beta \otimes \alpha$; and α is arbitrary *in $\langle x \rangle \varphi(\beta, x)$.*
We need to show under these conditions that we can prove $\beta : A \to A$ outside the box.
Obviously $\beta : A \to A$ must be derivable from axioms. So if we take the axiom $\mathbf{e} : A \to A$ then we can derive $\beta : A \to A$ provided $\mathbf{e} \le \beta$. β is not arbitrary, it satisfies the condition

$$\forall \alpha [\varphi(\beta, \alpha) \Rightarrow \alpha \le \beta \le \beta \otimes \alpha].$$

So we need the albegra to satisfy:

$$\bullet \bullet \forall \alpha [\varphi(\beta, \alpha) \Rightarrow \alpha \le \beta \le \beta \otimes \alpha] \Rightarrow \mathbf{e} \le \beta.$$

Subcase $(1, b)$
In this case we have outside the box $\gamma' : B$ and we know $\gamma' \le \beta \otimes \alpha$ and α is aritrary in $\langle x \rangle \varphi(\beta, x)$. We want to deive outside $\beta : A \to B$.
Thus we need the axiom

$$\mathbf{e} : B \to (A \to B)$$

We want to carry out elimination with $\gamma' : B$. For this we need $\varphi(\mathbf{e}, \gamma')$ which we assumed always holds. We will get $\mathbf{e} \otimes \gamma' : A \to B$ and from this we need to be able to deduce $\beta : A \to B$. We have to understand what this means. We can get $\beta : A \to B$ if $e \otimes \gamma' \le \beta$. Thus we need

$$\bullet \bullet \quad \forall \alpha \forall \gamma' [\varphi(\beta, \alpha) \Rightarrow \gamma' \le \beta \otimes \alpha] \Rightarrow \mathbf{e} \otimes \gamma' \le \beta$$

since $\mathbf{e} \otimes x = x$ in our algebras we can rewrite it as:

$$\forall \gamma' [\forall \alpha (\varphi(\beta, \alpha) \Rightarrow \gamma' \le \beta \otimes \alpha) \Rightarrow \gamma' \le \beta]$$

Subcase $(1, c)$
In this case we already have $\delta : A \to B$ outside the box. We want to get $\beta : A \to B$ outside the box. What we know about δ is that for an aribtarary $\alpha \in \langle x \rangle [\varphi(\delta, x) \wedge \varphi(\beta, x)]$, $\delta \otimes \alpha \le \beta \otimes \alpha$. If from this we can deduce $\delta \le \beta$ then we are done.
Thus we need

$$..\forall \alpha [\varphi(\delta, \alpha) \wedge \varphi(\beta, \alpha) \Rightarrow \delta \otimes \alpha \le \beta \otimes \alpha] \Rightarrow \delta \le \beta.$$

This is in fact one possible requirement on the algebra.

Subcase $(1, d)$
In this case we have $\varepsilon : Z$ outside the box and we want to get outside the box $\beta : (Z \to B) \to B$. We are given that $\varphi(\beta, \alpha)$ and $\varphi(\alpha, \varepsilon)$ hold and $\alpha \otimes \varepsilon \le \beta \otimes \alpha$. We need the axiom

$$\mathbf{e} : Z \to ((Z \to B) \to B).$$

Since we have $\varphi(\mathbf{e}, \varepsilon)$ we can get $\mathbf{e} \otimes \varepsilon : (Z \to B) \to B$ and if $\mathbf{e} \otimes \varepsilon \le \beta$ we are done. Thus the condition we need is:

$$\bullet \bullet \forall \alpha [\varphi(\beta, \alpha) \wedge \varphi(\alpha, \varepsilon) \Rightarrow (\alpha \otimes \varepsilon \le \beta \otimes \alpha)] \to (\varepsilon \le \beta).$$

Subcase (k, a)
In this case $\gamma : B$ *is obtained from* $\gamma' : B$ *inside the box. Thus we have* $\gamma' \leq \gamma$ *and since* $\gamma \leq \beta \otimes \alpha$ *we have* $\gamma' \leq \beta \otimes \alpha$ *and the induction hypothesis* $\beta : A \to B$ *can be obtained outside the box.*

Subcase (k, b)
This is the same as subcase (1, b).
Subcase (k, c)

$\gamma : B$ *is obtained by elimination from* $\varepsilon : X$ *and* $\delta : X \to B$ *where* $\varphi(\delta, \varepsilon)$ *holds and both* $\varepsilon : X$ *and* $\delta : X \to B$ *are outside. We can perform elimination outside and get* $\gamma : B$ *outside. Now we continue to find the condition as in case (1, b).*

Subcase (k, d)
In this case $\gamma = \delta \otimes \varepsilon$*, with* $\varepsilon : X$ *in the box and* $\delta : X \to B$ *outside the box.* $\varphi(\delta, \varepsilon)$ *holds. By the induction hypothesis we can have outside a proof of* $\varepsilon_0 : A \to X$ *where* $\varepsilon_0 \otimes \alpha = \varepsilon$.

We need to assume that our algebra allows residuation, *i.e.* $\forall \varepsilon, \alpha \exists x (\varepsilon = x \otimes \alpha)$*. In fact in many algebras there exists the minimal such* x.

We know that $\varphi(\beta, \alpha)$ *and* $\gamma = \delta \otimes \varepsilon = \delta \otimes (\varepsilon_0 \otimes \alpha) \leq \beta \otimes \alpha$*. We want to get* $\beta : A \to B$ *outside. We distinguish two possibilities.*

Subcase (d_1)
$\varepsilon_0 \leq \mathbf{e}$.
In this case $\vdash A \to X$*. We need the rule*

$$\frac{\vdash A \to X}{\vdash (X \to B) \to (A \to B)}$$

this means that we can get $\delta : A \to B$ *by modus ponens since* $\varphi(\mathbf{e}, \delta)$ *holds. We have* $\delta \otimes (\varepsilon_0 \otimes \alpha) \leq \beta \otimes \alpha$ *for arbitrary* α *such that* $\varphi(\delta, \varepsilon_0 \otimes \alpha) \wedge \varphi(\beta, \alpha)$ *hold.*

If we can deduce from this $\delta \leq \beta$ *then we finish. We need*

$$\bullet \varepsilon_0 \leq \mathbf{e} \wedge \forall \alpha (\varphi(\delta, \varepsilon_0 \otimes \alpha) \wedge \varphi(\beta, \alpha) \Rightarrow \delta \otimes (\varepsilon_0 \otimes \alpha) \leq \beta \otimes \alpha) \Rightarrow \delta \leq \beta.$$

Subcase (d_2)
Not $\varepsilon \leq \mathbf{e}$
Take the axiom

$$\mathbf{e} : (X \to B) \to ((A \to X) \to (A \to B))$$

Since $\varphi(\mathbf{e}, \delta)$ *holds we get by modus ponens*

$$\delta : (A \to X) \to (A \to B)$$

and assuming $\varphi(\delta, \varepsilon_0)$ *holds, we get* $\delta \otimes \varepsilon_0 : A \to B$.

We need now $\delta \otimes \varepsilon_0 \leq \beta$ *in order to get* $\beta : A \to B$ *outside.*

Thus the condition we need is

$$\bullet \sim \varepsilon_0 \leq \mathbf{e} \wedge \forall \alpha [\varphi(\delta, \varepsilon_0 \otimes \alpha) \wedge \varphi(\beta, \alpha) \Rightarrow \delta \otimes (\varepsilon_0 \otimes \alpha) \leq \beta \otimes \alpha] \Rightarrow \varphi(\delta, \varepsilon_0) \wedge \delta \otimes \varepsilon_0 \leq \beta$$

Subcase (k, e)
We have $\varepsilon : X$ *outside and* $\delta : X \to B$ *inside the box. By the induction hypothesis, we have that* $\delta = \delta_0 \otimes \alpha$ *and* $\delta_0 : A \to (X \to B)$ *can be obtained outside. We want to get* $\beta : A \to B$ *outside. Try the axiom*

$$\mathbf{e} : A \to (X \to B) \to (X \to (A \to B))$$

we can thus get

$$\delta_0 : X \to (A \to B)$$

and assuming $\varphi(\delta_0, \varepsilon)$ *holds we get* $\delta_0 \otimes \varepsilon : A \to B$*. We want* $\delta_0 \otimes \varepsilon \leq \beta$.

We therefore need the condition:

$$\bullet\bullet\ \forall\alpha[\varphi(\delta_0\otimes\alpha,\alpha)\Rightarrow(\delta_0\otimes\alpha)\otimes\varepsilon\leq\beta\otimes\alpha]\Rightarrow\delta_0\otimes\varepsilon\leq\beta\wedge\varphi(\delta_0,\varepsilon)$$

Subcase (k,f)
We have $\delta=\delta_0\otimes\alpha,\varepsilon=\varepsilon_0\otimes\alpha$ *and* $\delta\otimes\varepsilon\leq\beta\otimes\alpha$ *and outsie the box we have*

$$\varepsilon_0 : A\to X$$
$$\delta_0 : A\to(X\to B)$$

we want to get outside the box $\beta : A\to B$.

Take the axiom

$$\mathbf{e} : (A\to(X\to B))\to((A\to X)\to(A\to B)).$$

We need $\varphi(\mathbf{e},\delta_0)$ *to get*

$$\delta_0 : (A\to X)\to(A\to B)$$

and $\varphi(\delta_0,\varepsilon_0)$ *to get*

$$\delta_0\otimes\varepsilon_0 : A\to B$$

and we need $\delta_0\otimes\varepsilon_0\leq\beta$.

Thus the condition is

$$\bullet\bullet\ \forall\alpha(\varphi(\delta_0\otimes\alpha,\varepsilon_0\otimes\alpha)\Rightarrow(\delta_0\otimes\alpha)\otimes(\varepsilon_0\otimes\alpha)\leq\beta\otimes\alpha)\Rightarrow\delta_0\otimes\varepsilon_0\leq\beta\wedge\varphi(\delta_0.\varepsilon_0)$$

4.3 Algorithmic procedures for algebraic LDS

We cannot present a logical system without presenting also proof theoretic algorithmic procedure for it. In our case, it is not just a mattter that the very concept of a logic (as discussed in Chapter 1) requires algorithmic procedures. It is also that some of the natural operations and applications of our systems rely heavily on effective proof procedures. One such an example we have already encountered, is finding conditions on the algebra corresponding to an axiom. This really involves some sort of abduction, and algorithmic procedures are needed.

This section will give a goal directed system. we begin with some definitions and examples. See Definition 2.4.12, Theorem 2.4.13 and Example 2.4.14.

Definition 4.3.1 (Goal directed algorithmic proof for substructural logics) *Let* $\mathcal{A}$ *be an algebra and* $\mathbf{L}$ *be an implictional propositional language. We give a computation for the consequence relation of definition 4.2.3. We assume there is one modus ponens rule of the form*

$$\frac{\varphi(\beta,\alpha);\alpha : A;\beta : A\to B}{\beta\otimes\alpha : B}.$$

Let Δ *be a set of labelled formulas of the form* $\{t_i : A_i\}$ *and let* $s : B$ *be a goal (declarative unit). We define two metapredicates:*
Success $(\Delta,t,B,\mathsf{C},\mathsf{H})$.
Failure $(\Delta,t,B,\mathsf{C},\mathsf{H})$.

The success (failure) predicate means that from the database Δ *the goal* $t : B$ *succeeds, (resp. fails) where the state of the compuation is that the constraints* C *have to be satisfied and where the history of the computation (usually only previous goals) is* H.
C *is a consistent set of wffs in the language of the algebra* $\mathcal{A}$. H *is a list of triples, database, label and goals.*
The computation can go as follows. (It can be slightly varied):
• **Succ** $(\Delta,t,B,\mathsf{C},\mathsf{H})$ *if for some* $t'\leq t, t : B\in\Delta$ *and* C *is consistent in* $\mathcal{A}$ *and* $\delta_D?\delta(t)$ *succeeds.*
• **Succ** $(\Delta'=\Delta\cup\{s_0 : A_1\to\ldots\to(A_n\to q)\ldots)\},t,q,\mathsf{C},\mathsf{H})$ *for* q *atomic, if for some* $s_1,\ldots,s_n$ **Succ** $(\Delta,s_i,A_i,\mathsf{C}',\mathsf{H}')$ *for* $i=1,\ldots,n$, *where* $\mathsf{C}'=\mathsf{C}\cup\{\varphi(s_0,s_1),\varphi(s_0\otimes s_1,s_2),\ldots,\varphi((\ldots(s_0\otimes s_1)\otimes s_2)\ldots\otimes s_{n-1},s_n)\}\cup\{(\ldots(s_0\otimes s_1\otimes\ldots\otimes s_n)\leq t\}$ *and* $\mathsf{H}'=\mathsf{H}*(\Delta',t,q)$ *and* C' *is consistent.*

- **Succ** $(\Delta, t, A \to B, \mathsf{C}, \mathsf{H})$ *if for arbitrary* α *in* $\langle x\rangle\varphi(t,x)$ *we have* **Succ** $(\Delta \cup \{\alpha : A\}, t \otimes \alpha, B, \mathsf{C} \cup \{\varphi(t,\alpha)\}, \mathsf{H})$ *and* $\mathsf{C} \cup \{\varphi(t,\alpha)\}$ *is consistent.*

The clauses for the Failure predicate are as follows:

- **Fail** $(\Delta, t, A, \mathsf{C}, \mathsf{H})$ *if* C *is inconsistent.*
- **Fail** $(\Delta, t, q, \mathsf{C}, \mathsf{H})$ *for* q *atomic, if for each* $s_0 : A_1 \to (\ldots \to (A_n \to q)\ldots)$ *in* Δ *and each* $s_1, \ldots, s_n$ *we have for some* i, **Fail** $(\Delta, s_i, A_i, \mathsf{C}', \mathsf{H}')$ *where* C', H' *are as in the corresponding success clause.*
- **Fail** $(\Delta, t, A \to B, \mathsf{C}, \mathsf{H})$ *if for arbitrary* α *in* $\langle x\rangle\varphi(t,x)$ *we have* **Fail** $(\Delta \cup \{\alpha : A\}, t \otimes \alpha, B, \mathsf{C} \cup \{\varphi(t,a), \mathsf{H})$
- **Fail** $(\Delta, t, q, \mathsf{C}, \mathsf{H})$ *if* q *does not unify with any clause in* Δ.

Many systems can be presented using a form of the Restart rule.

Let R *be a formula in the metalanguage of* H *and* Δ *and* $t : q$, *choosing a database and goal* (Δ', s', q'). *Then the restart rule says*

- **Succ** $(\Delta, t, q, \mathsf{C}, \mathsf{H})$ *if* **Succ** $(\Delta \cup \Delta', s', q', \mathsf{C}, \mathsf{H})$ *where* (Δ', s', q') *is the database and goal chosen by* R.

To be continued ...

4.4 Consequence relations and flattening

4.5 Hilbert formulations

4.6 Semantics and Completeness

4.7 Combining algebraic LDSs

4.8 Tableau for algebraic LDS

Chapter 5

General Labelled Deductive Systems and their Fibred Semantics

5.1 Introduction

The previous two chapters dealt with algebraic *LDS*. These have the form $(\mathcal{A}, \mathbf{L})$, where $\mathcal{A}$ is an algebra of labels. All labels in an algebraic *LDS* are from $\mathcal{A}$ and all formulas are from $\mathbf{L}$. Thus a declarative unit has the form $t : A$, and a database has the form $\Delta = (D, \mathbf{f}, d)$ as defined in Definition 4.2.1. D is a diagram of the theory of $\mathcal{A}$, containing labels and some relations between them of the form $\varphi(x, y) \sim \varphi(x, y), x \leq y, \sim x \leq y$, and some equalities. The function $\mathbf{f}$ associates with each $t \in D$, a wff $\mathbf{f}(t)$.

We want to generalise the above notions. In general, the label is supposed to give more information about the formula it annotates. In the most general case, a label can itself be a database. Similarly, a formula is in general a declarative statement from a logic. In the most general case the declarative statement can be a database. Thus we must give a definition of an *LDS* which allows for declarative units of the form $\Delta_1 : \Delta_2$ where Δ_1 is a database serving as a label and Δ_2 is a database serving as a formula.

Examples 2.3.9, 2.3.10 and 2.3.11 have already illustrated the need for such concepts.

It is easy to understand what $\Delta_1 : \Delta_2$ means in the context of modal logic. Imagine an algebraic modal *LDS*. The declarative units have the form $t : A$, where t is a name for a possible world and A a formula holding at that world.

Imagine A has the form $\Diamond q$, q true in some accessible world. Then Figure 5.1 really means that at world t, there is an $s, t < s$ and $s : q$. Suppose we want to be more specific about what s is. I might write the database in Figure 5.2 , corresponding to the meaning of figure 5.1.
or in our database notation

$$\Delta = (\{t, t \leq s_0, s_0\}, \mathbf{f}, t\}$$

where $\mathbf{f}(t) = \top$ and $\mathbf{f}(s_0) = q$.

What would $x : \Delta$ mean?

It can only mean that the declarative contents of Δ must hold at the label x. But the declarative content of Δ is $t : \Diamond q$. To keep our notation consistent we must have $x = t$. Thus we can allow for

$$\cdot t : \Diamond q$$

Figure 5.1:

$$t\cdot \longrightarrow \cdot s_0 : q$$

Figure 5.2:

expressions of the form $t : (D, \mathbf{f}, d)$ provided $t = d$.

Let us now consider how we can use a database as labels. Consider again $t : \Diamond q$. The point t is a world. Let C be some formula which happens to be true at exactly world t (e.g. C can be 'the world here is t').

Then $C : \Diamond q$ means the same as $t : \Diamond q$. In the general case we can have a whole database, e.g. Δ_1

$$\Delta_1 = (D_1, \mathbf{f}_1, t)$$

'characterising' t, as the only t 'around' which Δ_1 holds. Thus we can write $\Delta_1 : \Diamond q$. We can now write $\Delta_1 : \Delta$ and it should mean $t : \Diamond q$.

In the general case we can write

$$(D_1, \mathbf{f}_1, d_1) : (D_2, \mathbf{f}_2, d_2)$$

we do not have to require $d_1 = d_2$, this will be expected in the semantical denotation.

The above process should be inductively iterated to obtain full generality.

To turn the above into a logical system, we need proper proof theory and semantics for it as well as other traditional investigations.

This is the task of this chapter.

5.2 Databases used as labels

Definition 5.2.1 (Metabases) *Let $(\mathcal{A}, \mathbf{L})$ be an algebraic* LDS *in the sense of definition 4.2.1. We define the notion of an $(\mathcal{A}, \mathbf{L})$ metabase as follows.*

1. *a* diagram *is a set (D, d) of labels, with $d \in D$ together with literals of the form $\varphi(x, y), \sim \varphi(x, y), x = y, x \neq y, x \leq y, \sim x \leq y$, for some $x, y \in D$.*

2. *A metabase of level 1 has the form $(D, \mathbf{f}, d)$ where (D, d) is a diagram and $\mathbf{f}$ is a function associating with ech $t \in D$ a declarative unit $f(t) = s : A$.*

   Note that the label s need not be the same as t.

3. *A metabase of level $n+1$ has the form $(D, \mathbf{f}, d)$, where (D, d) is as before and $\mathbf{f}$ is a function associating with each $t \in D$, the pair $\mathbf{f}(t) = \Delta_1^t : \Delta_2^t$, where Δ_i^t are metabases of level $\leq n$. We also allow $\mathbf{f}(t)$ to be of the form $s : \Delta_2^t$, where s is a label from $\mathcal{A}$ or to be of the form $\Delta_1^t : A$, where A is a formula of $\mathbf{L}$.*

Example 5.2.2 (Modal metabases) *We try to give a semantic interpretationfor modal metabases, to help give ourselves some intuition. This should be compared with Example 2.8.2 and Definition 2.8.3. Let (S, R, a, h) be a propositional Kripke structure. Let $t : A$ be a declarative unit, where t comes from an algebra $\mathcal{A} = (T, <)$ (really a set with an ordering) and A from a modal langauge (with $\Diamond$ and $\Box$). Let g be a function from labels into possible worlds of S. Then for $s \in S$ we have*

1. *$s \vDash^{h,g} t : A$ iff (def) $g(t) = s$ and $s \vDash^h A$ where $\vDash^h$ is the traditional Kripke model satisfaction.*

2. *Let $\Delta = (D, \mathbf{f}, d)$ be a metabase of level 1. Then $\mathbf{f}(t) = t' : A$, for $t \in D$, where t' is an algebraic label. Let $s \in S$. We write $s \vDash^{g,h} \Delta$ iff (def) the following holds:*

   - *$g(d) = s$*

- *g is a homomorphism from $(D, \leq, d)$ into (S, R, a) namely, for all $x, y \in D$, we have*
  $D \vdash x \neq y$ implies $g(x) \neq g(y)$
  $D \vdash x = y$ implies $g(x) = g(y)$
  $D \vdash x < y$ implies $g(s)Rg(y)$
  $D \vdash\sim (x < y)$ implies $\sim g(x)Rg(y)$

- *If $\mathbf{f}(t) = t' : A$ then $g(t) = g(t'))$ and $g(t) \vDash^h A$*

3. *Let Δ be of the form $(D, \mathbf{f}, d)$ such that $\mathbf{f}(t) = \Delta_1^t : \Delta_2^t$ for metabases Δ_i^t.*

   We say $s \vDash^{g,h} \Delta$ iff (def) the following holds

   - *$g(d) = s$*
   - *For all $t \in D$, if $g(t) \vDash^{g,h} \Delta_1^t$ then $g(t) \vDash^{g,h} \Delta_2^t$*
   - *g is a homomorphism from $(D, \leq, d)$ into (S, R, a).*

Example 5.2.3 (Uniquely declarative resource metabases) *This example will illustrate what can be done with metabases.*

Let $(\mathcal{A}, \mathbf{L})$ be an LDS *as in definition 4.2.1. Assume $\mathcal{A}$ is a resource algebra with φ and $\otimes$ and $\mathbf{L}$ is a language with $\rightarrow$ only.*

1. *A declarative unit has the form $t : A$, t a label, A a wff.*

2. *A level $(0, 1)$ metabase Δ of length n is a function $\mathbf{f}$ from the set $\{1, 2, \ldots, n\}$ associating with i a declarative unit $\mathbf{f}(i) = t_i : A_i$. We can also display Δ as $(t_1 : A_1; \ldots; t_n : A_n)$.*

3. *A level $(0, m + 1)$ metabase Δ of length n is a function $\mathbf{f}$ from $\{1, \ldots, n\}$ associating with each i a declarative expression $\mathbf{f}(i) = t_i : \Delta_i$, where t_i is a label and Δ_i is a metabase of level $(0, m'), m' \leq m$.*

4. *A labelled metabase of level 0 has the form $t : \Delta$, where t is a label and Δ a metabase of level $(0, m)$ for some m.*

5. *Let Δ_1 and Δ_2 be two metabases of level 0.*

$$\Delta_i = (t_1 : \Delta_1^i; \ldots; t_{n_i} : \Delta_{n_i}^i)$$

   Then $\Delta_1 * \Delta_2$ is defined as

$$(t_1 : \Delta_1^1; \ldots, t_{n_1}^1 : \Delta_{n_1}^1; t_1^1 : \Delta_1^2, \ldots, t_{n_2}^2 : \Delta_{n_2}^2)$$

6. *Let ϑ be a function associating with each pair of algebraic labels t, s a new label $\vartheta(t, s)$, we now define the notion of $\Delta \vdash_\vartheta s : B$, for a metabase Δ of level 0 and a declarative unit $s : B$. Recall that $\rightarrow$ has the elimination rule*

$$\frac{\alpha : A; \beta : A \rightarrow B; \varphi(\beta, \alpha)}{\beta \otimes \alpha : B}$$

   (a) *$t_1 : A_1 \vdash_\vartheta s : B$ iff $t_1 = s$ and $A_1 = B$*

   (b) *$t_1 : A_1; t_2 : A_2 \vdash s : B$ iff $s = t_1 \otimes t_2$ and $A_1 = A_2 \rightarrow B$ and $\varphi(t_1, t_2)$ hold.*

   (c) *$t_1 : A_1; \ldots; t_n : A_n; t_{n+1} : A_{n+1} \vdash s : B$ iff for some k and s_1, s_2 we have $t_1 : A_1; \ldots; t_k : A_k \vdash s_1 : A \rightarrow B$, and $t_{k+1} : A_{k+1}; \ldots; t_{n+1} : A_{n+1} \vdash s_2 : A$ and $s = s_1 \otimes s_2$ and $\varphi(s_1, s_2)$ hold.*

7. *Let $t : \Delta$ be a labelled metabase of level 0. We write $t : \Delta \vdash_\vartheta s : B$ iff for some $s_1, \Delta \vdash_\vartheta s_1 : B$ and $s = \vartheta(t, s_1)$.*

8. *Let $t : \Delta = t : (t_1 : \Delta_1; \ldots ; t_n : \Delta_n)$ be a labelled metabase of level 0. Let $s : B$ be a declarative unit. We write $t : \Delta \vdash_\vartheta s : B$ iff for some $s_1 : B_1; \ldots ; s_n : B_n$ we have $t_i : \Delta_i \vdash_\vartheta s_i : B_i$ and $t : (s_1 : B_1; \ldots ; s_n : B_n) \vdash_\vartheta s : B$.*

9. *A metabase may not prove anything. For example $\{t_1 : q; t_2 : q\}$ does not prove anything.*

   A metabase Δ of level 0 is said to have a unique declaration if there exists a unique declarative unit $s : B$ such tha $\Delta \vdash_\vartheta s : B$. We call such metabases uniquely declarative. *B is the formula of Δ and s its label. (The other alternatives are that Δ proves B with several labels or proves nothing.)*

10. *Let A be a wff and Δ a uniquely declarative metabase of level 0 with wff A. Then $\Delta : A$ is a metadeclarative unit (i.e. since we have that $\Delta \vdash s : A$ uniquely, instead of writing 's', we can put Δ instead). We could have written Δ alone since A is also determined by Δ but we might prefer this more convenient display.*

    (a) *A metabase of level 1 has the form*

    $$\Delta_1 : A_1, \ldots , \Delta_n : A_n$$

    where $\Delta_i : A_i$ are metadeclarative units.

    (b) *Modus ponens can be extended to metadeclarative units by*

    $$\frac{\Delta_1 : A; \Delta_2 : A \to B; \varphi(s, t)}{s \otimes t : B}$$

    where $\Delta_1 \vdash_\vartheta t : A, \Delta_2 \vdash_\vartheta s : A \to B$. (Note that each Δ_i has a unique label and formula.)
    we can similarly deal with

    $$t_1 : A, \Delta_2 : A \to B$$

    and

    $$\Delta_1 : A, s : A \to B$$

    (c) *Let Δ be a metabase of level 1. The notion of $\Delta \vdash_\vartheta s : B$ can be defined as for metabases of level 0. Similarly the notion of being uniquely declarative can be defined as for metabases of level 0.*

    (d) *A metabase of level $n+1$ has the form $\Delta = (\Delta_1 : \Gamma_1; \ldots ; \Delta_n : \Gamma_n)$ where Δ_i are uniquely declarative metabases of level $\leq n$ and Γ_n are also uniquely declarative metabases of level $\leq n$. Assume by induction that the notion of $\Delta \vdash_\vartheta s : B$ has been defined for metabases of level $\leq n$. We assume that Γ_i proves the unique formula of Δ_i. I.e. if $\Delta_i \vdash s_i : B_i$ then $\Gamma_i \vdash t_i : B_i$. We let*

    $$\Delta \vdash r : C \textit{ iff by definition } \{s_i : (t_i : B)\} \vdash_\vartheta r : C$$

    (e) *A metabase is a metabase of level n, for some n.*

Example 5.2.4 (Linked metabases) *This example continues the example of resource metabases, Example 5.2.3 by adding the concpet of* linking *of databases. This concept was mentioned in Examples 2.7.1 and 2.7.2.*

Consider a metabase Δ. Δ contains in it algebraic labels from the algebra $\mathcal{A}$. Suppose it contains somewhere among its labels a variable x (ranging over labels from the algebra). We can write it as $\Delta[x]$. In an intuitive sense, $\Delta[x]$ is a predicate on x, in the very complex LDS *langauge of the metabase. Its exact nature will be clear only when we give fibred semantics for metabases. At this stage we can think of it just as a 'metapredicate'. Suppose we have two metabases $\Delta_1[x]$ and $\Delta_2[y]$. We can link them by writing Link$(\Delta_1[x], \Delta_2[y], y/x)$. The geometry of the linkage is displayed in figure 5.3*

$$\begin{tikzcd}\Delta_1[x] \\ \Delta_2[y] \arrow[u]\end{tikzcd}$$

Figure 5.3:

Its meaning is that we have a metabase $\Delta_1[x]$ containing a label x satisfying the 'metapredicate' $\lambda y \Delta_2[y]$. This is similar to writing in classical logic the expression:

$$Q_1(x), \textit{ for } x \textit{ such that } Q_2(x) \textit{ holds}$$

we may ask, why not write $Q_1(x) \wedge Q_2(x)$, clearly this means the same? The answer is that sometimes it is more convenient to link, especially if x and y are buried deep inside different metabases whose languages are different!

A simple example where a natural linkage may occur may be the database

$$\Delta_1(x) = \{x : A\},$$

which is a declarative unit, where x is the reliability value of the wff A. If x is obtained by some other numerical model analysis in a language different from that of A, then the most natural way of representing this relation is by Link$(\{x : A\}, \Delta_2(y), x/y)$, where Δ_2 is the metabase supporting the analysis leading to the numerical value x.

Definition 5.2.5 (Semantics for metabases) *A general semantics structure has the form*

$$\mathbf{m} = (W, S, \leq, \varphi, \mathbf{e}, f, h, g, \mathbf{F})$$

where W is a set of labels and for each $x \in W$, we have $S^x \subseteq W, S^x \neq \varnothing \leq^x, \varphi^x$ are relations on S^x, $\mathbf{e}^x \in S^x$, f is a binary fuction on S^x and h^x is an assignment into S^x such that $(S^x, \leq^x, \varphi^x, \mathbf{e}^x, f^x, h^x)$ is a model in the sense of Definition 4.1.2.

g^x is a function associating with each atomic label t an element

$$g^x(t) \in S^x$$

The function $\mathbf{F}^x$ is a binary function $\mathbf{F}^x(y) = z$ associating a new z with each x and y. It is used to 'fibre' our semantics. Since metabases can serve as labels, we must 'fibre' the models of these labels with the current models.

We proceed to define the notion $y \vDash_x \Delta$, to be read, the database Δ holds at point $y \in S^x$ in the model $\mathbf{m}$.

Entailent between metabases Δ, Γ is defined

$$\Delta \vDash \Gamma \textit{ iff } \forall x, y[y \vDash_x \Delta \textit{ implies } y \vDash_x \Gamma]$$

Let us now define $\vDash_x$.

1. *For a basic declarative unit $a : q$ for q atomic and a atomic label, we have:*

$$y \vDash_x a : Q \textit{ iff } h^x(y, q) = 1 \textit{ and } g^x(a) = y$$

2. *$y \vDash_x a : A \to B$ (where A and B are wffs built up from atoms and $\to$) iff $g^x(a) = y$ and for all $y', y' \vDash_x A$ implies $f^x(y, h') \vDash_x B$*

3. *$y \vDash_x s : \Delta$ for $\Delta = (t_1 : \Delta_1; \ldots, t_n : \Delta_n)$ iff $g^x(s) = y$ and for all $i, g^{\mathbf{F}^x(y)}(t_i) \vDash_{\mathbf{F}^x(y)} \Delta_i$*
   We use the function $\mathbf{F}$ to move to the model $\vDash_z$ for $z = \mathbf{F}^x(y)$ and we evaluate $t_i : \Delta_i$ in there.

4. $y \vDash_x (r:$ *Link*$(\Delta_1(u), \Delta_2(\alpha), \alpha/u$ *iff* $g^x(r) = y$ *and* $y \vDash_x \Delta_1$ *and* $y \vdash_x \Delta_2(u)$.
Thus semantically Linking is like taking a conjunction.

5. *We now consider* $y \vDash_x \Delta : \Gamma$*, a case where the label is* Δ*, a metabase.*
We read it as $y \vDash_x a : \Gamma$ *for all labels* a *such that* $y \vDash_x a : \Delta$.

In general, given two *LDS* logics $(\mathcal{A}_1, \mathbf{L}_1)$ and $(\mathcal{A}_2, \mathbf{L}_2)$, we can allow one to be the algebra for the other Thus we can allow for declarative units of the form $\Delta_1 : \Delta_2$, where Δ_1 is a metabase of *LDS*$_1$ and Δ_2 of *LDS*$_2$.

If we use *LDS*$_1$ as a labelling algebra, we need to define the notions $\mathbf{e}, \leq, \varphi, f, \uplus$, etc. required by Definition 4.2.1.

Let us begin by examining what we need to define, for the case, of the same *LDS* i.e. the algebra of metabases Δ labelling a wff A where both A and Δ are from the same $(\mathcal{A}, \mathbf{L})$.

This, we hope, will give us clues for the general case.

Let $\mathcal{A}, \mathbf{e}, \varphi, f, \leq$ and $\uplus$ be given for a language $\mathbf{L}$ with $\to$. Let Δ_1, Δ_2 be two databases. Consider

$$\frac{\Delta_1 : A; \Delta_2 : A \to B; \Phi(\Delta_2, \Delta_1)}{F(\Delta_2, \Delta_1)}$$

We want to find a reasonable definition for Φ and F. Consider

$$t : A_1 \to (A_2 \to \ldots \to (A_n \to B)\ldots) \vdash t : A_1 \to (A_2 \to \ldots (A_n \to B)\ldots)$$

this should hold in the original *LDS*. According to the rules the above holds iff for all s_1 such that $\varphi(t, s_1)$ holds we have the database

$$\varphi(t, s_1); t : B_1; s_1 : A_1 \vdash f(t, s_1) : B_2$$

where $B_i = A_i \to (A_{i+1} \to \ldots \to (A_n \to B)\ldots)$. Continuing this process we get to the notion of a linear database of Example 4.1.6.

The process of increasing the antecedent database by use of introduction rule (adding $s_i : A_i$) indicates what $\Phi(\Delta_2, \Delta_1)$ and $F(\Delta_2, \Delta_1)$ should be

- A database has the form $\Delta = (D, \mathbf{f}, t_0)$, where

$$D = \{t_0, s_1, \ldots, s_n, \varphi(t, s_1), \varphi(t_1, s_2) \ldots \varphi(t_{n-1}, s_n), \ldots, t_i = f(t_{i-1}, s_i), i = 1 \ldots n\}$$

$$\mathbf{f}(s_i) = A_i, i = 1, \ldots, n.$$

- Two databases Δ_1, Δ_2 satisfy $\Phi(\Delta, \Delta')$ if $t'_0 = t_n$.
- $F(\Delta, \Delta') = \Delta''$ where Δ'' is obtained by appending Δ' to Δ. This is made possible by the condition $t_n = t'_0$.

There is another way of looking at the above. Consider

$$\varphi(t_1, t_2); t_1 : A \to B; t_2 : A \vdash t_1 t_2 : B.$$

Suppose

$$\begin{array}{l} \Delta_1 \vdash t_1 : A \to B \\ \Delta_2 \vdash t_2 : A \end{array}$$

If we have *cut* we must deduce

$$F(\Delta_1, \Delta_2) \vdash t_1 t_2 : B$$

where F is the way Δ_1, Δ_2 are 'put together' and $\Phi(\Delta_1, \Delta_2)$ is the compatibility predicate.

To be continued ...

5.3 General LDS

5.4 Fibred semantics

Chapter 6

Quantifiers and η-functions

6.1 Introduction and discussion

This chapter extends the metabox discipline to quantifiers and Skolem functions. The language is built up in the usual way from atomic predicates $P(x_1, \dots, x_n)$, from propositional connectives and the quantifiers $\forall$ and $\exists$. For the purpose of this section let us assume that the only propositional connective is $\rightarrow$. This will allow us to study the metabox behaviour of $\forall, \exists$ in the context of linear, relevant, intuitionistic and classical implications. The next section will give a system of predicate modal and temporal logic as an example of the metabox discipline.

The main problem with the quantifiers is how to handle the four main rules in the metabox. These are:

$\forall$ **Introduction:**

$$\frac{A(y)}{\forall x A(x)}$$

y free and satisfies some restrictions.

$\forall$ **Elimination:**

$$\frac{\forall x A(x)}{A(t)}$$

$\exists$ **Introduction**

$$\frac{A(t)}{\exists x A(x)}$$

$\exists$ **Elimination:**

$$\frac{\exists x A(x)}{A(\eta)}$$

where η is a Skolem constant subject to some rules and restrictions.

The way to discover the rules is to look at some examples, try to prove them using the metabox procedures and see what goes wrong, and what is needed.

The first feature we discover, which we motivate by the examples below, is that we need two types of variables. One type of variables, the ordinary type, are the usual ones, the ones which are used in quantifiers and which can appear free in formulas. These we denote by $x, y, z \dots$. The other type of variables arise from the proper treatment of Skolem functions. These variables we call η-variables. We use $u, v, w, \dots$ to denote these variables. the name η-variable comes from the use of the function η which is a generalised Skolem operator. Assume for example that $\exists x A(x, y)$ is considered true, ie is either among our assumpitons or is derived from them. Then we can skolemise and write $A(\mathbf{f}(y), y))$, where $\mathbf{f}$ is a Skolem function. For our purposes this is too crude. Hilbert ε operator is a bit better. We can consider $\varepsilon_x A(x, y)$ and write $A(\varepsilon_x A(x, y), y)$. For the

	1		Show $\exists x(A(x) \to \forall x A x))$
	2	$\varnothing$	Assumption
	3	$A(u) \to \forall x A(x),$	*from Box 1 and choose u later*
Box 1	4		Show $\forall x A(x)$
	5	$A(u)$	*Assumption*
	6	$\forall x A(x)$	*from box 2*
Box 2	7		*Show* $A(y)$
	8	*Introduce an arbitrary* y	*i.e. skolemise 6*
	9	$A(y)$	*Instantiate* $u = y$ *from 5 and 8*

Figure 6.1:

purpose of the metabox, special care must be taken with Skolem functions. In fact we need a logic for Skolem functions. In this new logic of Skolem functions we use the η-function and write

$$A(\eta x A(x, y), y)$$

Thus the η-function is a generalised sort of ε-function developed especially for the metabox.

The formula $A(u), u$ an η-variable, will be understood as meaning A (constant to be chosen). u is to be chosen at a later step in the metabox. Meanwhile it is put in as an η-variable u. This is best explained by an example.

We know that in any logic

$$\vdash \forall x A(x) \to \forall x A(x)$$

in classical logic we can pull the first quantifier out and get

$$\vdash_C \exists x(A(x) \to \forall x A(x))$$

This is not a theorem of intuitionistic logic. Let us try a metabox proof of this formula, see Figure 6.1.

Example 6.1.1 *A metabox to check whether*

$$\varnothing \vdash \exists x[A(x) \to \forall x A(x)]$$

in intuitionistic logic, figure 6.1

Here we get success. We should not succeed. We went wrong when we made $u = y$. The correct rule is to introduce in line 8 a Skolem function $y = \mathbf{f}(u)$ and hence the step $u = \mathbf{f}(u)$ will be blocked by the occur check. Thus formally the metabox discipline must tell us which $\{u_i\}$ are involved so that when we skolemise we make the Skolem function dependent on the u_i. Moreover, if some u_i are involved in an outer box it is not clear that we transfer (reiterate) them into an inner box. Different logics may have different disciplines.

Let us try another example. This time with and without Restart, see Example 3.4.6 for restart..

Example 6.1.2

$$\neg\neg\exists x A(x) \ ?\ \exists x \neg\neg A(x)$$

*should fail in intuitionistic logic and succeed in classical logic. Let us check it in linear logic and linear logic with restart (*linear classical *in 9.1.2). (See fig 6.2)*

	1		Show $a_1 : \exists x((A(x) \to \bot) \to \bot)$
	2	$a_1 : (\exists x A(x) \to \bot) \to \bot$	Assumption
	3	$\varnothing : \exists x A(x) \to \bot$	from Box 1
Box 1	4		Show $a_2 : \bot, \exists x((A(x) \to \bot) \to \bot)$
	5	$a_2 : \exists x A(x)$	Assumption
	6	$a_2 : A(c)$	Skolemising
	7	$a_2 : \exists x((A(x) \to \bot) \to \bot)$	from Box 2 and use restart
Box 2	8		Show $a_2 a_3 : \bot$
	9	$a_3 : A(u) \to \bot$	Assumption
	10	$a_3 : A(c) \to \bot$	instantiate, $u = c$
	11	$a_2 : A(c)$	reiteration
	12	$a_2 a_3 : \bot$	
exit	13	$a_2 : \exists x((A(x) \to \bot) \to \bot$	
exit	14	$\varnothing : \exists x A(x) \to \bot$	
	15	$a_1 \exists x((A(x) \to \bot) \to \bot)$	from Box 3
Box 3 :	16		$a_1 a_4 : \bot, u$ is an η variable
	17	$a_4 : A(u) \to \bot$	Assumption
	18	$a_1 : \bot$	from lines 2 and 3
	19	we cannot exit in Relevance of linear logic because the	
	20	label of $\bot$ is a_1 and it does not contain (use) the label of	
	21	box assumption (namely a_4) . However, we an exit in	
	22	intuitionistic and classical logic because these logics	
	23	disregard the labels.	
exit	24	$a_1 : \exists x((Ax)x \to \bot) \to \bot)$	

Figure 6.2:

	1		Show $a : \exists x A(x) \to B$
	2	$a : \forall x(A(x) \to B)$	Assumption
	3	$a : \exists x A(x) \to B$	from Box 1
Box 1	4		Show $ab : B$
	5	$b : \exists x A(x)$	Assumption
	6	$b : A(c)$,	Skolem constant
	7	$a : A(c) \to B$,	from line 2
	8	$ab : B$	
exit	9	$a : \exists x A(x) \to B$	

Figure 6.3:

	1		Show $a : \forall x(A(x) \to B)$
	2	$a : \exists x A(x) \to B$	Assumption
	3	$a : \forall x(A(x) \to B)$	from box 1
Box 1	4		Show $a : A(y) \to B$
	5	Introduce y	
	6	$a : A(y) \to B$	from Box 2
Box 2	7		Show $ab : B$
	8	$b : A(y)$	Assumption
	9	$b : \exists x A(x)$	Rule
	10	$ab : B$	
exit	11	$a : A(y) \to B$	

Figure 6.4:

Example 6.1.3 *The equivalence of*

$$\forall x(A(x) \to B) \leftrightarrow (\exists x A(x) \to B)$$

where x not free in B, holds in linear logic, as the following two boxes illustrate. (See figs 6.3 and 6.4)

The second box is (fig 6.4).

We are now ready to give a formal definition of the notion of a quantificational metabox. It is convenient to use Definition 2.4.1 and Theorem 2.4.2 of Chapter 2. It is sufficient to give the Goal Rules, Data Rules and Metabox rules for each of the quantifiers. Having done that, we will be able to add $\forall, \exists$ to any propositonal metabox system. Definition 6.1.4 here should be compared with the skolemization and visa rules of Definition 3.2.8. It is especially useful to compare how both definitions apply to the case of **S4** strict implication in view of the discussion presented in Example 2.2.5.

Definition 6.1.4 (Metabox Quantifier Rules) *1. The predicate language contains two types of variables, ordinary variables for the quantifiers denoted by $\{x, y, z, \ldots\}$ and η variables for*

Skolem constants denoted by $\{u, v, w, \ldots\}$. *It also contains the Skolem functor* η. *We define the notion of* well formed formulas (wffs) *and* terms *as follows:*

1.1. *Any* n*-place atomic predicate* $Q(x_1, \ldots, x_n)$ *with the indicated variables is a wff with the variables free.*

1.2. *If* $\sharp$ *is an* m*-place connective and* $A_i(x_{i,j})$ *are formulas with the indicated free variables, then* $\sharp(A_1, \ldots, A_n)$ *is a formula with the indicated (union) of free variables.*

1.3. *If* $A(x, y_i)$ *is a wff with an ordinary variable* x *free then*

$$\forall x A(x, y_i) \text{ and } \exists x A(x, y_i)$$

are wffs with the remaining y_i *variables free.*

1.4. *If* $\alpha : A(x, y_i)$ *is a labelled formula,* x *an ordinary variable and* u *is a set of* $\eta-$ *variables then* $\eta[\alpha : A(x, y_i), u]$ *is a term.*

2. *A quantified metabox system is comprised of the following components:*

2.1. *A finite partially ordered system of metabox names* $\{\mathbf{a}_i, <\}$, *as in (1) of Definition 3.4.9. In addition with each metabox name* $\mathbf{a}$, *a finite set of* η*-variables* $U(\mathbf{a})$ *is associated, together with a substitution* $\Theta\mathbf{a}$ *to these variables.*

The metabox system satisfies the conditions of Definition 3.4.9(5).

3. *The following are the Goal Rules, Data rules and Metabox rules for the quantifiers. Imagine we are at a certain line in a box* $\mathbf{a}$

3.1. *To show* $\alpha : \exists x A(x)$ *we open a new box* $\mathbf{b}$ *and choose a new* η *variable* u, *associate* $V(\mathbf{a}) \cup \{u\}$ *with* $\mathbf{b}$. *Let the assumptions and reiterations of* $\mathbf{b}$ *be the same as those of* $\mathbf{a}$ *together with whatever was already proved in* $\mathbf{a}$. *We want to show* $\alpha : A(u)$ *in box* $\mathbf{b}$.

3.2. *To show* $\alpha : \forall x A(x)$ *we start a box* $\mathbf{b}$ *and show*

$$\alpha : A(\eta x[A(x), U(\mathbf{a})])$$

The term $\eta x[A(x), U(\mathbf{a})]$ *is marked as having been first introduced in this box.*

3.3. *At any time in the forward proof at any box* $\mathbf{a}$ *we can modify the substitutiton* $\Theta_{\mathbf{a}}(u)$ *to be* $\Theta_{\mathbf{a}}(u) = t$. *This is allowed provided the occur check is satisfied.*

3.4. $$\frac{\alpha : \forall x A(x)}{\alpha : A(t)}$$

Subject to restrictions on the term t *and where it was first introduced.*

3.5. $$\frac{\alpha : A(t)}{\alpha : \exists x A(x)}$$

Subject to restrictions on the term t *which take into account where it was first introduced.*

3.6. $$\frac{\alpha : \exists x A(x)}{A(\eta x[\alpha : A(x), U(\mathbf{a})]}$$

The term $\eta x[\alpha : A(x), U(\mathbf{a})]$ *is marked as having been first introduced in box* $\mathbf{a}$.

3.7. *The metabox rules for the propositional connectives about what can be reiterated are also valid for the universal closures of the reiterated formulas but not for existential closures.*

Box a_1	1		Show $a_1 : \forall x A(x) \to \forall x B(x)$
	2	$a_1 : \forall x(A(x) \to B(x))$	assumption
	3	$a_2 : \forall x A(x) \to \forall x B(x),$	from box a_2
Box a_2	4		Show $a_1 a_2 : \forall x B(x)$
	5	$a_2 : \forall x A(x)$	assumption
	6	$a_1 a_2 : \forall x B(x)$	from box a_3
Box a_3	7		
	8	*Introduce* $y = \eta x B(x)$	
	9	$a_1 : \forall x(A(x) \to B(x))$	*Reiteration*
	10	$a_1 : A(y) \to B(y)$	$\forall\exists$*Rule*
	11	$a_2 : \forall y A(y)$	*Reiteration*
	12	$a_2 : A(y)$	$\forall\exists$*Rule*
	13	$a_1 a_2 : B(y)$	

Figure 6.5:

Example 6.1.5 *We show that in the logic* **W** *see Definition 9.2.1:*

$$\vdash \forall x[A(x) \to B(x)] \to [\forall x A(x) \to \forall x B(x)]$$

but

$$\nvdash \forall x A(x) \to [\forall x(A(x) \to B(x)) \to \forall x B(x)].$$

See fig 6.5.

The proof of $\forall x A(x) \to [\forall x(A(x) \to B(x)) \to \forall x B(x)]$ *is blocked by the reiteration restrictions of the logic* **W**.

Chapter 7

A General Theory of Structured Consequence Relations

7.1 Introduction

In previous chapters we introduced a systematic framework where the logical units are labelled formulas (of the form $t : A$, where t is a term in some algebra) and where the logical rules are rules for manipulating both formulas and their labels. The intuition behind *LDS* is that in $t : A, A$ carries declarative information and the label t, represents further information of a different nature which we do not want to "code" into A. It turns out that many monotonic, non-monotonic and probabilistic systems can be presented as *LDS* systems. An *LDS* database Δ is a constellation of labelled formulas with additional structure on the labels. For example Δ may be an algebra τ with a function $\delta(x), x \in \tau$, assigning formulas $A_x = \delta(x)$, for each $x \in \tau$.

Proof rules are given to define the notion $\Delta \vdash s : B$. This approach is developed in detail in other chapters. It is possible to have $\Delta \vdash s_i : B$ for several labels s_i, for example s_i may show from which part of Δ B is proved. In many applications one is interested in the notion of $\Delta \vdash B$, where we want to ignore the label and just give a yes or no answer to the query " Does B follow from Δ". In such cases we are actually *hiding* the labels and all we have is a structured database Δ proving or not proving a formula B. We can in this case define axiomatically the properties of the consequence $\Delta \vdash B$. We thus end up with a notion of consequence between Δ and B, where Δ is structured and the labels in Δ are hidden, and used only in the proof system. It turns out that this simplified notion is very useful and can be developed on its own as a direct extension of the notion of monotonic consequence relation. We call our studies the area of *structured consequence relation.* It can be developed independently of *LDS*, as a direct contribution to the abstract area of *axiomatic non-monotonic reasoning.*

The purpose of this chapter is to develop the notion of structured consequence relation and further refine the notion of a non-monotonic consequence relation to relate to non monotonic systems arising from resource boundedness, and controlled inference. The refinement is in the direction of structuring the assumptions.

The traditional notion of a database Δ and a logical unit of information A (on which consequence $\Delta \mid\sim A$ is defined) is that of a set Δ and a wff A.

Many non-monotonic databases are naturally structured. The database may be, for example, a list of assumptions, or a partially ordered set of assumptions, and consequences are proved from the database using an inferential (logical) discipline which takes into account such a structure. Non-monotonicity may arise because of these "resource" considerations and inferential restrictions.

We introduce the new notion of a non-monotonic consequence relation, which we call S-consequence relation (Structured Consequence Relation). A precise definition will be given in a later section. We need two auxiliary notions for our definition:

1. We have already indicated that we need to deal with structured databases. We thus have to

specify precisely what structures are allowed as databases, for the consequence relation to be defined. These must include the one formula database (A) and the empty database $\varnothing$.

2. The notion of a structured database must also include the concept of *structured addition of data*. By this we mean the notion of how to combine two databases together. The traditional way is to take their union but when the databases are already structured, the union has to be structured as well. We denote the structured addition of the databases Δ and Γ by $\Delta + \Gamma$. This binary operation, "+", has to be defined as part of the concepts underlying the consequence relation. It usually satisfies associativity

$$\Delta_1 + (\Delta_1 + \Delta_3) = (\Delta_1 + \Delta_2) + \Delta_2$$

and the empty database must satisfy

$$\varnothing + \Delta = \Delta + \varnothing = \Delta$$

Commutativity is not required to hold.

3. We also need a concept of substitution of one structured database Δ inside another, Γ, achieved by replacing one formula A in Γ by Δ. Formally we need to make sense of the symbol $\Gamma[A]$, reading Γ is a structured database which contains A somewhere inside, and the symbol $\Gamma[\Delta]$, which denotes the database resulting from substituting Δ for A inside Γ. These notions are needed to formulate the Cut Rule.

We can now define the three rules which we call **Identity**, **Surgical Cut** (or **Substitutional Cut**) and **Directional Monotonicity** as follows:

Identity

$$(A) \mid\!\sim A$$

Surgical (Substitutional) Cut

$$\frac{\begin{array}{c}\Delta \mid\!\sim A\\ \Gamma[A] \mid\!\sim B\end{array}}{\Gamma[\Delta] \mid\!\sim B.}$$

Directional Monotonicity (right hand side)

$$\frac{\Delta \mid\!\sim A}{\Delta + \Gamma \mid\!\sim A}$$

Directional Monotonicity (left hand side)

$$\frac{\Delta \mid\!\sim A}{\Gamma + \Delta \mid\!\sim A}$$

Definition 7.1.1 (Tentative Definition of Structured Consequence Relation) *Let* **L** *be a language. Let $\mathcal{L}$ be a family of order types of the form $(\tau, \leq)$. Assume that the empty set and one element set types are in $\mathcal{L}$. Assume that the notion of substitution of one order type τ for a point x in another order type $\tau'(x)$ is well defined (as in (3) above).*

We say that a relation $\Delta \mid\!\sim A$, (between ordered multisets Δ of wffs of an order type τ and a single wff A) is a S-consequence relation *iff it satisfies* **Identity** *and* **Surgical Cut**.

The next two sections will describe several logics in detail and section 4 will give the precise definitions of our general concepts.

We now give some examples of structured consequence relations

Example 7.1.2 (Hereditarily Finite Sets) *1. Let X be a set of atoms and let the family of wffs be the family of all Hereditarily Finite Sets built up from X, ie*

(a) Any finite subset of X is a wff.

(b) If A and B are wffs so are $A \cup B$ and $\{A\}$.

2. For a finite set of wffs Δ, let $\Delta \mid\!\sim A$ iff (definition) $A \in \Delta$.

3. Define the notion of substitution of one set for another to be as follows:
The result of substituting Δ for $y \in \Gamma$ is the set $(\Gamma - \{y\}) \cup \Delta$.
Define $\Delta + \Gamma$ as $\Delta \cup \Gamma$

4. It is easy to show that $\mid\!\sim$ satisfies **Reflexivity, Monotonicity** *and* **Surgical Cut**.

5. Note also that if our data strucutres were just subsets of $X$with substitution and $+$ defined as in (3), then **Reflexivity, Monotonicity** *and* **Surgical Cut** *are still satisfied.*

Example 7.1.3 (Hereditarily Finite Stacks) *1. Let X be a set of atoms. Define the notion of a wff by*

(a) $(a_1, \ldots, a_n)$ is a wff if $a_i \in X$ for all i

(b) If $A_1, \ldots, A_n$ are wffs so is $(A_1, \ldots, A_n)$.

(c) A database is any wff.

2. Define $(A_1, \ldots, A_n) \mid\!\sim A$ iff (definition) for some i, $A = A_i$.

3. Define strucutral substitution as follows:
The result of substituting for y in $(A_1, \ldots, A_n, y, B_1, \ldots, B_m)$ the list $(C_1, \ldots, C_k)$ is the list $(A_1, \ldots, A_n, C_1, \ldots, C_k, B_1, \ldots, B_m)$.

4. Define $\Delta + \Gamma$ to be the concatenation, in that order, of Δ and Γ.

5. It is easy to show that $\mid\!\sim$ satisfies **Identity, Directional Monotonicity** *and* **Surgical Cut**.

Example 7.1.4 *1. Let* **L** *be the language of classical or intuitionistic logic and let the data structures be sets. Define a consequence relation $\mid\!\sim_{H^0}$ by $\{A_1, \ldots, A_n\} \mid\!\sim_{H^0} B$ iff $B \vdash_H \bigvee_{i=1}^n A_i$ where $\vdash_H$ is provability in classiccal logic ($H = C$) or intuitionistic logic ($H = I$). We show that $\mid\!\sim_{H^0}$ is an S-consequence relation.*

It is obvious that it satisfies **Identity**, *we check the* **Surgical Cut** *rule.*

Assume, for example, that $\Delta, A \mid\!\sim_{H^0} B$ and $\Gamma \mid\!\sim_{H^0} A$. We must show that $\Delta, \Gamma \mid\!\sim_{H^0} B$. From our assumptions we have $B \vdash_H A \vee \bigvee \Delta$ and $A \vdash_H \bigvee \Gamma$.

We get therefore

$$B \vdash_H \bigvee \Gamma \vee \bigvee \Delta$$

hence by definition

$$\Delta, \Gamma \mid\!\sim_{H^0} B.$$

Note that the following holds for all A, B

$$A \vdash_H B \text{ iff } B \mid\!\sim_{H^0} A.$$

2. *Define another consequence relation* $\mid\!\sim_{H^1}$ *by* $\{A_1, \ldots, A_n\} \mid\!\sim_{H^1} B$ *iff for some* A_i, $B \vdash_H A_i$. *We show that* $\mid\!\sim_{H^1}$ *is a consequence relation. It is obvious that it satifies* **Identity**. *We check the* **Surgical Cut**.

   Assume $\Delta, A \mid\!\sim_{H^1} B$ *and* $\Gamma \mid\!\sim_{H^1} A$. *We must show that* $\Delta, \Gamma \mid\!\sim_{H^1} B$. *From our assumptions,* $B \vdash_H X$, *for some* $X \in \Delta \cup \{A\}$ *and* $A \vdash_H Y$ *for some* $Y \in \Gamma$. *We need to show that* $B \vdash_H Z$ *for some* $Z \in \Delta \cup \Gamma$. *Such* Z *exists because:*
   If $X \in \Delta$ *let* $Z = X$
   If $X = A$ *let* $Z = Y$.

   Clearly for all A, B,
   $A \vdash_H B$ *iff* $B \mid\!\sim_{H^1} A$.

Example 7.1.5 (Context Free Grammar) *Consider a context free grammar, with rules of the form*

$$y \mapsto x_1 \ldots x_n$$

where x_i *are either variables or terminals. Regard the rules as basic consequence relation axioms in the form*

$$x_1 \ldots x_n \vdash y$$

Let $a_1 \ldots a_m \mid\!\sim b$ *hold if* b *can be reduced in the grammar to* $a_1 \ldots a_m$.

$\mid\!\sim$ *certainly satisfies* **Identity** *and* **Surgical Cut** *(for the obvious substitutions), that is:*

$$\frac{\begin{array}{c} a_1 \ldots a_m \mid\!\sim b \\ c_1 \ldots b \ldots c_n \mid\!\sim c \end{array}}{c_1 \ldots a_1 \ldots a_m \ldots c_n \mid\!\sim c}$$

This should not be surprising because Horn clause logic can be viewed as context free grammar.

7.2 A general notion of an unlabelled logical system

We have seen that the general notion of a database is a configuration of formulas. In a previous chapter we introduced the logic **CL** (concatenation logic), and the configurations for they were linear orderings of formulas. The proof discipline from such databases would rely on the ordering in defining the allowable proof moves. In the case of **CL** we allowed modus ponens only if the minor premise was supported by a later part of the database than the implication. In the general case, other restrictions may come into play.

This section develops the general notions needed for S-consequence relation. We begin with definitions and examples.

Definition 7.2.1 1. *Let* **L** *be a langauge for well formed formulas.* **L** *will contain atomic propositions and connectives and the notion of a wff is defined inductively in the traditional way.*

2. *let* $\mathcal{L}$ *be a theory in first or higher order logic and let* $\mathcal{M}$ *be a class of classical models of* $\mathcal{L}$. *We assume that the structure with one point and empty relations is in* $\mathcal{M}$. *Assume that two model theoretic operations are defined on* $\mathcal{M}$ *and assume that* $\mathcal{M}$ *is closed under these operations. The operations are the following:*

   (a) If τ_1 *and* τ_2 *are two structures in* $\mathcal{M}$ *then an operation* $+$ *is available for constructing the structure* $\tau = \tau_1 + \tau_2$. *"+" usually satisfies associativity, but not necessarily in the general case. If* $\varnothing$ *is the empty set then* $+$ *is defined and* $\varnothing + \tau = \tau + \varnothing = \tau$.

   (b) Let τ_1 *and* τ_2 *be two disjoint structures in* $\mathcal{M}$ *and let* $x \in \tau_1$ *be a point in* τ_1. *Then a substitution operation* $\mathrm{Sub}^x_{\tau_2}\tau_1(x)$ *is defined which yields the result of substituting* τ_2 *in* $\tau_1(x)$ *for* x *(ie* $\lambda x \tau_1(x)[\tau_2]$ *is well defined). The result of the substitution is also in* $\mathcal{M}$. *Sub is associative, and behaves like substitution if done properly (no clash of variables).*

(c) *For some purposes (see section on structural connectives) we need to allow for the notion of* deletion. *That is we must know how to delete the point x from $\tau(x)$. We can regard deletion as substituting the empty structure $\varnothing$ for x, ie $\lambda x \tau(x)[\varnothing]$.*

3. *A database is any pair (τ, δ) where $\tau \in \mathcal{M}$ and δ is a function assigning for each $x \in \tau$ a wff $\delta(x)$.*

4. *A consequence relation for the pair $(\mathcal{M}, \mathbf{L})$ is any relation $\mid\!\sim$ between databases $\Delta = (\tau, \delta)$ and single wffs A, written in the form $\Delta \mid\!\sim A$, satisfying the following:*

   (a) **Identity**

$$(A) \mid\!\sim A$$

   (b) **Surgical Cut for $\mathcal{M}$**
   If

$$\Delta = (\tau_1, \delta_1), \Gamma = (\tau_2, \delta_2), \tau_1 \cap \tau_2 = \varnothing, x \in \tau_2 \text{ and } \delta_2(x) = A$$

   and if further

$$\begin{array}{l} \Delta \mid\!\sim A \\ \Gamma \mid\!\sim B \end{array}$$

   then

$$\Gamma[A] = (\tau_3, \delta_3) \mid\!\sim B$$

   where

$$\tau_3 = \text{Sub}^x_{\tau_1} \tau_2(x)$$

   and

$$\delta_3(y) = \begin{cases} \delta_1(y) & \text{if } y \in \tau_1 \\ \delta_2(y) & \text{if } y \in \tau_2. \end{cases}$$

Example 7.2.2 1. *Let $(T, <_T)$ be an ordering, and let $t \in T$. Let $(S, <_S)$ be another ordering with $T \cap S = \varnothing$. We define the substitution $[(T, <_T) \mid t \mid (S, <_S)]$ to be the ordering with $T \cup S - \{t\}$ as the set of points and order $<$ defined by*

$$\begin{array}{ll} x < y & \text{if } x, y \in T \text{ and } x <_T y \\ & \text{or } x, y \in S \text{ and } x <_S y \\ & \text{or } x \in T \text{ and } y \in S \text{ and } x < t \\ & \text{or } x \in S \text{ and } y \in T \text{ and } t < y. \end{array}$$

The idea of the definition is to take the ordering $(T, <_T)$ and to take a point $t \in T$ and take another ordering $(S, <_S)$ and to replace t by the entire set S. We now have an ordering with S in the middle. Thus all points of T which were before (after) t are now before (after) all the points of S. Inside S the ordering is the same as before, ie $<_S$.

2. *Define $(T, <_T) + (S, <_S)$ to be the ordering with all elements of T coming before all elements of S.*

3. *Let $\mathbf{L}$ be a language. Assume that part of the definition of $\mathbf{L}$ is the definition of allowable databases of the form $\{t : A_t \mid t \in T\}$, where $(T, <_T)$ is an order configuration. Assume that the order configurations of $\mathbf{L}$ are closed under the substitutions defined in the above. Let $\Vdash$ be a relation defined in $\mathbf{L}$ between data structures and formulas. Then the Cut Rule for $\Vdash$ is defined to be the following:*

   Surgical Cut Rule
   If $\{t : A_t \mid t \in T\} \vdash B$
   and $\{s : B_s \mid s \in S\} \vdash C$
   and $x \in S$ and $B_x = B$
   then $\{r : A_r \mid r \in S \cup T - \{x\}\} \vdash y : C$
   where the configuration on $S \cup T - \{x\}$ is $[(S, <_S) \mid x \mid (T, <_T)]$ defined above.

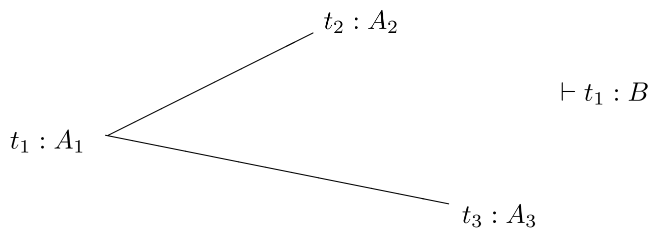


Figure 7.1:

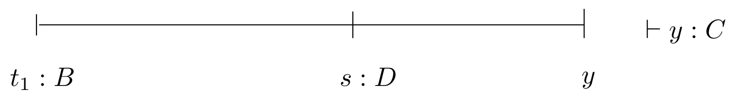


Figure 7.2:

Example 7.2.3 *Consider modal logic. A configuration is an ordered set of possible worlds. Thus if for example we have the configurations in figures 7.1 and 7.2 then the application of the cut rule will yield the configuration of figure 7.3*

We now want to propose a form for the deduction theorem. The discipline must tell us how a database Δ can prove an implicational formula of the form $A \to B$.

Let us understand $A \to B$ as a sort of deductive "functor" which "given" A can "yield" us B. Thus $\Delta \vdash_{\mathbf{CL}} A \to B$ means that if we "add" A to Δ ie "give" A to Δ, we can "get" or "yield" B from Δ "and" A.

In general application areas, Δ is a structured database; some configuration of formulas. Thus the notion of adding A to Δ needs to be defined. We need to specify where to fit A in the configuration of Δ. In the general case precise definitions must be given. Since there may be more than one way of "adding" A to Δ, we may have several possible versions of the deduction theorem as well as several possible implications.

Example 7.2.4 *Consider* **CL** *and its data structures of lists of wffs (see sections 2.4 and 10.1 for details). Let $\Delta = (A_1, \dots, A_n)$ and consider A. This new data item can be "added" to Δ in several ways:*

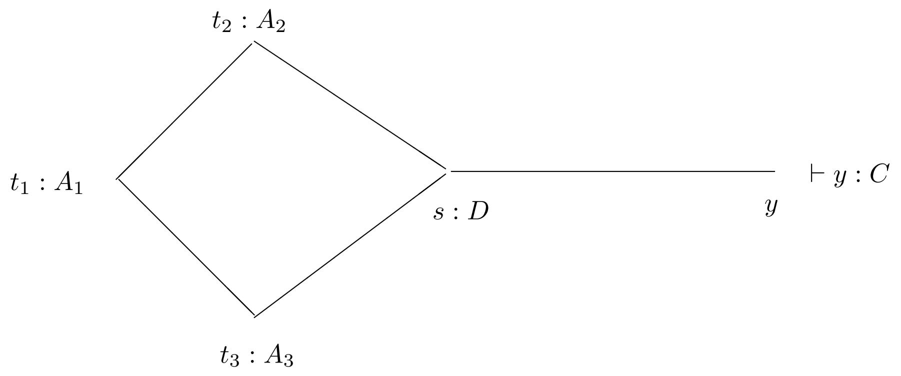


Figure 7.3:

1. *Add A to the end of the sequence to form*

$$(A_1, \ldots, A_n, A)$$

2. *Add A to the beginning of the sequence to form*

$$(A, A_1, \ldots, A_n)$$

3. *Add A to the middle of the sequence to form*

$$(A_1, \ldots, A_k, A, A_{k+1}, \ldots, A_n)$$

4. *Of course we can have combinations, for example, add A to the middle or the beginning, etc*

The **CL** *deduction theorem allows for A to be added to the end of the sequence. We can thus write $A \rightarrow_e B$ for the* **CL** *implication to indicate that given A it can give B provided A is added to the end of the data sequence. The next connective to consider is the implication $A \rightarrow_b B$ where "b" means "beginning". Here we expect a deduction theorem of the form*

$$(A, A_1, \ldots, A_n) \vdash B,$$

iff

$$(A_1, \ldots, A_n) \vdash A \rightarrow_b B$$

For example, since

$$A \rightarrow_b B \vdash A \rightarrow_b B$$

we will get in this logic

$$A, A \rightarrow_b B \vdash B.$$

In concatenation logic the implication must come earlier than the minor premise. In this logic, the minor premise comes first. The set of theorems of this logic, with $\rightarrow_b$, is unchanged; we get **CL** *again, except that in the semantics we now concatenate to the left. A different system is obtained if we have both implications (see [Wansing, 1990]). This is actually the Lambek Calculus. The Lambek Calculus can be defined as the smallest S-consequence relation with two implications $\rightarrow_e$ and $\rightarrow_b$ satisfying the appropriate Deduction Theorems for the list data structures.*

For example, since

$$A \rightarrow_e B, A \vdash B$$

we get that

$$A \vdash (A \rightarrow_e B) \rightarrow_b B$$

The stipulation that

$$A \rightarrow_e B \text{ iff } A \rightarrow_b B$$

gives us linear logic, with $\rightarrow$ being either of $\rightarrow_b$ or $\rightarrow_e$.

Both $\rightarrow_e$ and $\rightarrow_b$ satisfy a Deduction Theorem.

Having illustrated our concepts, we can now generalise the notion of the deduction theorem to arbitrary data constellations. To achieve that we also need to generalise the notion of implication.

Definition 7.2.5 *1. Let $\mathcal{M} = \{\tau_i\}$ be a class of structures τ_i, as in definition 7.2.1. Let φ be a wff in some language $\mathcal{L}$ capable of expressing properties of the structures in $\mathcal{M}$.*

Assume that for every τ in $\mathcal{M}$, φ defines a unique element (or possibly a unique set of elements) in τ.

The formula φ defines a contraction mapping on structures τ. Let τ' be the structure obtained from τ by taking out the points identified by φ. Thus, for example, for lists structures τ, if φ identifies the last element in the list then τ' is the list without its last element. We can denote the contraction mapping function by "$\parallel \varphi$" and write $\tau' = \tau \parallel \varphi$.

Thus, given two structures τ_1, τ_2 we may say that $\tau_2 = \tau_1 \| \varphi$ iff τ_2 is obtained by dropping out of τ_1 the elements identified by φ. We write $\tau[\|\varphi]^n$ to mean $\tau\|\varphi \ldots \|\varphi$, n times. For example, for lists τ and φ which identifies the last element of the list, we have $(t_1, \ldots, t_n) = (t_1, \ldots, t_n, x)\|\varphi$ and $\varnothing = (t_1, \ldots, t_n)[\|\varphi]^n$.

2. *Let a database discipline be given, involving a family of structures $\mathcal{M}$. A database Δ in this discipline is any pair (τ, δ) where τ is a finite element of $\mathcal{M}$ (eg a list) and δ associates with each $t \in \tau$ a formula $\delta(t) = A_t$. We also write $\Delta = \Delta'\|\varphi$, just in case $\Delta = (\tau, \delta), \Delta' = (\tau', \delta')$ and $\tau = \tau'\|\varphi$ and $\delta = \delta' \restriction \tau$, ie δ is the restriction of δ' to τ.*

Example 7.2.6 *Let $\mathcal{M}$ be the class of all finite lists including the empty list. Define + as concatenation. Let $\mathcal{L}$ be a language with "+" and "<". φ can be a formula in $\mathcal{L}$ which defines uniquely the last element of any finite list. Thus for $\tau = (a_1, \ldots, a_n, b)$, we have $\tau\|\varphi = (a_1, \ldots, a_n)$. These lists are the data structures of* **CL**.

Definition 7.2.7 (The Notion of the Deduction Theorem) *Let Δ, Δ' be two databases and assume that $\tau = \tau'\|\varphi$. Assume further that we have*

$$\Delta\|\varphi \mathrel{|\!\sim} A \to B \text{ iff } \Delta' \mathrel{|\!\sim} B$$

Consider a special connective denoted by $\to_\varphi$. We say that $\to_\varphi$ satisfies the Deduction Theorem with respect to *φ iff whenever $\Delta \mathrel{|\!\sim} B$ and $\Delta = (\tau, \delta)$ and τ has n elements then*

$$\Delta[\|\varphi]^n \mathrel{|\!\sim} \varnothing : \delta(t_1) \to_\varphi (\ldots \to_\varphi (\delta(t_n) \to_\varphi B) \ldots)$$

where $\{t_i\} = \tau[\|\varphi]^i - \tau[\|\varphi]^{i+1}$ i.e. t_i is the element identified by φ in $\tau[\|\varphi]^i$.

Our discussions so far presented the consequence relation in the form of $\Delta \mathrel{|\!\sim} A$, where A is a single formula on the right hand side. This is a Tarski type consequence relation. The general form of the consequence relation should be $\Delta \mathrel{|\!\sim} \Gamma$, where both Δ and Γ are structured databases. This is a Scott type consequence relation. In fact, there is no reason at all to assume that the same structures can serve on both sides of the turnstile. If we refer to the left hand side as *the data* and the right hand side as the goal, then the general Scott type S-consequence relation has the form $\Delta \mathrel{|\!\sim} \Gamma$, where Δ is a data structure and Γ is a goal structure. For example, Δ may be a set of formulas while Γ may be a list of formulas. This is indeed the case, for example, in Logic Programming, where we have

$$\Delta \mathrel{|\!\sim} (A_1, \ldots, A_n) \text{ iff } \Delta \mathrel{|\!\sim} A_1 \text{ and } \ldots \text{ and } \Delta \mathrel{|\!\sim} A_n$$

If Δ is a sequence, as in the case of **CL**, or a multiset, as in the case of linear logic, then we can have

$$\Delta \mathrel{|\!\sim} (A_1, \ldots, A_n) \text{ iff for some } \Delta_i, \Delta = \Delta_1 + \ldots + \Delta_n \text{ and } \Delta_i \mathrel{|\!\sim} A_i, i = 1, \ldots, n$$

where + is concatenation or multiset union, respectively.

Another example is the case of Gentzen systems for classical logic, where both Δ and Γ are sets and where we have $\Delta \mathrel{|\!\sim} \{A_1, \ldots, A_n\}$ iff $\Delta \mathrel{|\!\sim} A_1 \vee \ldots \vee A_n$.

In the above cases the Scott consequence relation $\Delta \mathrel{|\!\sim} \Gamma$, for Γ a complex structure, can be reduced to its Tarski part $\Delta \mathrel{|\!\sim} A$, where A a single formula. It is not clear whether we should insist on this property to hold in the general case.

To get an idea of our options, let our starting point be two structured Tarski type consequence relations $\mathrel{|\!\sim}_1$ and $\mathrel{|\!\sim}_2$ on the same language **L** with $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{M}_1$ and $\mathcal{M}_2$ the two structural languages satisfying the conditions of definition 7.2.1. We want to "extend" $\mathrel{|\!\sim}_1$ and $\mathrel{|\!\sim}_2$ to a Scott type consequence relation $\|\!\sim$, allowing for $\Delta\|\!\sim\Gamma$ where Δ is an $\mathcal{M}_1$ structured database and Γ is an $\mathcal{M}_2$ structured goal. Thus the allowable structures for $\|\!\sim$ are $\mathcal{M}_1$ for the data and $\mathcal{M}_2$ for goals. Clearly it must satisfy the rules of **Identity** and **Surgical Cut** in the following form:

$$A\|\!\sim A$$

and

$$\frac{\Delta[A]|\!|\!\!\sim\Gamma, \Delta'|\!|\!\!\sim A}{\Delta[\Delta']|\!|\!\!\sim\Gamma}.$$

We may wish to rename the Cut Rule as **Left hand side Surgical Cut**, to stress that cut is done on the left.

A **Right hand side Surgical Cut** would be

$$\frac{\Delta|\!|\!\!\sim\Delta[B] \quad B|\!|\!\!\sim\Gamma'}{\Delta|\!|\!\!\sim\Gamma[\Gamma']}$$

The question is, what other requirements do we have for $|\!|\!\!\sim$?

We obviously want to make use of both $|\!\!\sim_1$ and $|\!\!\sim_2$. The new consequence $|\!|\!\!\sim$ should extend them both in some sense. Since both consequence relations are general, the only structures they share for certain are the unit formulae structures A. Given Δ, Γ and A, we know what $\Delta \mid\!\sim_1 A$ is and what $\Gamma \mid\!\sim_2 A$ is, but no more.

The next two definitions tell us what is the general notion of a Scott type S-consequence relation and how to form a Scott type $|\!|\!\!\sim$ out of $|\!\!\sim_1$ and $|\!\!\sim_2$.

Definition 7.2.8 *Let* **L** *be a language for wffs and let $(\mathcal{M}_1, \mathcal{L}_1)$ and $(\mathcal{M}_2, \mathcal{L}_2)$ be two languages for data and for goal structures respectively. Note that $\mathcal{L}_1$ and $\mathcal{L}_2$ come with their own respective notions of structrual substitutions and structural additions $+_1$ and $+_2$ respectively.*

A relation $\Delta|\!|\!\!\sim\Gamma$ is said to be a (Scott type) S-consequence relation on the data and goal structures iff it satisfies the following two conditions:

Identity

$$A|\!|\!\!\sim A$$

Surgical Cut

$$\frac{\Delta[A]|\!|\!\!\sim\Gamma[B] \quad \Delta'|\!|\!\!\sim A \quad B|\!|\!\!\sim\Gamma'}{\Delta[\Delta']|\!|\!\!\sim\Gamma[\Gamma'].}$$

Note that some of the traditional formulations of the Cut Rule may not hold, for example

Internal Cut

$$\frac{\Delta + A \mid\!\sim \Gamma \quad \Delta \mid\!\sim A + \Gamma}{\Delta \mid\!\sim \Gamma}$$

Transitivity

$$\frac{\Delta \mid\!\sim A \quad \Delta + A \mid\!\sim B}{\Delta \mid\!\sim B}$$

may not hold.

Further note that **Identity** can hold only for single formulas. I do not know whether the rule of identity formulated for single formulas A would imply the rule of identity for all common structures Δ.

Definition 7.2.9 *Let $\mid\!\sim_1$ and $\mid\!\sim_2$ be two Tarski type S-consequence relations on structured databases with structures $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively, based on the same language $\mathbf{L}$, as in definition 7.2.1. Assume that $\mid\!\sim_1$ and $\mid\!\sim_2$ are* compatible, *namely they satisfy for all A, B.*

$$A \mid\!\sim_1 B \textit{ iff } B \mid\!\sim_2 A$$

Define $\mid\!\sim_{1,2}$ to be the following Scott type S-consequence relation, called the Minimal Amalgmation *of $\mid\!\sim_1$ and $\mid\!\sim_2$:*

1. *The data structures of $\mid\!\sim_{1,2}$ are those of $\mathcal{M}_1$. The goal structures of $\mid\!\sim_{1,2}$ are those of $\mathcal{M}_2$*
2. *Let $\Delta \mid\!\sim_{1,2} \Gamma$ hold iff (definition) for some wff A we have that both $\Delta \mid\!\sim_1 A$ and $\Gamma \mid\!\sim_2 A$ hold.*

Lemma 7.2.10 *Let $\mid\!\sim_1$ and $\mid\!\sim_2$ be two compatible Tarski type consequence relations. Let $\|\!\sim$ be their minimal amalgamation. Then $\|\!\sim$ is a Scott type consequence relation.*

Proof. It is clear that **Identity** holds. We check the rule of **Surgical Cut**.

Assume

$$\Delta[A] \|\!\sim \Gamma[B], \Delta' \|\!\sim A \text{ and } B \|\!\sim \Gamma',$$

We show

$$\Delta[\Delta'] \|\!\sim \Gamma[\Gamma'].$$

By definition there exist C_1, C_2, C_3 such that

$$\Delta[A] \mid\!\sim_1 C_1, \Gamma[B] \mid\!\sim_2 C_1, \Delta' \mid\!\sim_1 C_2, A \mid\!\sim_2 C_2, B \mid\!\sim_1 C_3, \Gamma' \mid\!\sim_2 C_3$$

all hold. From compatibility we get

$$C_2 \mid\!\sim_1 A \text{ and } C_3 \mid\!\sim_2 B.$$

By **Surgical Cut** we get

$$\Delta' \mid\!\sim_1 A \text{ and } \Gamma' \mid\!\sim_2 B.$$

Hence by **Surgical Cut** again we get

$$\Delta[\Delta'] \mid\!\sim_1 C_1 \text{ and } \Gamma[\Gamma'] \mid\!\sim_2 C_1.$$

Hence by definition we get

$$\Delta[\Delta' \|\!\sim \Gamma[\Gamma'].$$

■

Example 7.2.11 *Let $\Delta \vdash_H A$ be the classical (for $H = C$) or intuitionistic (for $H = I$) consequence relation, for the data structures of single formulas. Let $\mid\!\sim_{H^0}$ be the consequence relation of example 7.1.4(1). Form $\mid\!\sim_{H,H^0}$ as in the preceding definition. We show that what we get is classical (or intuitionistic) Scott type relation $\Vdash_H$ for sets as structures on both sides of the turnstile.*

We have $\Delta \mid\!\sim_{H,H^0} \Gamma$ iff for some $A, \Delta \vdash_H A$ and $\Gamma \mid\!\sim_{H^0} A$. But $\Gamma \mid\!\sim_{H^0} A$ iff $A \vdash_H \bigvee \Gamma$. Hence $\Delta \mid\!\sim_{H,H^0} \Gamma$ iff for some A, $\Delta \vdash_H A$ and $A \vdash_H \bigvee \Gamma$ and in classical or intuitionistic logic we can continue, iff $\bigwedge \Delta \vdash_H \bigvee \Gamma$, because we can take $A = \bigvee \Gamma$.

Example 7.2.12 *Let $\mathbf{L}$ be the language of intutionistic implication $\to$ and conjunciton $\wedge$. Let $\Delta \vdash_I A$ be the intuitionistic consequence relation for this language, with the goal structure of single formulas on the right. Let $\mid\!\sim_{H^1}$ be the consequence relation for $\mathbf{L}$, defined in example 7.1.4(2).*

Then $\mid\!\sim_{I,H^1}$ is the amalgamated consequence relation satisfying

$$\Delta \vdash_{I,H^1} \Gamma \textit{ iff for some } A \in \Gamma \textit{ we have } \Delta \vdash_I A.$$

In my book [Gabbay, 1981, Ch. 1] it was shown that $\mid\!\sim_{I,H^1}$ is the consequence relation arising from the Beth Semantics *for intuitionsitic logic. It was further shown that there are two Scott type consequene relations which agree (extend) the Tarski type $\Delta \mid\!\sim_I A$, for A a single wff. The maximal one, arising from the Kripke semantics, defined as $\Delta \vdash_I A_1 \vee \ldots \vee A_n$, and the minimal one, defined as in this example.*

Definition 7.2.13 *Let* $\mathbf{L}$ *be a language and let* $(\mathcal{M}_1, \mathcal{L}_1)$ *and* $(\mathcal{M}_2, \mathcal{L}_2)$ *be structures for the data and goal respectively. Let* $\mid\!\sim_i$ $i = 1, 2$ *be Tarski type* S*-consequence relations on the data structures* $\mathcal{M}_i$ *respectively with the goal structure being single wffs and let* $\|\!\sim$ *be a Scott type* S*-consequence on both data structures* $\mathcal{M}_1$ *and goal structures* $\mathcal{M}_2$*.*

(a) *We say that* $\|\!\sim$ agrees *with* $\mid\!\sim_1$ *in the data (respectively* $\|\!\sim$ agrees *with* $\mid\!\sim_2$ *in the goal) iff for all* Δ, Γ, A*, (1) holds (respectively (2) holds).*

1. $\Delta \|\!\sim A$ *iff* $\Delta \mid\!\sim_1 A$

2. $A \|\!\sim \Gamma$ *iff* $\Gamma \mid\!\sim_2 A$*.*

(b) *We say that* $\|\!\sim$ *agrees with* $(\mid\!\sim_1, \mid\!\sim_2)$ *iff* $\|\!\sim$ *agrees with* $\mid\!\sim_1$ *in the data and with* $\mid\!\sim_2$ *in the goal.*

Theorem 7.2.14 *Let* $\mid\!\sim_1$ *and* $\mid\!\sim_2$ *be two compatible Tarski type* S*-consequence relations, ie satifying for all* $A, B, A \mid\!\sim_1 B$ *iff* $B \mid\!\sim_2 A$*. Then there exist two Scott-type consequence relations, a maximal* $\|\!\sim^+$ *and a minimal* $\|\!\sim^-$ *which agree with* $(\mid\!\sim_1, \mid\!\sim_2)$*. In other words, for any* $\|\!\sim$ *which agrees with* $(\mid\!\sim_1, \mid\!\sim_2)$ *we have*

1. *for all* Δ, Γ
   $\Delta \|\!\sim^- \Gamma$ *implies* $\Delta \|\!\sim \Gamma$

2. *For all* Δ, Γ
   $\Delta \|\!\sim \Gamma$ *implies* $\Delta \|\!\sim^+ \Gamma$

Proof. Let $\|\!\sim^-$ be the minimal amalgamation of $\mid\!\sim_1$ and $\mid\!\sim_2$ of definition 7.2.9.

We show that $\|\!\sim^-$ agrees with $(\mid\!\sim_1, \mid\!\sim_2)$. Assume $\Delta \|\!\sim^- A$, then by definition for some B, $\Delta \mid\!\sim_1 B$ and $A \mid\!\sim_1 B$. Hence by compatibility $B \mid\!\sim_2 A$ and by the Cut Rule, $\Delta \mid\!\sim_1 A$.

Similarly assume $A \|\!\sim^- \Gamma$. Then for some B $A \mid\!\sim_1 B$ and $\Gamma \mid\!\sim_2 B$. By compatibility $B \mid\!\sim_2 A$ and by Cut, $\Gamma \mid\!\sim_2 A$.

We now show minimality. Assume that $\|\!\sim$ agrees with $(\mid\!\sim_1, \mid\!\sim_2)$. Assume that $\Delta \|\!\sim^- \Gamma$, we show that $\Delta \|\!\sim \Gamma$.

From our assumption we have that for some A

$$\Delta \mid\!\sim_1 A \text{ and } \Gamma \mid\!\sim_2 A$$

Hence from agreement $\Delta \|\!\sim A$ and $A \|\!\sim \Gamma$ and hence by Cut, $\Delta \|\!\sim \Gamma$.

We now have to prove the existence of $\|\!\sim^+$. To this end we show that for any finite set $\{\|\!\sim_i\}$ of consequence relations which agree with $\mid\!\sim_1$ and $\mid\!\sim_2$ one can define a consequence relation $\|\!\sim$ agreeing with $(\mid\!\sim_1, \mid\!\sim_2)$, with the property that for all i $\|\!\sim_i \subseteq \|\!\sim$.

An application of Zorn's Lemma will now yield the existence of $\|\!\sim^+$.

Let $\|\!\sim_i$ be give for $i = 1, \ldots, m$. We define the desired $\|\!\sim$. Define $\Delta \|\!\sim \Gamma$ to hold iff there exists a sequence of pairs $(\Delta_n, \Gamma_n), n = 1, \ldots, k$ such that $\Delta_k = \Delta, \Gamma_k = \Gamma$ and each element (Δ_n, Γ_n) in the seqeunce satisfies one of the following conditions:

1. For some $\|\!\sim_i, \Delta_n \|\!\sim_i \Gamma_n$ (ie (Δ_n, Γ_n) is an "axiom").

2. There exist earlier n_1, n_2, n_3 in the sequence such that:

   - $(\Delta_{n_1}, \Gamma_{n_1}) = (\Delta', A), A$ a wff.
   - $(\Delta_{n_2}, \Gamma_{n_2}) = (B, \Gamma'), B$ a wff.
   - $(\Delta_{n_3}, \Gamma_{n_3})$ has the form $(\Delta_{n_3}[A], \Gamma_{n_3}[B])$
     and $(\Delta_n, \Gamma_n) = (\Delta_{n_3}[A/\Delta'], \Gamma_{n_3}[B/\Gamma'])$.

Intuitively $\|\!\sim$ is essentially the sequent system with $\Delta \|\!\sim_i \Gamma$ as axioms, for $i = 1, \ldots, m$ and where the surgical cut is the inference rule.

It is clear from the definition of $\|\!\sim$ that **Identity** and **Surgical Cut** are satisfied and that $\|\!\sim_i \subseteq \|\!\sim$.

We now have to show that $\|\!\sim$ agrees with $(\mid\!\sim_1, \mid\!\sim_2)$. ie we have to show that for any Δ, Γ, A (1) and (2) hold:

1. $\Delta \Vert\!\!\sim A$ iff $\Delta \mid\!\sim_1 A$
2. $A \Vert\!\!\sim \Gamma$ iff $\Gamma \mid\!\sim_2 A$

We prove this by induction on the length of the sequence defining $\Vert\!\!\sim$, and on the assumption that $\Vert\!\!\sim_i, i = 1, \dots, m$ all agree with $(\mid\!\sim_1, \mid\!\sim_2)$.

For sequences of length 1 the claim follows form the assumption.

Consider a sequence of length k which justifies $\Delta \Vert\!\!\sim A$. Then we have the following situation from earlier members of the sequence:

- $\Delta' \Vert\!\!\sim A'$ and hence by the induction hypothesis $\Delta' \mid\!\sim_1 A'$
- $B \Vert\!\!\sim A$ and hence by the induction hypothesis $A \mid\!\sim_2 B$.
- $\Delta[A'] \Vert\!\!\sim B$ and hence $\Delta(A') \mid\!\sim_1 B$

We also know that $\Delta[\Delta'] \Vert\!\!\sim A$ and we want to show that it also agrees with $(\mid\!\sim_1, \mid\!\sim_2)$, ie that $\Delta[\Delta'] \mid\!\sim_1 A$. From compatibility we get $B \mid\!\sim_1 A$ and from Cut we get $\Delta[\Delta'] \mid\!\sim_1 A$.

Similarly consider a seqeunce of length k which justifies $A \Vert\!\!\sim \Gamma$. This means that earlier elements in the sequence are

- $A' \Vert\!\!\sim \Gamma'$ hence $\Gamma' \mid\!\sim_2 A'$
- $A \Vert\!\!\sim B$ hence $A \mid\!\sim_1 B$
- $B \Vert\!\!\sim \Gamma[A']$ and hence $\Gamma[A'] \mid\!\sim_2 B$

We also know that $A \Vert\!\!\sim \Gamma[\Gamma']$ and we want to show that $\Gamma[[\Gamma'] \mid\!\sim_1 A$. By compatibility $B \mid\!\sim_2 A$ and by Cut, $\Gamma[\Gamma'] \mid\!\sim_2 A$. ■

Example 7.2.15 (Symmetric Amalgamation) *Given a Scott type S-consequence relation $\mid\!\sim$, the two Tarski consequence relations derived from it, namely*

$$\Delta \mid\!\sim_1 A \textit{ iff } \Delta \mid\!\sim A$$
$$\Gamma \mid\!\sim_2 A \textit{ iff } A \mid\!\sim \Gamma$$

may not be identical or isomorphic, even in the case where Δ and Γ are based on the same data structures. We say that the Scott type $\mid\!\sim$ is a symmetrical amalgamation of the Tarski type $\mid\!\sim_0$ iff the two derived consequence relations $\mid\!\sim_1$ and $\mid\!\sim_2$ are the same and equal to $\mid\!\sim_0$. For a given $\mid\!\sim_0$, it is always possible to construct the symmetrical amalgamation. We cannot take the naive identity $\mid\!\sim_1 = \mid\!\sim_2 = \mid\!\sim_0$ because then the definition of amalgamation will yield
$\Delta \mid\!\sim \Gamma$ iff for some $A, \Delta \mid\!\sim_1 A$ and $\Gamma \mid\!\sim_2 A$ iff for some $A, \Delta \mid\!\sim_0 A$ and $\Gamma \mid\!\sim_0 A$.
If we let A = truth we get $\Delta \mid\!\sim \Gamma$ for all Δ, Γ.

What is needed is an isomorphism function $*$ such that

$$\Gamma \mid\!\sim_2 A \textit{ iff } \Gamma^* \mid\!\sim_1 A^*.$$

We thus get

$$\Delta \mid\!\sim \Gamma \textit{ iff for some } A, \Delta \mid\!\sim_1 A \textit{ and } \Gamma^* \mid\!\sim_1 A^*$$

now since we also have

$$A \mid\!\sim_1 B \textit{ iff } B \mid\!\sim_2 A$$

we get that $*$ must satisfy

$$A \mid\!\sim_1 B \textit{ iff } B^* \mid\!\sim_1 A^*.$$

In case of classical logic the mapping $*$ is negation

$$A \vdash B \textit{ iff } \neg B \vdash \neg A.$$

This notion is developed in the section on structural connectives.

The above examples have shown how to extend S-consequence relations to ones with a more complex goal structure by minimal amalgamation. When the original consequence relation $\mid\!\sim$ satisfies the Deduction Theorem, it is possible to extend $\mid\!\sim$ by making use of this fact. The Deduction Theorem is a left to right shift operator (studied in a later section). Thus by using shift operators we can give meaning to $\Delta \mid\!\sim \Gamma$ for Γ more complex, in terms of shifting Γ to the left and reducing the problem to a known one.

Thus for example one may write $\Delta \mid\!\sim \Gamma$ iff some boolean combination $\Delta_i \mid\!\sim \Gamma_i$ holds, where each Γ_i has less elements than Γ.

Classical logic can be viewed to be a special case of reducing $\{A_i\} \vdash_C \{B_j, C\}$ to $\{A_i, \neg C\} \vdash_C \{B_j\}$.

The Intuitionistic consequence, (see 2.2.2), cannot be viewed in this manner. We have no means of shifting wffs from the right hand side to the left hand side.

Let $\mid\!\sim'$ be a consequence relation with $\mathcal{M}'_1$ and $\mathcal{M}'_2$ being the structures for the data and goals respectively. Let $\mid\!\sim''$ be another consequence relation with structures $\mathcal{M}''_1$ and $\mathcal{M}''_2$ respectively. We want to define the composition $\mid\!\sim^*$ of $\mid\!\sim'$ and $\mid\!\sim''$. The data structures (goal structures) of $\mathcal{M}^*_1(\mathcal{M}^*_2)$ are obtained from $\mathcal{M}'_1$ and $\mathcal{M}''_1$ (respectively $\mathcal{M}'_2$ and $\mathcal{M}''_2$) by the repeated substitution of one data strucrure inside the other.

If the structures in $\mathcal{M}'_1$ are lists of points $(x_1, \ldots, x_n)$ and the strucures in $\mathcal{M}''_1$ are sets then the structures in $\mathcal{M}^*_1$ are hereditary lists of non-empty sets of points $(X_1, \ldots, X_n)$. If $\mathcal{M}'_1$ contains multisets of points then $\mathcal{M}^*_1$ is a hereditary set of multisets of non-empty sets of points.

Definition 7.2.16 (Composition of two S-consequence relations) *Let $\mid\!\sim'$ and $\mid\!\sim''$ be two consequence relations with structures $(\mathcal{M}'_1, \mathcal{M}'_2)$ and $(\mathcal{M}''_1, \mathcal{M}''_2)$ respectively. We assume further that we have available a notion of substitution of a τ' structure inside a τ'' structure and vice versae as needed below. We define the composition $\mid\!\sim^*$ of $\mid\!\sim'$ and $\mid\!\sim''$ as follows:*

1. *The structures of $\mathcal{M}^*_i$ of $\mid\!\sim^*, i = 1, 2$ are defined by induction.*
   (a) *Let $\mathcal{M}^*_i(0) = \mathcal{M}'_i \cup \mathcal{M}''_i$.*
   (b) *Assume $\mathcal{M}^*_i(k)$, for $k < n$ are all defined. Let $\tau \in \mathcal{M}^*_i(k)$ for some $k < n$ and $\tau'(x)$ (respectively $\tau''(x)$) is in $\mathcal{M}'_i$ (respectively $\mathcal{M}''_i$). Then $\tau'(\tau)$ (resp $\tau''(\tau)$) is in $\mathcal{M}^*_i(n)$.*
   (c) *Let $\mathcal{M}^*_i = \bigcup_n \mathcal{M}^*_i(n)$.*
   Note: *It is easy to show that if $\tau_1(x) \in \mathcal{M}^*_i$ and $\tau_2 \in \mathcal{M}^*_i$ then $\tau_1(\tau_2) \in \mathcal{M}^*_i$.*

2. *Let $\Delta = (\tau_1, \delta_1)$ and $\Gamma = (\tau_2, \delta_2)$ be two databses for $\mid\!\sim^*$.*

   We can assume by definition that for some $\alpha_1, \beta_1, \alpha_2, \beta_2, \varepsilon, \eta$:

$$\begin{aligned}
&\tau_1 = \beta_1(\alpha_1) \textit{ hence } \Delta = \Delta_0(\varepsilon)\\
&\tau_2 = \beta_2(\alpha_2) \textit{ hence } \Gamma = \Gamma_0(\eta), \textit{ where}\\
&\Delta_0(x) = (\beta_1(x), \delta_1), \Gamma_0(x) = (\beta_2(x), \delta_2)\\
&\epsilon_0 = (\alpha_1, \delta_1), \eta_0 = (\alpha_2, \delta_2).
\end{aligned}$$

   where β_1, β_2 are either in $\mathcal{M}'_1$ or in $\mathcal{M}''_i$ for $i = 1, 2$ respectively.

   Note that δ_i do not give a value to x.

   We say that $\Delta \mid\!\sim^* \Gamma$ iff for some C, D

$$\begin{aligned}
&\varepsilon = (\alpha_1, \delta_1) \mid\!\sim^* C\\
&\eta = (\alpha_2, \delta_2) \mid\!\sim^* D \textit{ and}\\
&\Delta_0[C] \mid\!\sim^* \Gamma_0[D]
\end{aligned}$$

Let us summarise our current view:

- A consequence relation is defined by stating a family of data structures and defining the notions of substitution on the data structure, and of adding (combining) data structures.

- Minimal conditions for a relation to be a consequence relation are those of **Identity** and **Surgical Cut**.

- The basic structured consequence relation can be extended either in the direction of non-monotonic consequence relations or in the direction of Resource Logics or both.

- We need structured semantics with structured family of worlds with structured preference relation to be an adequate semantics for **S** in the spirit of [Kraus *et al.*, 1990].

7.3 Structural connectives

Let $\mid\!\sim$ be a Scott type consequence relation on a language **L**. The consequence relation involves structures for the data and structures for goal. The purpose of this section is to show that given these structures, there are some natural connectives, called structural connectives, which one may wish to add into the language **L**, with porperties dictated by the structure. In fact, the entire consequence relation may be in some cases (eg linear logic, see next section) no more and no less than a reflection (via the structural connectives) of the properties of the data structures involved.

Let $\mid\!\sim$ be any Scott type consequence relation. Let $\mathcal{M}_1$ be structures for the data and let $\mathcal{M}_2$ be the structures for the goals. Let $+_1$ be structural addition for the data and $+_2$ for the goal. We can thus write

$$\Delta +_1 A \mid\!\sim B +_2 \Gamma$$

which may or may not hold.

We assume nothing special about $\mid\!\sim$, only that it satisfies the mimimal properties of **Identity** and **Surgical Cut**. We do not assume any special connectives in the language **L** of $\mid\!\sim$, and we aim to introduce several structural connectives.

It might be useful to think in terms of two examples, namely concatenation logic, where the structures are lists, and linear logic, where the structures are multisets. Both cases would have $\rightarrow$ in the language **L**. To be even more specific, think of the Tarski type consequence relations to be the relations $\vdash_{\mathbf{CL}}$ and $\vdash_{\mathbf{LL}}$ for **CL** and for **LL** of definitions 10.1.1 and 9.2.5 and 9.2.6 respectively, and assume that $\mid\!\sim$ is some Scott extension of one of them which agrees with it, the goal structures being the same as the data structures. Note further that there may be more than one such $\mid\!\sim$ agreeing with the Tarski type $\vdash_{\mathbf{CL}}$ or $\vdash_{\mathbf{LL}}$ of definitions 10.1.1 and 9.2.5 and 9.2.6, and that in the sequel we may choose one such $\mid\!\sim$ with special symmetry conditions.

Let us begin with a general Scott type consequence $\mid\!\sim$.

The first thing to bear in mind is that in the general case, the structural operators "$+_1$" and "$+_2$"which come with $\mid\!\sim$ are in the metalevel, on the structures of $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively, and not in the langauge **L** of $\mid\!\sim$ itself. For the special case of **CL**, "$+_i$" is concatenation and for the case of **LL**, "$+_i$" is multiset union. Let us now add another metalevel operator, a sort of notational shift operator "o", having the following properties:

- $\Delta +_1 A^o \mid\!\sim \Gamma$ means the same as $\Delta \mid\!\sim A +_2 \Gamma$
  and

- $\Delta \mid\!\sim B^o +_2 \Gamma$ means the same as $\Delta +_1 B \mid\!\sim \Gamma$
  In fact,
  $\Delta +_1 A^o \mid\!\sim B^o +_2 \Gamma$ should be the same as $\Delta +_1 B \mid\!\sim A +_2 \Gamma$.

Thus the operator "o" in "A^o" is used to indicate, in the metalevel, where the formula A should be. It has the status of a metalevel annotation, indicating that although the formula A appears in one place in the structure, it should be in another place. Thus $\Delta +_1 A^o \mid\!\sim \Gamma$ really says through the annotation "o" on A, that A is really on the right hand side, namely $\Delta \vdash A +_2 \Gamma$. For general structures the annotation may be problematic. If $\Delta[x]$ indicates that x stands somewhere in the structure Δ, then we know what $\Delta[A]$ means. We also have to say what $\Delta[A^o]$ means. We have to specify the following:

1. Since A^o in $\Delta[A^o]$ means that although A^o shows up in $\Delta[x]$ at the place x (ie we have $\Delta[A^o/x]$), A should really be somewhere else, we need to specify where A should be.

2. If A^o is not at place x in $\Delta[x]$, then there is *nothing* at x. Hence our notion of substitution should also allow for empty substitution $\Delta[\varnothing/x]$, which is really deletion.

If both Δ and Γ are multisets, as in the case of linear logic, then we have no such problem, otherwise a precise definition is required. Even in the case of **CL**, this is not simple, and requires care.

In **CL**, the data structures are lists $(A_1, \ldots, A_n)$. Of course $(A_1, \ldots, A_n, B)$ is generally not the same data structure as $(B, A_1, \ldots, A_n)$. We can mark B as B^o to mean that B should be *shifted* to the other end of the list. Thus

$$\begin{aligned}(A_1, \ldots, A_n, B^o) &= \text{def } (B, A_1, \ldots, A_n)\\ (B^o, A_1, \ldots, A_n) &= \text{def } (A_1, \ldots, A_n, B).\end{aligned}$$

Note that $\Delta_1 +_2 \Delta_2$ (and similarly $\Gamma_1 +_1 \Gamma_2$) is not meaningful, since $+_2$ (resp $+_1$) is a goal (data) structural addition and cannot be applied to the data (goal).

So far we have introduced several meta operations. We now want to put them in the object level. We have seen in connection with the Deduction Theorem in the previous section that it is possible to take a meta operation and add a connective for it in the object language. We now do just that for "$+_1$", "$+_2$" and "o" of the metalevel. We denote the object level connectives by "$\oplus_1$", "$\oplus_2$" and "$\odot$", respectively.

Our purpose is to study their obvious properties. The connection between $\{A+B\}$ and $\{A\oplus B\}$ is that they must be declaratively identical. This means that $A \oplus B$ should behave essentially like $A + B$ in all $\mid\!\sim$ contexts. We thus have the following rules:

Identity

$A +_1 B \mid\!\sim A \oplus_1 B$

$A \oplus_2 B \mid\!\sim A +_2 B$

The other direction of the identity of $A \oplus_i B$ with $A +_i B$, cannot be written directly but can be characterised by a Cut rule:

Surgical Cut

$$\frac{\Delta_1 +_1 (A +_1 B) +_1 \Delta_2 \mid\!\sim \Gamma_1 +_2 (C +_2 D) +_2 \Gamma_2}{\Delta_1 +_1 (A \oplus_1 B) +_1 \Delta_2 \mid\!\sim \Gamma_1 +_2 (C \oplus_2 D) +_2 \Gamma_2}$$

Example 7.3.1 *In the case of* **CL**, *if we add the structural connective $\oplus$ to the language, we get the following system* **CL**$(\oplus)$, *with the following two rules:*

$$(A, B) \mid\!\sim A \oplus B$$

which we get from **Identity**, *and we should also have the following from the* **Cut Rule**

$$\frac{(A_1, \ldots, A_n, A, B, B_1, \ldots, B_m) \mid\!\sim C}{(A_1, \ldots, A_n, A \oplus B, B_1, \ldots, B_m) \mid\!\sim C}$$

These we call **Structural Rules** *for the structural connective $\oplus$, because they arise from the fact that $\oplus$ is no more than an object level reflection of the structured database operation $+$.*

In classical logic the data structures are sets and conjunction "$\wedge$" serves as the structural connective.

In case of **CL** *we also have a* **Right hand side Deduction Theorem**, *which, when combined with the structural connective $\oplus$ will yield rules like*

$$A \oplus B \mid\!\sim C \textit{ iff } A \mid\!\sim B \to C$$

and

$$\varnothing \mid\!\sim A \to (B \to A \oplus B).$$

We now have to check which rules for $\odot$ follow from the properties of shift. We know that $\Delta +_1 A^\odot \mid\!\sim B^\odot +_2 \Gamma$ should hold iff $\Delta +_1 B \mid\!\sim A +_2 \Gamma$. We immediately get

- $A^{\odot\odot} \mid\!\sim A$
- $A \mid\!\sim A^{\odot\odot}$

Consider

$$\Delta +_1 C^o +_1 D^o \mid\!\sim \Gamma$$

This is the same as

$$\Delta \mid\!\sim (C +_2 D) +_2 \Gamma$$

which is the same as

$$\Delta +_1 (C +_2 D)^o \mid\!\sim \Gamma$$

We can thus have that $(C +_2 D)^o$ must be the same as $C^o +_1 D^o$. Similarly $(C +_1 D)^o$ must behave the same as $C^o +_2 D^o$. We thus get

- $(C \oplus_2 D)^\odot \mid\!\sim C^\odot \oplus_1 D^\odot$
- $(C \oplus_1 D)^\odot \mid\!\sim C^\odot \oplus_2 D^\odot$

We still have a problem of what to do (what meaning we can prove or have) to $A \oplus_1 B$ when it appears on the right hand side and $C \oplus_2 D$ when it appears on the left hand side. The typical expression is:

$$C \oplus_2 D \mid\!\sim A \oplus_1 B$$

clearly it should be the same as

$$(A \oplus_1 B)^\odot \mid\!\sim (C \oplus_2 D)^\odot$$

which is the same as

$$A^\odot \oplus_2 B^\odot \mid\!\sim C^\odot \oplus_1 D^\odot$$

which does not help, as again we get the wrong "$\oplus$" on the wrong side.

In general, there is nothing to be done. However, if we have the object level "$\rightarrow_r$" with the Deduction Theorem holding, a few more equivalences can be derived.

Consider

$$A +_1 B \mid\!\sim C$$

this holds iff

$$A \mid\!\sim B \rightarrow_r C$$

but also iff

$$A \mid\!\sim B^o +_2 C$$

Hence
$B \rightarrow_r C$ is equivalent to $B^o \oplus_2 C$.

The difference between the shift and the Deduction theorem is that the connective "$\rightarrow_r$" is in the object language **L**, while our "$+_2$" and "o" are in the metalevel. $\Delta \mid\!\sim B^o +_2 C$ is just another way of writing, in the metalevel, $\Delta +_1 B \mid\!\sim C$.

However, if we do have the object level connectives, "$\oplus_1$", and "$\oplus_2$", we can write down equivalences. We therefore get that

- $A \oplus_2 B$ is equivalent to $A^\odot \rightarrow_r B$
- $A \oplus_1 B$ is equivalent to $(A \rightarrow_r B^\odot)^\odot$.

Thus "$\oplus_1$", "$\oplus_2$" are reducible, in a system containing "$\to_r$", to the connectives "$\to_r$" and "$\odot$".

Let us now pause and summarise what we have learnt so far.

Suppose we start with a consequence relation $\mid\!\sim$ with $+_1$ for data and $+_2$ for goals. Suppose we introduce a shift operation which shifts from the right hand side of the data to the left hand side of the goals and vice versae, and suppose that we have an object level connectives $\odot$ for the shift and $\to_r$ satisfying right hand side Deduction Theorem. Then we can define in the object level the two operators "$\oplus_1$" and "$\oplus_2$" using "$\to_r$" and the object level "$\odot$" as follows:

$$\begin{aligned} &A \oplus_2 B = \text{def } A^{\odot} \to_r B \\ &A \oplus_1 B = \text{def } (A \to_r B^{\odot})^{\odot}. \end{aligned}$$

The above discussion showed how to add a structural connective $\oplus$ to reflect the notion $+$, which is part of the defintion of any consequence relation. Any specific consequence relation , allows for a family $\mathcal{M}$ of structures. Any of these structures $\tau \in \mathcal{M}$ can be reflected in the object level by a special connective $\textcircled{\tau}$. The way it is done is described in the next definition.

Definition 7.3.2 (Left Structural Connectives and Left Structural Rules) *Let $\mathcal{M}$ be a classs of structures and let $\mid\!\sim$ be a consequence relation structured on $\mathcal{M}$, as defined in 3.1. Let τ be a fixed structure in $\mathcal{M}$ and assume τ has n elements $a_1, \dots, a_n$. Indicate this fact by writing $\tau(a_1, \dots, a_n)$. Enrich the language of $\mid\!\sim$ with an additional n-place connective $\textcircled{\tau}(A_1, \dots, A_n)$ with the following* Left Structural Rules for $\textcircled{\tau}$.

$\textcircled{\tau}$**Identity**

$$(\tau(a_1, \dots, a_n), \delta) \mid\!\sim \textcircled{\tau}(\delta(a_1), \dots, \delta(a_n)).$$

$\textcircled{\tau}$**Surgical Cut**

Let $\Delta_1 = (\tau_1, \delta_1)$ and $\Delta = (\tau, \delta)$. Let $x \in \tau_1$ and let $\delta_1(x) = B$. Let $\Delta_2 = (\tau_2, \delta_2)$ be the result of structural substitution of Δ in Δ_1, ie $\Delta_2 = \Delta_1[B/\Delta]$.

This means that $\tau_2 = \lambda x \tau_1(x)[\tau]$.

Consider $\Delta'_1 = (\tau_1(x), \delta'_1)$ which is just like Δ_1 except that $\Delta'_1(x) = \textcircled{\tau}(\delta(a_1), \dots, \delta(a_n))$. Thus $\Delta'_1 = \Delta_1[B/\textcircled{\tau}(\delta(a_1), \dots, \delta(a_n))$.

The **Surgical Cut Rule** *is therefore*

$$\frac{\Delta_1[B/\Delta] \mid\!\sim A}{\Delta_1[B/\textcircled{\tau}(\delta(a_1), \dots, \delta(a_n)] \mid\!\sim A}$$

or if formulated with a slight abuse of notation:

$$\frac{\lambda x \tau_1(x)[\tau] \mid\!\sim A}{\lambda x \tau_1(x)[\textcircled{\tau}(\delta(a_1), \dots, \delta(a_n))] \mid\!\sim A}$$

The cut rule really ensures the other direction of the identity, by forcing $\textcircled{\tau}$ to behave properly.

We are now ready to explain what a **Structural Shift Connective** is and what **Structural Shift Rules** are.

Definition 7.3.3 (Structural Shift Connectives) *Let $\mid\!\sim$ be a Scott type S-consequence relation and let $\mathcal{M}_i$ and $\mathcal{L}_i$ be as in definition 7.2.8. Let φ_i and be the wff of $\mathcal{L}_i$ each identifying a unique point in each $\tau_i \in \mathcal{M}_i$. Further assume that for each $\tau(a_1, \dots, a_n)$ there exist a unique $\tau_2(a_1, \dots, a_n, z)$ such that $\tau = \tau_2 \| \varphi_2$. Assume $\tau' = \tau_1 \| \varphi_1$. We can write $\tau' = f_1(\tau_1)$, and $\tau_2 = f_2(\tau)$ where f_i are function symbols for the functional dependence which exists.*

Let $\odot^{\varphi_1}_{\varphi_2}$ be a new unary connective to the language of $\mid\!\sim$. Enrich $\mid\!\sim$ with this connective and the following structural shift rule:

(φ_1, φ_2) **Structural Shift Rule**
Let $\Delta_1 = (\tau_1(x), \delta_1)$ be the data structure with x the point identified in τ_1 by φ_1, and let $\Delta_2 = (\tau_2(y), \delta_2)$ be the goal structure with y the point identified by φ_2. Assume that $\delta_1(x)$ and $\delta_2(y)$ satisfy that either $\delta_1(x) = \odot^{\varphi_1}_{\varphi_2} A$ and $\delta_2(y) = A$ or $\delta_1(x) = A$ and $\delta_2(y) = \odot^{\varphi_1}_{\varphi_2} A$. For each of the above the following holds:

$$\Delta_1 \mathrel{|\!\!\sim} \Delta_2 \parallel \varphi_2 \text{ iff } \Delta_1 \parallel \varphi_1 \mathrel{|\!\!\sim} \Delta_2$$

The functions induced by φ_1, φ_2 define the two places, and the shift rule tells us that $\odot^{\varphi_1}_{\varphi_2} A$ shifts A from one place to the other place.

It is worth noting that with a bit of abuse of notation, that the shift operation can be written as

1. $\Delta_1[\odot A] \mathrel{|\!\!\sim} \Delta_2[\varnothing]$ iff $\Delta_1[\varnothing] \mathrel{|\!\!\sim} \Delta_2[A]$
2. $\Delta_1[A] \mathrel{|\!\!\sim} \Delta_2[\varnothing]$ iff $\Delta_1[\varnothing] \mathrel{|\!\!\sim} \Delta_2[\odot A]$

where $\varnothing$ is the empty set and its substitution at point x in $\Delta[x]$ means deletion.

Example 7.3.4 *Consider* **CL** *with its list structures. Let φ_1 define the last element of a list and let φ_2 define the second element of a list. Then if $\odot$ denotes $\odot^{\varphi_1}_{\varphi_2}$ we get to the following two shift rules:*

1. $(A_1, \ldots, A_m, \odot B) \mathrel{|\!\!\sim} (C_1 \ldots C_n)$
   iff $(A_1 \ldots, A_m) \mathrel{|\!\!\sim} (C_1, B, C_2, \ldots, C_n)$
2. $(A_1, \ldots, A_n) \mathrel{|\!\!\sim} (C_1, \odot B, C_2, \ldots, C_n)$
   iff $(A_1, \ldots, A_n, B) \mathrel{|\!\!\sim} (C_1, \ldots, C_n)$.

The shift operation suggest a new structural connective $\textcircled{!}A$. Its meaning can be motivated by the following observation. Given $\Delta_1(x)$ and $\Delta_2(y)$ the shift rule, stated intuitively, says for example that:

$$\begin{array}{l} \Delta_1[A] \mathrel{|\!\!\sim} \Delta_2(\varnothing) \\ \text{iff } \Delta_1[\varnothing] \mathrel{|\!\!\sim} \Delta_2[\odot A]. \end{array}$$

We understand the operation as shifting A from Δ_1 into Δ_2. After the shift, A is no longer present in Δ_1 but is recorded in Δ_2. That is why we write $\Delta_1[\varnothing]$. If we do not take A out Δ_1, we get a different rule namely

$$\begin{array}{l} \Delta_1[\textcircled{!}A] \mathrel{|\!\!\sim} \Delta_2[\varnothing] \\ \text{iff } \Delta_1[\textcircled{!}A] \mathrel{|\!\!\sim} \Delta_2[\odot A]. \end{array}$$

This means that although A is supposed to "shift" to the right hand side, it stays while "duplicating" itself. The connective "$\textcircled{!}$" indicates the duplications property.

Example 7.3.5 *Consider the previous example where we had*

$$\begin{array}{l} (A_1 \ldots A_m, B) \mathrel{|\!\!\sim} (C_1, \ldots, C_n) \\ \textit{iff } (A_1, \ldots, A_n) \mathrel{|\!\!\sim} (C_1^{\odot}, B, C_2, \ldots, C_n) \end{array}$$

If the formula B does not delete itslef when shifting, but duplicates itself we indicate this by writing $\textcircled{!}B$ instead of B.

$$\begin{array}{l} (A_1, \ldots, A_n \textcircled{!}B) \mathrel{|\!\!\sim} (C_1, \ldots, C_n) \\ \textit{iff } (A_1, \ldots, A_m, \textcircled{!}B) \mathrel{|\!\!\sim} (C_1 \odot B, C_2, \ldots, C_n) \end{array}$$

Thus $\textcircled{!}B$ at the last place on the left simply indicates an infinite number of B's. Of course we can shift the entire $\textcircled{!}B$ to the right and get

$$\begin{array}{l} (A_1, \ldots, A_m, \textcircled{!}B) \mathrel{|\!\!\sim} (C_1, \ldots, C_n) \\ \textit{iff } (A_1, \ldots, A_m) \mathrel{|\!\!\sim} (C_1 \odot \textcircled{!}B, C_2, \ldots, C_n) \end{array}$$

Some properties of $\textcircled{!}$ and $\odot$ can be obtained from the above meaning, especially if other connectives are presnt, for example

$$\textcircled{!}B + B \equiv \textcircled{!}B$$

Another structural connective is the proof-net like connective, which identifies copies. Consider again $\Delta_1[x]$ and $\Delta_2[y]$. Using a shift connective, we can either put $x = A$ and $y = \varnothing$ or put $x = \varnothing$ and $y = \odot A$. In either case we get A on the left ($\Delta_1[A]$) and nothing on the right.

Another possibility is to put A on the right and nothing on the left, ie $\Delta_1[\varnothing], \Delta_2[A]$. We can write this possibility also as $\Delta_1[\odot A], \Delta_2[\varnothing]$.

What we should *not* write is

$$\Delta_1[\odot A] \mathrel{|\!\sim} \Delta_2[A]$$

because here we are duplicating A twice on the right. We could of course add an additional marker say $\dagger\dagger$, to indicate that these two copies of A should really be *one* copy. So we can write:

$$\Delta_1[\odot \dagger_A \dagger] \mathrel{|\!\sim} \Delta_2[\dagger_A\dagger]$$

where $\dagger\dagger$ marks the fact that there is only one copy. Thus $(\odot\dagger_A\dagger, \dagger_A\dagger)$ is not the same as $(\dagger_A\dagger, \odot\dagger_A\dagger)$ becuase the first corresponds to A on thge right and the second corresponds to A on the left.

Example 7.3.6 *To continue the previous example,*

$$(A_1, \ldots, A_n, \odot \dagger_A \dagger) \mathrel{|\!\sim} (C_1, \dagger_A\dagger, C_2, \ldots, C_n)$$

is the same as

$$(A_1, \ldots, A_m) \mathrel{|\!\sim} C_1, A, C_2, \ldots, C_n)$$

while

$$(A_1, \ldots, A_m, \dagger_A\dagger) \mathrel{|\!\sim} (C_1, \odot \dagger_A \dagger, C_2, \ldots, C_m)$$

is the same as

$$(A_1, \ldots, A_m, A) \mathrel{|\!\sim} (C_1, \ldots, C_n).^1$$

We now turn out attention to the so called "additive" connectives. As before, we start with a general consequence relation $\mathrel{|\!\sim}$.

Consider now the case where we have that both $\Delta \mathrel{|\!\sim} \Gamma[A_1]$ and $\Delta \mathrel{|\!\sim} \Gamma[A_2]$ hold. We can abbreviate the above situation by writing in the metalevel $\Delta \mathrel{|\!\sim} \lambda x\Gamma(x)[A_1 \wedge A_2]$ or just $\Delta \mathrel{|\!\sim} \Gamma[A_1 \wedge A_2]$ where "$\wedge$" is a metaleval symbol, just like "$+_1$", "$+_2$" and "o". Similarly if we have $\Delta[A_1] \mathrel{|\!\sim} \Gamma$ and $\Delta[A_2] \mathrel{|\!\sim} \Gamma$ we can abbreviate it using "$\vee$" and write $\Delta[A_1 \vee A_2] \mathrel{|\!\sim} \Gamma$. Note that "$\wedge$" and "$\vee$" only abbreviate other notation and are not operations on the data. We could modify the data structures if we want to accommodate the respective operations but that is not necessary.

Note that $\Delta[A_1 \wedge A_2] \mathrel{|\!\sim} \Gamma$ is not meaningful, as we have not said what it means when "$\wedge$" appears on the left. Similarly for $\Delta \mathrel{|\!\sim} \Gamma[A_1 \vee A_2]$.

The following hold by definition

$$\Delta \mathrel{|\!\sim} A \wedge B \text{ iff } \Delta \mathrel{|\!\sim} A \text{ and } \Delta \mathrel{|\!\sim} B$$

$$A \vee B \mathrel{|\!\sim} \Gamma \text{ iff } A \mathrel{|\!\sim} \Gamma \text{ and } B \mathrel{|\!\sim} \Gamma.$$

Example 7.3.7 *Suppose we add the structural connectives $\bigcirc\!\!\!\!\vee$ and $\bigcirc\!\!\!\!\wedge$ to the language; what are the obvious properties it must satisfy? In the general case we cannot say much beyond the obvious, so let us see what we get in the special case of linear logic. Here the Deduction Theorem comes into play:*

---

^{1}The perceptive reader may ask why we are doing all of this shifting. Instead of writing plainly $(C, A) \mathrel{|\!\sim} B$, for example, we code it as $C \mathrel{|\!\sim} (B, \odot A)$, putting A on the right and then *marking* it by $\odot$ to indicate that it should really be on the left. Then to confuse matters even further, we write A on the left anyway and mark it so that the duplication is cancelled, ie $(C, \dagger_A \dagger \mathrel{|\!\sim} (B, \odot \dagger_A \dagger)$.

Why do we need this geometry of illusion? The answer can be found in understanding the annotation as indicating *resource*. A structured database presents formulas to be used in the deduction in a way compatible with their structured layout. For linear logic the layout is a multiset and so each formula is to be used exactly once. In the course of the proof the formulas may be scattered about and/or duplicated. We need annotations to keep track of what is happening. This is why these connectives are useful.

1. *$\Delta \mathrel{|\!\sim} A \to B \owedge C$ iff $\Delta, A \mathrel{|\!\sim} B \owedge C$*
   iff $\Delta, A \mathrel{|\!\sim} B$ and $\Delta, A \mathrel{|\!\sim} C$
   iff $\Delta \mathrel{|\!\sim} A \to B$ and $\Delta \mathrel{|\!\sim} A \to C$
   iff $\Delta \mathrel{|\!\sim} (A \to B) \owedge (A \to C)$.

2. *$\Delta \mathrel{|\!\sim} A \ovee B \to C$*
   iff $\Delta, A \ovee B \mathrel{|\!\sim} C$
   iff $\Delta, A \mathrel{|\!\sim} C$ and $\Delta, B \mathrel{|\!\sim} C$
   iff $\Delta \mathrel{|\!\sim} A \to C$ and $\Delta \mathrel{|\!\sim} B \to C$
   iff $\Delta \mathrel{|\!\sim} (A \to C) \owedge (B \to C)$.

3. *From the Identity Rule we get, since $A \ovee B \mathrel{|\!\sim} A \ovee B$ that*
   $A \mathrel{|\!\sim} A \ovee B$
   $B \mathrel{|\!\sim} A \ovee B$

4. *Similarly since by the Identity Rule $A \owedge B \mathrel{|\!\sim} A \owedge B$ we get that:*
   $A \owedge B \mathrel{|\!\sim} A$
   $A \owedge B \mathrel{|\!\sim} B$

5. *Since $A \mathrel{|\!\sim} A \ovee B$ we get $(A \ovee B)^{\odot} \mathrel{|\!\sim} A^{\odot}$ and similarly $(A \ovee B)^{\odot} \mathrel{|\!\sim} B^{\odot}$. Hence we get $(A \ovee B)^{\odot} \mathrel{|\!\sim} A^{\odot} \wedge B^{\odot}$.*

6. *Since $A \owedge B \mathrel{|\!\sim} A$ we get $A^{\odot} \mathrel{|\!\sim} (A \owedge B)^{\odot}$. Similarly $B^{\odot} \mathrel{|\!\sim} (A \owedge B)^{\odot}$ and therefore $A^{\odot} \ovee B^{\odot} \mathrel{|\!\sim} (A \owedge B)^{\odot}$.*

7. *Since $(A \to C) \owedge (B \to C) \mathrel{|\!\sim} A \to C$ and $A \oplus_1 (A \to C) \mathrel{|\!\sim} C$ we get by* **Surgical Cut** *that $A \oplus_1 ((A \to C) \owedge (B \to C)) \mathrel{|\!\sim} C$ and similarly $B \oplus_1 ((A \to C) \owedge (B \to C)) \mathrel{|\!\sim} C$ hence $(A \ovee B) \oplus_1 ((A \to C) \owedge (B \to C)) \mathrel{|\!\sim} C$ and therefore*
   $(A \to C) \owedge (B \to C) \mathrel{|\!\sim} A \ovee B \to C$.

So far our study of structural connectives and shift connectives used no special properties of $\mathrel{|\!\sim}$ beyond the existence of $\to_r$ in the object language. We shall see that we are going to need some symmetry assumptions on $\mathrel{|\!\sim}$. Suppose we start with the Tarski $\mathrel{|\!\sim}_{\mathbf{LL}}$ of definition 10.1.1 and 9.2.5 and 9.2.6 and assume that $\mathrel{|\!\sim}_1$ and $\mathrel{|\!\sim}_2$ are both Scott type and are compatible with $\mathrel{|\!\sim}_{\mathbf{LL}}$. Everything we have said and done so far would apply to both $\mathrel{|\!\sim}_1$ and $\mathrel{|\!\sim}_2$ equally. We can thus add the structural connective "$\odot$" and get all the equivalences mentioned earlier to hold. However there might be a difference between the case of $\mathrel{|\!\sim}_1$ and that of $\mathrel{|\!\sim}_2$, depending on their properties.

Example 7.3.8 *Consider the consequence relation $\mathrel{|\!\sim}_{\mathbf{LL}^1}$ defined by $\Delta \mathrel{|\!\sim}_{\mathbf{LL}^1} A$ iff for some $B \in \Delta$, $A \mathrel{|\!\sim}_{\mathbf{LL}} B$, where Δ is a multiset.*

This is a consequence relation as can be shown easily..

Let $\mathrel{|\!\sim}_1$ be the amalgamation of $\mathrel{|\!\sim}_{\mathbf{LL}}$ and $\mathrel{|\!\sim}_{\mathbf{LL}^1}$. Clearly $\Delta \mathrel{|\!\sim}_1 \Gamma$ iff for some $A \in \Gamma, \Delta \mathrel{|\!\sim}_{\mathbf{LL}} A$ holds. This means that $\mathrel{|\!\sim}_1$ is monotonic in Γ.

Let us add to the $\mathrel{|\!\sim}_1$ the shift "$\odot$" and consider the following: $A \mathrel{|\!\sim}_1 B, C$ iff $A \mathrel{|\!\sim}_1 B^{\odot} \to C$ iff $A, B^{\odot} \mathrel{|\!\sim}_1 C$. If we continue the chain using the fact that $\mathrel{|\!\sim}_1$ agrees with $\mathrel{|\!\sim}_{\mathbf{LL}}$ we get that the above holds iff $A, B^{\odot} \mathrel{|\!\sim}_{\mathbf{LL}} C$. We can now get that $\mathrel{|\!\sim}_{\mathbf{LL}}$ is monotonic in $B^{\odot}$, ie $\Delta \mathrel{|\!\sim}_{\mathbf{LL}} C$ implies $\Delta, B^{\odot} \mathrel{|\!\sim}_{\mathbf{LL}} C$. This is so because $\Delta \mathrel{|\!\sim}_{\mathbf{LL}} C$ implies $\Delta \mathrel{|\!\sim}_1 B, C$ and hence $\Delta, B^{\odot} \mathrel{|\!\sim}_{\mathbf{LL}} C$ from the equivalence chain. Although $\mathrel{|\!\sim}_{\mathbf{LL}}$ itself is not monotonic, there is nothing wrong with monotonicity in $B^{\odot}$, because we know that $\Delta, B^{\odot} \mathrel{|\!\sim}_{\mathbf{LL}} C$ really means $\Delta \mathrel{|\!\sim}_1 B, C$. However, it does force us syntactically to note which wff is of the form B and which is of the form $B^{\odot}$, and excludes writing $B^{\odot\odot}$ instead of B. Clearly this is not desirable. We therefore need to assume further properties on the $\mathrel{|\!\sim}$ which agrees with $\mathrel{|\!\sim}_{\mathbf{LL}}$, namely the property that it is symmetrical, *namely $A \mathrel{|\!\sim} \Gamma$ iff $\Gamma^{\odot} \mathrel{|\!\sim}_{\mathbf{LL}} A^{\odot}$.*

In terms of definition 7.2.9 we want $\mathrel{|\!\sim}$ to be the amalgamation of $\mathrel{|\!\sim}_{\mathbf{LL}}$ with itself.

We now proceed with a series of definitions leading to the notion of self amalgamation.

Definition 7.3.9 (The dual of a consequence relation) *Let $\mid\!\sim$ be a consequence relation in language $\mathbf{L}$. Consider a dual language $\mathbf{L}^*$ defined as follows:*

1. *The atoms of $\mathbf{L}^*$ are all atoms of the form q^*, where q is an atom of $\mathbf{L}^*$.*
2. *The connectives of $\mathbf{L}^*$ are all connectives of the form $\sharp^*$, where $\sharp$ is a connective of $\mathbf{L}$.*
   Let $*$ be a maping from $\mathbf{L}$ to $\mathbf{L}^*$ defined by
3. *$(q)^* = q^*$, for q atomic*
4. *$(\sharp(A_1, \ldots, A_n))^* = \sharp^*(A_1^*, \ldots, A_n^*)$, for a connective $\sharp$.*
5. *Define $\Delta^* \mid\!\sim^* A^*$ iff def $\Delta \mid\!\sim A$.*

Examples 7.3.10 *In classical logic let $\wedge$ and $\vee$ be duals and let $A^* = \neg A$.*

7.4 A Case study: What is the logic of linear logic

A lot has been said about linear logic. From our point of view, the system is very simple; it is the logic based on the data structures of multisets, satisfying the Deduction Theorem and fortified with additional structural and shift connectives.

It is our purpose in this section to present linear logic from this point of view.[2]

We start with the data structures of multisets for both the data and goal. Consider the language with $\to$ only and consider the smallest Scott type consequence relation $\mid\!\sim$ satisfying the deduction theorem for $\to$.

We know that the smallest Tarski type consequence relation $\mid\!\sim_1$ for multisets satisfying the deduction theorem does characterise linear logic. Thus it is clear that

$$\Delta \mid\!\sim_1 A \text{ iff } \Delta \mid\!\sim \{A\}.$$

What is not clear is the nature of the consequence $\Delta \mid\!\sim_2 A$ defined by

$$\Delta \mid\!\sim_2 A \text{ iff } \{A\} \mid\!\sim \Delta.$$

We shall address this problem later.

Given $\mid\!\sim$, let us add to the language the structural connectives $\oplus_1, \oplus_2$ and the shift connective $\odot$, as we did in the previous section, when we were discussing such connectives for an arbitrary consequence relation. We get for our case that the following holds, where $X \equiv Y$ means both $X \mid\!\sim Y$ and $Y \mid\!\sim X$:

1. $A^{\odot\odot} \equiv A$
2. $A \oplus_2 B \equiv A^\odot \to B$
3. $A \oplus_1 B \equiv (A \to B^\odot)^\odot$
4. $(A \oplus_1 B)^\odot \equiv A^\odot \oplus_2 B^\odot$
5. $(A \oplus_2 B)^\odot \equiv A^\odot \oplus_1 B^\odot$

We can also add the "of course" connective $\textcircled{!}$ and proof net markers. Let us agree that $\dagger_n\dagger$ is the corresponding connective on the right. We thus have:

[2]The reader is cautioned that the phrase "from our point of view" is important. Intuitionistic logic, from our point of view, is essentially the smallest monotonic consequence relation with the Deduction Theorem for $\to$, with sets as structures and the structural and shift connectives. However, intuitionistic logic arose in a context of far greater importance and motivation in the historical development of logic. Similarly, linear logic has its own historical context and to understand its role one has to go back as early as 1976 [Girard, 1976]. For these reasons, Avron [Avron, 1988b] should be viewed in context. I cannot make up my mind at the moment whether the excitement, generated in computer science circles by intuitionistic and by linear logic, is indeed justified. A clear case should be made for each logic whether it is best suited for its proposed computer science applications and the indiscriminate application of these logics may obscure their intrinsic logical value.

6. $A \oplus_1 \textcircled{!}_1 \equiv \textcircled{!}_1 A$

7. $A \oplus_2 \textcircled{!}_2 A \equiv \textcircled{!}_2 A$

8. $A \oplus_1 B^{\dagger_n \dagger} \mid\!\sim C \oplus_2 (B^{\odot})^{\dagger_n \dagger}$
   is the same as
   $A \oplus_1 B \mid\!\sim C$

9. $A \oplus_1 B^{\odot \dagger_n \dagger} \mid\!\sim C \oplus_2 B^{\dagger_n \dagger}$
   is the same as
   $A \oplus_1 B^{\odot} \mid\!\sim C$.

The moral of the story is that the properties of all of these connectives are determined by the structure and their geometric meaning.

The additives $\textcircled{\wedge}$and $\textcircled{\vee}$can be added to linear logic as in the example 7.3.7. When added to the language alongside $\oplus_1$ and $\oplus_2$, one can form formulas with arbitrary nestings of $\textcircled{\wedge}, \textcircled{\vee}, \oplus_1, \oplus_2$ within themselves. Given the meaning of these connectives, the corresponding meatalevel structures are the composition of two consequence relations, one for the additives over sets and one for the multiplicatives over multisets, as in 7.2.16.

In the case of the additives of linear logic, the structures τ^* were multisets of sets (not hereditary). These can be regarded as a family of structures $\{\tau\}$ of $\mathcal{M}'$ (ie multisets) obtained by choosing all possible points from the sets of points in τ^*. For example if τ^* is $(\{a, b\}, \{c, d\}, \{e\})$ then the following set of τ's are associted.

$(a, c, e), (a, d, e), (b, c, e)$ and (b, d, e).

We write formally $\tau \in \tau^*$ to indicate the connection. We can define the function δ^* to give values (wff) to each point. Thus databases and goals become lists of sets of formulas or multisets of sets of formulas.

For example, if Δ^* is $(\{A, B\}, \{C, D\}, \{E\})$ we get the following associated databases

$$(A, C, E), (A, D, E), (B, C, E) \text{ and } (B, D, E).$$

We write again $\Delta \in \Delta^*$ to indicate the association.

Our goal is to define a consequence relation $\Delta^* \mid\!\sim^* \Gamma^*$ by stipulating: $\Delta^* \mid\!\sim^* \Gamma^*$ iff def. for all $\Delta \in \Delta^*, \Gamma \in \Gamma^*, \Delta \mid\!\sim \Gamma$.

We need to show that we get an S-consequence relation and for that we need to define the *Surgical Cut Rule*; we need to define substitution of one structure in another. We refer back to a general definition of the *composition* of two S-consequence relations.

We now present a theorem about the Hilbert formulation of linear logic with $\to$ and $\neg$ which shows that $\odot$ can be taken as $\neg$ and that $\neg$ can be mapped into the implicational fragment.

Theorem 7.4.1 *Let* $\mathbf{LL}(\neg)$ *be the extension of* $\mathbf{LL}$ *of 10.1.1 and 9.2.5 and 9.2.6 with the unary symbol* $\neg$ *and the following axioms:*

$$\begin{array}{l} \neg\neg A \leftrightarrow A \\ (\neg(A \to \neg B) \to C) \leftrightarrow (A \to (B \to C)) \\ (A \to B) \to (\neg B \to \neg A) \end{array}$$

Let $\bot$ *be an arbitrary atom of* $\mathbf{LL}$. *Let* $\varphi^{\bot}(A)$ *be a translation from* $\mathbf{LL}(\neg)$ *into* $\mathbf{LL}(\bot)$ *defined as follows:*

$$\begin{array}{l} \varphi^{\bot}(q) = (q \to \bot) \to \bot, q \textit{ atomic} \\ \varphi^{\bot}(A \to B) = \varphi^{\bot}(A) \to \varphi^{\bot}(B) \\ \varphi^{\bot}(\neg A) = \varphi^{\bot}(A) \to \bot \end{array}$$

Then the following holds:

$$\mathbf{LL}(\neg) \vdash A \textit{ iff } \mathbf{LL} \vdash \varphi^{\bot}(A).$$

Proof. In Section 10.7. ■

Remark 7.4.2 *The previous theorem yields the following:*

1. *Semantics for* **LL**$(\neg)$*, through the semantics for* **LL**.
2. *It shows that* $\neg, \oplus_1$ *and* $\oplus_2$ *(ie all the structural connectives of linear logic) are already definable from the impliction* $\rightarrow$*, via the "internal" translation* $\varphi^{\perp}$*, for a fixed atom* $\perp$.
3. *It shows that* $\neg$ *can also be regarded as a negation, and not merely as a shift operator.*

Example 7.4.3 (Cut Free Formulation of Linear Logic) *In this section, we started with a consequence relation* $\vdash_{\mathbf{LL}}$ *for multisets as data, added the obvious structural and shift connectives together with the additives, defined the symmetric amalgamation and got a conseqeunce relation* $\mid\!\sim$*. Certain properties of* $\mid\!\sim$ *were listed in the previous example. It is possible to list enough properties of* $\mid\!\sim$ *to enable one to derive the* Cut Rule*. When this is done one gets a cut free formulation of the sytem* $\mid\!\sim$*. Although we have been referring to* $\mid\!\sim$ *as linear logic, we have not proved that* $\mid\!\sim$ *is indeed the system known in the literature as linear logic. To show that, all we need to do is to take a known formulation of linear logic, say* $\vdash_1$*, see e.g. [Girard, 1987], and prove that all* $\vdash_1$ *rules are valid in* $\mid\!\sim$*. This will show that* $\vdash_1 \subseteq \mid\!\sim$*. If we prove the Cut Elimination for* $\vdash_1$ *this will show that* $\mid\!\sim \subseteq \vdash_1$*. Thus essentially Cut Elimination for* $\vdash_1$ *establishes that* $\mid\!\sim$ *is indeed linear logic.*

7.5 Case study: What is negation

The notion of negation is basic to any formal or informal logical system. When any such system is presented to us, it is presented either as a system without negation or as a system with some form of negation. In both cases we are supposed to know intuitively whether there is no negation in the system or whether the form of negation presented in the system is indeed as claimed Yet the notion of what is negation in a formal system is not clear. When we see a unary connecive $*A$, (A a wff) together with some other axioms for some additional connectives, how can we tell whether $*A$ is indeed a form of negation of A? Are there some axioms which $*$ must satisfy in order to qualify A as a negation?

Let L be any propositional logical system and let $\vdash_L$ be its provability relation. We do not specify how L is presented to us, it can be as a Hilbert style system with axioms and rules, or as a ntural deduction system or by semantics etc. The main point is that we have the provability relation of L, namely:

$$A_1, \ldots, A_k \vdash_L B$$

between a finite set Δ of A_j and a signle B satisfying the following three conditiosn:

1. $\Delta \vdash A$ for A in Δ (reflexivity).
2. If $\Delta \vdash A$ and $\Delta \supseteq \Delta$ then $\Delta' \vdash A$ (monotonicity).
3. If $\Delta \vdash A$ and $\Delta \cup \{A\} \vdash B$ then $\Delta \vdash B$. (Transivity or cut).

In fact any relation $\vdash$ on wffs satisfying 1, 2 and 3 can be regarded as a logical system.

Our strategy is to give several candidate definitions of what should constitute a negation in a system and test them against rou intuitions and against known examples. The examples we look at are as follows:

Example 7.5.1 *(a) Let us consider the following system in a language with* $\neg$ *and* $\rightarrow$.

1. $A \rightarrow (B \rightarrow A)$
2. $[A \rightarrow (B \rightarrow C)] \rightarrow (A \rightarrow B) \rightarrow (A \rightarrow C)]$
3. $\neg\neg A \rightarrow A$
4. $A \rightarrow \neg\neg A$

Rules

5. *Modus Ponens*

$$\frac{A, A \to B}{B}$$

Question: *Is $\neg$ a form of negation in this system?*
(b) Let us make life more difficult by adding more axioms to our system. To get the idea of what to add, first we need disjunctions and conjunctions (the sytem has only $\neg$ and $\to$). So let us see what can be taken as disjunction.

$$(a \to b) \to b = \neg(a \to b) \vee b = (a \wedge \neg b) \vee b = a \vee b$$

so let $a \vee b =$ def.$(a \to b) \to b$. This is in fact a well known definition of $\vee$ in terms of $\to$.
Also let $a \wedge b =$ def.$\neg(\neg a \vee \neg b) = \neg((\neg a \to \neg b) \to \neg b)$.
Take the following rule:

6.

$$\frac{\vdash A \to B}{\vdash \neg B \to \neg A}$$

and the further axioms:

7. $\neg(A \vee B) \leftrightarrow \neg A \wedge \neg B$

8. $\neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$

9. $((A \to \neg\neg B) \to A) \to A$. *(This axiom says $(A \to B) \vee A$).*

Is $\neg$ a negation in this system?
(c) We can ask further:
If we also add the axiom

10. $A \to (\neg A \to B)$.

does this make $\neg$ a negation? (we shall see that Answer is no for cases (a) and (b) and yes for case (c).)

It seems from the above example that this question does not have an immediate simple answer. Remember that we cannot just write a set of axioms for negation and say that anything satifying these axioms is a negation. f we write too many axioms we may only get classical negation, and even that is not guaranteed because maybe we do not know how the negation axioms are supposed to interact with other connectives e.g. $\to$.

Let us look at more examples:

Example 7.5.2 *Consider the system* **L**3 *below of Wajsberg. It axiomatises the 3 valued logic of Łukasiewicz with $\to$ and $\neg$.*
Axioms:

W1 $A \to (B \to A)$

W2 $(A \to B) \to ((B \to C) \to (A \to C))$

W3 $(\neg B \to \neg A) \to (A \to B)$

W4 $((A \to \neg A) \to A) \to A$

The inference rule is modus ponens.
Question: *Can one determine on the basis of $\vdash_{\mathbf{L}3}$ whether $\neg A$ is a negation in* **L**3*?*

Example 7.5.3 *Consider a third system denoted by* **LS3**. *Its language contains an additonal connective Θ besides $\neg$ and $\to$. It is obtained from* **L3** *by adding the axioms:*

$\Theta 1$ $\Theta A \to \neg\Theta A$

$\Theta 2$ $\neg\Theta A \to \Theta A$.

Questions:

1. *Is $\neg$ a negation in this system? Is Θ a negation?*
2. *If $\neg$ is considered a negation in* **L3**, *does it have to be considered a negation in the extension* **LS3***?*

Armed with this stock of examples we now move to a formulation and some possible solutions of our problem.
Problem P:
Given a consequence relation $\vdash$ (satisfying 1, 2, and 3), and a connective $*A$ in the language of $\vdash$, are there any criteria on the relationsihip between $\vdash$ and $*$ which will agree with our intuitions regarding the qustion of when * is to be considered a form of negation?

Carnap and Church discussed whether a syntactical characterisation of negation was possible. Carnap thought it was possible, Church thought not.

A basic intution regarding the meaning of $\neg A$ is that A *does not hold* or A is *not wanted* or A is *excluded* or even A is *not confirmed*. Thus if **L** is a system with a candiate $*A$ for negation, we cannot hope to have A, $*A$ considtent together. This leads us to our first attempt in answering problem P.

We must specify a set Θ of unwanted wffs. the wffs of Θ are not allowed to be true. This is normal and natural for any database. For example we do not want two lecturers to be assigned to the same classroom at the same time. In a formal system **L**, one can take $\Theta =$ *falsity* or one can take Θ to be certain conjunctions of atoms etc. So to get negation into a system we must have a set of unwanted wffs Θ. This set may be different for diffent negations. The connective (*1) may be a negation because of $\Theta 1$ and (*2) may be a negation because of $\Theta 2$ and so on.

We are thus led to the following defintion:

Definition 7.5.4 (Negation as syntactical inconsistency: 1st attempt) *Let $\vdash$ be the provability relation of a system and $*A$ be a connective. We say * is a form of negation if there is a fixed non-empty set of wffs Θ^* which is not provably equivalent to the set of all wffs, such that for any $\cdot :$ and any A the following holds:*

$$\Delta \vdash *A \textit{ iff } \exists y \in \Theta^*(\Delta, A \vdash y).$$

i.e. A is negated by Δ becasue A leads to some unwanted y in Θ^*.

Lemma 7.5.5 *Let $*$ be a negation in the logical system $\vdash$. Then the set $\{x : \varnothing \vdash *x\}$ is non-empty.*

Proof. Since * is a negation, let $q \in \Theta^*$, then $\varnothing \vdash^* q$, since $q \vdash q$.

The above is purely syntactic (in terms of $\vdash$) definition. So to check whether $*A$ of an axiom system is a negation, look for a Θ^* and try to prove the above equivalence. Note that the equivalence must hold for any Δ and A.

We may ask ourselves, how do we find a Θ^*? The answer is that if such a Θ exists, (i.e. * is a negation according to the above definition) then it follows from lemma 7.5.5 that Θ^* can be taken as:

$$\Theta^* = \{C \mid \varnothing \vdash^* C\}$$

where $\varnothing$ is the empty set. ■

Lemma 7.5.6 *Assume * is a negation with a Θ^* according to definition 7.5.4, then for any Δ and any A, (a) is equivalent to (b):*

a. $\Delta, A \vdash C$ *for some* C *such that* $\varnothing \vdash^* C$.

b. $\Delta, A \vdash B$ *for some* $B \in \Theta^*$.

Proof. Let $B \in \Theta^*$ then since $B \vdash B$ we get by definition 7.5.4 that $\varnothing \vdash^* B$. This shows that (b) implies (a).

Assume for some C such that $\varnothing \vdash^* C$, we have $\Delta, A \vdash C$. Since $\varnothing \vdash^* C$, we have that for some $B \in \Theta^*$,

$$C \vdash B.$$

By monotonicity of $\vdash$

$$\Delta, A, C \vdash B$$

and by cut using $\Delta, A \vdash C$ we get

$$\Delta, A \vdash B,$$

This proves the lemma. ■

We can thus modify definition 7.5.4 as follows:

Definition 7.5.7 (Negation as syntactical inconsistency: modified first attempt) *Let* $\vdash$ *be a logical system and* $\neg A$ *be a connective. We say that* $\neg$ *is a form of negation in* $\vdash$ *iff for any* Δ *and any* A *the following holds.*

$$\Delta \vdash \neg A \textit{ iff for some } C \textit{ such that } \varnothing \vdash \neg C \textit{ we have } \Delta, A \vdash C.$$

The above definition seems theoretically sound and acceptable. All we have to see now is whether it takes care of all the currently known and agreed upon negations. We will see later that further modifications are necessary. For this reason we continue to use Θ^* itslef and not $\{C \mid \varnothing \vdash^* C\}$. Note that Θ^* may contain wffs containing * itself. We do not nee dto exclude this possibility. In fact for classical logic we can take $\Theta^* = \{q_0 \wedge \neg q_0\}$ for some atom q_0 and we all know that in classical logic $\Delta \vdash \neg A$ iff $\Delta, A \vdash q_0 \wedge \neg q_0$ holds, and so classical negation is a negation. So is intuitionistic negation because the same equivalence holds.

According to 7.5.4, the $\neg$ defined in example 7.5.1 axioms 1-9 is not a negation. One can see this by taking the following interpreation and verifying that all axioms 1-9 of example 7.5.1 are valid. In this interpretation there are two worlds h and e (heaven for h and earth for e). $\neg A$ is *true* in one if A is *false* in the other. $\rightarrow$ is the usual truth functional implication. All axioms and rules are valid; i.e. we have $\neg A$ iff A is true in e and h under any assignment to the atoms. Now we can see that $\neg A$ is not a negation of A, since it just says that A is false in the other world. $A \wedge \neg A$ is not a negation of A, since it just says that A is false in the other world. $A \wedge \neg A$ can be consistent, as A could be true in this world (e.g. e) and false in the other world (e.g. h). the rule of definition 7.5.4 for negation does not apply here. If $\neg$ were a negation, then for some Θ, and for all Δ, A we would have:

$$\Delta \vdash A \Leftrightarrow \Delta, A \vdash y, \text{ for some } y \in \Theta.$$

In particular for any $y \in \Theta$ we get $\vdash \neg y$. Let p be atomic then since

$$\neg p \vdash \neg p$$

we get $\neg p, p \vdash y$ for some $y \in \Theta$, and therefore:

$$\neg p \wedge p \vdash y \text{ for some } y \in \Theta, \text{ and hence by definition } \vdash \neg(p \wedge \neg p).$$

Since p is an atom we cannot have the above since $p \wedge \neg p$ is consistent, i.e. since $\nvdash A \wedge \neg A$. Thus the $\neg$ above is not a negation.

If we add axiom 10 (of example 7.5.1) i.e. $A \rightarrow (\neg A \rightarrow B)$ we get $e = h$ and $\neg$ becomes classical negation. We can take $\Theta = (\neg q_0 \wedge q_0)$ and derive from the axioms that

$$\vdash \neg A \leftrightarrow (A \rightarrow (\neg q_0 \wedge q_0)).$$

In fact the above additional axiom says simply $\vdash \neg(A \wedge \neg A)$.

Let us check now whether $\neg$ in the system **L3** of example 7.5.2 is indeed a form of negation. This system axiomatises Łukasiewicz 3 valued logic. There are 3 truth values, 1 (*truth*), 1/2, and 0 (*falsity*). The truth tables for $\neg$ and $\rightarrow$ are as follows:

$$\neg x = 1 \text{ and } x \rightarrow y = \text{ Min } (1, 1 + y - x).$$

The idea of the definition for $x \rightarrow y$ is that if $x \leq y$ then $x \rightarrow y$ is true. (Like $0 \rightarrow 1$ in classical logic). If $x > y$ then $x - y$ is the measure of falsity of $x \rightarrow y$ and so the value of $x \rightarrow y$ is $1 - (x - y)$. $\neg x = 1 - x$ is just the mirror image of the truth value.

Conjunction $x \wedge y$ and disjunction $x \vee y$ have the definition below. They are definable from $\rightarrow$ by:

$$\begin{aligned} x \vee y = &\text{ def. } (x \rightarrow y) \rightarrow y = \text{ Min}(x, y). \\ x \wedge y = &\text{ def. } \neg(\neg x \vee \neg y) = \text{ Max } (x, y). \end{aligned}$$

there is no doubt that $\neg x$ is a form of negation in this system, because $\neg x = 1 - x$. The farther x is from the truth the nearer $\neg x$ is to the truth.

Let us write $A_1, \ldots, A_n \vDash B$ in this system to mean that under any assignment:

Min(value A_j) $\leq$ value B, and $\vDash B$ to mean that under any assignment value $B = 1$.

Notice that the relation $\vDash$ defined semantically above, fulfils the criteria for a logical system. The deduction theorem, however, is not valid for $\vDash$.

The Wajsberg axiom system is complete in the sense that the following holds:

$$A_1, \ldots, A_n \vDash B \text{ iff } \bigwedge A_j \rightarrow B$$

If we define $A_1, \ldots, A_n \vdash B$ to mean that $\vdash \bigwedge A_j \rightarrow B$ it then follows that $\vdash B$ iff value $B = 1$ under all assignment.

Our definition 7.5.4 of what a negationis should give us that $\neg$ is a negation. Suppose $\neg$ is indeed a negation according to definition 7.5.4. Then there exists a fixed Θ such that for any Δ and any A of the logic **L3**, we have:

$$\Delta \vdash \neg A \text{ iff } \Delta, A \vdash B(\text{ for some } B \in \Theta). \text{ Necessarily } \Theta \neq \varnothing.$$

Take any $B \in \Theta$ and $\Delta = \varnothing$ then $\vdash \neg B$ iff $B \vdash y$ for some $y \in \Theta$; but since $y = B \in \Theta$ and $B \vdash B$ we get $\vdash \neg B$ for all $B \in \Theta$.

One can verify by looking at the axioms the following lemma:

Lemma 7.5.8 *If $\vdash A$ then value $(A) = 1$ under all assignments.*

Proof. The above is true for the axioms and is preserved under modus ponens and substitution. Thus we conclude that for any $B \in \Theta$, value $B = 0$ under all assignments. Now consider an atom q, certainly

$$\neg q \vdash \neg q$$

hence for some $B \in \Theta$

$$\neg q, q \vdash B$$

hence under all assignments Min (value $\neg q$, value q) $\leq$ value B. In particular for any assignment h with $h(q) = 1/2$ we get $h(B) \geq 1/2$. This contradicts the previous conclusion that value $B = 0$ always. We therefore need to improve our definition 7.5.4 of negation. ■

Our basic idea in defining negation was that $A \vdash \neg B$ holds iff A, B togetehr lead to some undesirable result Θ.

$$\text{i.e. } A, B \vdash \Theta.$$

However the way the above is written is that A and B are 'combined' together via conjunction, i.e. $A \wedge B$. It is quite possible that A, BB can be combined together via a different connective e.g. some connective $\mathbf{C}(A, B)$. Thus $A \vdash \neg B$ holds if $\mathbf{C}(A, B) \vdash \Theta$. $\mathbf{C}$ is a connective which 'brings out' the effect A and B can have together. Of course $\mathbf{C}(x, y)$ is not an arbitrary connective. It must be monotonic and satisfy some obvious properites. $\mathbf{C}(x, y)$ must say more than just $x \wedge y$, and satisfy the conditions listed in definition 7.5.9 for it.

Definition 7.5.9 (Negation as a potential syntactic inconsistency: first attempt) *Let L be a system with a provability relation $\vdash$ and let $*$ be a unary connective of L. We say $*$ is a form of negation in L iff there exist a non empty set of wffs Θ which is not provably equivalent to the set of all wffs, and a binary connective $\mathbf{C}(x, y)$ s.t. the following holds for any D and A.*

$$D \vdash^* A \textit{ iff } \mathbf{C}(D, A) \vdash y \textit{ for some } y \in \Theta.$$

$\mathbf{C}$ *must satisfy the following: (*truth *is any provable such formula; such formulae exist if $*$ is a negation. See lemma 7.5.5.)*

1. $\mathbf{C}(x, y) \vdash x$
2. $\mathbf{C}(x, y) \vdash y$
3. $\mathbf{C}(\top, y) = \mathbf{C}(y, \top) = y$
4. $$\frac{x \vdash x'}{\mathbf{C}(x, y) \vdash \mathbf{C}(x', y)} \qquad \frac{y \vdash y'}{\mathbf{C}(x, y) \vdash \mathbf{C}(x, y')}$$

where $A = B$ abbreviates $A \vdash B$ and $B \vdash A$.

Remark 7.5.10
1. *We get from the above that (in case that a* falsity *can be defined in the logic): $\mathbf{C}(\perp, y) = \mathbf{C}(x, \perp) = \perp$.*
2. *Definition 7.5.9 was given D for a single formula, if L has conjunction then we can take $\Delta \vdash^* A$.*

For our negation in the system L3 let $\mathbf{C}(x, y) = \neg(x \to \neg y)$, and let $\Theta = \{\text{falsity}\} = \{\neg(y_0 \to y_0)\}$. Clearly, by the definition of $\vdash$, $x \vdash \neg y$ iff $\vdash x \to \neg y$ iff value $(x \to \neg y) = 1$ in all assignments, iff value $\neg(x \to \neg y) = 0$ in all assignments, iff $\neg(x \to \neg y \vdash$ falsity. The truth table for $\mathbf{C}(x, y) = \neg(x \to \neg y)$ is Max (0, value x + value $y - 1$). As can be seen, since the truth function of $\mathbf{C}(x, y)$ is $Max(0, x + y - 1)$ we get:

a. $\mathbf{C}(x, y) \leq x$

b. $\mathbf{C}(x, y) \leq y$

c. $\mathbf{C}(1, y) = y = \mathbf{C}(y, 2)$

d1. $x \leq x' \Rightarrow \mathbf{C}(x, y) \leq \mathbf{C}(x', y)$

d2. $y \leq y' \Rightarrow \mathbf{C}(x, y) \leq \mathbf{C}(x, y')$

These correspond to the conditions of definition 7.5.9 and hence $\neg$ in the 3 valued logic is a negation. In fact the above definitions of $\neg, \to$ and $\mathbf{C}(x, y)$ as $\neg(x \to \neg y)$ show that $\neg$ is a negation in all Lukasiewicz many valued logics.

Now that we have changed the definition of negation in a formal system we have to check whether the $\neg$ of example 7.5.1 is still not considered a negation. So assume that $\neg$ is a negation in the system of example 7.5.1, the system with axioms 1-9. Then for some Θ and $\mathbf{C}$ the condition of definition 7.5.9 holds, namely for all D, A.

$$D \vdash \neg A \text{ iff } \mathbf{C}(D, A) \vdash B \text{ for some } B \in \Theta$$

We shall show that

$$(\pounds) \qquad A \to \neg(x \to x) \vdash \neg A$$

using $\mathbf{C}$ and Θ, and this is impossible because in our two world model $(\pounds)$ says that if A is false in one world, A is false in the other world also. Thus if we prove $(\pounds)$ then this shows that no $\mathbf{C}$, Θ can exist and $\neg$ is not a form of negation.

We now proceed to prove (*£*):

$$\begin{array}{l}\text{Since } \mathbf{C}(y,z) \vdash y \wedge z \text{ we get}\\ \mathbf{C}(A \to \neg(x \to x), A) \vdash \neg(x \to x)\end{array}$$

Hence by definition of ¬

$$\mathbf{C}(\mathbf{C}(A \to \neg(x \to x), A), x \to x) \vdash B, \text{ for some } B \in \Theta.$$

Since $x \to x$ is *truth* and $\mathbf{C}(y, \textit{ truth }) = y$, we get

$$\mathbf{C}(A \to \neg(x \to x), A) \vdash B, \text{ for some } B \text{ in } \Theta$$

and hence by definition of ¬ we get:

$$(\textit{£}) \qquad A \to \neg(x \to x) \vdash \neg A$$

Example 7.5.11 (The system of relevant logic R) *Consider a language with → only and the following set of axioms and rules defining the system R →.*
Rule

$$\textit{modus ponens} \qquad \frac{\vdash A, \vdash A \to B}{\vdash B}$$

Axioms

R1. $A \to A$

R1. $(A \to B) \to ((C \to A) \to (C \to B))$

R3. $(A \to (B \to C)) \to (B \to (A \to C))$

R4. $(A \to (A \to B)) \to (A \to B)$

The above system was introduced by Church and Moh (1950, 1951). Church called it 'weak positive implicational calculus'. They proved the following deduction theorem for the system.

1. **Deduction Theorem for** $R \to$
   If there exists a proof of B from $A_1, \ldots, A_n$ in which all $A_1, \ldots, A_n$ are used in arriving at B then there exists a proof of $A_n \to B$ from $A_1, \ldots, A_{n-1}$ satisfying the same conditions. The above calls for the following definition of $\vdash_{R\to}$.

2. **Definition of** $\vdash_{R\to}$

$$\begin{array}{l}A_1, \ldots, A_n \vdash_{R\to} B \textit{ iff}\\ \vdash_{R\to} A_1 \to (A_2 \to \ldots \to (A_n \to B)\ldots)\end{array}$$

   One can see by axiom R3 that the above is independent of the order of $\{A_j\}$. The above system is identical with the implicational relevance logic of Anderson and Belnap. It satifies the conditions of a logical system. Negation ¬ is introduced into R → to obtain R(→, ¬) via the Ackermann negation axioms. These axioms are used to introduce negation not only into R → but also into all neighbouring systems.

Ackermann axioms for negation

AN1 $(A \to \neg B(\to (B \to \neg A)$

AN2 $(A \to \neg A) \to \neg A$

AN3 $\neg\neg A \to A$

The following can be proved

AN4 $A \to \neg\neg A$

AN5 $(A \to B) \to (\neg B \to \neg A)$

See [Anderson and Belnap, 1975, pp. 20–21, 107–109] for details.

The above definition of negation is indeed negation according to our definition 7.5.9 of negation. Meyer 1966 has shown that if we add to $R \to$ a symbol $\bot$ (*falsity*) with the additional axiom

R5 $((A \to \bot) \to \bot) \to A$

we get a system equivalent to $R(\to, \neg)$,with $\neg$ via the interpretation

1. $\neg A =$ def. $A \to \bot$.

The following must be proved.

2. $D \vdash \neg A$ iff $D, A \vdash \bot$
   i.e. $D \vdash A \to \bot$ iff $D, A \vdash \bot$
   or equivalently by definition of $\vdash_{R\to}$:

$$\vdash D \to (A \to \bot) \text{ iff } D \to (A \to \bot), \text{ which is correct.}$$

Remark 7.5.12 *Technically,if the system $R(\to, \neg)$ is formulated with $\neg$ and without $\bot$, can we find an $\bot$ such that $\neg A = A \to \bot$? In classical logic one can take $\bot = q_0 \wedge \neg q_0$ or if conjunction is not available, one takes $\bot = \neg(a_0 \to q_0)$ for some fixed q. We cannot do the same for $R(\to, \neg)$,because if we take $\bot = \neg(q_0 \to q_0)$ for some fixed atom q_0, we will not have anough axioms on $\to$ to be able to use $\bot$ as needed. We will have to add axiom R4 for this new $\bot = \neg(q_0 \to q_0)$ and then show that no new theorems can be proved for any wffs not containing q_0. Thus we see that definition 7.5.9 is not quite right in the sense that the system considered may be too weak to show that it has a negation. In other words a connective * may indeed be a negation in the system $\vdash$, but $\vdash$, may be too weak to prove the definition 7.5.9. In fact, a connective $\mathbf{C}(x, y)$ required by definition 7.5.9 may not be definable in the language of the system, but only in an extension. Intuitively if * is a negation in a conservative extension, then we can and should regard it a negation in the system itself. We are thus led to the following definition:*

Definition 7.5.13 (Negation as a potential syntactic inconsistency modified attempt) *1.*
Let $L1$ and $L2$ be logical systems such that the language of $L2$ extends the language of $L1$.

We say $L2$ is a conservative extension of $L1$ iff the following holds for any Δ, A in the language of $L1$.

$$\Delta \vdash_{L1} A \text{ iff } \Delta \vdash_{L2} A$$

*2. We say that * is a negation in $L1$ iff for some conservative extension $L2$ and some Θ and $\mathbf{C}$ in $L2$ satisfying the conditions of definition 7.5.9, we have that for any D, A of $L1$ the following holds:*

$$D \vdash_{L1} *A \text{ iff } \mathbf{C}(D, A) \vdash_{L2} B, \text{ for some } B \in \Theta$$

We have now to check whether this new definition of negation turns the connective $\neg$ of example 7.5.1 axioms 1-9 into a negation. (Recall that we found that $\neg$ is not a negation). The answer is no; $\neg$ is still not a negation. The reason is that for any conservative extension of the system in example 7.5.1 the two world interpretation (with the e world and the h world) is still valid. So the argument for showing that no $\mathbf{C}$ and Θ can make $\neg$ into a negation still goes through.

Example 7.5.14 *We now give another example illustrating the need for definition 7.5.13.*
Consider the language of classical propositional logic and its conseqeunce relation $\vdash$.
Let $\vdash_1$ be defined as

$$\Delta \vdash_1 \text{ iff } \Delta \neq \varnothing \text{ and } \Delta \vdash A.$$

$\vdash_1$ is a consequence relation. However, $\neg$ is not a negation in $\vdash_1$, according to definition 7.5.4, since for any non-empty Θ that we choose we would have to have for $B \in \Theta$ that $\varnothing \vdash_1 \neg B$ since certainly $B \vdash_1 B$ contrary to definition of $\vdash_1$. But this is counter intuitive since certainly $\Theta = \{q \wedge \neg q\}$ should be acceptable.

The example is certainly pathological and definition 7.5.13 handles it nicely. However, in our view a more satisfactory solution to this particular problem is to require the following additional property to be fulfilled by a consequence relation.

4. $\Delta \vdash A$ iff $\forall x(\Delta, x \vdash A)$. (Coherence).

We now investigate the possibility that there might be negations for which Θ depends on D. This is quite intuitive, since it says that what we do not want, Θ depends on the data, D, which we have. In fact it turns out that we *cannot* have a notion of Θ dependent on D, for a coherent consequence relation. (See example 7.5.14 above).

Proposition 7.5.15 *Let $\vdash$ be a monotonic logical system with conjunction $\wedge$, and negation $\neg$ satisfying the following conditions:*

1. *For any D thre exists $\Theta(D)$, dependent on D, such that for any A the following hold:*
2. *$D \vdash \neg A$ iff $\exists y \in \Theta(D)(D, A \vdash y)$*
3. *$\Delta \vdash A$ iff $\forall x(\Delta, x \vdash A)$.*

Then there exists an N (independent of D) such that (1) holds, i.e. $N = \Theta(D)$).

Proof. We prove the proposition by means of two lemmas.

Lemma 7.5.16 *Let $\vdash, \neg$ and $\Theta(D)$ be as in proposition 7.5.15. Let $N(D)$ be the set*

$$N(D) = \{y \mid D \vdash \neg y\}$$

then $\neg$ is a negation satisfying equation 7.5.15 (2) with $N(D)$ as a set of unwanted sentences.

Proof. Very much as in Lemma 7.5.6, we show that, for any D and A:

(*) $\exists y \in \Theta(D)(D, A \vdash y)$ iff $\exists z \in N(D)(D, A \vdash z)$

(a) Assume $D, A \vdash y$ for some $y \in \Theta(d)$.

By 7.5.15(2) we get that $D \vdash \neg A$ and hence $A \in N(D)$ and therefore there exists a $z \in N(D)$, namely $z = A$ such that $D, A \vdash z$.

(b) Assume $D, A \vdash z$, for some $z \in N(D)$.

Since $z \in N(D)$ we therefore have that $D \vdash \neg z$. Hence by 7.5.15(2) again there exists a $y \in \Theta(D)$ such taht $D, z \vdash y$. We now have

$$D, A \vdash z \text{ and } D, z \vdash y$$

by monotonicity we get:

$$D, A, z \vdash y$$

and by the cut rule we get

$$D, A \vdash y.$$

This completes the proof of lemma 7.5.16. Note that the proof in part (2) above can be modified to show that $D, a \vdash B$ and $D, A \vdash \neg B$ implies $D \vdash \neg A$. ■

Remark 7.5.17 *We draw several conclusions from lemma 7.5.16.*

1. *First that if $\neg$ is indeed a negation dependent on D (via $\Theta(D)$) then equation 7.5.15 (2) is really an uniformative tautology. By lemma 7.5.16, $\Theta(D)$ can be taken as $N(D) = \{y \mid D \vdash \neg y\}$ and equation 7.5.15 (3) becomes:*

   $$D \vdash \neg A \text{ iff } (D \vdash \neg y \text{ and } D, A \vdash y)$$

   which is trivially true for $y = A$.

   Note that for the case where Θ was fixed (independent of D) we got that $D \vdash \neg A$ iff $\exists y$ ($\vdash \neg y$ and $D, A \vdash y$) which is more informative.

2. *The second conclusion is that Θ is dependent on D in a special way.*

   As D gets stronger, Θ incrases. This is not intuitive! Why should (a priori) what we do not want increase with the database?

   This property follows since we have:

   $$\frac{D' \vdash D, D \vdash \neg A}{D' \vdash \neg A}$$

   We thus get that

   $$D' \vdash D \Rightarrow N(D') \supseteq N(D).$$

3. *The third conclusion follows from the proof of lemma 7.5.16 and the assumption 7.5.15 (3). We get the following for $\neg$:*

   $$(c1) \qquad \frac{D, A \vdash B; D, A \vdash \neg B}{D \vdash \neg A}$$

   Furthermore, sicne we saw in (b) that $D' \vdash D \Rightarrow N(D') \supseteq N(D)$, we can get that

   $$(c2) \qquad \frac{D \vdash \neg A}{D \vdash \neg(A \wedge B)}$$

   The reason is that if $D, A \vdash y, y \in N(D)$, then certainly $DB, A \vdash y$ and since $D, B \vdash D$, we have $y \in N(D, B)$ and hence $D \vdash \neg(A \wedge B)$.

We now proceed to use lemma 7.5.16 to prove proposition 7.5.15 namely that $\neg$ can be taken to be a negation with a fixed Θ (independent of D). We assumed that the language contains conjunction $\wedge$. $\wedge$ satisfies the three axioms:

$$\begin{array}{l} A \wedge B \vdash A \\ A \wedge B \vdash B \\ A, B \vdash A \wedge B. \end{array}$$

Lemma 7.5.18 *Let $\vdash$ be a system with negation $\neg$, satisfying the rule:*

1. $$\frac{D, A \vdash B; D, A \vdash \neg B}{D \vdash \neg A}$$

   Then for $N = \{B \wedge C \mid B \vdash \neg C\}$ we have for any D, A.

2. *$D \vdash \neg A$ iff $\exists y \in N(D, A \vdash y)$.*

Proof.

1. Assume $D \vdash \neg A$. We are looking for a y such tat $y \in N$ and $D, A \vdash y$. Let $y = D \wedge A$. Certainly $D, A \vdash D \wedge A$ and $D \wedge A \in N$ since $D \vdash \neg A$.

2. Assume that for some $y \in N$, we have $D, A \vdash y$. y is then equal to some $B \wedge C$ with $B \vdash \neg C$. Since $D, A \vdash B \wedge C$ we get $D, A \vdash C$. Since $B \vdash \neg C$ we get $D, A \vdash \neg C$ and hence by rule 1, $D \vdash \neg A$.

Proof of Proposition 7.5.15
Assume the conditions of proposition 7.5.15 for $\vdash$ and $\neg$ hold. By conclusion (c1) of remark 7.5.17 the conditions of lemma 7.5.18 hold and hence $\neg$ is a negation with a fixed $\Theta = N$.

The above considerations show that there is no hope for a formulation of a negation $\neg$ with a Θ dependent on the database, within the framework of monotonic logics. The assumption that $\wedge$ is available does not restrict generality since $\wedge$ can always be added to the language and definition 7.5.13 for negation be used. ■

7.6 Negation and structured consequence relations

In the previous section we addressed the problem of what is negation in the context of a monotonic consequence relation. By this we mean a relation between formulas of the form $\Delta \vdash A$, where Δ is a finite set of formulas satisfying reflexivity, monotonicity and cut.

The basic idea was that a connective $*$ is to be considered a negation if there exists a theory Θ_* such that for any Δ and any A we have:

$$\Delta \vdash *A \qquad \text{iff} \qquad \Delta, A \vdash B \text{ for some } B \in \Theta_*$$

Θ_* is considered the set of *unwanted* sentences and any A is *negated* by Δ
(i.e. $\Delta \vdash *A$) if it leads, "together with" Δ, to an unwanted B in Θ_*. We refinef this definition by imposing conditions on Θ_* and refining the notion of "together with".

The *LDS* notion of a database (or theory) is no longer a set formulas, but can be a multiset or a list, or even an arbitrary structure. Monotonicity no longer holds and we need to extend the basic notions involved in deciding whether a connective '$*$' is a negation.

Our first step should be to set up the scene in which the concept of negation is to be characterized, namely, introduce the new notion of a non-monotonic consequence relation, which we call S-consequence relation (Structured Consequence Relation). A precise definition is given in an earlier chapter. Recall that we need three auxiliary notions for our definition:

1. We have already indicated that we need to deal with structured databases. We thus have to specify precisely what structures are allowed as databases, for the consequence relation to be defined. These must include the one formula database (A) and the empty database $\varnothing$.

2. The notion of a structured database must also include the concept of *structured addition of data*. By this we mean the notion of how to combine two databases together. The traditional way is to take their union, but when the databases are already structured, the union has to be structured as well. We denote the structured addition of the databases Δ and Γ by $\Delta + \Gamma$. This binary operation, "+", has to be defined as part of the concepts underlying the consequence relation. It will usually satisfy associativity:

$$\Delta_1 + (\Delta_2 + \Delta_3) = (\Delta_1 + \Delta_2) + \Delta_3$$

and the empty database must satisfy:

$$\varnothing + \Delta = \Delta + \varnothing = \Delta$$

Commutativity is not required to hold.

3. We also need a concept of substitution of one structured database Δ inside another, Γ, achieved by replacing one formula A in Γ by Δ. Formally, we need to make sense of the symbol $\Gamma[A]$, reading Γ is a structured database which contains A somewhere inside, and the symbol $\Gamma[\Delta]$, which denotes the database resulting from substituting Δ for A inside Γ. These notions are needed to formulate the Cut Rule.

We can now define the three rules which we call **Identity**, **Surgical Cut** (or **Substitutional Cut**) and **Directional Monotonicity** as follows:

Identity

$$(A) \mid\!\sim A$$

Surgical (Substitutional) Cut

$$\frac{\Delta \mid\!\sim A \qquad \Gamma[A] \mid\!\sim B}{\Gamma[\Delta] \mid\!\sim B}$$

Directional Monotonicity (right hand side)

$$\frac{\Delta \mid\!\sim A}{\Delta + \Gamma \mid\!\sim A}$$

Directional Monotonicity (left hand side)

$$\frac{\Delta \mid\!\sim A}{\Gamma + \Delta \mid\!\sim A}$$

Since $+$ need not be commutative, we may distinguish between directional data additions $+^r$, $+^l$ defined by $\Delta +^r \Gamma = \Delta + \Gamma$, $\Delta +^l \Gamma = \Gamma + \Delta$.

Definition 7.6.1 (Tentative Definition of Structured Consequence Relation) Let **L** be a language. Let $\mathcal{L}$ be a family of order types of the form $(\tau, \leq)$. Assume that the empty set and one element set types are in $\mathcal{L}$. Assume that the notion of substitution of one order type τ for a point x in another order type $\tau'(x)$ is well defined (as in (3) above). We say that a relation $\Delta \mid\!\sim A$ (between ordered multisets Δ of wffs of an order type τ and a single wff A) is an *S-consequence relation* iff it satisfies **Identity** and **Surgical Cut**.

Definition 7.6.2 (Negation Relative to Data Addition) Let $\mid\!\sim$ be a structured consequence relation. Let $+$ denote addition of data. Let $*$ be a unary connective. We say $*$ is a negation relative to $+^r$ (resp. $+^l$) iff there exists a set of wffs Θ_* such that for all Δ and A

$$\Delta \mid\!\sim *A \qquad \text{iff} \qquad \text{for some } B \in \Theta_* \ \Delta + A \mid\!\sim B \text{ (resp. } A + \Delta \mid\!\sim B)$$

We require that Θ_* is not the set of all wffs, nor does it contain theorems (i.e. any $B \in \Theta_*$ which satisfies $\varnothing \mid\!\sim B$). We say that $*$ is a negation relative to $+$ iff it is a negation relative to $+^r$ or $+^l$.

This definition is parallel to Definition 7.5.4 of the previous section. As remarked in that section, there may be systems containing negation $*$ which are not expressive enough to find a corresponding Θ_*. We need a slight modification of the above definition which involves conservative extensions.

Definition 7.6.3 *Let $\mid\!\sim$ be a consequence relation, $+$ addition of data and $*$ a unary connective. We say that $*$ is a negation in $\mid\!\sim$ relative to $+^r$ (resp. $+^l$) iff in some conservative extension $\mid\!\sim_1 \supseteq \mid\!\sim$, there exists a Θ_* such that for all Δ, A of the language of $\mid\!\sim$ we have*

1. $\Delta \mid\!\sim *A$ *iff* *for some $B \in \Theta_*$ $\Delta + A \mid\!\sim_1 B$ (resp. $A + \Delta \mid\!\sim_1 B$)*
2. *In every conservative extension of $\mid\!\sim$, $*$ is a negation relative to $+^r$ (resp. $+^l$) in the sense of Definition the previous definition.*

We assume Θ_* is not the set of all wffs and does not contain theorems.

Example 7.6.4 Consider Łukasiewicz' many-valued logic.

1. Consider the consequence relation $\mid\!\sim$ as defined by:

$A_0, \dots, A_n \mid\!\sim B$ iff for all assignments h of truthvalues to the atoms we have

$$\max(0, \sum_{i=0}^{n} h(A_i) - n) \leq h(B)$$

2. Let $\vdash$ be the consequence relation defined by:

   $A_0, \dots, A_n \mid\!\sim B$ iff $\min(h(A_i)) \leq h(B)$

   then clearly if

$$C(X,Y) \stackrel{\text{def}}{=} \neg(X \to \neg Y)$$

   then

$$A_1, A_2 \mid\!\sim B \text{ iff } C(A_1, A_2) \vdash B$$

The consideration in Remark 7.5.10 of the previous section, shows that $\neg$ is a negation in $\mid\!\sim$ relative to '$+_1$' being union and is also a negation in $\vdash$ relative to '$+_2$' being 'C'.

With condition 2 in the previous definition, being a negation relative to $+^r$ resp. $+^l$ is preserved under conservative extensions.

Lemma 7.6.5 *Let $\mid\!\sim$ be a consequence relation with negation $*$ relative to data addition $+^r$ (resp. $+^l$). Then $*$ is a negation relative to $+^r$ (resp. $+^l$) in every conservative extension $\mid\!\sim_2$.*

Proof. Since $*$ is a negation in $\mid\!\sim$, there is a conservative extension $\mid\!\sim_1$ that satisfies the conditions of Definition 7.6.2. Suppose $\mid\!\sim_2$ is a conservative extension of $\mid\!\sim$. Let $\mid\!\sim_3$ be the composition of $\mid\!\sim_1$ and $\mid\!\sim_2$, defined as follows:

1. $\Delta \mid\!\sim_3^0 A$ if $\Delta \mid\!\sim_i A$ for $i = 1, 2$
2. $\Delta \mid\!\sim_3^{n+1} A$ if for some $C, \Gamma_1[C]$ and Γ_2 we have $\Gamma_1(C) \mid\!\sim_3^m A$ and $\Gamma_2 \mid\!\sim_i C$ and $m \leq n$ and $i = 1, 2$ and $\Delta = \Gamma_1[\Gamma_2]$.
   In other words we require closure under the Surgical Cut Rule.
3. Let $\Delta \mid\!\sim_3 A$ iff for some $n, \Delta \mid\!\sim_3^n A$.

Then clearly $\mid\!\sim_3$ is a conservative extension of both $\mid\!\sim_2$ and $\mid\!\sim_1$. Assume without loss of generality that $*$ is a negation relative to $+^r$ in $\mid\!\sim_3$ in the sense of definition 7.6.1. Thus, in the language of $\mid\!\sim_3$ there is a set Θ_* such that for all Δ and A:

$$\Delta \mid\!\sim_3 *A \qquad \text{iff} \qquad \Delta + A \mid\!\sim_3 B \text{ for some } B \in \Theta_*.$$

Hence, in particular, for all Δ and A in the language of $\mid\!\sim_2$:

$$\Delta \mid\!\sim_2 *A \qquad \text{iff} \qquad \Delta + A \mid\!\sim_3 B \text{ for some } B \in \Theta_*.$$

■

Definition 7.6.6 If the consequence $\mid\!\sim$ has several $+_i$ available, we can let $*$ be an $\vee$-negation (resp. $\wedge$-negation) relative to $\{+_i\}$ iff for some Θ_* we have for all Δ and A:

$$\begin{aligned} \Delta \mid\!\sim *A \quad \text{iff} \quad & \text{for some } i \text{ (resp. for all } i) \text{ and some } B \in \Theta_* \\ & (\Delta +_i A \mid\!\sim B \text{ or } A +_i \Delta \mid\!\sim B) \end{aligned}$$

So far, we assumed Θ_* is given to us as a set of unwanted wffs. We did not specify how Θ_* is given. It may be the case that Θ_* is *generated* by another consequence relation $\mid\!\sim_*$. Thus we may have

$$B \in \Theta_* \text{ iff } \varnothing \mid\!\sim_* B$$

Clearly, if Θ_* is given as a set, then the relation $\mid\!\sim_*$ can be taken as the membership relation, i.e.

$$\Gamma \mid\!\sim_* B \text{ iff } B \in \Gamma \cup \Theta_*$$

This membership relation is reflexive, monotonic and satisfies cut.

Definition 7.6.7 Let $\mid\!\sim$ be a consequence relation in a language $\mathbf{L}$. Let $\mid\!\sim_1$ be a conservative extension of $\mid\!\sim$ in language $\mathbf{L}_1 \supseteq \mathbf{L}$. Let $*$ be a unary connective of $\mathbf{L}$ and $+$ be data addition in $\mathbf{L}$. We say that $*$ is a negation in $\mid\!\sim$ relative to $+^r$ (resp. $+^l$) iff for some consequence relation $\mid\!\sim_*$ of $\mathbf{L}_1$ the following holds:

1. The theorems of $\mid\!\sim_*$ do not contain all wffs of $\mathbf{L}$ and do not contain any theorem of $\mid\!\sim$ in $\mathbf{L}$.
2. For all Δ and $A, \Delta \mid\!\sim *A$ iff for some B of $\mathbf{L}_1$ such that $\varnothing \mid\!\sim_* B$, we have $\Delta + A \mid\!\sim_1 B$ (resp. $A + \Delta \mid\!\sim_1 B$).
3. In every conservative extension of $\mid\!\sim$, $*$ is a negation relative to $+^r$ (resp. $+^l$) in the sense of Definition 7.6.1.

The following lemma says that Θ_* can be taken as $\{B \mid \varnothing \mid\!\sim *B\}$.

Lemma 7.6.8 *Let $\mid\!\sim$ be a consequence relation and let $*$ be a negation relative to $+$, through a Θ_* in some conservative extention $\mid\!\sim_1$. Suppose without loss of generality that $*$ is a negation relative to $+^r$. Then (1) is equivalent to (2):*

(1) $\Delta + A \mid\!\sim_1 B$ *for some* $B \in \Theta_*$

(2) $\Delta + A \mid\!\sim_1 C$ *for some* C *such that* $\varnothing \mid\!\sim_1 *C$.

Proof.

(1) $\Rightarrow$ (2): If $\Delta + A \mid\!\sim_1 B$ and $B \in \Theta_*$, then since $\varnothing + B = B \mid\!\sim_1 B$ we get $\varnothing \mid\!\sim_1 *B$ and (2) holds.

(2) $\Rightarrow$ (1): If $\Delta + A \mid\!\sim_1 C$ and $\varnothing \mid\!\sim_1 *C$, then for some $C_1 \in \Theta_*$, $C \mid\!\sim_1 C_1$ and hence by **Cut**, $\Delta + A \mid\!\sim_1 C_1$. This yields (1).

■

Definition 7.6.9 Let $\mid\!\sim$ be a consequence relation with addition $+$ and $*$ a unary connective. We say $*$ is a *context dependent negation relative to* $+$ if for every Δ, A there exists a $\Theta_*(\Delta, A)$ (dependent on Δ and A) such that we have:

$$\begin{array}{lll} \Delta \mid\!\sim *A & \text{iff} & \text{for some } B \in \Theta_*(\Delta, A) \\ & & (\Delta + A \mid\!\sim B \text{ or } A + \Delta \mid\!\sim B) \end{array}$$

Remark 7.6.10 The preceding definition should be compared with the parallel definition of the previous section for the monotonic case. There, for every Δ, Θ_* depended on Δ only, i.e. for all Δ and A

$$\Delta \mid\!\sim *A \text{ iff } \exists B \in \Theta_*(\Delta), \Delta, A \mid\!\sim B$$

In our definition, Θ_* depends on A as well. For the monotonic case, since $\Delta + A + A$ is equivalent to $\Delta + A$, this is only a slight difference.

Further note that because of **Cut**, we have $C \mathrel{|\!\!\sim} *A$ and $C' \mathrel{|\!\!\sim} C$ imply $C' \mathrel{|\!\!\sim} *A$. It is reasonable to assume that $C' \mathrel{|\!\!\sim} C$ implies $\Theta_*(C') \supseteq \Theta_*(C)$.

Example 7.6.11 Consider positive intuitionistic logic (for the language of $\to, \wedge$). To add intuitionistic negation we need the axioms

$$\begin{array}{l} A \to (\neg A \to B) \\ (A \to \neg B) \to (B \to \neg A) \end{array}$$

With these axioms we can let $\bot \overset{\text{def}}{=} \neg(q \to q)$ and interpret $\neg A$ as $A \to \bot$. The axiom for $\bot$ that we get is

$$\bot \to B$$

Here $\Theta_* = \{\bot\}$ and

$$\Delta \vdash \neg A \qquad \text{iff} \qquad \Delta, A \vdash \bot$$

The above are the well known results for obtaining $\neg$ from a falsity constant $\bot$. What can we do in order to add $\neg$ which is a context dependent negation?, i.e. for any A we add $\bot_A$ with

$$\begin{array}{lll} \neg A \leftrightarrow (A \to \bot_A) & & \\ A \vdash \neg B & \text{iff} & A, B \vdash \bot_B \end{array}$$

Consider the axiom

$$(B \to \neg B) \to \neg B$$

It was shown by R. Valerius [Valerius, 1989] that positive intuitionistic logic with $\to, \wedge$ and the above axiom allows for a conservative extension with falsities $\bot_B$, for each B and the equivalence

$$\neg B \leftrightarrow (B \to \bot_B)$$

Theorem 7.6.12 *Let $\mathrel{|\!\!\sim}$ be a consequence relation, $+$ addition and $*$ a unary connective, being a context dependent negation relative to $+$. Assume that $\oplus$ is a structural connective corresponding to $+$, i.e. we have*

1. $A + B \mathrel{|\!\!\sim} A \oplus B$
2. $\dfrac{\Delta + A + B + \Delta' \mathrel{|\!\!\sim} D}{\Delta + A \oplus B + \Delta' \mathrel{|\!\!\sim} D}$

Assume $*$ is a negation relative to $+^r$ which is context dependent with $\Theta_*(\Delta, A)$. Let

$$\mathcal{N}_*(\Delta, A) = \{*B \oplus B \mid \Delta \mathrel{|\!\!\sim} *B \text{ and } A \mathrel{|\!\!\sim} B\}$$

Then (1) iff (2).

(1) For some $Y \in \Theta_*(\Delta, A)$, $\Delta + A \mathrel{|\!\!\sim} Y$

(2) For some $Y \in \mathcal{N}_*(\Delta, A)$, $\Delta + A \mathrel{|\!\!\sim} Y$

Proof. To show (1) $\Rightarrow$ (2):

$$\Delta + A \mathrel{|\!\!\sim} Y \qquad \text{iff} \qquad \Delta \mathrel{|\!\!\sim} *A$$

Since $*A + A \vdash *A \oplus A$ we get by **Cut**

$$\Delta + A \mathrel{|\!\!\sim} *A \oplus A$$

which yields (2). To show (2) $\Rightarrow$ (1): Assume $\Delta + A \mathrel{|\!\!\sim} *B \oplus B$ for some B such that $\Delta \mathrel{|\!\!\sim} *B$ and $A \mathrel{|\!\!\sim} B$. Since $\Delta \mathrel{|\!\!\sim} *B$, there exists a $Y \in \Theta(\Delta, A)$ such that $\Delta + B \mathrel{|\!\!\sim} Y$.Since $A \mathrel{|\!\!\sim} B$, we get by **Cut**, $\Delta + A \mathrel{|\!\!\sim} Y$, which is condition (1). ■

If a negation $*$ is context dependent with Δ and A, its being a negation does not depend on $\Theta_*(\Delta, A)$.

Theorem 7.6.13 *Let $\mid\!\sim$ be a consequence relation, $+$ addition and $*$ a negation relative to $+$ which is context dependent with Δ and A. Let $\mid\!\sim_1$ be $\mid\!\sim$ conservatively extended by $\oplus$. Suppose without loss of generality that $*$ is a negation relative to $+^r$ in $\mid\!\sim_1$. Assume that $C' \mid\!\sim C \Rightarrow \Theta_*(C') \supseteq \Theta_*(C)$. Then there is a set $\mathcal{N}_*$ of wffs in the language of $\mid\!\sim_1$ such that $\mathcal{N}_*$ is independent of Δ and A and*

$$\Delta \mid\!\sim_1 *A \quad \textit{iff} \quad \Delta + A \mid\!\sim_1 B \textit{ for some } B \in \mathcal{N}_*$$

Proof. Set $\mathcal{N}_* = \{(B \oplus C) \mid B \mid\!\sim_1 *C\}$. If $\Delta = A_1 + \ldots + A_n$, then $\oplus\Delta$ denotes $A_1 \oplus \ldots \oplus A_n$. $\Rightarrow$: If $\Delta \mid\!\sim_1 *A$, then for $B = \oplus\Delta \oplus A$ we have $\Delta + A \mid\!\sim_1 B$ and $B \in \mathcal{N}_*$. $\Leftarrow$: Assume $B \in \mathcal{N}_*$ and $\Delta + A \mid\!\sim_1 B$. Then for some $(B_1 \oplus C_1)$, $B = (B_1 \oplus C_1)$ and $B_1 \mid\!\sim_1 *C_1$. Since $*$ is a negation in $\mid\!\sim_1$ in the sense of Definition7.6.1, there is a set Θ_* in the language of $\mid\!\sim_1$ and $C_2 \in \Theta_*(B)$ such that $B_1 + C_1 \mid\!\sim_1 C_2$. Hence $B_1 \oplus C_1 \mid\!\sim C_2$. Applying **Cut** to this and $\Delta + A \mid\!\sim_1 (B_1 \oplus C_1)$ yields $\Delta + A \mid\!\sim_1 C_2$. Hence$\Delta \mid\!\sim *A$. ■

Remark 7.6.14 Let $\mid\!\sim$ be a consequence relation, $+$ addition of data and $*$ a negation relative to $+$. Assume $*$ is a negation relative to $+^r$. Then for every A, $\varnothing \mid\!\sim *(*A \oplus A)$:

$$\begin{array}{ll} & *A \mid\!\sim *A \\ \text{iff} & *A +^r A \mid\!\sim B \\ \text{iff} & (*A \oplus A) \mid\!\sim B \\ \text{iff} & \varnothing \mid\!\sim *(*A \oplus A) \end{array}$$

for some set Θ_* and some $B \in \Theta_*$. Thus, $*$ cannot be negation relative to data addition $+$, if for some A, neither $\varnothing \mid\!\sim *(*A \oplus A)$ nor $\varnothing \mid\!\sim *(A \oplus *A)$.

Example 7.6.15 (Strong negation) *Consider the following construction. Let $\mathbf{L}$ be a language with some connectives $\{\sharp_1, \ldots, \sharp_m\}$. Let ρ be a new unary logical connective. For each connective $\sharp(q_1, \ldots, q_n)$ of $\mathbf{L}$ let $\psi_\sharp(q_1, \ldots, q_n, \rho q_1, \ldots, \rho q_n)$ be a wff of the language $\mathbf{L} + \rho$ (usually not containing $\sharp$). Further assume that some complexity measure can be defined on $\mathbf{L} + \rho$ such that for all $A_1, \ldots, A_n$, the complexity of $\psi_\sharp(A_1, \ldots, A_n, \rho A_1, \ldots, \rho A_n)$ is strictly less than that of $\rho\sharp(A_1, \ldots, A_n)$. Let $\mathbf{H}$ be any logical system in the language $\mathbf{L}$. We call $\mathbf{H}_\rho$* a rewrite extension *of $\mathbf{H}$ in $\mathbf{L} + \rho$ iff (intuitively) $\mathbf{H}_\rho$ is the smallest extension of $\mathbf{H}$ satisfying the following axioms for all connectives $\sharp$, and for k fixed:*

($\rho_\sharp$) $\rho\sharp(A_1, \ldots, A_n) \leftrightarrow \psi_\sharp(A_1, \ldots, A_n, \rho A_1, \ldots, \rho A_n)$

(ρ_k) $\rho^k A \leftrightarrow A$

For example, let $\mathbf{H}$ be the intuitionistic positive propositional calculus with $\wedge, \vee, \rightarrow$. Let "$\rho$" be "$\sim$" ($\sim$ is intended to be Nelson's strong negation). Let $\mathbf{N}$ be the system with:

$$\begin{array}{rcl} \sim (A \rightarrow B) & \leftrightarrow & A \wedge \sim B \\ \sim (A \vee B) & \leftrightarrow & \sim A \wedge \sim B \\ \sim (A \wedge B) & \leftrightarrow & \sim A \vee \sim B \\ \sim\sim A & \leftrightarrow & A \end{array}$$

The complexity is lexicographic in the nestings of $\wedge, \vee, \rightarrow$ within $\sim$, where "$\rightarrow$" is stronger than "$\wedge$", "$\vee$". Given such an extension, we can add negation $*$ to the system by letting $\Theta_* = \{\wedge_{i=1}^{k-1} \rho^i B\}$.

$$\Delta \mid\!\sim *A \textit{ iff for some } B, \Delta + A \mid\!\sim \rho^i B, \textit{ for all } i = 1, \ldots, k-1.$$

In the case of $\sim$, we get

$$\Delta \vdash *A \textit{ iff for some } B, \Delta \mid\!\sim B \wedge \sim B.$$

If we want a single atomic falsity $\bot_*$ for $*$ (i.e. $\Theta_* = \{\bot_*\}$), we can stipulate the additional axiom:

$$A, \rho A, \ldots, \rho^{k-1} A \mid\!\sim B$$

for any A and B. This will allow us to identify the left hand side with some constant $\bot_*$. Denote this system by $\mathbf{H}_k^+$. In the case of $\sim$, we need

$$A \wedge \sim A \vdash B$$

and thus since all $A \wedge \sim A$ are equivalent, they can be identified as $\bot_\sim$. In the case of $\bot_\sim$, the negation $*$ becomes intuitionistic negation. The only reason for that is the fact that $\bot_* \vdash B$ holds for any B. It has nothing to do with the axioms on $\sim$. We know that intuitionistic negation $\neg$ can be defined from any *fixed propositional constant $\bot$ provided we have the additional rule:*

$$\bot \vdash B, B \text{ arbitrary.}$$

In the case of $\mathbf{N}_2^+$, $\bot_\sim$ can serve as the $\bot$. If the intuitonistic system already contains falsity $\bot$ we can add:

$$\sim \bot = \top$$
$$\sim \top = \bot$$

In this case we will have two falsities $\bot$ and $\bot_\sim$.

It is interesting to note that it so happens that for any wff A, there exist atoms q_j such that in $\mathbf{N}$ we have $\sim A \wedge A \vdash \vee_j \sim q_j \wedge q_j$. This can be proved by induction on A. Thus the additional axiom $\sim A \wedge A \vdash B$, for B arbitrary, can be restricted to A atomic. This has ramifications to the semantics for strong negation.

Chapter 8

Metalevel Features in LDS

8.1 Introduction

In Chapter 1 we claimed that in *LDS*, the metalevel features are done by the labels while the object level features are done by the formulas of the logic. The meaning of the above statement, as well as notions of metalevel have to be clarified and formalised. This is the task of this Chapter.

We clarify the fundamental concepts involved in the relationship between meta-language and object language features. We will also study in more detail a meta-language **HFP**, presented in a later section, and examine the claim that it can function as a general purpose meta-language. Our conceptual framework will revolve around two pure notions. The notions of a *proper meta-language* and that of an object language $\mathbf{L}_1$ *implementing meta-language features* for a language $\mathbf{L}_2$.

Given a language $\mathbf{L}$, we understand by a proper meta-language $\mathbf{M}$ to the language $\mathbf{L}$, any language which is able to name every well formed symbol of $\mathbf{L}$ and to describe all the logical operations of $\mathbf{L}$. Thus formulas φ of $\mathbf{L}$ become terms t_φ of $\mathbf{M}$ and logical relations of $\mathbf{L}$ become predicates in $\mathbf{M}$. $\mathbf{M}$ must have axioms to ensure that every interpretation of $\mathbf{M}$ gives rise, through the embedding of $\mathbf{L}$ in $\mathbf{M}$, to an interpretation of $\mathbf{L}$. For a given $\mathbf{L}$, $\mathbf{M}$ is not unique. Many languages can serve as a meta-language for $\mathbf{L}$, provided they are strong enough. In computer language terms, any strong enough language, e.g. BASIC, can be used to write an interpreter for any other language. In such a case one *simulates* one language in another. The other pure notion of one language, $\mathbf{L}_1$, serving as an object language, $\mathbf{L}_2$, is more difficult to explain. Consider the relationship between $\mathbf{L}$ and $\mathbf{M}$. $\mathbf{L}$ is a sublanguage of $\mathbf{M}$ in a very strong sense. $\mathbf{M}$ has names for everything in $\mathbf{L}$ and axioms about $\mathbf{L}$. Imagine a more equal relationship between two languages. Suppose $\mathbf{L}_1$ and $\mathbf{L}_2$ are both sublanguages of a language $\mathbf{L}_{1,2}$. $\mathbf{L}_{1,2}$ is not a meta-language to $\mathbf{L}_1$. It is just a richer language. For example, classical predicate logic, $\mathbf{L}_1$, augmented with additional dummy predicates being the language $\mathbf{L}_2$ gives us $\mathbf{L}_{1,2}$ which is certainly not a meta-language. Consider $\mathbf{L}_{1,2}$ and consider some mixed axioms $\Delta_{1,2}$ in $\mathbf{L}_{1,2}$ affecting both $\mathbf{L}_1$ and $\mathbf{L}_2$. Any interpretation of $\mathbf{L}_{1,2}$ satisfying the mixed axioms, will induce an interpretation on $\mathbf{L}_1$ alone. Similarly, we can consider a proper meta-language $\mathbf{M}_1$ of $\mathbf{L}_1$, with names for $\mathbf{L}_1$ symbols, etc. as discussed before. In the language $\mathbf{M}_1$ we can write axioms $\Delta_{\mathbf{M}_1}$ about $\mathbf{L}_1$, restricting its possible interpretations. It may be the case that the family of possible pure interpretations of $\mathbf{L}_1$ allowed by $\Delta_{1,2}$ (as induced from interpretations of $\mathbf{L}_{1,2}$ satisfying $\Delta_{1,2}$) are the same as the possible interpretations of $\mathbf{L}_1$ allowed by $\Delta_{\mathbf{M}_1}$ (as induced from interpretations of $\mathbf{M}$ satisfying $\Delta_{\mathbf{M}_1}$). In this case we say that $\mathbf{L}_{1,2}$ and $\Delta_{1,2}$ is an object language implementation of $\Delta_{\mathbf{M}_1}$. The justification for 'object language' is that $\mathbf{L}_{1,2}$ is not meta to $\mathbf{L}_1$, it is just an extension. Consider for example two interpreters $\mathbf{L}_1$ and $\mathbf{L}_2$ which are linked in some way. Thus each interpreter has its allowable options of how to run. However, since they are linked, not all of these individual options can be realized. $\mathbf{L}_2$ can choose to run in certain ways which will *limit* the options of $\mathbf{L}_1$ because they are linked. Suppose we use a proper meta-language $\mathbf{M}_1$ to talk and describe the options of $\mathbf{L}_1$ and suppose further that we express, through a wff φ_1 of $\mathbf{M}_1$, our wish to have $\mathbf{L}_1$ run only in certain ways. So φ_1 (if true) limits the runs (or options) of $\mathbf{L}_1$. There is another way of limiting the runs of $\mathbf{L}_1$, through its linkage with $\mathbf{L}_2$.

It may be that $\mathbf{L}_2$ can be restricted by condition δ_1 so that $\mathbf{L}_1$ (when linked with $\mathbf{L}_2$ satisfying δ_1) can run exactly in a way which satisfies φ_1. If this is the case, then we say that $\mathbf{L}_2$, through its object language restriction δ_1 and its linkage with $\mathbf{L}_1$, can *implement* φ_1, i.e. implement the meta-statement φ_1. It may be that for each φ_n of $\mathbf{M}_1$, there exists a way of restricting $\mathbf{L}_2$ by δ_n which through the link with $\mathbf{L}_1$, allows $\mathbf{L}_1$ to run only in a way which satisfies φ_n. In that case we have object language implementation of the set $\{\varphi_n\}$ of the language $\mathbf{M}_1$.

To make these concepts more concrete, think of an LDS system. The logic $\mathbf{L}$ of the formulas is $\mathbf{L}_2$ and the algebra $\mathcal{A}$ of labels is $\mathbf{L}_1$. When linked together they form an LDS system. The meta-language $\mathbf{M}$ is a language capable of talking about the proof theory of the logic $\mathbf{L}$. Here is a concrete example: let $\mathbf{L}$ be intuitionistic implicational logic. Let the proof theory for $\mathbf{L}$ be given using modus ponens ($A, A \to B \vdash B$) and $\to$ Introduction (to show $A \to B$, assume A and show B). $\mathbf{M}$ will be a meta-level language capable of talking about proofs of $\mathbf{L}$, containing predicates which can describe how many times an assumption A was used in a given proof of B. Let φ be a formula of $\mathbf{M}$ saying that each assumption is used at most once. Some proofs satisfy φ, some do not. We can add a labelling algebra $(A, \cup)$ where A is a set of atomic labels and $\cup$ is a multiset union of atomic labels and use the labels to keep trace exactly which assumptions are used in the proof. The system becomes:

- Label all assumptions by different atomic labels.
- Formulate modus ponens as
$$\frac{\alpha : A; \beta : A \to B \text{ and } \alpha \cap \beta = \varnothing}{\alpha \cup \beta : B}.$$
- Formulate $\to$ introduction as the following:
to show $\alpha : A \to B$ assume $x : A$ with a new atomic label x and show $\alpha \cup \{x\} : B$.
- Say $\Delta \vdash B$ if we label all wffs of Δ with different atomic labels we can prove $\alpha : B$ with α a subset of the set of these labels.

This LDS system implements the meta-level condition φ on the proofs.

Another example is from resolution in classical logic. Suppose we have a classical language $\mathbf{L}_1$ and a set Δ of clauses which is inconsistent. The resolution machinery may derive the empty clause but may not be equipped to give a set of substitutions into the clauses of Δ which can yield propositional inconsistency. Let $\mathbf{M}_1$ be a meta-language in which we demand this set. This demand can be implemented by adding a dummy predicate $Q(x_1, x_2, \ldots)$ with the appropriate variables (language $\mathbf{L}_2$) as a disjunct to all clauses of Δ. The resolution machine can stop when unable to eliminate Q and from the various instantiations of Q the substitutions can be obtained.

The value of metalanguage features and of the study of object level implementations of metalanguage features lies in developing computational implementations of *LDS*. There are many theorem provers for classical logic. There are also other implementations of logical languages. If we can express in a generic way the general *LDS* discipline in one of these logics we will immediately have a computational system for *LDS*. Since many non-monotonic logics can be expressed nicely in *LDS*, we will get a computational capability for non-monotonic logics, an area highly famous for its computationally intractable systems.

It is best that we see some examples:

We begin with modal logic. We consider Example 2.2.3. The language contains propositional variables $p, q, r, \ldots$, the classical connectives $\wedge, \vee, \to, \perp$ and the modalities $\Box$ and $\Diamond$. The labels are atomic labels from a set $\{t_1, s_1, t_2, s_2, \ldots\}$ and we have the configuration relation $<$ on labels. A database has the form of a set of labelled formulas called assumptions and a configuration on the labels involved in the assumptions. In Example 2.2.3 the database was:

	Assumptions	**Configuration**
(1)	$t : \Box\Box B$	$t < s$
(2)	$s : \Diamond(B \to C)$	

The deduction rules allowed us to move from one database and configuration to another, obtained by adding assumptions and labels and extending the configuratin according to certain rules. The rules used are the rules of the particular *LDS* system for the above language.

We shall use the above language (above *LDS*) as our example of an *object language.* We will develop a *metalanguage* for it. We will do the construction in detail. We will then give another example of language metalanguage, for relevance logic (2.2.1). After seeing these two examples in detail we will be ready for general definitions of language metalanguage concepts and the main body of this section. We aim at the central concept of "*object level implementation of a metalevel feature*". This will take time to develop and define. We first define the usual metalevel concepts, then try to simplify them, then try to do them in the object level and then explain how it is done in principle.

So, let us begin.

We want a metalanguage of the modal language, ie for the *LDS* of 2.2.3. A metalanguage should be strong and rich enough to talk about all the natural movements we make in the object language. Let us choose two sorted classical predicate logic as our metalanguage. This is a wise choice. The logic is well known, well understood and has many theorem provers available. If we can find it suitable it will be advantageous for us. How can it ever be unsuitable? There may be two possible reasons, one – it may not be mathematically expressive enough and two – its natural structure may not be intuitively compatible with the natural structure of the object language. Fortunately for our case, for the modal logic we are considering, classical logic is very suitable.

First order classical logic, however, is not general enough to serve as a metalanguage. We need the higher order language of Hereditarily Finite Predicates (**HFP**). This language will be formally defined later in this section. Meanwhile, we mention that it has the feature of allowing atomic predicates to have formulas as part of the predicate. For example, the metapredicate $\mathbf{Hold}(\varphi, t)$ reading "φ is true at t" is such a predicate. Another predicate is $\mathbf{Derive}(\varphi, \psi)$, reading "$\psi$ is derivable (in some logic) from φ"

We start by assigning formal predicate symbols and terms (names) to denote the properties and constants of the object language. Let us denote the label t by "t" and $<$ by "$<$". For each formula A let "A" be a term naming A. We let $\mathbf{Data}(x, y)$ be a two sorted binary predicate of classical **HFP** intended to formalise the relation "$t : A$ is in the database", written as $\mathbf{Data}(t, A)$. Let $\mathbf{Derive}(\Delta, x, y)$ be another relation intended to capture the notion of "$t : A$ is derived using the *LDS* rules from the database Δ".

Note that formally if t, A are considered the ground type 0 (level 0) then **Data** is if level 1 while **Derive** is of level 2, where generally level $n + 1$ can name and talk about all formulas of levels $m \leq n$.

The database of 2.2.4 can be described in the metalanguage by the wff φ.

$$\varphi = \mathbf{Data}(t, \Box\Box B) \wedge \mathbf{Data}(s, \Diamond(B \to C)) \wedge (t < s)$$

To the above sentence, we have to add a set of axioms (in **HFP** logic) describing the relation of derivation and the naming relation. These are:

(D1) $\mathbf{Derive}(\varphi, t, A \wedge B) \leftrightarrow \mathbf{Derive}(\varphi, t, A) \wedge \mathbf{Derive}(\varphi, t, B)$

(D2) $\mathbf{Derive}(\varphi, t, A \vee B) \leftrightarrow \mathbf{Derive}(\varphi, t, A) \vee \mathbf{Derive}(\varphi, t, B)$

(D3) $\mathbf{Derive}(\varphi, t, A \to B) \leftrightarrow \mathbf{Derive}(\psi, t, B)$ where $\psi = \varphi \wedge \mathbf{Data}\ (t, A)$

(D4) $\mathbf{Derive}(\varphi, t, \bot) \to \mathbf{Derive}\ (\varphi, x, Y)$

(D5) $\mathbf{Derive}(\varphi, t, \Box A) \to \mathbf{Derive}(\varphi, s, A)$ whenever $\varphi \vdash t < s$.

(D6) $\mathbf{Derive}(\varphi, s, B) \to \mathbf{Derive}(\varphi, t, \Diamond B)$ whenever $\varphi \vdash t < s$

(D7) $\mathbf{Derive}(\varphi, t, \Diamond A) \to \exists s\ \mathbf{Derive}(\varphi \wedge (t < s), s, A)))$

(D8) $\mathbf{Derive}(\varphi \wedge \mathbf{Data}(t, A), t, A)$.

Note that the above gives us intuitionistic derivability. Note the clause for deriving a disjunction. We do not include disjunctions in the data. To obtain classical derivability we need to add:

(D9) **Derive**(φ, t, A), where A is a substitution instance of a truth functional tautology.

Let Δ be the (infinite) theory containing the universal closure of the D axioms. We would hope that for any labelled wff, $r : A$, eg $t : \Diamond\Diamond B$ we would have:

$\Delta \vdash$ **Derive** (φ, r, A) in classical **HFP** iff $t : A$ is derivable in the system of 2.2.4.

Our difficulty at the moment is that **HFP** is not mathematically well defined. However, let us continue despite that. The mathematics will come later in this chapter.

8.2 Classical logic as a metalanguage

What we now want to do is to define the meta features formalised by **Data** and **Derive**, in first-order classical logic, which serves as an object language.

To achieve that we associate it with every atomic formula A of the *LDS* (without the label) a unary formula $A^*(t)$ of first order predicate logic, with one free variable t. We now translate the metapredicate **Data**(t, A) for arbitrary A as follows: ($*$ is the translation):

1. **Data**$(t, A)^* = A^*(t)$ for A atomic.
2. **Data**$(t, A \wedge B)^* = A^*(t) \wedge B^*(t)$.
3. **Data**$(t, A \to B)^* = A^*(t) \to B^*(t)$.
4. **Data**$(t, \bot)^* = \bot$.
5. **Data**$(t, \Box A)^* = \forall s(t < s \to A^*(s))$.
6. **Data**$(t, \Diamond A)^* = \exists s(t < s \wedge A^*(s))$.
7. $(t < s)^* = t < s$

We can now translate **Derive**(φ, t, A) as follows: ($\sharp$ is the translation).

8. **Derive**$(\varphi, t, A)^\sharp = \varphi^* \to A^*(t)$.

It is easy to verify that properties (D1)-(D5), (D7)-(D8) of the **Derive** predicate hold for the translation. We check property (D6):

We have to show that

Derive$(\varphi, t, \Diamond A)^\sharp \to \exists s$ **Derive**$(\varphi \wedge t < s, s, A)^\sharp$

After translation we get

$\varphi^* \to \exists s(t < s \wedge A^*(s)) \to \exists s(\varphi^* \wedge t < s \to A^*(s))$

Obviously this holds.

Let us summarise our steps. We started with a metalanguage **HFP** in which we represented our logic *LDS*.

HFP is a *proper* metalanguage, it names and describes the object language.

Our next step was to take another object language, in our case predicate logic, (call it the second object language) and connect it via translations $*$ and $\sharp$ to the first object language *LDS*. **Data** was translated directly via the $*$ translation and **Derive** was basically translated as classical implication "$\to$".

The properties of **Data** and **Derive** are *mirrored* in the 2nd object language. In a sense the 2nd object language *implements* some meta properties of *LDS* as expressed in the metalanguage.

The reader should think, for example, of a debugger to a program. The debugger is another object program which implements some metaproperties of the program (via interrupts). The following diagrams and tables illustrate our ideas (See figs 8.1, 8.2 and 8.3):

We now proceed to express relevance logic in the object language of classical logic in the same way we did for modal logic. Consider a first order predicate logic with the monadic predicates

Object Language LDS	**Metalanguage HFP**
$t : A$	$\textbf{Data}(t, A)$
LDS proof steps	Properties of the **Derive** meta predicate.
additional metaconditions yielding other logics eg transitivity of the accessibility relation.	Axioms written in **HFP**

Figure 8.1:

Object Language LDS	**2nd Object Language Predicate Logic**
$t : A$	$A^*(t)$
LDS Proof steps	Predicate logic proof steps
Additional meta conditions on the logic	Additional formulas of predicate logic or restrictions on the proof steps of predicate logic. It is quite possible that the *LDS* conditions cannot be expressed in predicate logic, in which case our capabilities are limited

Figure 8.2:

LDS vs Predicate Logic	**Program vs Debugger**
Properties to be studied: **Data** and **Derive**	Properties to be studied: contents of global variables registers
Means are the translations $*$ and $\sharp$	Means are the interrupts
Exact meaning of what we see can be expressed in **HFP**	Exact meaning to be expressed in a proper metalanguage which can express the program semantics.

Figure 8.3:

of the form $A^*(x)$, the predicate $atom(x)$ and the binary function "$\otimes$". We allow for a finite set of constants $\mathbf{C}_m = \{c_0 = \varnothing, c_1, \ldots c_m\}$, and equality "$=$". Let $A(x, y_1, \ldots, y_n)$ be a formula of predicate logic with x, y_i free.

Let $\forall^*, \exists^*$ be the quantifiers defined as follows:

$\forall^* x A(x)$ iff (def) $\forall x(\bigwedge_{i=1}^m x \neq y_i \wedge atom\ (x) \wedge x \neq \varnothing \to A(x))$

$\exists^* x A(x)$ iff (def) $\exists x(\bigwedge_{i=1}^m x \neq y_i \wedge atom\ (x) \wedge x \neq \varnothing \wedge A(x))$.

Notice that the meaning of $\forall^*, \exists^*$ is context dependent on the free variables of A.

It is intended that $\mathbf{C}_m$ represents the set of atomic lables and that more complex labels are generated by the function symbol "$\otimes$". If $t_1 : A$ and $t_2 : A \to B$ are labelled formulas then by applying modus ponens we get the labelled formula $(t_2 \otimes t_1) : B$. This was represented informally as $t_2 t_1 : B$. "$\otimes$" is a general binary operation and different axioms on $\otimes$ can give it properties of different logics. Associativity of "$\otimes$" seems to correspond to the logic **CL** of 9.2.5 and 10.3.9. For the case of relevance logic the labels are sets of atomic labels. In this case the binary operation "$\otimes$" is set union on labels. Thus if t_1 and t_2 are labels so is t_1 and t_2. we continue to use "$\otimes$" rather than "$\cup$" because for a weaker logic than relevance "$\otimes$" could mean other operations. We need to use equality "$=$" as well, since we have function symbols.

We can now translate from any propositional resource logic (such as **CL**) into the monadic predicate logic as follows:

With each atomic proposition A of the resource logic we associate a unary predicate $A^*(t)$ of classical logic. $A^*(t)$ will represent "$t : A$". The intuitive reading of $A^*(t)$ is that "A is labelled t". The $*$ translation can be extended to any wff of relevance logic as follows:

$$[t : (A \to B)]^* = \forall^* x(A^*(x) \to B^*(t \otimes x))$$

Recall that $\forall^*$ ranges over atomic labels, and is context dependent.

We need axioms on $\otimes$ to give it the correct meaning (of union in the case of relevance logic and just associativity in the case of **CL**).

Let **L** be a resource logic, eg **CL**. Let $\varphi_{\mathbf{L}}$ be a classical logic formula representing the labelling discipline of **L**. Such a formula must be shown to exist and be supplied by us. For example, $\varphi_{\mathbf{CL}}$ is:

$$\forall xyz(atom\ (x) \wedge atom\ (y) \wedge atom\ (z) \to (x \otimes y) \otimes z = x \otimes (y \otimes z)) \wedge \wedge$$

$$\forall x(atom\ (x) \to (x \otimes \varnothing = \varnothing \otimes x = x))$$

Let B be a formula of the resource logic **L**. Assume n is the number of all subformulas of B. Then we have:

$\mathbf{L} \vdash B$ iff in classical logic $\varphi_{\mathbf{L}} \vdash B^*_{\varnothing}$.

For the case of relevance logic **R**, $\varphi_{\mathbf{R}}$ also says that $\otimes$ is commutative and idempotent (ie $\varphi_{\mathbf{R}}$ gives all axioms true of set union).

Let us now try the left transitivity axiom of **CL**, namely

$$\varnothing : (B \to C) \to ((A \to B) \to (A \to C))$$

the $*$ translation is

$$\begin{array}{l}
\forall^* x((B \to C)^*_x \to ((A \to B) \to (A \to C))^*_{\varnothing \otimes x} \\
= \forall^* x(\forall^* y(B(y) \to C(x \otimes y)) \to \forall^* z((A \to B)^*_x \to (A \to C)^*_{x \otimes z})) \\
= \forall^* x(\forall^* y(B(y) \to C(x \otimes y)) \to \forall^* z(\forall^* u(A(u) \to B(z \otimes u)) \to \\
\forall^* s(A(s) \to C((x \otimes z) \otimes s))))
\end{array}$$

This sentence should follow from $\varphi_{\mathbf{CL}}$.

To prove the formula in classical logic from $\varphi_{\mathbf{CL}}$, we skolemise and get that the following assumptions must entail the conclusion.

1. atom (x_0)
2. $\forall^* y(B(y) \to C(x_0 \otimes y))$

3. atom (z_0)
4. $\forall^* u(A(u) \to B(z_0 \otimes u))$
5. atom (s_0)
6. $A(s_0)$

The conclusion is

$$C((x_0 \otimes z_0) \otimes s_0)$$

To prove this we substitute s_0 in (4) and get from (4) and (6) $B(z_0 \otimes s_0)$. We then substitute $(z_0 \otimes s_0)$ in (2) and get $C(x_0 \otimes (z_0 \otimes s_0))$. $\varphi_{\mathbf{CL}}$ ensures associativity and thus the desired conclusion is obtained.

We now try to prove left transitivity, namely try to prove

$$\varnothing : (A \to B) \to ((B \to C) \to (A \to C))$$

We translate in stages. The $*$ translation is:

$$\begin{aligned}\forall^* x \quad & ((A \to B)^*_x \to ((B \to C) \to (A \to C))^*_{\varnothing \otimes x}) \\ & = \forall^* x[\forall^* y(A(y) \to B(x \otimes y)) \to \forall^* z((B \to C)^*_z \to (A \to C)^*_{x \otimes z})) \\ & = \forall^* x(\forall^* y(A(y) \to B(x \otimes y)) \to \\ & \forall^* z(\forall^* u(B(u) \to C(z \otimes u)) \to \forall^* s(A(s) \to C((x \otimes z) \otimes s))))\end{aligned}$$

This sentence does not follow from $\varphi_{\mathbf{CL}}$. However, the sentence should be a theorem of classical logic, given the boolean axioms on $\otimes$ as a set union, ie $\varphi_{\mathbf{R}}$ proves this sentence.

The importance of the above translation into classical logic is twofold. First it allows us to have as a metalanguage for *LDS* a very familiar and comfortable logic, namely classical logic. Second it allows us to develop automated dedution capability for the new logics via existing automated deduction machines and techniques in classical logic. We may have to modify existing machines but in many cases this is worthwhile. Of course there will always be the need to do automated deduction directly on the new logic and not through translation. The translation however adds another dimension to our understanding of the new logics.

8.3 Linked predicate languages: classical logic as a metalanguage

So far we have discussed, to various degrees of detail, three seemingly independent families of concepts:

- We discussed the proposed *LDS* discipline.
- We discussed at the beginning of this chapter the possibility of two languages $\mathbf{L}_1$ and $\mathbf{L}_2$ being linked in some way and through that link one can implement some meta-level features of the other. This was shown to be a way of looking at *LDS*, namely that we are linking the labels of the algebra with the formulas.

  This linkage of $\mathcal{A}$ and $\mathbf{L}$ can be done for any two predicate systems within predicate logic.
- We also saw how an *LDS* system say $(\mathcal{A}, \mathbf{L})$ can be translated into two sorted classical logic, with two sorts, one for the labels and one for formulas. We thus have $t : A(x)$ is translated into $A^*(t, x)$. t is a term in $\mathcal{A}, A$ a wff in $\mathbf{L}$. The sort t is from the algebra and so has an algebraic language for manipulating it. A^* is a predicate of two sorts.

There is a more general way of looking at what we are doing, which is quite independent of the above particular examples. It has to do with linking two languages via the sharing of variables. The task of this section is to develop this theme.

We show how to prepare (separate, present) classical logic as a two sorted system so that it can serve as a target for translation. For this we need to 'link' predicate languages, which we denote by **G** and **L** to emphasize the complete generality.

Obviously a lot depends on the mechanism of 'linking'. We are going to clarify this notion for the case of two predicate logic theories. This will suffice for the handling of reduction of *LDS* or temporal logic into classical logic. **G** acts as object level language implementing meta-level restrictions on **L**.

Consider the first order language **G** with a binary relation $<$ and variables $\{t, s, \ldots\}$. Think of it as a theory of the flow of time. Consider the predicate language **L**, with variables $\{x_1, x_2, \ldots\}$ and predicates $A(x_1, \ldots, x_n)$, $B(y_1, \ldots, y_m)$. The formal operation of what we have done was to 'combine' these two languages by replacing **L** by $\mathbf{L}^*$, where $\mathbf{L}^*$ is a two sorted language, with variables $\{t, s, \ldots\}$ and $\{x_1, x_2, \ldots\}$ and taking as atomic predicates the two sorted predicates $A^*(t, x_1, \ldots, x_n)$, $B^*(s, y_1, \ldots, y_n)$, etc.

We allow for common quantification and mixed wffs. Thus we can write for example the mixed formula $\varphi(t)$

$$\varphi(t) = \forall s(t < s \to A^*(t, x_1, \ldots, x_n))$$

which you will recognize as the truth table for '$GA(x_1, \ldots, x_n)$ holds at t'.

Formally φ is just a formula in the mixed language. Let us give a quick definition to clarify our concepts before we continue our discussion.

Definition 8.3.1 (the two sorted language $\mathbf{L}^*_k(\mathbf{G})$) *Let* **L** *and* **G** *be the two languages. Both* **L** *and* **G** *have atomic predicates. Let* $\mathbf{L}^*_k$ *be obtained from* **L** *by replacing each atomic predicate* $R(x_1, \ldots, x_n)$ *of* **L** *by* $R^*(t_1, \ldots, t_k, x_1, \ldots x_n)$ *where* t_i *are variables of a new sort. Thus the resulting* $\mathbf{L}^*_k$ *is a two sorted language, with the new sort for t-variables. We allow quantification over t-variables. The intention is that the t-variables will be from the language* **G**.

1. *The atomic wffs of* $\mathbf{L}^*_k(\mathbf{G})$ *are of the form* $R^*(t_1, \ldots, t_k, x_1, \ldots, x_n)$, *where* $R(x_1, \ldots, x_n)$ *is an atomic wff of* **L** *and* $t_1, \ldots t_k$ *are terms of* **G**, *or of the form* $Q(t_1, \ldots, t_m)$, *where* $Q(t_1, \ldots, t_m)$ *is an atomic wff of* **G**.

2. *If* φ *and* ψ *are wffs of the language with t free in* φ *and x free in* φ *then* $\varphi \wedge \psi, \varphi \vee \psi, \varphi \to \psi, \sim \varphi, \forall t\varphi, \forall x\varphi, \exists t\varphi, \exists x\varphi$ *are wffs of the language.*

Example 8.3.2 1. *Let* **G** *be the language of order* $<$ *and let* **L** *be classical predicate logic.* $\mathbf{L}^*_1(\mathbf{G})$ *gives a language with atoms* $R^*(t, x_1, \ldots x_n)$, *which we used in the previous example to mean 'R is true at t'.*

2. *The language* **G** *can be a meta-language. Consider a propositional language, e.g. the propositional language with* $\neg, \wedge, \vee$ *and* $\to$. *For each wff A of the propositional language introduce a constant* τ_A. *Introduce the operations* $\mathbf{f}_\sim(\tau_A) = \tau_{\sim A}$ $\mathbf{f}_\wedge(\tau_A, \tau_B) = \tau_{A\wedge B}$ *and similarly* $\mathbf{f}_\vee$ *and* $\mathbf{f}_\to$. *Let* **G** *be the language with* τ_A *as terms and* $\{\mathbf{f}_\sim, \ldots\}$ *as functions. The language* $\mathbf{L}^*_1(\mathbf{G})$ *talks about the formulas of the propositional language as labels* $R^*(\tau_A, x_1, \ldots x_n)$ *represents* $A : R(x_1, \ldots x_n)$, *where A acts as a label.*

The above linkage is still not the most general case. To get an idea of what we need, consider the following set-up. Consider a distributed system of stations $s_1, s_2, s_3, \ldots$ related in some manner. Each station has its own internal structure and language **L**. Assume they all use the same language and logic. We also need a global configuration language to describe how these stations are related. This language denoted by **G** (for global) need not be the same logic as **L**. For example **G** can be based on intuitionistic logic, while **L** is based on classical logic. **L** may be a Horn clause logic programming language while **G** could be a form of Petri net langauge. The stations s_i need to communicate, so they would have output and input ports which can be accessed by the other stations. For simplicty assume that there are some variables $x_1, \ldots, x_k$ which can be accessed by the stations. This means that the language **G** can talk about x_i.

If we want to link these two languages into one, we can form the syntactic combination $\mathbf{L}^*_k(\mathbf{G})$ of the previous definition. The additional feature to worry about is that the logic of **G** and **L** may not be the same. Consider the following cases: ($A^*(t,x)$ could mean 'at station t mailbox x is down', and $B^*(t,x)$ can mean 'at station t printer x is down'.)

1. $A^*(t,x) \to B^*(s,y)$.
2. $A^*(t,x) \to B^*(t,y)$.
3. $A^*(t,x) \to B^*(s,x)$.
4. $A^*(t,x) \to B^*(t,x)$.

In cases 2 and 4 we can say that this is an internal statement of station t and hence the implication follows the logic of **L**. In case 3 we can use the logic of **G** as this is a relation between stations.

What do we do in case 1?

The simplest course of action is to understand it as a relationship between stations and hence use the logic of **G**. We thus have a simple rule. Any formula with *different* station constants in it is to be manipulated logically according to **G** but if the constants are equal, it is to be manipulated according to **L**. In fact once we agree to that, different stations can have different logics, $\mathbf{L}_t$. They must all be based on the same language.

We have one restriction on the system, that is the configuration logic must be sound with respect to the station logics. In symbols, $\Delta \vdash_{\mathbf{G}} A$ in the configuration logic implies $\Delta \vdash_{\mathbf{L}} A$ in the station logic. The reason for that is unification. We must be able to reason about two possibly different stations in **G** and if we identify them our reasoning (now in **L**) must still be valid.

We now consider the Horn clause fragment of $\mathbf{L}^*_k(\mathbf{G})$. The modal and temporal logics of the case studies used the language **G** with $<$, the linking of classical predicate logic with the theory of order $<$. In fact, we were operating in the Horn clause fragment of $\mathbf{L}^*_1(<)$. We should therefore define the Horn clause fragment of $\mathbf{L}^*_k(\mathbf{G})$ in general.

Definition 8.3.3 (The Horn clause fragment of $\mathbf{L}^*_k(\mathbf{G})$) *1. Any atom of the form $R^*(t_1,\ldots,t_k,x_1,\ldots,x_n)$ of $\mathbf{L}^*_k$ and $Q(t_1,\ldots,t_m)$ of* **G** *is in the Horn clause fragment and in the generated body fragment.*

2. If A and B are in the Horn clause fragment or in the generated body fragment then so are $A \wedge B, \forall xA, \exists xA, \forall tA, \exists tA$.

3. If A is in the generated body fragment and B is in the Horn clause fragment then $A \to B$ is in the Horn clause fragment.

Example 8.3.4 *1. If $A(t,x), B(s,y)$ are atomic, then $A' = \forall t\exists xA$ is in the atomic fragment and $C = A' \to \exists s\forall yB$ is in the Horn clause fragment. In classical logic, one can write C as:*

$$\exists t\forall x\exists s\forall y(A \to B)$$

because we can pull the quantifiers to the front and hence the previous definition can be simplified. In intuitionistic logic this is not possible and so we need clause 3 of the previous definition and the notion of atomic fragment, in order not to be committed to classical logic.

2. Notice that for $k = 1$ and **G** *the theory of order $<$, the Horn clause fragment of $\mathbf{L}^*_1(<)$ gives us the Horn fragment for temporal logic.*

The above discussion about linking two languages **L** and **G** is a general discussion of how a language **G** can be linked to a language **L** and 'influence it' by forming $\mathbf{L}^*_k(\mathbf{G})$. We gave no particular meaning to this operation beyond the 'meta-level' one. We did see that this linkage did generalize the special construction of **PC** in the previous section. **G** corresponded to the time order relation $<$ and **L** corresponded to the predicate logic. The linkage formulated in Definition 8.3.1 allows for linking arbitrary **G** and **L**.

Can we give modal or temporal meaning to $\mathbf{L}^*_k(\mathbf{G})$ for arbitrary **G** and **L**?

Furthermore, we have the following diagram, which we want to generalize:

$$\begin{tikzcd}
\begin{array}{l}\text{Temporal}\\ \text{logic }\mathbf{TL}\end{array} \arrow[r, leftrightarrow] \arrow[d] & \begin{array}{c}\text{Possible world}\\ \text{semantics with }<\end{array} \arrow[d] \\
\begin{array}{c}\mathbf{PC}=\mathbf{L}_1^*(<)\\ \text{where }\mathbf{L}=\text{ predicate logic}\end{array} \arrow[r, leftrightarrow] & \begin{array}{c}\text{First order language}\\ \mathbf{G}(\text{with }<)\end{array}
\end{tikzcd}$$

We want to complete the following diagram:

$$\begin{tikzcd}
\begin{array}{l}\text{What language?}\\ \text{Is it }\mathbf{Modal}_m(\mathbf{L},\mathbf{G})\end{array} \arrow[r, leftrightarrow] \arrow[d] & \begin{array}{l}\text{Possible world}\\ \text{semantics corresponding}\\ \text{to }\mathbf{G}\end{array} \arrow[d] \\
\begin{array}{c}\mathbf{L}_m^*(\mathbf{G})\\ \mathbf{L}=\text{ predicate logic}\end{array} \arrow[r, leftrightarrow] & \begin{array}{c}\mathbf{G}\text{ an arbitrary}\\ \text{predicate logic}\end{array}
\end{tikzcd}$$

What would be the counterpart of **TL** for arbitrary **G** and **L**?

Obviously we must generalize the notion of a temporal connective based on time ordering $<$ to some general modal connective based on some language **G**.

We are now ready to define a modal logic **Modal**(**L**, **G**) of **L** and **G**. To explain our options, let us reconsider the basic temporal connectives F and P. The truth table for $FA(x)$ is:
$FA(x)$ is true at t iff for some $s > t, A(x)$ is true at s.

If $A(x)$ is atomic, we translate it into $A^*(t, x)$, understanding $A^*(t, x)$ as '$A(x)$ is true at t' and thus we get:

$$FA(x) \text{ is true at } t \text{ iff } \exists s(t < s \wedge A^*(s, x))$$

or

$$(FA(x))^*_t = \exists s(t < s \wedge A^*(s, x)).$$

In $\mathbf{L}_1^*(<), <$ is the atomic relation of the order. $A^*(t, x)$ is an atomic relation of $\mathbf{L}_1^*$. 'F' represents the existential quantifier on the variable s in $t < s \wedge A^*(s, x)$ and it is denoted by 'F'. The existential quantifier on t would be $\exists t(t < s \wedge A^*(t, x))$ and that would correspond to 'P'.

Let us generalize this situation to $\mathbf{L}_1^*(\mathbf{G})$. Here the atoms $A^*(t, x)$ are still the same, but the atoms of **G** could be any $Q(t_1, \ldots, t_m)$. The parallel construction would be:

$$\exists t_i(Q(t_1, \ldots, t_i, \ldots, t_m) \wedge A^*(t_i, x)).$$

We could introduce for each atom $Q(t_1, \ldots, t_m)$ of **G** and each $1 \leq i \leq m$ the existential connective $\Diamond_{Q,i}A(x)$ to mean

$$(\Diamond_{Q,i}A)^*_t = \text{ (def) } \exists t_i[Q(\cdots t_i \cdots) \wedge A^*(t_i, x)].$$

Similarly we can define $\Box_{Q,i}A$

$$(\Box_{Q,i}A)^*_t = \text{ (def) } \forall t_i[Q(\cdots t_i \cdots) \to A^*(t_i, x)].$$

These correspond to 'G' and 'H' of temporal logic. To summarize we have:

$$\begin{array}{ll} \Box_{<,2} & = G \\ \Box_{<,1} & = H \\ \Diamond_{<,2} & = F \\ \Diamond_{<,1} & = P \end{array}$$

Let us consider what *Since*(A, B) does in this context.

$S(A, B)$ is true at t iff $\exists s < t[A$ is true at s and $\forall y(s < y \wedge y < t$ implies B is true at $y)]$.

Let ψ_S be the following formula in a language with $<$ and the two *new monadic* predicates Q_1 and Q_2:

$$\psi_S(t, Q_1, Q_2) = \text{ (def) } \exists s < t[Q_1(s) \wedge \forall y(s < y \wedge y < t \to Q_2(y))].$$

Then we have $S(A, B)$ is true at t iff $\psi_S(t, A^*, B^*)$, where A^*, B^* are considered monadic predicates in the **G** sort.

To generalize the above, let $\mathbf{G}^+_1$ be the extension of the language of **G** with new monadic predicates $\{Q_1, Q_2, \ldots\}$. Let $\psi(t, Q_1, \ldots, Q_n)$ be a formula of $\mathbf{G}^+_1$ with 1 free variable t and the n new monadic predicates. Let $A^*_1, \ldots, A^*_n$ be n formulas of $\mathbf{L}^*_1$, which can be considered monadic in the sort t. We can introduce the n place connective $\sharp(A_1, \ldots, A_n)$ with the truth table

$\sharp(A_1, \ldots, A_n)$ true at t iff $\psi(t, A^*_1, \ldots, A^*_n)$.

We will now give the general definition and an example.

Definition 8.3.5 *Let* **G**, **L** *be two languages. We construct the modal language* $\mathbf{Modal}_m(\mathbf{L}, \mathbf{G})$ *as follows:*

1. *Let* $\mathbf{G}^+_m$ *be the language* **G** *supplemented with an infinite sequence of* new *m place atomic predicates* $\{R_1, R_2, \ldots\}$.

2. *Let* $\mathbf{L}^*_m(\mathbf{G})$ *be the language obtained from* **L** *as in Definition 8.3.1. We can regard each atomic formula* $A^*(t_1, \ldots, t_m, x_1, \ldots, x_n)$ *as m-place atomic in the variables* $t_1, \ldots, t_m$.

3. *Let* $\mathcal{G}$ *be any model of* **G**. *We regard* $\mathcal{G}$ *as a model of possible worlds. With each* $t_1, \ldots, t_m \in \mathcal{G}$, *associate a model* $\mathcal{L}_{(t_1,\ldots,t_m)}$ *of* **L**. *Assume all the* $\mathcal{L}_{(t_1,\ldots,t_m)}$ *models have the* same *domain* D *(constant domain semantics). We refer to* $(\mathcal{G}, \mathcal{L}_{(t_1,\ldots,t_m)})$ *as a* modal model.

4. *For each wff* $\psi(t_1, \ldots, t_m, R_1, \ldots, R_n)$ *of* $\mathbf{G}^+_m$ *with* t_i *free and* R_i *new atomic predicates, associate a n place connective* $\sharp_\psi$. *The full modal language* $\mathbf{Modal}_m(\mathbf{L}, \mathbf{G})$ *is* $\mathbf{L} \cup \{\sharp_\psi \mid \psi \in \mathbf{G}^+_m\}$.

Definition 8.3.6 *Given a modal model* $(\mathcal{G}, \mathcal{L}_{(t_1,\ldots,t_m)})$ *and an assignment h of* **L** *into D, a model* $\mathcal{G}^h$ *of* $\mathbf{G}^+_m$ *is induced on* $\mathcal{G}$ *as follows.*

Let $A^*(t_1, \ldots, t_m, x_1, \ldots, x_n)$ *be an atom of* $\mathbf{L}^*_k$. *It can be viewed, for* $x_1, \ldots, x_n$ *fixed, as an atomic predicate of* $\mathbf{G}^+_m$. *Let the extension of this predicate in* $\mathcal{G}$ *be* $\{(t_1, \ldots, t_m) \mid \mathcal{L}_{(t_1,\ldots,t_m)} \vDash A(h(x_1), \ldots, h(x_n))\}$.

We define the notion of a formula $A(x_1, \ldots, x_n)$ *of* $\mathbf{Modal}_m(\mathbf{L}, \mathbf{G})$ *holding at the indices* $(t_1, \ldots, t_m)$, *notation* $\|A\|^h_{t_1,\ldots,t_m} = 1$, *as follows:*

1. $\|A\|^h_{(t_1,\ldots,t_m)} = 1$ *iff (def)* $\mathcal{L}_{t_1,\ldots,t_m} \vDash A(h(x_1), \ldots, h(x_n))$.
   $\|A\|^h_{t_1,\ldots,t_m} = 0$ *otherwise.*

2. $\|A \wedge B\|^h = 1$ *iff* $\|A\|^h = \|B\|^h = 1$.

3. $\| \sim A\|^h = 1$ *iff* $\|A\| = 0$.

4. *$\|\exists x A(x)\|^h = 1$ iff for some $d \in D$, and h_1, such that $h_1(x) = d$ and h_1 agrees with h on all other variables we have $\|A(x)\|^{h_1} = 1$.*

5. *$\|\sharp_\psi(A_1, \ldots, A_n)\|^h_{t_1,\ldots,t_n} = 1$ iff $\mathcal{G}^+_m \vDash \psi(t_1, \ldots, t_m, Q_1, \ldots, Q_n)$ where $\mathcal{G}^+_m$ is the extension of $\mathcal{G}$ to a model of the language $\mathbf{G}^+_m$ by the assignment $Q_i = \{(t_1, \ldots, t_m) \mid \|A\|^h_{t_1,\ldots,t_m} = 1\}$.*

6. *Let $\mathcal{K}$ be a class of $\mathbf{G}$ models $(\mathcal{G}, (s_1, \ldots, s_m))$ with a distinguised seqeunce of $s_i \in \mathcal{G}$. Let φ be a formula of $\mathbf{Modal}_m(\mathbf{L}, \mathbf{G})$. We say $\mathcal{K} \vDash \varphi$ iff for all models $(\mathcal{G}, \mathcal{L}_{(t_1,\ldots,t_m)})$ and all h, $\|A\|^h_{s_1,\ldots,s_m} = 1$.*

Example 8.3.7 *To show the generality of the previous definition, we give an example from resource logics. Let $\mathcal{G}$ be a commutative semigroup with multiplication $*$ and a unit $\mathbf{e}$. That is, we have the axioms:*

1. *$\forall xyz((x * y) * z = x * (y * z))$.*

2. *$\forall xy(x * y = y * x)$.*

3. *$\forall x(\mathbf{e} * x = x)$.*

Let $\mathbf{L}$ be classical propositional logic. Consider the formulas $\psi(t, Q_1, Q_2)$ and $\varphi(t, Q_1, Q_2)$ where:

$$\psi(t, Q_1, Q_2) = \ def\, \forall y[Q_1(y) \to Q_2(t * y)]$$

$$\varphi(t, Q_1, Q_2) = \ def\, \exists xy[Q_1(x) \wedge Q_2(y) \wedge t = (x * y)].$$

Then the logic with the connectives $\sharp_\psi(A, B)$ and $\sharp_\varphi(A, B)$ is linear logic with linear implication and external conjunction.

$$\sharp_\psi(A, B) = A \multimap B$$

$$\sharp_\varphi(A, B) = A \otimes B.$$

The semantics induced on these connectives by the general definition can be directly defined as follows.

The set of possible worlds is a semigroup $\mathcal{G}$. The propositional assignment h gives a truth value to every atom at a point i.e. $h(t, q) \in \{0, 1\}$ for $t \in \mathcal{G}$, q atomic. The truth conditions for $\multimap$ and $\otimes$ are as follows:

$$\|A \multimap B\|^h_t = 1 \text{ iff } \forall y(\|A\|^h_y = 1 \text{ implies } \|B\|^h_{t*y} = 1)$$

$\|A \otimes B\|^h_t = 1$ iff for some x, y we have $\|A\|^h_x = 1$ and $\|B\|^h_y = 1$ and $t = x * y$.

We have $\vDash A$ iff for all $\mathcal{G}$ and all h, $\|A\|^h_1 = 1$.

8.4 The meta-language HFP: computational classical logic

The previous two sections studied first the possiblity of classical logic serving as a meta-language, and second the general mechanism of linking two languages $\mathbf{G}$ and $\mathbf{L}$, so that the linkage can implement meta-language features of the modal system based on $\mathbf{G}$. In both cases our examples used classical logic for $\mathbf{L}$. Since different languages $\mathbf{G}$ seem to serve as the object language which implements the meta features involved turning classical logic $\mathbf{L}$ into different new logics $\mathbf{L}^*_1(\mathbf{G})$, we ask ourselves, is there a convenient general purpose language $\mathbf{M}$ which is specially suited for expressing meta language features, and which will contain all the $\mathbf{L}^*_1(\mathbf{G})$?

The answer is yes. The language we have in mind is the language **HFP**.[1]

The relevance of the existence of **HFP** to the Debate is as follows. We claim that classical logic or its variants are universal languages at least from the point of view of automated deduction. So

[1]In fact, the Logic Programming Community, when working in extensions of logic programming, are actually working in **HFP**, more specifically in Horn clause **HFP** which, you will be surprised to learn, is essentially Micro-PROLOG. Unfortunately, no one uses Micro-PROLOG anymore.

given for example an arbitrary logic **N** we can translate it into $\mathbf{L}^*_k(\mathbf{G})$, for some suitable **G** (usually obtained from the semantics or equivalent LDS proof theory of **N**), an appropriate dimension k and classical logic **L**. This method suffers from some disadvantages. The target system is many sorted and may be computationally inconvenient. We also get for different $\mathbf{N}_1, \mathbf{N}_2, \ldots$ different target many sorted languages $\mathbf{L}^*_{k(i)}(\mathbf{G}_i), i = 1, 2, \ldots$. We would like one very nice language which is convenient and can serve any logic **N**. We propose the language **HFP** and we will show at the end of this section how any logic **N** and any $\mathbf{L}^*_k(\mathbf{G})$ can be nicely translated into **HFP**.

We begin by describing the intuition behind the language **HFP**. Consider first propositional temporal logic, formalized in the usual way, via the addition of extra connectives to classical propositional logic. Assume the connectives are FA and PA, 'A will be true' and 'A was true' respectively.

In symbols:

$$\|FA\|_t = 1 \text{ iff } \exists s > t \quad \|A\|_s = 1$$

$$\|PA\|_t = 1\text{iff } \exists s < t \quad \|A\|_s = 1.$$

Suppose we want to choose the first moment s_0 such that $s_0 > t$ and $\|A\|_{s_0} = 1$, assume such a moment exists. This moment is a function of t and A. Let us denote it by $s_0 = f^1_1(A, t)$. Here f^1_1 is a two place function, taking a formula and a point and yielding a point.

Consider now the connective F itself, this connectives takes a formula A yields a formula FA, thus its function is $R^1_0(A)$.

The functionality of f^1_1 and R^1_0 can be described set theoretically. If X is the set of all points in which A is true, we get

$$f^1_1(X, t) = \text{ min } (X \cap \{s \mid s > t\})$$

$$R^1_0(X) = \{s \mid \exists t \in X(s < t)\}.$$

In general we can have general connectives and functions of the form f^m_n, R^m_n, taking m formulas and n points and yielding a point or a formula respectively.

To give another example, consider the connective $S(A, B)$. Its meaning is that since a point in the past where A was true, B has been true all the time. We can now form a new connective $S^x_y(A, B)$ reading since a point t in the past which is different from x in which A is true, B has been continuously true at all points since which are greater than y. This connective has the form $S(A, B, x, y)$.

The language **HFP** is a general language in which tables and connectives can be described. It allows for predicates R^m_n and function symbols f^m_n to take both formulas and terms as arguments.

The use of **HFP** is not restricted to the meta-level of modal and temporal languages. It is also the meta-language of actual PROLOG programs. Although officially logic programming deals with the Horn clause fragment of classical logic (possibly with negation by failure), namely with clauses of the form

$$\bigwedge A_i \to B$$

where A_i and B are atoms, in practice the PROLOG programmer uses **HFP** as the language. (Negation by failure can be incorporated using a meta-level failure predicate).

Expressions like *Hold*(A, B), *Demo*$(A(x), B(y))$ are written in PROLOG. The above are expressions of **HFP**. It may be thought that *Demo* and *Hold* are metapredicates. This is partially correct. They have special properites. The variables x and y in *Demo*$(A(x), B(y))$ can be quantified and unified. Thus *Demo* is not like the general metapredicates of a pure meta-language **M** but rather special and we claim it is like in **HFP**. (We shall see later in this section how to quantify on seemingly object level variables occurring in a meta-level predicate.)

We see in Definiton 8.4.3 below that **HFP** is given modal semantics. So the question is whether this semantics is adequate for the way PROLOG uses the language. We will be in a better position to answer in the next section. The answer in general is no, PROLOG predicates are not in general modal connectives, but under certain restrictions they can be viewed as such though this is not the most convenient way of looking at them.

We are now ready to define the general meta-language **HFP**, of *Hereditarily Finite Predicates.*

Definition 8.4.1 (The Language HFP) *The language contains the classical connectives $\sim, \wedge, \vee, \rightarrow$ and the variables $\{x, y, z, \ldots\}$ for terms and variables $\{X, Y, Z, \ldots\}$ for formulas, and the quantifiers $\forall$ and $\exists$. There is a list of atomic predicates R_n^m allowing us to form $R_n^m(\psi_1, \ldots, \psi_m, x_1, \ldots, x_n)$, where ψ_i are formula variables or formulas and x_j are term variables or terms . There are function symbols f_n^m allowing us to form $f_n^m(\psi_1, \ldots, \psi_m, x_1, \ldots x_n)$, where ψ_i are formula variables and x_j are term variables. We define now the notions of a* wff *and a* term *of the language* **HFP**.

1. *$\top$ is a formula with no free variables.*
2. *x is a term with x free. Y is a formula with Y free.*
3. *If $\psi_i(x_1^i, \ldots x_{k(i)}^i), i = 1, \ldots, m$, are formulas with free variables $\{x_1^i, \ldots, x_{k(i)}^i\}$ and $F_j(\overline{x}_1, \ldots, \overline{x}_{r(j)}), j = 1, \ldots, n$, are terms with $\{\overline{x}_1^j, \ldots, \overline{x}_{r(j)}^j\}$ as free variables and R_n^m is a predicate symbol with (m, n) places and f_n^m is a function symbol with (m, n) places then:*
   - (a) *$R_n^m(\psi_1, \ldots, \psi_m, F_1, \ldots, F_n)$ is a formula with the above free variables, namely $(\overline{x}_j^i, x_s^t), i = 1, \ldots, m, j = 1, \ldots, k(i), t = 1, \ldots, n, s = 1, \ldots, r(t)\}$.*
   - (b) *$f_n^m(\psi_1, \ldots, \psi_m, F_1, \ldots, F_n)$ is a term with the same free variables as in (a).*

   The free variables mentioned can be either term or formula variables.
4. *If ψ_1, ψ_2 are wffs, then so are $\sim \psi_1, \psi_1 \wedge \psi_2, \psi_1 \rightarrow \psi_2, \psi_1 \vee \psi_2$. $\sim \psi_1$ has the same free variables as ψ_1, and $\psi_1 \wedge \psi_2, \psi_2 \vee \psi_2$ and $\psi_1 \rightarrow \psi_2$ have a set of free variables which is the union of the sets of free variables of ψ_1 and ψ_2.*
5. *If $\psi(x, y_i)$ is a formula with free variables $\{x, y_i\}$ being either term or formula variables then $\forall x\psi, \exists x\psi$ are formulas with the free variables $\{y_i\}$.*

Example 8.4.2 (Hold predicate) *To give some examples consider the* **Hold** *predicate. This has the form* **Hold** (ψ, t) = *'ψ is true at time t'.*

We can now write **Hold**(**Hold**$(\psi, t), s)$

'At time s it was true that ψ was true at t';

compare this sentence with **Thought**(**Thought**$(\psi, 0), 5)$

'At time 5 people thought that at time 0 it was thought that ψ was true';

consider **Thought-F**(ψ, t) *which reads:*

'At time t, ψ was considered true in the future'.

Thus we can write a rule: If at any time you expect to get a budget for a Research Assistant, then hire one. (It may be that you never get the budget!).

$\forall t[$**Thought-F**$($*budget, t*$) \rightarrow$ **Hire**$(t)]$.

We now define several types of models for this language.

Definition 8.4.3 (modal semantics) *Let D be a nonempty set. A function h is an assignment for* **HFP** *iff the following holds:*

1. *If x is an individual term variable then $h(x) \in D$. If x is a formula variable then $h(x) \subseteq D$.*
2. *If R_n^m is a predicate symbol then $h(R_n^m)$ is a function with domain $(2^D)^m \times D^n$ and range 2^D.*
3. *If f_n^m is a function symbol then $h(f_n^m)$ is a function with domain $(2^D)^m \times D^n$ and range D.*

4. *Given a formula $\psi(x_1, \ldots, x_n)$ with free variables $x_1, \ldots x_n$ and a term $F(x_1, \ldots, x_n)$ with free variables $x_1, \ldots, x_n$ we define the value $h^*(F) \in D$ and $h^*(\psi) \subseteq D$ as follows:*
   (a) $h^*(x) = h(x)$.
   (b) $h^*(\top) = D$.
   (c) $h^*(R_n^m)(\psi_1, \ldots, \psi_m, F_1, \ldots, F_n) =$
   $h(R_n^m)(h^*(\psi_1), \ldots, h^*(\psi_m), h^*(F_1), \ldots, h^*(F_n))$.
   (d) $h^*(f_n^m(\psi_1, \ldots, \psi_m), F_1, \ldots, F_n) =$
   $h(f_n^m)(h^*(\psi_1), \ldots, h^*(\psi_m), h^*(F_1), \ldots, h^*(F_n))$.
   (e) $h^*(\sim \varphi) = D - h^*(\varphi)$
   $h^*(\varphi \wedge \psi) = h^*(\varphi) \cap h^*(\psi)$.
   (f) $h^*(\forall x \psi(x)) = \bigcap_{d \in D} h^*_{x/d}(\psi(x))$
   where $h_{x/d}$ is the function which differs from h only by having $h_{x/d}(x) = d$
   i.e.
   $h_{x/d}(y) = h(y)$ *for* $y \neq x$
   $h_{x/d}(y) = d$ *for* $y = x$.
   Similarly
   $h^*(\exists x \psi(x)) = \bigcup_{d \in D} h^*_{x/d}(\psi(x))$.

5. (a) *A model is a domain and an assignment, (D, h).*
   (b) *A formula ψ holds in a model if $h^*(\psi) = D$.*
   (c) *A formula ψ is valid if it holds in all models.*

Example 8.4.4 1. *Consider classical propositional temporal logic. We can associate with any atomic proposition q a 0-place predicate Q of* **HFP**. *The domain D is time and assume an order $<$ on D. Q is a $(0,0)$ predicate. Let $\mathbf{F}$ be the $(1,0)$ predicate $\mathbf{F}(Q)$. Let $h_<(\mathbf{F})(Y)$, for $Y \subseteq D$ be the set $Y^+ = \{s \mid \exists y \in Y s < y\}$. Then if $h_<(Q)$ is the set where q is true then $h^*_<(FQ)$ is the set of points where Fq is true.*

2. *Consider the example in (a) above and consider the connective (predicate) $\mathbf{F}(Q, x)$. $\mathbf{F}$ is a $(1,1)$ predicate. It corresponds in the propositional temporal logic to a labelled connective $F^x q$. We may have for example $F^x q$ true at t iff $t < x$ and q true at x. In* **HFP** *the assignment would be:*
$$h_<(\mathbf{F})(Y, x) = \{t \mid t < x \textit{ and } x \in Y\}.$$

3. *Notice that the* **Hold** *predicate of Example 8.4.2 should be of type $\mathbf{Hold}_0^1$, i.e.* **Hold**(A) *if the modal semantics associates a set with it. Thus* **Hold**$(A) \equiv A$, *i.e.* $h($**Hold**$) \equiv$ *identity.*

The above discussion gave to **HFP** modal semantics. This semantics is adequate for describing the modal and temporal applications of **HFP**, as it talks about truth conditions of connectives. However, it does not seem satisfactory in explaining the way **HFP** predicates are used in practical PROLOG programs. Whenever we write in PROLOG a clause of the form

$$\mathit{Hold}(\mathit{Demo}(A(x), B(y)) \wedge A(z)) \rightarrow B(z).$$

We are using **HFP** expressions involving predicates of predicates. Sometimes we even write clauses of the form
$$X \rightarrow P(X)$$
which play an important role in metaprogramming.

For example the clause
$$\mathbf{Hold}(\varphi) \text{ if } \varphi$$
may look more familiar to the reader.

The logical status of these phrases and their semantics have to be explained. The main characteristic feature of such use is that the variables in the clauses can unify and change value. Thus it is not immediately obvious whether one can say that in the expression $\varphi(A(x), x)$, $A(x)$ is a name of a formula. If $A(x)$ represents the wff $A(x)$ (i.e. what we really have is $\varphi($'$A(x)$', $x)$, then the 'x' in $A(x)$ cannot change or unify, certainly not in uniformity with the other x. In practice of course when 'x' is unified to be 'a', we get $\varphi(A(a), a)$ and not $\varphi($'$A(x)$'$, a)$.)

We thus need to explain the above for the case of **HFP**, because like in PROLOG, **HFP** allows for x to be free in $\varphi(A(x), x)$ in both places.

Notice that in the modal semantics the above are interpreted as connectives. So for example $\varphi(B, y)$ can be interpreted as the connective saying: *B will always be true but not later than time* y, and $A(x)$ can be interpreted as saying: *Time x is in the future (of now).* Together we form $\varphi(A(x), x)$ to mean: *Time x will always be in the future but not later than time* x, which is really a temporal tautology.

We now proceed to give the *meta-level semantics* for **HFP**, as opposed to the *modal semantics.* Our strategy is to begin with ordinary predicate logic, describe a meta-language **M** for it and identify **HFP** as part of the meta-level description of predicate logic in **M**.

Definition 8.4.5 (Term translation) *Let* **L** *be predicate logic with predicate symbols* $\{P_i, i = 1, 2, 3, \ldots\}$ *which are* n_i *place; variables* $\{x_i, y_i, \ldots\}$ *and constants* $\{a_i, b_i, \ldots\}$*. The language* **L** *may contain the classical connectives and in general a stock of m-place connectives of the general form* $\sharp(A_1, \ldots, A_m)$*. We refer to* **L** *as* the object language*. A language* **M** *is said to be* a meta-language *for* **L** *if it is capable of naming the symbols of* **L** *and contains the following representing terms (this translation is referred to as* term translation*):*

1. *For each symbol of* **L** *there exists a possibly unique name which is a term of* **M**. *The naming function is denoted by quotation marks.*

   - *variables* x_i *are named '*x_i*'*
   - *constants* a_i *are named '*a_i*'*
   - *predicate symbols* P_i *are named '*P_i*'.*

2. *For each* P_i*, which is* n_i *place predicate, there exists a function* $\mathbf{f}_i$ *of the meta-language* **M***, satisfying the following:*
   $\mathbf{f}_i$*('*P_i*','*t_1*', ..., '*t_{n_i}*') = '*$P_i(t_1, \ldots t_{n_i})$*'.*
   In fact, we can take the above as a definition of the name of $P_i(t_1, \ldots, t_{n_i})$*.*

3. *There exist functions* $\mathbf{f}_\wedge, \mathbf{f}_\vee, \mathbf{f}_\rightarrow, \mathbf{f}_\sim, \mathbf{f}_\forall$ *and* $\mathbf{f}_\exists$ *and* $\mathbf{f}_\sharp$ *for each m-place connective, satisfying equations as follows:*

   - $\mathbf{f}_\wedge$ *('A', 'B') = '*$A \wedge B$*'*
   - $\mathbf{f}_\vee$ *('A', 'B') = '*$A \vee B$*'*
   - $\mathbf{f}_\rightarrow$ *('A', 'B') = '*$A \rightarrow B$*'*
   - $\mathbf{f}_\sim$ *('A') = '*$\sim A$*'*
   - $\mathbf{f}_\forall$ *('A', 'x') = '*$\forall x A(x)$*'*
   - $\mathbf{f}_\exists$*('A', 'x') = '*$\exists x A(x)$*'*
   - $\mathbf{f}_\sharp$ *('*A_1*', ..., '*A_m*') = '*$\sharp(A_1, \ldots, A_m)$*'.*

4. *There are functions* **Sub**, **s** *and* **n** *satisfying the following, for terms t:*
   n*(t) = 't'*
   s*('t') = t*
   Sub *('*$A(x)$*', 'x',* $\mathbf{n}(t)$*) = '*$A(t)$*'*
   where t is any term and 't' is its name.

Note that **Sub** is a substitution function. We can assume that $\mathbf{f}_\sharp, \mathbf{f}_i, \mathbf{f}_\wedge, \mathbf{f}_\vee$, $\mathbf{f}_\sim$, $\mathbf{f}_\exists$ and $\mathbf{f}_\forall$ are actual function symbols of **M** and thus actually define the terms which name the formulas. **Sub** cannot be assumed as a function symbol because then a formula might have several names. In general meta-level language with functions representing substitution, the names will not be unique. Several function symbols including the substitution function symbol can combine to yield names for a formula in several different ways. If we choose a unique name via our formula construction functions as we did, the substitution function may not be representable.

So we have to assume that **Sub** is somehow definable in **M**. An alternative is to take **Sub** as a function symbol and add axioms to **M** to make things right.

Lemma 8.4.6 *Let $\varphi(x_1, \ldots, x_n)$ be a formula, then there exists a term $\mathbf{f}_\varphi(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of* **M** *with free variables $\mathbf{x}_1, \ldots, \mathbf{x}_n$ (ranging over names) such that*
$\mathbf{f}_\varphi$ *(‘a_1’, …, ‘a_n’) = ‘$\varphi(a_1, \ldots, a_n)$’.*

Proof. By induction on the structure of φ. For atomic $\varphi = P_i(x_1, \ldots x_{n_i})$, we have

$$\mathbf{f}_\varphi = \text{ (def) } \mathbf{f}_i(\text{‘}P_i\text{’}, \mathbf{x}_1, \ldots, \mathbf{x}_{n_i}).$$

If there exist terms $\mathbf{f}_A$ and $\mathbf{f}_B$ for A and B then $\mathbf{f}_{A \sharp B} = \mathbf{f}_\sharp(\mathbf{f}_A, \mathbf{f}_B)$ for $\sharp$ standing for $\wedge, \vee, \rightarrow$ and similarly for $\sim$ and for a general m-place $\sharp$.

Let $\varphi = \forall x \psi(x)$. Let $\mathbf{f}_\psi$ be given. Then $\mathbf{f}_\varphi = \mathbf{f}_\forall(\mathbf{f}_\psi, \text{‘}x\text{’})$. Similarly for $\varphi = \exists x \psi(x)$. ■

Definition 8.4.7 (meta-level semantics) *Let $\mathcal{M}$ be a model of* **M**. *Let* **Hold** *be a unary predicate of* **M**. *We can derive from $\mathcal{M}$ and* **Hold** *a model $\mathcal{A}$ of* **L** *by defining*

$$\mathcal{A} \vDash P(a_1, \ldots, a_n) \textit{ iff (def) } \mathcal{M} \vDash \mathbf{Hold}(\text{‘}P(a_1, \ldots, a_n)\text{’}).$$

We called **Hold** *a Hold predicate because we expect that for any $\varphi(a_1, \ldots, a_n)$ we have:*

$$\mathcal{A} \vDash \varphi(a_1, \ldots, a_n) \textit{ iff } \mathcal{M} \vDash \mathbf{Hold}(\text{‘}\varphi(a_1, \ldots, a_n)\text{’}).$$

This will not be the case unless we require axiomatically in **M** *that* **Hold** *satisfies some axioms for example:*

$$\mathbf{Hold}(\text{‘}A\text{’}) \wedge \mathbf{Hold}(\text{‘}B\text{’}) \leftrightarrow \mathbf{Hold}(\mathbf{f}_\wedge(\text{‘}A\text{’}, \text{‘}B\text{’}))$$

and similarly for the other connectives and quantifiers.

Note the quantifier axioms are:

$$\forall \mathbf{x} \mathbf{Hold}(\mathbf{f}_{\varphi(x)}(\mathbf{x})) \leftrightarrow \mathbf{Hold}(\mathbf{f}_{\forall x \varphi(x)})$$

$$\exists x \mathbf{Hold}(\mathbf{f}_{\varphi(x)}(\mathbf{x})) \leftrightarrow \mathbf{Hold}(\mathbf{f}_{\exists x \varphi(x)}).$$

Remark 8.4.8 *Note that if we have* **Sub** *then we can define the diagonal function by:*

Diag *(‘$A(x)$’, ‘x’) = def* **Sub** *(‘$A(x)$’, ‘x’, $\mathbf{f}_\sim$ (‘$A(x)$’)).*

Recall that using the **Diag** *function one can prove Tarski's inconsistency theorem, that there exists no truth predicate for* **M** *in* **M**. *Thus we cannot have a* **Hold** *predicate satisfying the axiom*

Hold *(‘A’) $\leftrightarrow$ A*

and allow **Diag** *to apply to* **Hold** *as well, because these conditions yield Tarski's inconsistency theorem. We need not worry however about this aspect. First, our aim is to provide semantics for* **HFP** *and* PROLOG *practice in order to do our ‘computational classical logic’, rather than construct self reflective languages which have their own truth predicates. So we do not mind to have the* **Hold** *predicate not to apply to itself. See the Perlis papers [Perlis, 1985, Perlis, 1988] for a good coverage of the subject. Second, in the Horn clause fragment of* **HFP** *negation is not available and it is* *possible to have a truth predicate in the object level for the object language itself, without leading to contradiction. The worst we can get are loops, if we use negation as failure. We cannot diagonalize with A('$\neg A$'), with $\neg$ being negation as failure because its meaning is procedural. Even when it is given a declarative semantics, say* $\mathbf{sem}(A)$*, we will get the formula* $A(\sim \mathbf{sem}(`A'))$ *which is not a direct diagonalization. More directly we can define a* **fail** *connective via a program Δ, then we can only have* $\mathbf{fail}_\Delta$ *as a connective and we get* A('$\mathbf{fail}_\Delta A$') *for the diagonal function. In this case we get stratification of* **fail** *and we do not have a universal object level diagonalization.*

We shall not pursue this topic any further, and will continue with the business at hand.

We are now in a position to explain what a formula of **HFP** stands for. We refer to the interpretation presented in the sequel as *the term interpretation* for **HFP**. Compare with Definition 8.4.5. The object language **L** of that definition is **HFP** and the meta-language **M** of the definition is classical logic. Note that since **HFP** extends classical logic we can say we are translating it into itself.

Suppose we name each letter x, P_i, a by itself. This can technically mean that we identify **L** as part of **M**. Thus P_i, x_i and a_i are already terms of **M**. We let 'a_i'$= a_i$, 'P_i' $= P_i$ and 'x' $= x$ in **M**, for these particular symbols. We can define predicates **Pred**(t), **Term**(t), **Var**(t), **wff**(t) and **Free** of **M** with the axioms:

- **Term** $(a_i), i = 1, \ldots$
- **Var** $(x_i), i = 1, \ldots$
- **Pred** $(P_i), \ldots$
- **wff** $(\mathbf{f}_i(P_i, a_1, \ldots, a_{n_i})$
- **wff** $(\mathbf{f}_i(P_i, x_1, \ldots, x_{n_i}))$
- **wff** $(\mathbf{f}_i(P_i, \alpha_1, \ldots, \alpha_{n_i})$, where α_i are metalinguistic variables representing a choice of either a variable x or a term a of **L**. So for each such choice we get an axiom of **M**.
- **Free** $(\mathbf{f}_i(P_i, \alpha_1, \ldots, x_j, \ldots, \alpha_{n_i}), x_j)$
- **wff** $(A) \wedge \mathbf{wff}(B) \to \mathbf{wff}(\mathbf{f}_\sharp(A, B))$
- **wff** $(A) \wedge \mathbf{Free}(A, x) \to \mathbf{Free}(\mathbf{f}_\sharp(A, \mathbf{y}), x)$
  where $\mathbf{y}$ is a metavariable and $\sharp$ is one of $\wedge, \vee, \to, \sim$.
- **wff** $(A) \wedge \mathbf{Free}(A, x) \to \mathbf{wff}(\mathbf{f}_\forall(A, x))$
- $\mathbf{wff}(A) \wedge \mathbf{Free}(A, x) \to \mathbf{wff}(\mathbf{f}_\exists(A, x))$
- $\mathbf{wff}(A) \wedge \mathbf{Free}(A, \mathbf{y}) \wedge x \neq \mathbf{y} \to \mathbf{Free}(\mathbf{f}_\forall(A, x)), \mathbf{y})$
  where $\mathbf{y}$ is a metavariable, i.e. a variable of **M**.

Having named formulas by themselves we can essentially regard the functions $\mathbf{f}_i(P_i, x_1, \ldots x_{n_i}), \mathbf{f}_\wedge, \mathbf{f}_\to$ etc. to be the connectives themselves, taken as functions. So $\mathbf{f}_\varphi$('x_1', ... 'x_n') will be '$\varphi(x_1, \ldots, x_n)$' which is $\varphi(x_1, \ldots, x_n)$.

If we consider $\mathbf{f}_\varphi(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ with $\mathbf{x}_i$ metavariables ranging over names, then in the convention of expressions of **L** naming themselves we get that it is equal to

$$\varphi(\mathbf{x}_1, \ldots, \mathbf{x}_n).$$

Thus we can unify and change $\mathbf{x}_i$ since they are metavariables.

Consider $\varphi(\mathbf{x}, \mathbf{y})$. This is really a term of **M**. We can certainly substitute in it the term $A(\mathbf{y})$. We thus get $\varphi(A(\mathbf{y}), \mathbf{y})$ which is an expression of **HFP**. It really stands for the term $\mathbf{f}_\varphi(\mathbf{f}_A(\mathbf{y}), \mathbf{y})$ of the meta-language.

Satisfaction in a model $\mathcal{A}$ of $\varphi(A(\mathbf{y}),\mathbf{y})$ means according to the meta-level semantics the satisfaction in a model $\mathcal{M}$ of:

$$\mathbf{Hold}(\mathbf{f}_\varphi(\mathbf{f}_A(\mathbf{y}),\mathbf{y})).$$

Since the meta-language is classical logic, the above term will look more familiar if we write it as $\mathbf{f}_1(\mathbf{f}_2(y),y))$, using two unary Skolem functions, $\mathbf{f}_1=\mathbf{f}_\varphi$ and $\mathbf{f}_2=\mathbf{f}_A$. Clearly in classical logic we can have unary predicates such as $Q(x)$. We can consider the formulas $Q(\mathbf{f}_2(y))$ and $Q(\mathbf{f}_1(\mathbf{f}_2(y),y))$. These are two non-equivalent formulas of classical logic. In an arbitrary model, one formula may be true while the other may be false, unless we have the equality axiom $\mathbf{f}_2(y)=\mathbf{f}_1(\mathbf{f}_2(y),y)$.

We need one more definition to clarify our meta-level concepts. Let Q be a classical unary predicate and let $\mathbf{g}(x_1,\ldots,x_n)$ be an n-place function. Let $\mathcal{M}$ be a model of classical logic with domain D. Define the Q denotation of $\mathbf{g}$ in $\mathcal{M}$ to be $\{(d_1,\ldots,d_n)\mid d_i\in D$ and $\mathcal{M}\vDash Q(\mathbf{g}(d_1,\ldots,d_n)\}$.

For the case where $Q=\mathbf{Hold}$, and $\mathbf{f}_2,\mathbf{f}_2$ are as above we can examine more closely the connection between the satisfaction of $A(d)$ and $\varphi(A(d),d)$. The model $\mathcal{A}$ has as its domain D terms of the meta-language $\mathbf{M}$ which are names of wffs of $\mathbf{L}$. $A(d)$ holds in $\mathcal{A}$ iff $\mathbf{Hold}(A(d))$ holds in $\mathcal{M}$.

The set $\{d\in D\mid\mathcal{M}\vDash\mathbf{Hold}(A(d))\}$ is the denotation of 'A' (i.e. of $\mathbf{f}_A$) and is a subset of the domain D. The denotation of 'φ' (i.e. $\mathbf{f}_\varphi$) is similarly a subset of $D\times D$. For $d_0\in D,\mathbf{f}_A(d_0)=A(d_0)$ '$A(d_0)$' is a term in D. $\mathcal{A}\vDash A($'$A(d_0)$'$)$ may or may not hold, i.e. $\mathcal{M}\vDash\mathbf{Hold}(A(A(d_0)))$ may or may not hold. There is *no* connection in general between the denotation of 'A' and the term '$A(d_0)$'. We do not necessarily have $\mathbf{f}_A(d_0)\in\{d\in D\mid\mathcal{M}\vDash\mathbf{Hold}(\mathbf{f}_A(d))\}$.

Thus in the model $\mathcal{M}$ and consequently in $\mathcal{A}$ the predicate 'A' has three 'manifestations' or 'footprints', one as $\{d\in D\mid\mathcal{M}\vDash\mathbf{Hold}(A(d))\}$ one as a function symbol $\mathbf{f}_A$ and one as a term 'A' in D (i.e. the name of $\mathbf{f}_A$). **Hold** ('A') also has a value in $\mathcal{M}$, so does $\mathbf{Hold}($'$A($ 'A'$)$'$)$ etc. These values are all independent. An **HFP** axiom of the form $X\to B(X)$ for X formula variable can be taken to mean in $\mathbf{M}$ the axiom **Hold** $(X)\to\mathbf{Hold}(B(X))$, which is a connection axiom of the **Hold** in $\mathbf{M}$. For example the axiom $\mathbf{Hold}(A(y))\to\mathbf{Hold}(A(A(y)))$, means $\mathbf{Hold}(\mathbf{f}_A(y))\to\mathbf{Hold}(\mathbf{f}_A(\mathbf{f}_A(y)))$.

We can in $\mathbf{M}$ read 'A' as representing the set $\{y\mid\mathcal{A}\vdash A(y)\}$. Terms of the form $\mathbf{f}_\varphi($'A'$,\mathbf{y})$ can be written in the meta-language and we can understand them in the object language as $\varphi(\lambda xA(x),y)$.

Example 8.4.9 *The modal and temporal semantics provides us with examples of the above situation. A formula may be true at points of the interval* $t=[1,2]$ *of time but whether it is true at the interval itself is another matter. For example, it is not true that one crossed the Atlantic at any moment at the interval* $[1,2]$ *but one may crossed the Atlantic at* $[1,2]$. *Connections between values at intervals and values at points are usually given as persistence criteria. For example, the criterion* if φ holds at an interval it holds at all subintervals, *can be expressed using our language, through axioms like:*

$$\varphi(\text{`}A\text{'},y)\to\forall x\subseteq y\ \varphi(\text{`}A\text{'},y)$$

or

$$\forall x(A(x)\to B(x))\to[\varphi(\text{`}B\text{'},y)\to\varphi(\text{`}A\text{'},y)]$$

where 'A' is the name of the predicate 'cross the Atlantic' and 'B' is the name of 'sail a boat' and φ is some metapredicate, for example $\varphi=$**Hold**.

The above freedom we still have with terms allows us to extend the expressive power of **HFP**. First let us take any binary atomic predicate of **HFP**, let us call it $P(X,Y)$ where X and Y are formula variables.

Form the formulas

$$P^2(A_1,A_2)=\text{ (def) }P(A_1,A_2)$$

$$P^{n+1}(A_1,\ldots,A_{n+1})=\text{ (def) }P(P^n(A_1,\ldots,A_n),A_{n+1}).$$

In the language $\mathbf{M}$, there are no axioms connecting $\mathbf{Hold}(P(X,Y))$ with $\mathbf{Hold}(P(P(X,Y),Z))$ and so in $\mathcal{M}$, $\mathbf{Hold}(P^n(X_1,\ldots,X_n))$ has no connection with $\mathbf{Hold}(P^m(X_1,\ldots,X_m))$ for $m\neq n$. However, this trick allows us to regard 'P' as a sequence predicate (i.e. without a fixed number of places) with $P(X_1,\ldots,X_n)$ holding in $\mathcal{A}$, iff $\mathbf{Hold}(P^n(X_1,\ldots,X_n))$ holds in $\mathcal{M}$.

We can thus write our predicate as a set of sequences $(P,X_1,\ldots,X_n)$, where n is arbitrary.

We can similarly treat function symbols f and write $(f, x_1, \ldots, x_n)$. If we do the above for any pure formula predicate of the language **HFP**, we can identify the following Lisp-like fragment of **HFP**.

Definition 8.4.10 (the sublanguage $\mathbf{HFP}_{Lisp}$) *The sublanguage contains formula variables $X_1, X_2, \ldots$, term variables and Lisp-predicate letters $P_1, P_2, P_3, \ldots$.*

The general notion of a wff of $\mathbf{HFP}_{\mathbf{Lisp}}$, with free variables is defined in the same way as in the case of Definition 8.4.1 of **HFP**, *except that we replace the formation rule for atoms 8.4.1(3) by the following formation rule ($3_{\mathbf{Lisp}}$): If n is arbitrary and $\varphi_1, \ldots, \varphi_n$ are formulas, or predicate letters or formula variables and P is a predicate letter then $(P, \varphi_1, \ldots, \varphi_n)$ is a formula. We allow the case $n = 0$ in which case (P) is a formula.*

The reader should note the translation $*$ from $\mathbf{HFP}_{\mathbf{Lisp}}$ into **HFP**. Each P of $\mathbf{HFP}_{\mathbf{Lisp}}$ is read in **HFP** as a binary predicate.

$(P)^* = P(\mathbf{truth}, \mathbf{truth})$
$(P, \varphi)^* = P(\mathbf{truth}, \varphi^*)$
and

$$(P, \varphi_1, \ldots, \varphi_n)^* = P^n(\varphi_1^*, \ldots, \varphi_n^*).$$

HFP can be translated into a fragment of $\mathbf{HFP}_{\mathbf{Lisp}}$, via the translation $\sharp$:

$$P(\varphi_1, \ldots, \varphi_n)^\sharp = (P, \varphi_1^\sharp, \ldots, \varphi_n^\sharp).$$

We are now interested in defining the Horn-clause fragment of **HFP** and of $\mathbf{HFP}_{\mathbf{Lisp}}$.

Definition 8.4.11 (Horn Clause fragment of HFP) *We follow closely the defining clauses of 8.4.1 and of 8.4.10. Read 'wff' as 'wff in the the Horn fragment':*

1. *Same as in 8.4.1*

2. *Same as in 8.4.1*

3. or $3_{\mathbf{Lisp}}$. *Same as in 8.4.1 and 8.4.10 with the restriction that the formulas involved (i.e. $\varphi_1, \ldots, \varphi_n, \psi_1, \ldots, \psi_m, F_1, \ldots, F_n$) are all taken from the Horn fragment.*

4.
   1. *If ψ_1 and ψ_2 are in the Horn fragment, so is $\psi_1 \wedge \psi_2$.*
   2. *If ψ_i, φ are atomic i.e. the result of applying clause (3) or ($3_{\mathbf{Lisp}}$) respectively, then $\wedge\psi_i \to \varphi$ is in the Horn fragment.*

5. *Same as in 8.4.1 and 8.4.10*

The next question to ask is:

Do we have an implementation of the Horn fragment?

The answer is surprisingly yes, we do have in fact an old implementation, as the next remark shows.

Remark 8.4.12 *The reader is referred to K. Clark and F. McCabe's book [Clark and McCabe, 1984] on Micro-*PROLOG. *Chapter 9 describes exactly the Micro-*PROLOG *system language which is the Horn clause fragment of $\mathbf{HFP}_{\mathbf{Lisp}}$. Thus there exists implementations of the Horn clause fragment of this language. Micro-*PROLOG *on the IBM PC gives access to this language. Later versions of LPA-*PROLOG *concentrated on supporting Edinburgh syntax and therefore the original Micro-*PROLOG *syntax has been suppressed in these later versions. Another implementation for the full* **HFP** *is essentially the rewrite language* **PLL** *of E. Babb [Babb, 1990].*

The discussion of the section so far showed that if we name in **M** elements of **HFP** by themselves, we get that all formulas of **HFP** are terms in **M**. The term functions $\mathbf{f}_i, \mathbf{f}_\wedge, \mathbf{f}_\vee, \mathbf{f}_\to, \mathbf{f}_\forall, \mathbf{f}_\sim, \mathbf{f}_\exists$, and **Sub** of Definition 8.4.5 all become familiar operations:

1. The naming function λx'x' is the identity.

2. $\mathbf{f}_i(P_i, t_1, \ldots, t_{n_i}) = P_i(t_1, \ldots, t_n)$
   Thus $\mathbf{f}_i$ is the P_i application function.

3. $\mathbf{f}_\wedge$ is conjunction $\wedge$
   $\mathbf{f}_\vee$ is disjunction $\vee$
   $\mathbf{f}_\to$ is implication $\to$
   $\mathbf{f}_\sim$ is negation $\sim$
   $\mathbf{f}_\forall$ is universal quantification
   $\mathbf{f}_\exists$ is existential quantification.

4. **Sub** is the substitution function
   s, **n** are the identity function
   $\mathbf{f}_\varphi$ for a formula φ becomes an application function for φ (just like $\mathbf{f}_i$ is for P_i).

The above functions however are functions in **M** and can therefore take any term of **M**. Assume P and Q are two unary predicates of **L**. Their names in **M** are also 'P' and 'Q'. Given a term a of **L** its name is also 'a'. The application functions $\mathbf{f}_P$ and $\mathbf{f}_Q$ of **M** allow us to form:

$$\mathbf{f}_P(\text{`}P\text{'}, \text{`}a\text{'})$$

and we are assured that:

$$\mathbf{f}_P(\text{`}P\text{'}, \text{`}a\text{'}) = \text{`}P(a)\text{'} = P(a) = \mathbf{f}_P(P, a).$$

We can, however, formally obtain the following terms of **M**.

1. $\mathbf{f}_P(\text{`}Q\text{'}, \text{`}a\text{'})$.

2. $\mathbf{f}_P(\text{`}P\text{'}, \text{`}Q\text{'})$.

We have no axioms in **M** to tell us what the above should mean. The **Hold** predicate of **M** would be true or false of the above terms without any restrictions. We do have axioms on **Hold** to assure us that e.g.

$$\mathbf{Hold}(\text{`}P(a)\text{'} \wedge \text{`}Q(a)\text{'}) \leftrightarrow \mathbf{Hold}(\text{`}P(a)\text{'}) \wedge \mathbf{Hold}(\text{`}Q(a)\text{'})$$

but nothing more, except for some general properties of **Hold**.

To extend our semantic meaning to terms of the form (1) and (2) above, we can first say that $\mathbf{f}_P, \mathbf{f}_Q$ should be the same, namely Application Functions. We can thus replace the n_i place function $\mathbf{f}_i$ by $\mathbf{App}_i$ and get

$$\mathbf{App}_{n_i}(\text{`}P_i\text{'}, t_1, \ldots, t_{n_i}) = \text{`}P_i(t_1, \ldots, t_{n_i})\text{'}.$$

Thus all we need are the functions $\mathbf{App}_1, \mathbf{App}_2, \ldots$. So both $\mathbf{f}_Q$ and $\mathbf{f}_P$ are $\mathbf{App}_1$. This view allows us to 'compare' with general λ-abstraction languages.

Furthermore, one can code $\mathbf{App}_3(x, y, z)$ as $\mathbf{App}_2(\mathbf{App}_2(x, y), z)$ and thus all one needs is a binary **App** function. Further, there is no reason why we cannot take application to be concatenation. Thus $\mathbf{f}_i$ can be read as concatenation function. This also allows us to regard all predicates as unary predicates. $R(x, y)$ is understood either as $R(x)(y)$ or as $R((x, y))$. Probably the former is more convenient.

The above syntactical moves do not necessarily involve new semantical commitments. They can be handled as just a matter of notation. We are still left with the problem of the semantical meaning of $P(Q)$ or $P(P)$. If we understand Q to name the set $\{x \mid Q(x)\}$, then we can understand $P(Q)$ as expressing a property of this set. We can now express persistence conditions like

$$P(Q) \to \forall x(Q(x) \to P(x)).$$

In general, $P(Q)$ is independent of $P(t_i)$, for t_i such that $Q(t_i)$ holds. We can see however, that we are touching on the problem of λ-calculus models.

We can still go on and ask what is the meaning of **App**(‘a’, ‘Q’). This question has now more than one option, since we have already agreed to read ‘Q’ as a name for $\{x \mid Q(x)\}$. We can thus read ‘a’ as a higher predicate on sets Q. It is true of Q iff it is an element of Q:

$$\mathbf{Hold}(a(Q)) \text{ iff } \mathbf{Hold}(Q(a)).$$

‘a’ can thus be identified with the set of all its properties. This is similar to the Montague semantics.

It is now up to us whether to allow in **HFP** (say in an extension which we can call $\mathbf{HFP}^+$) formulas of the form $P(Q)$ or not. If we do, we can still use the functions of the meta-language **M** but we would need a semantical interpretation, possibly some sort of λ-calculus model. We will not pursue this line of research in this Chapter. PROLOG practice, which uses **HFP** formulas, does not frequently write expressions like:

Demo (*Demo*, *Demo*).

Let Δ be an **HFP** theory. Formally extend Δ to a saturated theory Δ'. Thus whenever $\Delta' \vdash \exists x A(x)$ then for some terms $t, \Delta' \vdash A(t)$ and whenever $\Delta' \vdash A \vee B$ then either $\Delta' \vdash A$ or $\Delta' \vdash B$. We can now define $\mathbf{Hold}_{\Delta'}(A)$ iff $A \in \Delta'$ and get a model for the **Hold** predicate of **M** as applied to the wffs of **HFP** regarded as **M** terms.

The term interpretation of **HFP** can be understood in another way, as notation for generating predicates. To illustrate our view, consider conjunction. It can generate predicates by letting $[P \wedge Q](x)$ mean $P(x) \wedge Q(x)$. This is done in the same way that $[\sim A](x)$ is $\sim A(x)$ and $[\Box A](x) = \Box A(x)$.

If we push this point of view further we can regard any formula $\varphi(x,y)$ of **HFP** as a functional $\lambda xy\varphi$ which takes formulas A, B to form a new formula $\varphi(A,B)$. This view, although *mathematically* compatible with the term semantics, is not *conceptually* compatible with the intuitive way we read PROLOG. For example, we read *Demo*(A,B) as B can succeed from A and regard *Demo* (*Demo*, x) as meaningless.

Example 8.4.13 *We can now show how an arbitrary* LDS *can be translated into the Horn clause fragment of* **HFP**. *Assume the languages* **G** *and* **L** *of* LDS *are distinct. Consider a version of* **HFP** *based on the union of the languages of* **G** *and* **L** *together with the additional predicate* $\mathbf{Label}(x,y)$. *we translate* $\alpha : A$ *as* $\mathbf{Label}(\alpha, A)$ *and all axioms and rules are translated as is. For example*

$$\frac{\alpha : A,\ \beta : A \Rightarrow B}{\beta * \alpha : B}$$

is translated as

$\mathbf{Label}(\alpha : A) \wedge$ **Label** $(\beta, A \Rightarrow B) \rightarrow$ **Label** $(\beta * \alpha, B)$.

8.5 How to give an LDS formulation for a logic

Previous sections discussed *LDS* formulations of various logics. To widely apply the *LDS* discipline we must be able to give an *LDS* formulation to an arbitrary logic. It seems that many logics can be naturally presented through their algebraic semantics. A typical example is many valued logics, where the algebraic presentation is the most natural. Thus if we want to claim that a general logic can be presented as an *LDS* in a nice way, we must investigate suitable methods of using the algebraic semantics of a logic to turn it into an *LDS*. We are restricting ourselves to propositional algebraic semantics for two reasons. First because almost all known and useful non-classical logics use the usual quantifiers $\forall$ and $\exists$ and differ in their propositional part. Second we find it too difficult and complex to handle general non-standard quantifiers.

We begin by describing what we mean by algebraic semantics for a logic. We want a very general definition that would equally apply to many systems, whether they are monotonic or non-monotonic. There are many systems which have none of the classical connectives and which satisfy only some very restricted proof theory. There is no agreed standard definition of what is algebraic logic or semantics. Our primary concern in this Chapter is translation into classical logic, so any good definition will do. As it turns out our definitions are of independent interest. The new notion of algebraic logics for general consequence relations will be studied in [Gabbay, 1992b].

Logics can be identified via a general consequence relation $\Delta \mid\!\sim A$. This consequence relation will in general be non-monotonic. So we may not know, or have much to say about their consequence relation. We need minimal conditions on $\mid\!\sim$ to enable us to define a reasonable algebraic semantics for the logics which will allow us to define good translations into other logics.

Experience leads us to the following definition (which should be compared with more traditional definitions).

Definition 8.5.1 *Let* **L** *be a language with atomic propositions and connectives.*

1. *A unitary consequence relation is a relation $\mid\!\sim$ between finite (including empty $\varnothing$) sets of wffs Δ and single wffs A, which satisfies the following:*

   - Identity

   $$A \mid\!\sim A$$

   - Unitary cut

   $$\frac{\Delta \mid\!\sim A; A \mid\!\sim B}{\Delta \mid\!\sim B}.$$

   Note that we do not require reflexivity $(\Delta, A \mid\!\sim A)$ nor do we require any full version of cut such as:

   $$\frac{\Delta \mid\!\sim A; \Gamma, A \mid\!\sim B}{\Gamma, \Delta \mid\!\sim B}.$$

   Also note that we need not have substitutivity of equivalents, namely $A \mid\!\sim B$ and $B \mid\!\sim A$ need not imply $\varphi(A) \mid\!\sim \varphi(B)$, for arbitrary wff $\varphi(q)$. Many modal systems fail to have this property, as well as intuitionistic systems with Nelson's strong negation.

   We call the consequence relation unitary because we allow for unitary cut, the 'unit' being the single A in the rule $A \mid\!\sim B, \Delta \mid\!\sim A$ implies $\Delta \mid\!\sim B$. Also Δ being a set of single, unannotated formulas which is unstructured, as opposed to Δ being a list of wffs or a network of wffs etc.

2. *A Hilbert consequence relation is a relation defined only for $\varnothing \mid\!\sim A$. A singleton consequence relation is a relation defined only for $\Delta \mid\!\sim A$, where $\Delta = \{B\}$ or $\Delta = \varnothing$.*

3. *Logical systems are usually formulated either as a Hilbert consequence, in which case we have only the notion of $\varnothing\mid\!\sim A$ but not $A\mid\!\sim B$, or as a singleton or unitary consequence. When a Hilbert $\mid\!\sim_1$ is given, there exists the smallest and biggest singleton $\mid\!\sim_2$ which agrees with it (i.e. $\varnothing\mid\!\sim_1 A$ iff $\varnothing\mid\!\sim_2 A$, for all A). This will be proved later.*

   Many nonmonotonic logics do not satisfy transitivity and unitary cut. We need to weaken these notions. Here is an example of a more general consequence relation in a language with $\to$ which satisfies the following rule in addition to identity but is not required to satisfy unitary cut or transitivity:

   $$\frac{\begin{array}{l} A \mid\!\sim B \to C \\ X \mid\!\sim B \end{array}}{A \mid\!\sim X \to C}.$$

Definition 8.5.2 *1. A logic algebra is any algebra of the form* $\mathbf{a} = (A, f_i, \leq, T)$*, where A is a non-empty set (corresponding to the formulas of the logic) and $f_i, i = 1, \ldots, k$, are functions on A, with arity n_i (corresponding to the connectives of the logic) and $\leq$ is a reflexive and transitive relation on A (corresponding to the converse of the consequence* *relation of the logic) and $T \subseteq A, T \neq \varnothing$ is a subset of distinguished elements (corresponding to the theorems of the logic) satisfying:*

$$(*) \quad \forall x, y \in A \; [y \in T \text{ and } x \leq y \text{ implies } x \in T].$$

2. *A subset $F \subseteq A$ is a* filter *if $F \neq \varnothing$ and for all x, y ($x \in F$ and $x \leq y$ implies $y \in F$).*

3. *A function $f(x_1, \ldots, x_n)$ in a logic algebra is* extensional *iff*

$$x_i \leq y_i \wedge y_i \leq x_i, i = 1, \ldots, n \text{ imply } f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n).$$

4. *A function $f(x_1, \ldots, x_n)$ in a logic algebra is* directional *if it is either monotonic up or down in each of its variables.*

5. *An algebra is* directional *if all of its functions are directional.*

Definition 8.5.3 1. *A connective $\sharp(A_1, \ldots, A_n)$ is* extensional *iff the following holds:*

$$\frac{x_i \mid\!\sim y_i, y_i \mid\!\sim x_i, i = 1 \ldots n}{\sharp(x_1, \ldots, x_n) \mid\!\sim \sharp(y_1, \ldots, y_n)}.$$

2. *A connective $\sharp(A_1, \ldots, A_n)$ is directional in A_1 iff either*

   (a) Upward monotonicity

$$\frac{p \mid\!\sim q}{\sharp(p, A_2, \ldots) \mid\!\sim \sharp(q, A_2, \ldots)}.$$

   (b) Downward monotonicity

$$\frac{p \mid\!\sim q}{\sharp(q, A_2 \ldots) \mid\!\sim \sharp(p, A_2 \ldots)}.$$

3. *A logic is extensional (directional) iff all of its connectives are extensional (directional, resp.) in all their variables.*

Example 8.5.4 1. *Intuitionistic logic is directional. $A \to B$ is directional up in B and down in A. So are all known substructural implications such as relevant, linear and Lambek implications.*

2. *Extensionality allows us to give neighbourhood semantics for the logic or to give it a Lindebaum–Tarski algebraic semantics. In the case of a language with one modal operator $\Box$, and the classical propositional connectives $\neg, \wedge, \vee$ and $\to$, extensionality of $\Box$ means we can give $\Box$ the usual neighbourhood semantics. Both $\Box$ and $\Diamond$ are monotonic upwards but satisfy different algebraic conditions.*

3. *Let $\sharp(x)$ be a connective and suppose it is both directional up and down in x. Then essentially $\sharp(x)$ does not depend on x.*

   To see this, assume we have either truth $\top$ or falsity $\bot$ in the language. Then for any x we have

$$\bot \mid\!\sim x \mid\!\sim \top.$$

   Hence we get $\sharp(\bot) \mid\!\sim \sharp(x) \mid\!\sim \sharp(\top)$ and $\sharp(\top) \mid\!\sim \sharp(x) \mid\!\sim \sharp(\bot)$.

   This means $\sharp(x)$ is independent of x.

   In case neither $\top$ nor $\bot$ are in the language, we can prove 'equivalence', namely $\sharp(y) \mid\!\sim \sharp(x)$ and $\sharp(x) \mid\!\sim \sharp(y)$, only for x, y that are R-connected, where R is the transtive closure of $\leq$ and its converse. This is enough to allow us for Definition 8.5.8, (1) to go through.

We now describe algebraic semantics for a logic $\mathbf{L}$, with consequence relation $\mid\!\sim$. The semantics will be considerably simplified for the case of directional logics.

Definition 8.5.5 *1. Let $\mathbf{L}$ be a logic with consequence $\mid\!\sim$ and connectives $\sharp_1, \ldots, \sharp_k$ with arities $r_1, \ldots, r_k$ respectively. Let $\mathbf{a} = (A, f_1, \ldots, f_k, \leq, T)$ be an algebra with functions f_i with arities r_i resp.*

(a) An algebraic assignment for $\mathbf{L}$ into $\mathbf{a}$ is a function $h^{\mathbf{a}}$ assigning for each atomic q an element $h^{\mathbf{a}}(q) \in A$.

(b) $h^{\mathbf{a}}$ can be extended to arbitrary wffs of $\mathbf{L}$ by

$$h^{\mathbf{a}}(\sharp_i(A_1, \ldots, A_{r_i})) = f_i(h^{\mathbf{a}}(A_1), \ldots, h^{\mathbf{a}}(A_{r_i})).$$

(c) We write $(\mathbf{a}, h^{\mathbf{a}}) \vDash A$ iff $h^{\mathbf{a}}(A) \in T$. We write $A \vDash_{\mathbf{a},h^{\mathbf{a}}} B$ iff $h^{\mathbf{a}}(B) \leq h^{\mathbf{a}}(A)$. In future we omit the '$\mathbf{a}$'.

2. An algebra is connected if any two elements are connected by the transitive closure of $R = (\leq \cup \geq)$ (i.e. of $\leq$ and its converse). This condition relates to the connectivity mentioned in Example 8.5.4 (3).

Theorem 8.5.6 (Completeness for general logics) *Let $\mid\!\sim$ be a consequence relation then $A \mid\!\sim B$ iff for all directional algebras $\mathbf{a}$ and all assignments $h^{\mathbf{a}}$, we have $A \vDash_{h^{\mathbf{a}}} B$.*

Proof.

1. Soundness is immediate from the reflexivity and transitivity of $\leq$.

2. Completeness is obtained by taking the *canonical algebra* with $A =$ set of all wffs and defining $x \leq y$ iff $y \mid\!\sim x$. We let $T = \{B \mid \varnothing \mid\!\sim B\}$. Let $f_\sharp(A_1, \ldots, A_n) = \sharp(A_1, \ldots A_n)$ and let h be the assignment giving $h(q) = q$.

One can prove by induction on A that $h(A) = A$. ■

Theorem 8.5.7 *Let $\mid\!\sim$ be an extensional consequence relation. Then it is complete for the class of all extensional algebras.*

Proof. Immediate from the definitions and the canonical model of the previous theorem. ■

Definition 8.5.8 *1. Let $\mathbf{L}$ be a directional logic with consequence $\mid\!\sim$ and connectives $\sharp_1, \ldots, \sharp_k$ with arities $r_1, \ldots, r_k$. Let $\mathbf{a} = (A, f_i, \leq, T)$ be a connected logic algebra with function symbols $f_i, i = 1, 2, \ldots$ with arities $r_1, r_2, \ldots$ resp (same as the connectives). Let $h^{\mathbf{a}}$ be a function (assignment) assigning for each atom q of $\mathbf{L}$ a filter $F_q \subseteq A$. Assume f_i are directional in the same way as $\sharp_i$. We can extend $h^{\mathbf{a}}$ to all wffs of $\mathbf{L}$ inductively as follows. Let $\sharp_j(A_1 \ldots, A_{n_j}, B_1, \ldots, B_{m_j})$ be a connective monotonic down in A_i and monotonic up in B_i. We let $h^{\mathbf{a}}(\sharp_j(A_1, \ldots, A_{n_j}, B_1, \ldots, B_{m_j})) = \{y \mid$ for all $x_1, \ldots, x_{n_j}$ such that $x_i \in h^{\mathbf{a}}(A_i)$ there exist y_k such that $y_k \in h^{\mathbf{a}}(B_k), i = 1, \ldots, n_j$ and $k = 1, \ldots, m_j$, we have $f_j(x_1, \ldots x_{n_j}, y_1, \ldots, y_{m_j}) \leq y\}$.*

If a connective is monotonic both up and down in a slot we regard it as monotonic down. (We could regard it as montonic up—it does not matter.)

We show that the definition given here leads to the same assignment filter set no matter what we choose.

Let $\sharp(A, B, C)$ be a connective where the B slot is directional both ways, up and down.

We have two options.

(a) Taking B as directional up:
$S_1 = h(\sharp(A, B, C) = \{y \mid$ for all x_1, x_2 such that $x_1 \in h(A), x_2 \in h(B)$, there exists a $x_3 \in h(C)$ such that $f_\sharp(x_1, x_2, x_3) \leq y\}$.

(b) *Taking B as directional down:*
$S_2 = h(\sharp(A,B,C)) = \{y \mid$ *for all* x_1 *such that* $x_1 \in h(A)$, *there exists* x_2, x_3 *such that* $x_2 \in h(B), x_3 \in h(C)$ *such that* $f_\sharp(x_1,x_2,x_3) \leq y\}$.

We want to show that $S_1 = S_2$ *under the assumption that the algebra is connected (as in Definition 8.5.5).*

First note that clearly if $y \in S_1$ *then* $y \in S_2$. *Assume* $y \in S_2$. *Show* $y \in S_1$. *Since* $y \in S_2$ *then for every* $x_1 \in h(A)$, *there are* $x_2 = \alpha(x_1) \in h(B)$ *and* $x_3 = \beta(x_1) \in h(C)$ *such that* $f_\sharp(x_1, \alpha(x_1), \beta(x_1) \leq y$. *We want to show that for every* $x_1, x_2, x_1 \in h(A), x_1 \in h(B)$ *there exists a* $\gamma(x_1,x_2) \in h(C)$ *such that* $f_\sharp(x_1,x_2,\gamma(x_1,x_2)) \leq y$. *We let* $\gamma(x_1,x_2) = \beta(x_1)$. *we must show that for all* $x_1 \in h(A), x_2 \in h(B), f_\sharp(x_1,x_2,\beta(x_1)) \leq y$.

Since the second slot (x_2 *slot in* $f_\sharp$*) is directional both up and down and since we assumed the algebra is connected, we get in view of the discussion in Example 8.5.4 (3) that*

$$f_\sharp(x_1,x_2,\beta(x_1)) \leq f_\sharp(x_1,\alpha(x_1),\beta(x_1)) \leq y.$$

Thus we have shown that $S_1 = S_2$.

2. *An algebraic model for* $\mid\!\sim$ *is a pair* $(\mathbf{a}, h^{\mathbf{a}})$, *as in 1. above. An algebraic semantics is a class of algebraic models.*

3. *Let* $\mathbf{m} = (\mathbf{a}, h^{\mathbf{a}})$ *be an algebraic model and let* A, B, Δ *be wffs, we write:*

$$\begin{array}{rcl} A \mid\!\sim_{\mathbf{m}} B & \textit{iff} & h^{\mathbf{a}}(A) \subseteq h^{\mathbf{a}}(B) \\ \mid\!\sim B & \textit{iff} & h^{\mathbf{a}}(B) \subseteq T \\ \Delta \mid\!\sim_{\mathbf{m}} B & \textit{iff} & \bigcap_{A\in\Delta} h^{\mathbf{a}}(A) \subseteq h^{\mathbf{a}}(B) \end{array}$$

4. *Let* $\mathcal{K}$ *be a class of algebraic models. We say*
   - $A \mid\!\sim_{\mathcal{K}} B$ *iff* $A \mid\!\sim_{\mathbf{m}} B$ *for all* $m \in \mathcal{K}$.
   - $\mid\!\sim_{\mathcal{K}} B$ *iff* $\mid\!\sim_{\mathbf{m}} B$ *for all* $\mathbf{m} \in \mathcal{K}$.
   - $\Delta \mid\!\sim_{\mathcal{K}} B$ *iff* $\Delta \mid\!\sim_{\mathbf{m}} B$ *for all* $\mathbf{m} \in \mathcal{K}$.

5. *Let* $\mathbf{L}$ *be a language and* $\mid\!\sim$ *a directional consequence relation. We define the canonical algebra* $\mathbf{a}_{\mid\sim}$ *as follows:*
   - $A_{\mid\sim}$ = *the set of all wffs of* $\mathbf{L}$.
   - $f_\sharp$ *for each connective* $\sharp$ *of* $\mathbf{L}$ *is defined by*

   $$f_\sharp(A_1,\ldots,A_n) = \sharp(A_1,\ldots,A_n).$$

   - $A \leq B$ *is defined as* $B \mid\!\sim A$.
   T *is* $\{B \mid \varnothing \mid\!\sim B\}$.
   - *Let* $h^{\mathbf{a}}(q) = \{A \mid A \mid\!\sim q\}$. $h^{\mathbf{a}}(q)$ *is a filter because if* $A \in h^{\mathbf{a}}(q)$ *(i.e.* $A \mid\!\sim q$*) and* $A \leq B$ *(i.e.* $B \mid\!\sim A$*) then certainly* $B \in h^{\mathbf{a}}(q)$. *Also* $h^{\mathbf{a}}(q)$ *is non empty (since* $q \mid\!\sim q$*).*

Definition 8.5.9 (Term translation for a consequence relation) *Let* $\mathbf{L}$ *and* $\mid\!\sim$ *be as in the previous Definition. Let* A *be a wff of* $\mathbf{L}$ *with atoms* $q_1,\ldots,q_n$. *Let* $z_1,\ldots,z_n$ *be variables of classical logic associated with* $q_1,\ldots,q_n$. *In fact, let* q^* *be the variable associated with* q. *With each connective* $\sharp$ *of* $\mathbf{L}$ *associate a function symbol* $f_\sharp$ *of the same arity. We associate with any formula* A *a term of classical logic* A^* *by structural induction as follows:*

- $(q_i)^* = z_i$
- $(\sharp(A_1,\ldots,A_n))^* = f_\sharp(A_1^*,\ldots,A_n^*)$.

Consider the classical language with the function symbols $\{f_\sharp\}$, the binary relation symbol $\leq$ and the unary predicate symbol T. Then the translation $* : A \mapsto A^*$, translates each formula A into a term in this language. The consequence assertion $A {\mid\!\sim} B$ is translated into $B^* \leq A^*$ and the assertion $\varnothing {\mid\!\sim} A$ is transated into $A^* \in T$.

The consequence relation rules of the form

$$\frac{A_i \mid\!\sim B_i, i = 1, \ldots, k}{\sharp_1(C_1, \ldots, C_{n_1}) \mid\!\sim \sharp_2(D_1, \ldots, D_{n_2})}$$

are transated into classical logic Horn clauses of the form

$$\bigwedge_{i=1}^{k} (B_i^* \leq A_i^*) \to f_{\sharp_2}(D_j^*) \leq f_{\sharp_1}(C_j^*)$$

where '$\to$' is classical implication.

For example, axioms schema such as $A \Rightarrow (B \Rightarrow A)$ become expressions like

$$f_\Rightarrow(x, f_\Rightarrow(y, x)) \in T.$$

Modus ponens becomes

$$x \in T \wedge f_\Rightarrow(x, y) \in T \to y \in T.$$

Lemma 8.5.10 (Completeness) *In the canonical model,*

$$h^{\mathbf{a}}(B) = \{A \mid A \mid\!\sim B\}.$$

Proof. By induction.

1. The lemma holds for atomic B.

2. Assume the lemma holds for $A_1, \ldots, A_n, B_1, \ldots, B_m$ we show the lemma holds for

$$B = \sharp(A_1, \ldots, A_n, B_1, \ldots, B_m)$$

We assume $\sharp$ is monotonic down in A_i and up in B_k.

$$\begin{aligned} h^{\mathbf{a}}(B) = \{y \mid & \text{ for all } X_1, \ldots, X_n, \text{ such that } X_i \mid\!\sim A_i \\ & \text{there are } Y_1 \ldots Y_m, \text{ such that } Y_k \mid\!\sim B_k \\ & \text{we have } y \mid\!\sim \sharp(X_1, \ldots, X_n, Y_1, \ldots, Y_m)\}. \end{aligned}$$

We want to show

$$h^{\mathbf{a}}(B) = \{y \mid y \mid\!\sim \sharp(A_1, \ldots, B_m)\}.$$

Assume that $y \in h^{\mathbf{a}}(B)$ and show that $y \mid\!\sim B$. From the assumptions we see that for $X_i = A_i$, there are $Y_i \mid\!\sim B_i$ such that

$$y \mid\!\sim \sharp(A_i, Y_i).$$

Since $\sharp$ is monotonic up in Y_i we get $y \mid\!\sim B$.

Assume $y \mid\!\sim B$, we show $y \in h^{\mathbf{a}}(B)$. Let X_i be such $X_i \mid\!\sim A_i$, we need to find Y_k such that $y \mid\!\sim \sharp(X_i, Y_k)$. Take $Y_k = B_k$. We need to show $y \mid\!\sim \sharp(X_i, B_k)$. Since $\mid\!\sim$ is monotonic down in A_i, we get that $X_i \mid\!\sim A_i$ yield $\sharp(A_i, B_k) \mid\!\sim \sharp(X_i, B_k)$ and hence $y \mid\!\sim \sharp(X_i, B_k)$, by transitivity

This completes the proof of the lemma. Note that we proved that each filter is generated by a minimal element. ■

Example 8.5.11 *Let $X \twoheadrightarrow Y$ be a binary connective. Let $\mid\!\sim$ be the smallest consequence relation which makes $\twoheadrightarrow$ directional down in X and up in Y,*

$$\frac{X \mid\!\sim X'}{X' \twoheadrightarrow Y \mid\!\sim X \twoheadrightarrow Y} \qquad \frac{Y \mid\!\sim Y'}{X \twoheadrightarrow Y \mid\!\sim X \twoheadrightarrow Y'}$$

and satisfies the two standard conditions

$$A \mid\!\sim A \text{ and } \frac{A \mid\!\sim B; B \mid\!\sim C}{A \mid\!\sim C}.$$

The semantics for this system involve algebras of the form $\mathbf{a} = (A, f_{\twoheadrightarrow}(x,y), \leq, T, h)$ with the truth condition on $\twoheadrightarrow$ being:

$t \vDash A \twoheadrightarrow B$ iff for all $x \vDash A$ there exists a $y \vDash B$ such that $f_{\twoheadrightarrow}(x,y) \leq t$.

We have $A \mid\!\sim B$ iff for all $\mathbf{a}$ and all $t \in A^{\mathbf{a}}, t \vDash A$ implies $t \vDash B$.

This semantics should be compared with the more traditional semigroup semantics for substructural implications. A model has the form $(A, *, \leq, h)$ where $(\mathcal{A}, *)$ is a semigroup (associative?) and $\leq$ an ordering. The clause for $\twoheadrightarrow$ is

$$t \vDash A \twoheadrightarrow B \text{ iff } \forall x \vDash A, t * x \vDash B.$$

The two semantics will coincide if we take $\leq$ as identity and let

$$f_{\twoheadrightarrow}(x,y) = (\text{The } z \text{ such that})(z * x = y).$$

The previous definition of canonical model and the completeness theorem allow us to give a semi-algebraic possible world semantics for directional logics. The following definition establishes the connection.

Remark 8.5.12 (A more general completeness theorem) *The completeness theorem for Kripke-like semantics can be generalized. We need not assume the consequence relation be transitive. The following conditions are sufficient:*

- *$A \mid\!\sim A$ identity*
- *Directionality of all connectives (as in Definition 8.5.3)*
- *Downward strengthening: Let $\sharp(q,\ldots)$ be any connective which is downward monotonic in q, then for any A and B*
$$\frac{A \mid\!\sim \sharp(q,\ldots); B \mid\!\sim q}{A \mid\!\sim \sharp(B,\ldots)}.$$

Let $\mathbf{a}$ be any algebra satisfying reflexivity, directionality and downward strengthening. Let $t \in A^{\mathbf{a}}$, define a semi-filter I_t to be the largest *set Y satisfying $(*)$ below:*

$$(*) \qquad \forall s \in A[s \leq t \text{ iff } \forall x \in Y(s \leq x)].$$

Clearly $I_t^0 = \{t\}$ is such a set and the family of such sets is closed under union.

Let h be an assignment. Require h to satisfy:

$$(**) \qquad h(t,q) = 1 \text{ iff } (\forall s \in I_t)h(s,q) = 1.$$

Then in the canonical model, we have for such h and any A

$$t \vDash A \text{ iff } t \mid\!\sim A.$$

The proof is similar to that of Lemma 8.5.10. Downward strengthening plays the role of transitivity in the proof. In fact, if $\mid\!\sim$ is transitive then downward strengthening follows and each semifilter I_t is equal to $\{s \mid t \leq s\}$.

Definition 8.5.13 (semi-algebraic possible world semantics) *Let* $\mathbf{L}$ *be a language with connectives* $\sharp_1, \ldots, \sharp_k$ *with arities* $r_1, \ldots, r_k$ *respectively. A semi-algebraic model for* $\mathbf{L}$ *has the form* $(A, R_1, \ldots, R_k, \leq, T, h)$ *where* A *is a set of possible worlds,* $T \subseteq A, T \neq \varnothing$ *and each* R *is a relation on* A *of arity* $1 + r_i$. h *is an assignment from the atoms of the language into subsets of* A. *The following must hold.*

1. $x \leq y$ *and* $y \in T$ *imply* $x \in T$.

2. $t \in h(q)$ *and* $t \leq s$ *imply* $s \in h(q)$.

3. *Each* R_i *is directional in all its variables, in the same way as* $\sharp_i$*, respectively, and* R_i *is directional up in its first variable.*

Note that for a connective $\sharp(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ *there corresponds a relation* $R_\sharp(t, x_1, \ldots, x_n, y_1, \ldots, y_m)$. *The meaning of* $R_\sharp$ *in terms of the algebraic function* $f_\sharp(x_1, \ldots, x_n, y_1, \ldots, y_m)$ *and* $\leq$ *is*

$$R_\sharp(t, x_1, \ldots, x_n, y_1, \ldots, y_m) \textit{ iff } f_\sharp(x_1, \ldots, x_n, y_1, \ldots, y_m) \leq t.$$

Note that $R_\sharp$ *satisfies that for each* $(x_1, \ldots, x_n, y_1, \ldots, y_m)$ *there exists a minimal* t *such that* $R_\sharp(t, x_1, \ldots, x_n, y_1, \ldots, y_m)$ *holds. Conversely, the function* $f_\sharp$ *can be retrieved from such an* $R_\sharp$ *in the presence of* $\leq$.

Thus if $f_\sharp$ *is monotonic up in* y_j *and monotonic down in* x_i*, then* $R_\sharp$ *will be monotonic up in* x_i *and* t *and monotonic down for* y_j *(inverse direction for* x_i *and* y_j*).*

Define the following first truth table for the connectives $\sharp(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ *as follows: (*$\sharp$ *is directional down in* X_i *and up in* Y_i*):*

$t \vDash \sharp(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ *iff for all* t_i *such that* $t_i \vDash X_i$ *there exist* s_j *such that* $s_j \vDash Y_j$ *such that* $R_i(t, x_1, \ldots, x_n, y_1, \ldots, y_m)$ *holds.*

We say $A \mid\!\sim B$ *holds in the model if whenever* $t \vDash A$ *then* $t \vDash B$.

We say $\vDash A$ *in the model if whenever* $t \vDash A$ *then* $t \in T$.

Note that we can also assume that $R_\sharp$ *satisfies that for every* $t, x_1, \ldots, x_n$ *there exist unique* $y_1, \ldots y_m$ *such that* $R_\sharp(t, x_1, \ldots, x_n, y_1, \ldots, y_m)$. *The reason being that the filters* $\{y \mid y \vDash Y\}$ *have a minimal element and* $R_\sharp$ *is monotonic down in* y_i. *So* $R_\sharp$ *holds for some* y *iff it holds for the minimal* y. *This last statement entails that we can get completeness for* $R_\sharp$ *and the following truth table:*

$t \vDash \sharp(X_1, \ldots, X_n, Y_1, \ldots Y_m)$ *iff for all* t_i *such that* $t_i \vDash X_i$ *and all* s_j *such that* $R_\sharp(t_1, \ldots t_n, s_1, \ldots, s_m)$ *we have for all* $j, s_j \vDash Y_j$.

Theorem 8.5.14 *Let* $\mid\!\sim$ *be the smallest directional consequence relation, then* $\mid\!\sim$ *is complete for the semi algebraic semantics.*

Proof. Use the canonical model. ■

So far we have been mainly concerned with algebraic theorems preparing an almost arbitrary logic for translation into classical logic, not with the theory of partially ordered algebraic logics (cf. [Gabbay, 1993a, Gabbay, 1992b, Blok and Pigozzi, 1989]). We need the algebras as tools in order to translate any general logic, about which we do not know much. We found it convenient to apply our algebraic methods to singleton consequence relations $\mid\!\sim$, i.e. the kind defined for $A \mid\!\sim B$ and $\varnothing \mid\!\sim B$, which give rise to the ordering relation $\leq$ in the logic algebra. Many logics, however, are presented as Hilbert systems, and so we cannot apply our methods as we have only the relation $\varnothing \mid\!\sim_1 A$, for A wff. Suppose we are given a Hilbert presentation of a logic $\mathbf{L}_1$. This means that we are defining only the notion of $\varnothing \mid\!\sim_1 A$ (or just $\mid\!\sim_1 A$). We need to investigate a means of deriving from $\mid\!\sim_1$ a consequence relation $\mid\!\sim_2$ such that $A \mid\!\sim_2 B$ is always defined and $\mid\!\sim_2$ is a conservative extension, i.e. $\varnothing \mid\!\sim_2 A$ iff $\varnothing \mid\!\sim_1 A$ holds. In algebraic terms, given (A, f_i, T) can we define a suitable $\leq$ on A? In implicational logics with $\Rightarrow$ satisfying the deduction theorem this can be easily done: $A \mid\!\sim_2 B$ iff $\varnothing \mid\!\sim_1 A \Rightarrow B$.

Lemma 8.5.15 *Let* $\mid\!\sim_1$ *be a Hilbert conseuence relation (i.e. defined for* $\varnothing \mid\!\sim_1 A, A$ *wff). Then there exists the minimal* $\mid\!\sim_-$ *and maximal* $\mid\!\sim_+$ *singleton consequence relations which agree with it.*

Proof.

1. Let $\Delta \mid\!\sim_- B$ be defined by

$$\begin{array}{rl} \Delta \mid\!\sim_- B \text{ iff} & (1)\Delta = \varnothing \text{ and } \varnothing \mid\!\sim B \\ \text{or} & (2)\Delta = \{B\}. \end{array}$$

   We show $\mid\!\sim_-$ is a consequence relation:

   - clearly $A \mid\!\sim_- A$ holds
   - assume $\Delta \mid\!\sim_- A$ and $A \mid\!\sim_- B$. By definition $A = B$ and so $\Delta \mid\!\sim_- B$.

   We show $\mid\!\sim_-$ is minimal. Suppose $\mid\!\sim_1$ agrees with $\mid\!\sim$. We show $\Delta \mid\!\sim_- A$ implies $\Delta \mid\!\sim_1 A$. If $\Delta = \{A\}$, then clearly $A \mid\!\sim_1 A$. If $\Delta = \varnothing$, again by agreement $\varnothing \mid\!\sim_1 A$.

2. From (1) it is clear that the set of all singleton consequence relations agreeing with $\mid\!\sim$ is nonempty. Define $\mid\!\sim_+$ as follows.

   $A \mid\!\sim_+ B$ iff there exist $\mid\!\sim_0, \ldots \mid\!\sim_k$ agreeing with $\mid\!\sim$ and $A_1, \ldots, A_k$ such that

$$A \mid\!\sim_0 A_1, A_1 \mid\!\sim_1 A_2, \ldots, A_k \mid\!\sim_k B.$$

   We show that $\mid\!\sim_+$ is a consequence relation:

   - clearly $A \mid\!\sim_+ A$ because $A \mid\!\sim_- A$ holds.
   - assume $\Delta \mid\!\sim_+ B$ and $B \mid\!\sim_+ C$ we show $\Delta \mid\!\sim_+ C$. Then for some $\mid\!\sim_0, \ldots, \mid\!\sim_k$ and $\mid\!\sim_{k+1}, \ldots, \mid\!\sim_{m+1}$ and $A_1, \ldots, A_k, A_{k+1}, \ldots, A_m$ we have

$$\Delta \mid\!\sim_0 A_1, \ldots, A_k \mid\!\sim_k B, B \mid\!\sim_{k+1} A_{k+1}, \ldots, A_m \mid\!\sim_{m+1} C.$$

     But then by definition $\Delta \mid\!\sim_+ C$.

   We now show $\mid\!\sim_+$ agrees with $\mid\!\sim$

   - Clearly $\varnothing \mid\!\sim B$ implies $\varnothing \mid\!\sim_+ B$.
   - assume $\varnothing \mid\!\sim_+ B$ and show $\varnothing \mid\!\sim B$.

   Since $\varnothing \mid\!\sim_+ B$, there exist $A_1, \ldots, A_k$ and $\mid\!\sim_0, \ldots, \mid\!\sim_k$ such that $\varnothing \mid\!\sim_0 A_1, \ldots, A_k \mid\!\sim_k B$ hold. We prove $\varnothing \mid\!\sim B$ by induction on k.

Case $k = 0$
In this case $\varnothing \mid\!\sim_0 B$ and since $\mid\!\sim_0$ agrees with $\mid\!\sim$ we get $\varnothing \mid\!\sim B$.
Case $k > 0$
Consider the initial sequence

$$\varnothing \mid\!\sim_0 A_1 \text{ and } A_1 \mid\!\sim_1 A_2.$$

Since $\mid\!\sim_0$ agrees with $\mid\!\sim$ which agrees with $\mid\!\sim_1$, we get $\varnothing \mid\!\sim_1 A_1$ and hence by transitivity, we get $\varnothing \mid\!\sim_1 A_2$.

We now have a shorter sequence

$$\varnothing \mid\!\sim_1 A_2, A_1 \mid\!\sim_2 A_3, \ldots, A_k \mid\!\sim_k B.$$

Hence, by the induction hypothesis $\varnothing \mid\!\sim B$.
This completes the induction. Clearly $\mid\!\sim_+$ is maximal. Hence, the Lemma is proved. ■

The above considerations do not give us an algorithm for finding for a given Hilbert system $\mid\!\sim_1$ a singleton consequence relation $\mid\!\sim_2$ which agrees with it. Of course, we can always take the minimal one $\mid\!\sim_1^-$, but the minimal one is not particularly infomative. The stronger the consequence relation the better. One can try to find one using a theorem prover as the next example shows.

Example 8.5.16 *Let* **L** *be a propositional language with some connectives and let* **H** *be a Hilbert system for* **L** *with schematic rules of the form* $\rho_i, (i = 1, \ldots, m)$:

$$\rho_i : \quad \frac{\mid\!\sim \varphi_1^i; \ldots; \mid\!\sim \varphi_{k_i}^i}{\vdash \varphi_i}.$$

For example, Łukasiewicz formulation of the implicational fragment of classical logic (using the binary connective $\Rightarrow$*) is formulated by the schemas*

$$\rho_i : \quad \frac{\mid\!\sim P; \mid\!\sim P \Rightarrow Q}{\mid\!\sim Q}$$

$$\rho_2 : \quad \frac{\varnothing}{\mid\!\sim ((P \Rightarrow Q) \Rightarrow R) \Rightarrow ((R \Rightarrow P) \Rightarrow (S \Rightarrow P))}.$$

We can systematically look (by complexity) for a formula $\Psi(X, Y)$ *(binary) and try and prove using the term translation of Definition 8.5.9 and a classical theorem prover, that the following holds:*

- $\mid\!\sim_1 \Psi(X, X)$
- $\mid\!\sim_1 \Psi(X, Y)$ *and* $\mid\!\sim_1 \Psi(Y, Z)$ *imply* $\mid\!\sim_1 \Psi(X, Z)$
- $\mid\!\sim_1 X$ *and* $\mid\!\sim_1 \Psi(X, Y)$ *imply* $\mid\!\sim_1 Y$.

For a Ψ *satisfying the above we can let*

$$A \mid\!\sim_2 B \textit{ iff (definition) } \mid\!\sim_1 \Psi(A, B)$$

$$\varnothing \mid\!\sim_2 B \textit{ iff } \mid\!\sim_1 B.$$

In fact, since singleton consequence relations are closed under intersection, if we have several such Ψ_i *we can let* $A \mid\!\sim_2 B$ *be* $\bigwedge_i \mid\!\sim_1 \Psi(A, B)$.

We now ask how do we find such a Ψ *automatically? We can let* Ψ *range over all wffs of the language in order of increasing complexity and then for this* Ψ*, using the term translation of Definition 8.5.9 and a classical theorem prover see if each* Ψ *satisfies the required condition.*

A good heuristic is to look at the Hilbert rules of the form

$$\frac{\mid\!\sim_1 \varphi_1; \ldots; \mid\!\sim_1 \varphi_k}{\mid\!\sim_1 \varphi}$$

and try and substitute into φ_i *and* φ *some formula that will turn the above rule into the form*

$$\frac{\mid\!\sim_1 A; \mid\!\sim_1 \Psi_1(A, B); \ldots \mid\!\sim_1 \Psi_m(A, B)}{\mid\!\sim_1 B}.$$

We can now check whether the metapredicate

$$\bigwedge_{i=1}^{m} [\mid\!\sim_1 \Psi_1(X, Y]$$

is transitive, in which case, we define

$$A \mid\!\sim_2 B \textit{ iff } \bigwedge_{i=1}^{m} \mid\!\sim_1 \Psi_i(A, B)$$

$$\varnothing \mid\!\sim_2 B \textit{ iff } \mid\!\sim_1 B.$$

Applying the above procedure to Łukasiewicz logic, we immediately find that $\Psi(A) = (\mathrm{def}) A \Rightarrow B$ *is a candidate. We need to check whether*

- $\mid\!\sim_1 A \Rightarrow B$ *and* $\mid\!\sim_1 B \Rightarrow C$ *imply* $\mid\!\sim_1 A \Rightarrow C$
- $\mid\!\sim_1 A \Rightarrow A$.

These can be checked using the term translation.

Example 8.5.17 *The previous example dealt with the problem of finding a singleton consequence relation* $\Delta\mid\!\sim_2 B$ $(\Delta = \{B\}$ *or* $\Delta = \varnothing\}$ *agreeing with a Hilbert consequence* $\varnothing\mid\!\sim_1 B$. *We can continue this line of thought and try to extend a singleton consquence* $\Delta\mid\!\sim_2 B$ *to a unitary consequence* $\Delta\mid\!\sim_3 B$, Δ, *arbitrary which agrees with it.*

Assume now that a proper $\mid\!\sim$ *is given, where* $A\mid\!\sim B$ *and* $\varnothing\mid\!\sim B$ *are defined for arbitrary* A *and* B. *The second concept we define is that of a theory. We defined a theory* Δ *as a set of wffs and gave the notion* $\Delta \mid\!\sim A$ *the semantic meaning:*

$\Delta \mid\!\sim A$ *iff for all* $\mathbf{a}$ *and for all* $t \in A^{\mathbf{a}}$, *if* $t \vDash B$, *for all* $B \in \Delta$ *then* $t \vDash A$.

There is a more general notion of a theory. Let $\Psi(X, Y)$ *be formulas defined using the connectives of the logic. Assume* Ψ *is monotonic down in* X *and up in* Y. *Using* Ψ *we can define the following notion of a theory* Δ.

- *A theory* Δ *is a sequence of formulas* $(A_1, \ldots, A_n)$
- $(A_1, \ldots, A_n) \mid\!\sim A = def\ \varnothing \mid\!\sim \Psi(A_1, \Psi(A_2, \ldots, \Psi(A_n, A)\ldots))$.

This definition gives the deduction theorem for $\mid\!\sim$:

$$A_1, \ldots, A_n \mid\!\sim \Psi(A, B) \text{ iff } A_1, \ldots, A_n, A \mid\!\sim B.$$

Consider for example the connective $\twoheadrightarrow$ *of example 8.5.11. We have*

$$A_1, \ldots, A_n \mid\!\sim B \text{ iff } \varnothing \mid\!\sim A_1 \twoheadrightarrow (\ldots, \twoheadrightarrow (A_n \twoheadrightarrow B)\ldots).$$

The notion of a theory can be generalized. Let $\tau = \{\sharp_1, \ldots, \sharp_k\}$ *be a set of monotonic upwards connectives of the language. Close* τ *under substitution. Let* $\varphi(x_1, \ldots, x_n)$ *be a wff generated from* τ *with n-places. Then the formula expression* $[\varphi(A_1, \ldots, A_n)]$ *is a* φ*-theory of* $(A_1, \ldots, A_n)$ *with* $\varphi[A_1, \ldots, A_n] \mid\!\sim B$ *iff* $\varnothing \mid\!\sim \Psi(\varphi(A_1, \ldots, A_n), B)$.

Other definitions of algebraic logics and theories existing in the literature do not use the partial order but only the set of designated elements. One example is ($\mathbf{a}$ *an algebraic model):* $\Delta \vDash_{\mathbf{a}} A$ *iff for every assignment* h, *if* $h(B) \in T$ *for all* $B \in \Delta$ *then* $h(A) \in T$.

We can now formulate, through the semantics, any directional logic as an *LDS*. This is the task of the rest of this section.
`Dov to Continue ...`

8.6 Labelling the metalevel (nested labelling systems)

This book adopts the view that the basic unit of logical information is the labelled formula $t : A$.

We futher adopted the view that a labelled deduction system is mathematically just like an ordinary deductive system except that all formulas are labelled and the proof rules tell us how to manipulate the labels as well as the formulas.

If we accept ths point of view and its consequences, we must allow for the following:

1. Well formed formulas of the form $(t : A) \to (s : B)$.
2. Labelling disciplines for such formulas, for example:
   $t : (s : A)$
   $t : ((s_1 : A) \to (s_2 : B)$

3. Allow for arbitrary iterations of nested $\to$ labelling.
4. Study reduction rules for reducing nestings of labels.

In order to have a meaningful interpretation for nested labelling systems, we need to have a creative point of view of labelled formulas. Let us think of $t : A$ as an algebraic resource annotation. Imagine we have a proof system $\vdash_0$ with a syntactical algorithmic notion of how $A_1, \ldots, A_n \vdash_0 B$ holds. We take an algebra $\mathcal{A}$ and label the assumptions as $t_1 : A_1, \ldots, t_n : A_n$ and execute the proof to obtain B with a label $s = \mathbf{t}(t_1, \ldots, t_n)$, where t is a function symbol of the labelling algebra dependent on the proof path of B from $A_1, \ldots, A_n$. Thus we write

$$t_1 : A_1, \ldots, t_n : A_n \vdash_{\mathcal{A}} s : B.$$

We can have double labelling, from two different algebras, $\mathcal{A}_1$ and $\mathcal{A}_2$ to get

$$(t_i^1, t_i^2) : A_i \vdash_{\mathcal{A}_1 \otimes \mathcal{A}_2} (\mathbf{t}^1(\mathbf{t}_1^1, \ldots, t_n^1), \mathbf{t}^2(y_1^2, \ldots, t_n^2)) : B$$

We can now decide whether we accept B as proved by looking at the labels. This use of the labels filters proofs according to algebraic resource considerations.

What would the wff $(t : A) \to (s : B)$ mean in this context? We can understand it as a metastatement saying that if A can be obtained with reousrce t, then B can be obtained with resource s. Thus we have:

$$\Delta \vdash_{\mathcal{A}} (t : A) \to (s : B) \text{ iff } \Delta; t : A \vdash_{\mathcal{A}} s : B.$$

If Δ is aan *LDS* database, then the above is meaningful.

To understand the mechanisms involved in nested labelling, we can begin with a consequence relation $\vdash_0$ between formulas, written $A_i \vdash_0 B$. We can turn it into an *LDS* by adding labels and a labelling discipline, thus defining a consequence relation $\vdash_1$ on labelled formulas of the form $t_i : A_i \vdash_1 s : B$.

There does not necessarily have to be any obvious relationship between $\vdash_1$ and $\vdash_0$, though there may indeed be a connection, depending on the nature of the labelling discipline of $\vdash_1$. We may have a connection, for example, of the form

$$A \vdash_0 B \text{ iff } \varnothing : A \vdash_1 \varnothing : B$$

or of the form:

$$A \vdash_0 B \text{ iff } \varnothing : A \vdash s : B, \text{ for some } s.$$

Now that we have $\vdash_1$ we can again add a labelling discipline to get a system $\vdash_2$. $\vdash_2$ again may not be connected in any way with $\vdash_1$. In fact, the set of labels of $\vdash_2$ may be different from the labels of $\vdash_1$.

Let us see what happens when the labelling discipline and the labels used in moving form $\vdash_0$ to $\vdash_1$ is used again to move from $\vdash_1$ to $\vdash_2$. Do we get $\vdash_2 = \vdash_1$? To consider the question meaningfully let us start with a familiar $\vdash_0$ discipline. Let $\vdash_0$ be linear logic.

This means that we start with the unlabelled version of linear logic, then pass on to a labelled version and then label again. Let us see what happens.

The axioms of linear logic are:

$$A \to A$$
$$(A \to B) \to ((C \to A) \to (C \to B))$$
$$(A \to (B \to C)) \to (B \to (A \to C))$$

and the rules are MP and RT namely

$$\frac{A, A \to B}{B}$$

and

$$\frac{A \to B}{(B \to C) \to (A \to C)}.$$

The rule RT can be derived from the other axioms and rules. The notion of $\vdash A$ is defined in the usual way for the logic, namely $\vdash A$ iff there exists a sequence $A_0, \ldots, A_n = A$ such that each element in the sequence is either a substitution instance of an axiom or is obtained from two previous elements by the use of a rule, in this case modus ponens.

A theorem of the logic is any A such that $\vdash A$.

Note that we still have to define the notion $A \vdash B$, as we defined only what a theorem is, ie $\varnothing \vdash B$.

We say $A_1, \ldots, A_n \vdash_0 B$ iff there exists a sequence of formulas $D_1, \ldots, D_m = B$ such that each element of the sequence is either a theorem or an assumption A_i or is obtained from previous elements of the sequence using modus ponens.

We now impose a labelling discipline on this system. Label each theorem by $\varnothing$ and label each assumption by a new atomic label. Propagate the labels the usual way,

$$\begin{array}{l} \alpha : A \\ \beta : A \to B \\ \hline \beta\alpha : B \end{array}$$

In this manner, whenever

$$A_1, \ldots, A_n \vdash_0 B$$

We have

$$a_1 : A_1, \ldots, a_n : A_n \vdash_1 \alpha : B$$

where α is a sequence of $x_i \in \{a_i\}$ indicating which assumptions were used in the derivation.

Let us now use $\vdash_1$ together with the same labelling system to get $\vdash_2$. The assumptions have the form $\alpha_i : (\beta_i : A_i)$ (we are labelling an already labelled formula). We agree as before that assumptions get new atomic labels. Label propagation through modus ponens should follow the same pattern.

$$\begin{array}{l} \alpha_1 : (\alpha_0 : A) \\ \beta_1 : (\beta_0 : A \to B) \\ \hline \beta_1\alpha_1 : (\beta_0\alpha_0 : B) \end{array}$$

We similarly have that whenever $\alpha_1 : A_1, \ldots, \alpha_n A_n \vdash_1 \alpha : B$ then for some β and atoms x_i

$$x_1 : (\alpha_1 : A_1), \ldots, x_n : (\alpha_n : A_n) \vdash_2 \beta : (\alpha : B).$$

β indicates from which assumptions $x_i : (\alpha_i : A_i)$ was $\alpha : B$ derived.

Our problem is how to interpret $\alpha_1 : (\alpha_0 : A)$. Can we meaningfuly collapse it to some $\gamma : A$? ie, reduce $\vdash_2$ to $\vdash_1$?

Let us examine in more detail what can happen. Consider the assumptions:

$$\begin{array}{l} a_0 : A_0 \\ b_0 : A_0 \to B_0 \\ c_0 : B_0 \to C_0 \end{array}$$

Using modus ponens we get that the above proves in $\vdash_1$ $c_0b_0a_0 : C_0$.

If we were to label again, we would get

$$\begin{array}{l} a_1 : (a_0 : A_0) \\ b_1 : (b_0 : A_0 \to B_0) \\ c_1 : (c_0 : B_0 \to C_0) \end{array}$$

We use modus ponens to get

$$c_1b_1a_1 : (c_0b_0a_0 : C_0).$$

The two labels are practically identical. Thus labelling again with the same labelling discipline gives us nothing. In fact the example shows that $\alpha : (\beta : A)$ can really be taken as $(\alpha, \beta) : A$ and the two labelling manipulations can be done side by side, on each co-ordinate.

However, if we use a different labelling discipline on the α label we get composition of the metaleval meaning of each label. For example the β label may record relevance and the α label may record possible worlds. This way we get modal relevance logic, as presented in example 2.2.4.

1. Bring in internalisation of consequence
2. Temporalising

Part III

Labelled Resource Logics

Chapter 9

Resource Logics

9.1 Introduction

This chapter will illustrate the use of the Metabox to define some known logics and a new class of logics called *Resource Logics*. We understand the term to cover all logics where it is important to know exactly from which assumptions and using which inference rules and at what order a conclusion is drawn. In this case the labels function as names to the assumptions (data) and the deductive process propagates the labels to the conclusion through an agreed labelling discipline. Thus $\Delta \vdash t : A$ means that A can be proved from Δ using those assumptions and rules in Δ as indicated by the label t. In a more sophisticated labelling discipline the labels can be used to control the deductive process and thus function as a metalevel device. The label propagation mechanism will depend on what the labels denote. It could be probabilities, it could be priorities, it could be an indication of the origins or update version of the data. Well known among such logics are relevance and linear logics. In relevance logic $\Delta \vdash A$ requires A to be proved using *all* the assumptions in Δ and in linear logic it is further required that the assumptions in Δ are used not more than once.

Derivation can be controlled in two pure ways (and of course in a mixed way). One can control the use of assumptions. For example, one may have two assumptions in the database, A_1 and A_2, and the control allows the algorithmic proof procedure to use exactly one of them.

Another example is when we require all assumptions to be used exactly once. This resource restriction gives rise to Linear Logic. In general, the subclass of *Resource Logics*, which restricts only the number of assumptions used (as compared with how they are used) are known as substructural logics. Thus Linear Logic, Relevance Logic and their neighbours are Resource Logics in our sense.

Another important subclass of *Resource Logics* are the *Prioritised Logics*, where the main restrictions are on how to use the inference rules. The reader may question whether such a distinction makes sense, after all, all inference rules come with conditions on how to use them, and why can't we regard the so called "controlled inference" as part of the rules. The answer is that the control is on the global use of the rules and cannot be attached to any single rule. An extreme example of controlled inference logics are *Defeasible Logics* (see the appropriate chapter) and the logics known as *Ordered Logics.* These logics arise naturally in non-monotonic systems handling several distributed knowledge bases and having to reconcile possibly conflicting information.

Natural priorities among rules give rise to *prioritised logics.* In general *resource* -logics are a mixture of both, resource restrictions and inference control.

A simple example of a mixture is when we do not allow for any formula $A \to B$ to be used both as a ticket and as a minor premiss of a modus ponens. With this restiction,

$$(A \to (A \to B)) \to (A \to B)$$

is still a theorem (though not a theorem of linear logic) but

$$((C \to A) \to C) \to ((C \to A) \to A)$$

is not a theorem.

In general Resource Logics use the labels to observe and record the use of the assumptions, while *prioritised* -logics use the lables to observe and control the use of rules.

The whole way of thinking involved in *Resource*-logics is different from the traditional ones. One can no longer apply logical rules or even use assumptions in any order or as often as one wants. Certain sequnces are allowed and certain are not, according to some controlling mechanism. Some logicians strive, in fact, for complete symmetries in derivations and would therefore reject the idea of *prioritised*-Logics. Perhaps a theorem can be proved that every *prioritised*-Logic can be turned into a symmetrical logic with more connectives. We shall address this problem later. Girard's [Girard, 1987] linear logic is an example of a Resource (bounded) logic for which there exists a symmetrical proof (net) theory.

The general form of the *Resource*-logic is most naturally cast as a form of Labelled Deductive system (*LDS*). If we are going to have control over the use of resource or inference then we must be able to keep the accounting right of what we used and how we used it. If the accounting is to be complete then special notational systems must be employed and this is *LDS*. We do have, in a Gentzen formulation, the possibility of labelling formulas by placing them to the left or to the right of the turnstile "$\vdash$", and also by allowing multisets, for example we can write: $A, A \vdash B, B, C$. One can go further and allow sequences to the right or left of $\vdash$ or even more complex structures. This is nothing but geometrical labelling and one may as well take on the general case and say that we have a system of labels, symbolised in the form $(L, *, <)$ and that what we are dealing with are labelled wffs of the form $\alpha : A, \alpha$ the label, A the wff.

Another example of rule restriction is not to use rule R_1 after rule R_2 has been used. In default logic for example we may allow the use of a default rule to get a conclusion A to generate more conclusions via modus ponens (ie $A, A \to B \vdash B$) only if A was not obtained by default and we may not allow A to generate more defaults. This is inference control. The most general case is in defeasible reasoning and in ordered logic where the rules are classified according to strength and rules can defeat others, for example the database:

1. Birds fly

2. Big birds do not fly

3. Tweety is a big bird

allows us to deduce both Tweety flies and Tweety does not fly. One way to get the right answer is to say that rules are stronger if their antecedants are logically stronger and we must use the stronger rule to decide between any contradictory pair of derivable atoms of the form $\{p, \sim p\}$. This is definitely a *prioritised*-logic.

The remainder of this Chapter will study relevance implication logic and its related family of logics. The basic labelling and metabox mechanism for these logics is designed to record which of the assumptions were used in the deductive process, how many times and how. This should be compared with other resource logics which record priority or reliability of assumptions. For this reason all new assumptions are labelled by new atomic labels. The inference mechanism is the $\to$ introduction (metabox) rule (this is the goal Rule for $\to$) and the $\to$ elimination (modus ponens) rules (the Data Rule for $\to$) which can be used with various restrictions.

Schematically, these are:

Data Rule for $\to$ ($\to E$)

$$\frac{\begin{array}{l}\alpha : A\\ \beta : A \to B\end{array}}{(\beta * \alpha) : B}$$

provided $\psi_{MP}(\alpha, \beta)$ holds

α is the label of the minor assumption. β is the label of the ticket. $\beta * \alpha$ is the label of the resulting formula. $*$ is a convenient binary function for generateing new labels. In most cases $*$ is concatenation of sequences and α and β are sequences. In some cases α, β are multisets or sets and $*$ is union $\cup$. For any individual logic, $*$ will be specified. ψ_{MP} is a side condiiton on α and β such as $\alpha \cap \beta = \varnothing$ or such as $\alpha \neq \varnothing, \psi_{MP}$ is a formula in some metalanguage suitable for describing these conditions.

Goal Rule for $\rightarrow$ ($\rightarrow I$)

To show $\gamma : A \rightarrow B$ use a box

$$\boxed{\begin{array}{ll} \alpha : A & \text{assumption} \\ & \\ \vdots & \text{Reasoning using} \\ \vdots & \text{local box rules} \\ & \\ \beta : B & \end{array}}$$

exit $\gamma : A \rightarrow B$ provided $\psi_I(\alpha, \beta, \gamma)$

α is a new atomic label, in whatever is considered atomic in the labelling discipline. $\psi_I(\alpha, \beta, \gamma)$ is the exit condition, telling us how α, β, γ are related. The most general is the requirement $\beta = \gamma * \alpha$. In case $*$ is set union we may have for example $\alpha \subseteq \beta$ and $\gamma = \beta - \alpha$.

There are two more parameters involved here which are not mentioned explictly. These are:

1. Properties of $*$, eg commutativity, associativity, etc. These properties may help us satisfy side conditions such as $\psi_I(\alpha, \beta, \gamma)$,

2. Deductive rules in the box itself, like what can be reiterated from outer boxes, how can the reiteration be used etc.

In general there is a trade-off between the properties of $*$ and the deductive conditions. For example the condition $\alpha \cap \beta = \varnothing$ in ψ_{MP} corresponds to the deductive condition of not using an assumption more than once. Details of the possible trade-offs will become evident in the sequel.

We begin with a list of Hilbert type axioms for future use. Some of the axiomatisation results are already available in Anderson and Belnap's book. However it is necessary for us to present the metabox discipline in detail. The reader should also note that the metabox presentation and axiomatisation of a logic (eg **W**) yields the logic as a "module" ready to be combined with other metabox logics (e.g. relevance and modality) without any cost.

Definition 9.1.1 (Resource Axioms) *First define $(A^k \rightarrow B)$ to be:*

$$A^0 \rightarrow B = def B$$

$$(A^{n+1} \rightarrow B) = def A \rightarrow (A^n \rightarrow B)$$

Consider the following stock of axioms and rules:

1. Identity
$A \rightarrow A$

2. Right Transitivity
$(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$

2a. Right Transitivity Rule, (RT-Rule):

$$\frac{A \rightarrow B}{(B \rightarrow C) \rightarrow (A \rightarrow C)}$$

3. left transitivity
$(A \rightarrow B) \rightarrow ((C \rightarrow A) \rightarrow (C \rightarrow B))$

4. Distribution

a. $(A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C))$

b. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$

4m, n (m, n) distribution
$(A^m \to B) \to ((A^n \to (B \to C)) \to (A^{m+n} \to C))$

5m, n (m, n) contraction
$(A^m \to B) \to (A^n \to B)$

6β β-deduction
$\beta \to (\alpha \to \beta)$
Provided α, β satisfy the condition below:
α, β-condition possibilities:

(a) $\beta(\varnothing)$*: no condition on* α *or* β*.*

(b) $\beta(\to) : \beta = C \to D, \beta$ *has an implicational form,* α *arbitrary.*

(c) $(\mathbf{H})$*:* $\vdash_{\mathbf{H}} \alpha$ *in the logic* $\mathbf{H}$*,* β *arbitrary.*

7γ γ-Permutation
$(A \to (\gamma \to C)) \to (\gamma \to (A \to C))$
Provided γ satisfies the condition below.
Possibilities for γ:

(a) $\gamma(\varnothing)$*: no condition on* γ

(b) $\gamma(\to) : \gamma$ *has the form of an implicational* $C \to D$*.*

(c) $\gamma(\mathbf{H})$*:* $A \nvdash_{\mathbf{H}} \gamma$*.*

8. Restart (Peirce's Rule)
$((A \to B) \to A) \to A$

9_k k-mingle
$A^k \to (A \to A)$

10. Modus Ponens:

$$\frac{A, A \to B}{B}$$

11. Ackerman Type Rules

a. $$\frac{B, A \to (B \to C)}{A \to C}$$

b. $$\frac{B, (B \to A) \to C}{A \to C}$$

12. LC-Axiom
$((A \to B) \to C) \to (((B \to A) \to C) \to C)$

13. MV-Axiom
$((A \to B) \to B) \to ((B \to A) \to A)$

14. *Rose's Axiom R_k*
$((A^k \to B) \to A) \to A)$

The following explains the metabox and labelling meaning of the axioms. Axioms (1) and (3) (identity and left transitivity), and rule 2a correspond to the labelling system where $+$ is associative and can therefore be regarded as concatentation. When we perform modus ponens of $\alpha : A$ with $\beta : A \to B$, we get the label $\beta\alpha : B$ in this order. When we open a box, assume $a : A$ we must be able to prove $\gamma a : B$ in order to exit with $\gamma : A \to B$. See 9.2.6.

The corresponding metabox deductive rules are that only implications can be reiterated and that they must be used in the deductive process as tickets only and in the same order as the order of nesting (reiteration from an inner box must be used earlier than from outer box). See 10.3.9.

Axiom 2 allows for reiterations to be used in any order. See 10.3.4. 9.2.6(b) gives the corresponding labelling restrictions for **W**.

Axioms (4) and (5) both have to do with how many times an assumption can be used in a box. In labelling terms this means the question whether the labels are sets and $+$ is taken as a set union or the labels are multiset and $+$ is taken as multiset union. The two axioms (4) an (5) are not equivalent unless axiom (7) is present.

Axiom (7) is commutativity of $+$. In metabox terms (7) allows us to use reiterations as minor premisses and not only as tickets.

Axiom 6 has to do with the question of whether the assumption $\alpha : A$ opening a box has to be used at all in the proof of $\beta : B$ (ie can we exit with $\gamma : A \to B$ without the proof of B using A). In labelling terms it means $\alpha \subseteq \beta$.

Axiom 8 has to do with the restart rule. If we have a box within a box (see fig 9.1),

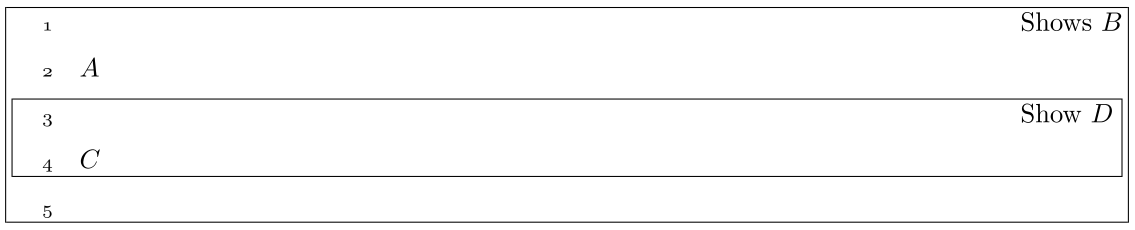


Figure 9.1:

can we show B in the inner box instead of D? In other words, do we read the above as equivalent to (see fig 9.2)

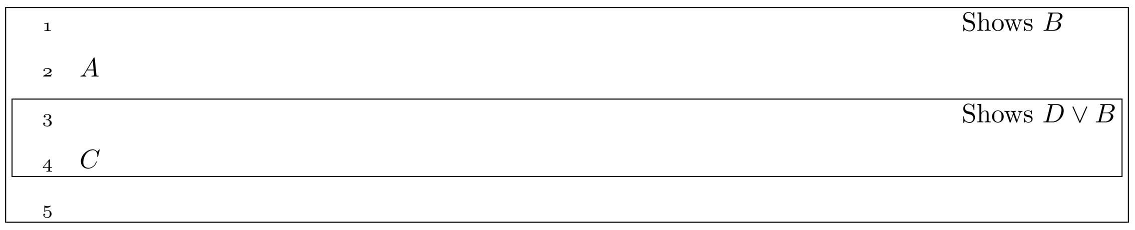


Figure 9.2:

Usually Peirce's rule yields classical logic. We show that it can be added to these weaker logics without turning them into classical logic.

Axiom 9 is a labelling axiom. It has to do with how we label several occurrences of A. Do we identify them as one occurrence and give them the same label?

Rule 11 has to do with the reasoning process. Is $A \to ((B \to B) \to A)$ a theorem? Can we reason inside a box and get $(B \to B)$ from nowhere? In labelling terms this is the condition $\alpha \neq \varnothing$, restricting the modus ponens rule.

Axiom 12 is the axiom for Dummett's LC. When added to intuitionistic logic and when added to many valued logics it gives linearity of the truth values.

Axioms 13 and 14 are axioms added to obtain Lukasiewicz's many valued logic. Rose's axiom R_k helps axiomatise Lukasiewicz $k + 1$ valued logic. Notice that for $k = 1$ we get Peirce's rule. Detailed discussions will be given in the appropriate chapters.

Definition 9.1.2 *Using the axioms of 9.1.1, the table below defines some new and some known Hilbert type systems. All systems allow for substitution and modus ponens.*

System Name	**Axioms**	**Comments**
CL	(1), (3), (2a)	Concatenation logic
W	(1), (2), (3)	Considered the weakest logic for which a reasonable deduction theorem exists.
$\mathbf{T_E}$	(1), (2), (3), (4)	Ticket entailment. (4b) can be proved from (1)-(3) (4a). (4a) is equivalent in $\mathbf{T_E}$ to $5_{(2,1)}$.
Linear Logic **LL**	(1), (2), (3),	(7 $\gamma(\varnothing)$)
Relevance **R**	(1)-(4),	(7 $\gamma(\varnothing)$)
Entailment **E**	(1)-(4),	(7 $\gamma(\to)$)
Strict Implication	(1)-(4),	(7 $\gamma(\varnothing)$) (6 $\beta(\to)$)
Intuitionistic Logic	(1)-(4),	(7 $\gamma(\varnothing)$) (6 $\beta(\varnothing)$)
Classical Logic	(1)-(4), (8)	(7 $\gamma(\varnothing)$) (6 $\beta(\varnothing)$)
Linear Classical	*Linear Logic* with (8)	A new logic
Relevance Classical	**R** with (8)	A new logic
H *Relevance*	(1)-(4),	(7 γ (**H**)) A new logic
LE *Linear Entailment*	(1)-(3),	(7 $\gamma(\to)$) A new logic
WC	(1)-(3), (8)	A new logic
B *Left Transitivity*	(1), (3) and (11)	A new logic

9.2 Basic resource logics

Definition 9.2.1 (The Logic W) *Consider the system* **W** *of metabox discipline of* $\to$ *with the following rules:*

1. *The basic forward rule is modus ponens. The labels are sets of atomic labels and label propagation is done according to the rule* MP1 *(*modus ponens for linearity*).*

   $$\begin{array}{l} \alpha : A \\ \underline{\beta : A \to B} \\ \alpha \cup \beta : B \end{array} \quad \textit{provided } \alpha \cap \beta = \varnothing \textit{ and } \alpha \neq \varnothing$$

   This is really the resource condition for linearity. If $\alpha \cap \beta \neq \varnothing$ *then both* A *and* $A \to B$ *used the same resource (eg* C*) and hence* B *cannot be derived because it would be using* C *more than once. The condition* $\alpha \neq \varnothing$ *ensures we do not use logical theorems in boxes. The meaning will become clearer later. In terms of Definitions 3.4.2 and 3.4.9, we have that* $\psi_{MP}(\alpha, \beta, \gamma)$ *is:* $\alpha \neq \varnothing \wedge (\alpha \cap \beta = \varnothing) \wedge \gamma = \alpha \cup \beta$.

2. *Box Goal Rule:*
   Assume we are in a box named b. *In line* k *of this box we want to show* $A \to B$. *To show* $A \to B$ *open a box and name it* a. *Assume* A *with a* new *atomic label* $\{a\} : A$, *also written* $a : A$. *Show* B *with label* β. *The box is successful if* $a \in \beta$. *Exit the box with* $\gamma : A \to B$, *where* $\gamma = \beta - \{a\}$.

   This is the condition of relevance logic. To show $A \to B$ *we must assume* A *and prove* B *in a way which uses* A.

In the proof of B *we can use some reiterations. The next rule tells us what reiterations we can use.*

3. *Metabox Reiteration Rule:*
   Given an outer box and an inner box as in the diagram (fig 9.3):

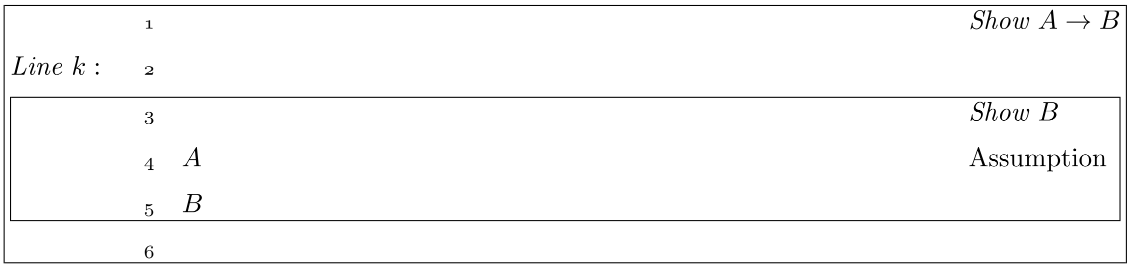


Figure 9.3:

The candidates for the allowed reiteration from the outer box into the inner box are all formulas which are assumptions, earlier reiterations or proved (in lines $m \leq k-1$*) and which have the form* $C \to D$. *We allow to bring in as reiteration any* $C \to D$. *The reiteration* $C \to D$ *can be used in the inner box, however,* only as a ticket *ie in a deduction of the form*

$$\begin{array}{l} \quad C \\ \underline{C \to D} \quad \textit{reiteration} \\ \quad D \end{array}$$

and not in a deduction of the form

$$\begin{array}{l} (C \to D) \to X \\ \underline{C \to D \qquad\qquad} \quad \textit{reiteration} \\ X \end{array}$$

In terms of Definition 3.4.2 we have defined $\mathbf{R}'$ *of 2.5.1 and the way modus ponens works in the inner box.*

Note that in the Box Goal Rule of the previous definition we name the boxes with completely new names and use completely new labels on new assumptions. We will have $b < a$, ie a is subordinate to b. In fact, since in the logic **W**, each new box arises only in order to show some implication $A \to B$ and it has one assumption $a : A$, we can achieve more economy of notation if we name the box as "a" as well. Thus the relation $<$ can be extended to labels. Call the original (most external box) as *box 0*. Let the assumption be numbered and named

$$\begin{array}{rl} Line1: & n_1 : A_1 \\ Line2: & n_2 : A_2 \\ & \vdots \\ & etc \end{array}$$

For the inner boxes, whenever we start a new box to show $A \to B$, we name it eg *Box a* and name the assumption $a : A$ thus (fig 9.4)

Line $k : A \to B$ from Box a

Box a	1		Show B
$(k, 1)$:	2	$a : A$	

Figure 9.4:

Each formula in any box has a label α. If $\alpha = \{a_1 \ldots a_n\}$ one can tell exactly which boxes it "used". The relation $\leq$ can be extended to labels as follows:

1. $x > n_i$, for all i since n_i are labels of the outer box.
2. $a > b$ if a names a subordinate box to b (remember that labels $\neq n_i$ are also names of boxes).
3. Let $\alpha \leq \beta$ if max $\alpha \leq$ max β

Thus when $\alpha : A, \beta : B$ are compared then $\alpha \leq \beta$ means that B was proved in a more inner box than A. Note that the metabox condition (3) of definition 9.2.1 is equivalent to the requirement that whenever we perform modus ponens

$$\frac{\begin{array}{l}\alpha; A \\ \beta : A \to B\end{array}}{\beta\alpha : B}$$

we have max $\beta <$ max α.

Definition 9.2.2 *Let* Δ *be a labelled database.*

(a) Let $\Delta \vdash_{\mathbf{W}} \alpha : Q$ *mean that there is a box proof of* Q *from* Δ *with label* $\alpha : Q$.

(b) Let $\Delta \vdash_{MP1} \alpha : Q$ *mean that* Q *can be proved from* Δ *with label* α *using only the rule of linear modus ponens MP1.*

Example 9.2.3 *Let us check the deduction of* $(A \to (B \to C)) \to ((B \to (A \to C))$ *(see fig 9.4)*

Note that if we relinquish the requirement that reiterations must be used only as tickets the box proof above will go through for $B = B_1 \to B_2$. *That is we can prove in that case that*

$$\varnothing \vdash (A \to ((B_1 \to B_2) \to C)) \to (((B_1 \to B_2) \to (A \to C)).$$

We shall see that the new condition gives rise to the system **LE** of Linear Entailment.

Box a_1	1		*Show* $B \to (A \to C)$
	2	$a_1 : A \to (B \to C)$	Assumption
	3	$a_1 : B \to (A \to B)$	from Box a_2
Box a_2	4		*Show* $A \to C$
	5	$a_2 : B$	Assumption
	6	$a_2 a_3 : A \to C$	from Box a_3
Boxa_3	7		*Show* C
	8	$a_3 : A$	Assumption
	9	$a_1 : A \to (B \to C)$	*reiteration*
	10	$a_1 a_3 : B \to C$	*modus ponens*
	11	*To obtain* C *we must bring in* B.	
	12	*We can do that only if* B *has the*	
	13	*form* $B_1 \to B_2$. *Even if* B *has*	
	14	*this form we cannot use it with*	
	15	$B \to C$ *because whatever reiterations*	
	16	*we bring in must be used as tickets.*	

Figure 9.5:

Box a_1	1		*show* $(B \to C) \to (A \to C)$
	2	$a_1 : A \to B$	*assumption*
	3	$a_2 : (B \to C) \to (A \to C)$	from Box a_2
Box a_2	4		*show* $A \ \to C$
	5	$a_2 : B \to C$	*assumption*
	6	$a_2 a_1 : A \to C$	from Box a_3
Box a_3	7		*show* C
	8	$a_3 : A$	*assumption*
	9	$a_1 : A \to B$	*reiteration*
	10	$a_1 a_3 : B$	*modus ponens*
	11	$a_2 : B \to C$	*reiteration*
	12	$a_2 a_1 a_3 : C$	*modus ponens*
exit	13	$a_2 a_1 : A \to C$	
exit	14	$a_1 : (B \to C) \to (A \to C)$	
exit	15	$\varnothing : (A \to B) \to ((B \to C) \to (A \to C))$	

Figure 9.6:

Example 9.2.4 *Consider:*

$$\varnothing \vdash (A \to B) \to ((B \to C) \to (A \to C)).$$

Try the following box (fig 9.6):

The logic **W** can be weakened, to obtain the logic **CL**, concatenation logic, the most basic logic of resource, defined below. Both logics are further studied in sections 4 and 5 of this chapter.

Definition 9.2.5 *Consider the following logics, defined by varying the restrictions of 9.2.1.*

1. *The logic* **CL** *obtained by strengthening the modus ponens rule of (1) of 9.2.1 and requiring that a modus ponens on A and $A \to B$ is allowed only if the reiteration $A \to B$ comes from an earlier box than the earliest of the reiterations used in proving A. In the notation of (2) of 9.2.1, this means that if a is the name of the current box then max $\beta \leq$ min $\alpha - \{a\}$.*

2. *The logic* $\mathbf{T}_{\mathbf{E}}$ *is obtained by abandoning the restriction $\alpha \cap \beta = \phi$ in the modus ponens.*

3. *Linear logic* **LL** *is obtained by abandoning restriction (3) of definition 9.2.1 and allowing for any formula to be reiterated and be used in any way, either as a ticket or as a minor. The only remaining restriction in the box discipline of linear logic is that formulas be used exactly once. See example 9.2.3*

4. *The logic* **LE** *of linear entailment allows for reiteratins to be used as minors but the reiterations themselves can be only of implicational form.*

Remark 9.2.6 *(a) Consider a refinement on the labelling and Modus Ponens which yields the logic* **CL** *(of 9.2.5 above) in a different way. Consider the following discipline. Labels are sequences of atomic labels. Propagation of labels follows the rule:*

$$\frac{\begin{array}{l}\alpha : A\\ \beta : A \to B\end{array}}{\beta\alpha : B}$$

where "$\beta\alpha$" is the result of concatentation of the sequences and provided $\alpha \neq \varnothing$.

To show $\gamma : A \to B$, we open a box with assumption $a : A$, "a" a new atomic label. If we get B in the box with label γa, then we can exit with $\gamma : A \to B$. Notice that B must be obtained with "a" appended to the right hand side of the sequence.

Here are the boxes proving the proposed axioms of **CL**.

Show $(A \to B) \to ((C \to A) \to (C \to B))$. See fig 9.7.

Compare the proof of the axiom $(A \to B) \to ((B \to C) \to (A \to C))$ as given in 9.2.4. The label propagation does follow our rules, however, the proof cannot go through because the exit from box a_2 is not legal in our discipline. The label a_2 is not the right hand most one. Similarly 9.2.3 cannot go through. $(A \to (B \to C)) \to (B \to (A \to C))$ is blocked. We can bring in B in Box a_3 of the above example and get $a_1a_3a_2 : C$ but we cannot exit box a_3 again because the label a_3 is not the right and most one. Compare with 2.2.8.

Remark 9.2.7 *The purpose of this remark is to show that there is a trade-off between label propagation rules and metabox reiteration rules. In the logic* **W**, *we can reiterate only implications. However, we can modify the labeling system to achieve the same effect. To see this, we now give a different labelling discipline for* **W**. *A label has the form of a pair α, a where α is a multiset and a is an atomic label. Essentially α, a is nothing but a multiset with one designated element. New assumptions are labelled $\varnothing, a$. We allow for the designated element not to exist. We adopt the following two rules:*

Box a_1	1		*Show* $(C \to A) \to (C \to B)$
	2	$a_1 : A \to B$	Assumption
	3	$a_1 : (C \to A) \to (C \to B)$	from Box a_2
Box a_2	4		$C \to B$
	5	$a_2 : C \to A$	Assumption
	6	$a_2 a_3 : C \to B$	from Box a_3
Box a_3	7		*Show* B
	8	$a_3 : C$	Assumption
	9	$a_2 : C \to A$	*reiteration*
	10	$a_2 a_3 : A$	*modus ponens*
	11	$a_1 : A \to B$	*reiteration*
	12	$a_1 a_2 a_3 : B$	*modus ponens*
exit	13	$a_1 a_2 : C \to B$	
exit	14	$a_1 : (C \to A) \to (C \to B)$	
exit	15	$\varnothing : (A \to B) \to ((C \to A) \to (C \to B))$	

Figure 9.7:

1. $$\frac{\begin{array}{l}\alpha, a : A \\ \beta, b : A \to B\end{array}}{\alpha \cup \beta \cup \{b\}, a}$$

provided $(\beta \cup \{b\}) \cap (\alpha \cup \{a\}) = \varnothing$;

and the box rule:

2. $$\boxed{\begin{array}{l}\varnothing, a : A \\ \\ \\ \gamma, b : B\end{array}}$$

exit $\gamma' : A \to B$*, provided (1)* $\gamma' = \gamma$ *and* $a = b$ *or (2)* $\gamma' = \gamma - \{a\}$ *and* $b = \varnothing$ *and* $a \in \gamma$.

In the box rule a *must be the designated element of the label of the conclusion. If the assumption is labelled* $\varnothing, a$ *and the conclusion is* $\gamma, \varnothing$ *then we can still exit with* $\gamma - \{a\}, \varnothing$ *provided* $a \in \gamma$.

The proof of $(A \to B) \to ((B \to C) \to (A \to C))$ *will go through. See fig 9.8.*

The proof of $(A \to (B \to C)) \to (B \to (A \to C))$ *will not go through. See fig 9.9 and compare with fig 9.5.*

If we drop the side condition on b in the box rule (b) above and read the labels as sets, we get the logic $\mathbf{T_E}$. Consider for example $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$. See fig 9.10.

Remark 9.2.8 *[The logic* **BR1***] This logic is what we consider the logic of basic resource. The intuition behind it is that each assumption can be used at most once but it is allowed that assumptions are not used at all. It can be defined as a Hilbert system with modus ponens and the axioms*

	1	$\varnothing, a_1; A \to B$	Assumption
	2	$\varnothing, a_2 : B \to C$	Assumption
	3	$\varnothing, a_1 : A \to B$	*reiteration*
	4	$\{a_1\}, a_3 : B$	*MP1*
	5	$\varnothing, a_2 : B \to C$	*reiteration*
	6	$\{a_1, a_2\}, a_3 : C$	*MP1*
exit	7	$\{a_1, a_2\}, \varnothing : A \to C$	
exit	8	$\{a_1\}, \varnothing : (B \to C) \to (A \to C)$	
exit	9	$\varnothing, \varnothing : (A \to B) \to ((B \to C) \to (A \to C))$	

Figure 9.8:

1	$\varnothing, a_1 : A \to (B \to C)$	Assumption
2	$\varnothing, a_2 : B$	Assumption
3	$\varnothing, a_3 : A$	Assumption
4	$\varnothing, a_1 : A \to (B \to C)$	*reiteration*
5	$\{a_1\}, a_3 : B \to C$	*MP1*
6	$\varnothing, a_2 : B$	*reiteration*
7	$\{a_1, a_2\}, a_2 : C$	*MP1*
8	*We cannot exit because a_3 is not*	
9	*the rightmost element in the sequence*	
10	*of the label of C*	
11		
12		

Figure 9.9:

	1	$\varnothing, a_1 : A \to (B \to C)$	assumption
	2	$\varnothing, a_2 : A \to B$	assumption
	3	$\varnothing, a_3 : A$	assumption
	4	$\varnothing, a_2 : A \to B$	reiteration
	5	$\{a_2\}, a_3 : B$	MP
	6	$\varnothing, a_1 : A \to (B \to C)$	reiteration
	7	$\{a_1\}, a_3 : B \to C$	MP
	8	$\{a_1, a_2, a_3\}, a_3 : C$	MP
exit	9	$\{a_1, a_2\}, \varnothing : A \to C$	
exit	10	$\{a_1\}, \varnothing : (A \to B) \to (A \to C)$	
exit	11	$\varnothing, \varnothing : (A \to (B \to C)) \to ((A \to B) \to (A \to C))$	

Figure 9.10:

(1)-(3), $6_{\beta(\varnothing)}, 7_{\gamma(\varnothing)}$ of 9.1.1. It is essentially intuitionistic logic without the distribution axiom. If we drop $6_{\beta(\varnothing)}$ from **BR1***, we get linear logic.*

The labelling discipline for **BR1** *is as follows: Labels are multisets of atomic labels. Assumptions always get new atomic labels. Modus ponens propagates labels in the usual way with the following side condition:*

$$\frac{\alpha : A \qquad \beta : A \to B}{\beta \cup \alpha : B} \quad \textit{provided } \alpha \cap \beta = \varnothing.$$

To show $\gamma : A \to B$ we open a box, assume $\{a\} : A$, with a new atomic label a and show $\beta : B$. It is not required that A be used in the proof (ie $a \in \beta$). We exit with $\gamma = \beta - \{a\}$.

Example 9.2.9 *We show that in the system* **CL***, the following rule holds:*
RT-Rule *(right transitivity rules):*

$$\frac{\vdash A \to B}{\vdash (B \to C) \to (A \to C)}$$

where $\vdash D$ means $\varnothing \vdash \varnothing : D$.

We show that by looking at the **CL** *metabox discipline. Since we assume $\vdash A \to B$, we assume that there exists a box of the form (see fig 9.11)*

Box a_2	1		
l_1	2	$a_2 : A$	*assumption*
	3	$\vdots$	
	4	$\vdots$	
l_k	5	$a_2 : B$	
exit	6	$\varnothing : A \to B$	

Figure 9.11:

We want to show $(B \to C) \to (A \to C)$. Fig 9.12 below gives the successful box.

	1		*Show* $A \to C$
	2	$a_1 : B \to C$	Assumption
	3	$a_1 : A \to C$	from box
l_1	4	$a_2 : A$	Assumption
	5	*continue with proof lines*	
	6	$l_2, \ldots, l_k$ *as in box* a_2 *of Figure* 9.11	
l_k	7	$a_2 : B$	
m_1	8	$a_1 : B \to C$	*reiteration*
m_2	9	$a_1 a_2 : C$	*modus ponens*
exit	10	$a_1 : A \to C$	
exit	11	$\varnothing : (B \to C) \to (A \to C)$	

Figure 9.12:

9.3 Peirce's axiom and the restart rule

Definition 9.3.1 (The Logic WC) *We now show the box conditions corresponding to the additional axiom (Peirce's Axiom)*

$$((A \to B) \to A) \to A.$$

We modify condition 2 of the box rules of 9.2.1 by allowing a box to show $A \to B$ to be successful if either B is proved or some earlier goal of an external box is proved. This rule I call the Restart Rule.

Restart Rule
As in Fig 9.13:

	1		Show previous goals $B_1, \ldots, B_n$
	2		Show B
	3	$a_1 : A$	Assumption
	4	$\vdots$	
	5	$\vdots$	
	6	$\alpha : B_i,$	$a_1 \in \alpha,$ for some i such that $i \in \{1, \ldots, n\}$
exit	7	$\alpha - \{a_1\} : A \to B$	

Figure 9.13:

Note that we exit with:

$$A \to B \text{ with label } \alpha - \{a_1\}$$

This is best explained by example.

Example 9.3.2 *From the assumptions $(A \to B) \to A$ show A. See fig 9.14.*

Note that although this axiom, Peirce's Axiom, usually gives us classical logic, it does not give us that much more in this case. $(A \to (B \to C)) \to (B \to (A \to C))$ still cannot be proved.

	1		*Show A*
	2	$a_1 : ((A \to B) \to A)$	*assumption*
	3	$\varnothing : A \to B$	*from box*
	4		*show B, previous goal A*
	5	$a_2 : A$	Assumption
	6	$a_2 : A$	*repetition of previous line*
	7	*success by restart*	
	8	$a_2 : A$	
	9	*Success of box because we showed an earlier goal.*	
exit	10	$\varnothing : (A \to B)$	
	11	$a_1 : A$	*modus ponens with (3) and (2)*

Figure 9.14:

Example 9.3.3 *Let us re-examine 9.2.3 in the logic* **WC** *(see fig 9.15).*

Example 9.3.4 *Let us check whether we can prove $A \to ((A \to B) \to A)$. (See fig 9.16)*

We now show the correspondence between Peirce's axiom and the Restart Rule. Suppose we have a box of the form (see fig 9.17):

The external box wants to show $A \to B$. In the course of the proof we want to show $C \to D$. We open a box to show C and succeed in the box to show B. We really need to show D. How do we proceed? The Restart Rule says, since B was a previous goal of an external box we can succeed (accept B instead of D).

To justify the Restart Rule, imagine for a moment that we had the extra assumption $a_2 : (B \to D)$ in the external box. Then everything is alright. We bring it in (reiteration) and use it as a ticket with the $\alpha : B$ which we already have and get $\alpha, a_2 : D$. Since $a_1 \in A$ already, we can exit with $(\alpha - \{a_1\}), a_2 : C \to D$.

You may say that this is fine, but the problem which we have solved is not the original problem, but the following one (see fig 9.18):

or equivalently (see fig 9.19):

This is not the original problem. However, if we have all instances of the Peirce's Axiom, as further assumptions, we can use $((B \to D) \to B) \to B)$ and modus ponens and get B.

Theorem 9.3.5 *Let Δ be a labelled database and assume $\Delta \vdash_{\mathbf{WC}} \alpha : Q$ (using the Restart Rule). Then there exist another box proof of $\alpha : Q$ from $\Delta \cup \Delta'$, where Δ' is a finite set of substitutions of Peirce's axiom all labelled $\varnothing$.*

Proof. By induction on the number n of nested boxes which use Restart.
Case $n = 0$
There is nothing to prove.
Case $n \geq 1$
Assume that Restart is used in m inner boxes. We eliminate the use of Restart systematically. Choose an inner box, say Box 2. We have the situation (see fig 9.20):

If in the inner box Restart is not used (ie D is proved) then we do nothing. Consider a box where B is proved, and an appeal to Restart is made. We want to prove B from A in **W** with the use of the extra assumption $((B \to D) \to B) \to B$ and without Restart.

Replace box 1 by the following proof

Box a_1	1		*Show* $B \to (A \to C)$
	2	$a_1 : A \to (B \to C)$	Assumption
	3	$a_1 : B \to (A \to C)$	from Box a_2
Box a_2	4		*Show* $A \to C$, *previous goal* $B \to (A \to C)$
	5	$a_2 : B$	Assumption
	6	$a_1 a_2 : A \to C$	from Box a_3
Box a_3	7		*Show* C, *Previous goals* $B \to (A \to C), A \to C$
	8		*Show* C
	9	$a_3 : A$	
	10	$a_1 : A \to (B \to C)$	
	11	$a_1 a_3 : B \to C$	
	12	*B cannot be brought in.*	
	13	*The previous goals are of no use here.*	
	14		
	15		

Figure 9.15:

1		*Show* $(A \to B) \to A$
2	$a_1 : A$	Assumption
3	$a_1 : A \to (B \to A)$	from Box
4		*Show* A
5	$a_2 : A \to B$	Assumption
6	*We cannot bring A in here. Even if we could,*	
7	*$A \to B$ will not have been used in the proof.*	
8		

Figure 9.16:

	1		Show B
	2	$a_1 : A$	Assumption
$line(l, l')$	3	$C \to D$	from Box
	4		Show D, previous goal B
	5	$b_1 : C$	Assumption
	6	$\vdots$	
	7	$\alpha : B$	
	8		

Figure 9.17:

1		Show B
2	$a_1 : A$	Assumption
3	$a_2 : B \to D$	Assumption
4		Show D
5	C	Assumption
6	$\vdots$	
7	D	
8		

Figure 9.18:

	1		Show $(B \to D) \to B$
	2	$a_1 : A$	Assumption
	3	$a_1 : (B \to D) \to D$	from Box a_2
Box a_2	4		Show B
	5	$a_2 : B \to D$	Assumption
	6	$a_1 a_3 : B$	from Box a_3
Box a_3	7		Show D
	8	$a_3 : C$	Assumption
	9	$\vdots$	
	10	$a_1 a_3 : D$	
	11	$a_1 a_3 : B$	
	12	$a_1 : (B \to D) \to B$	

Figure 9.19:

Box 1	1		Show B
	2	$a_1 : A$	Assumption
	3	$a_2 : B \to D$	Assumption
	4	Proof lines	
	5	leading to Box 2	
	6	below	
	7	some label:$C \to D$	from Box 2
Box 2	8		Show D
	9	$b_1 : C$	Assumption
	10	$\vdots$	
	11	B	
	12		

Figure 9.20:

line -1 $\varnothing$: $((B \to D) \to B) \to B$ (assumption)
(or reiteration if we put all instances of Peirce's Axiom at the very beginning of the entire deduction.)
Line 0: show $A \to B$ from box 1* (see fig 9.21)
The above modification can be done for every inner box of the form (see fig 9.22)
We can thus not use Restart, but we need extra instances of Peirce's axiom as axioms.
This completes the induction step and the proof of 9.3.5. ∎

Remark 9.3.6 *We now consider the metabox discipline for Rose's axiom R_n, namely:*

$$((A^n \to B) \to A) \to A).$$

We modify the Restart Rule essentially as follows. Suppose we are at an inner box and the box goal is to show B with some label. The goal A was a goal of some outer box. The Restart rule for R_n allows one to prove, in the inner box, the goal A n-times, possibly with different labels $\alpha_1, \ldots, \alpha_n$. These n proofs of A then count as a proof of B with label $\alpha_1\alpha_2 \ldots \alpha_n$. Note that for $n = 1$ this is the old restart rule for Peirce's axiom. To show B we show $\alpha : A$ and thus get $\alpha : B$, with the same label α.

The reader may ask what does it mean to show A n times? If A can be shown at least once, with label α, then one can count that n times and say we shown B with label $\alpha\alpha \ldots \alpha n$ times.

Example 9.3.7 *Show $((A \to (A \to B)) \to A) \to A$ in the logic with R_2. Use the following box: (see fig 9.23)*

Notice that in the inner box we could have also chosen to show A twice with label a_3. This would have given us $a_3a_3 : B$ which would have not *allowed us to exit from the a_2 box.*

Theorem 9.3.8 *The R_n-restart rule characterises the axiom R_n.*

Proof. We show soundness. Suppose in the inner box we want to show A. Since $\varnothing : ((A^n \to B) \to A) \to A$ is a theorem of the logic R_n, if we show $\alpha : (A^n \to B) \to A$ we can deduce by modus ponens $\alpha : A$.

So let us try to show $\alpha : (A^n \to B) \to A$. How do we do that? We open a box a_1 (see fig 9.24):

Note that a_1 will not be discharged until the exit of box a_1. The restart rule allows us not to write box a_1 explicitly. Completeness follows from the fact that axiom R_n can be proved using the restart rule R_n. ∎

Box 1$*$	1		Show B
	2	$a_1 : A$	Assumption
	3	$(B \to D) \to B.$	from box $1 * *$
Box $1 * *$	4		Show B
	5	$\varnothing : B \to D$	Assumption
	6	$\vdots$ proceed as in proof	
	7	$\vdots$ lines leading to	
	8	$\vdots$ Box 1 in figure 9.20	
	9	some label: $C \to D$	from Box 2$*$
Box 2$*$	10		Show D
	11	$b_1 : C$	Assumption
	12	$\vdots$	
	13	proceed as in box 1, figure 9.20	
	14	$\vdots$	
	15	B	
	16	proof of box 1 stops here.	
	17	We continue:	
	18	$B \to D$	reiteration
	19	D	MP1
exit	20	with $C \to D$	
	21	$\vdots$	
	22	$\vdots$ proceed as in outer box 1 of figure 9.20	
	23	$\alpha : B$	
exit	24	$(B \to D) \to B$	
	25	$\varnothing : ((B \to D) \to B) \to B$	reiteration of axiom
	26	$a_1 : B$	MP1
exit	27	$A \to B$	

Figure 9.21:

1		show $C_i \to D_i$
2		
3		
4		

Figure 9.22:

	1		*Show* $a_1 : A$
	2	$a_1 : A \to (A \to B)) \to A$	Assumption
	3	$\varnothing : A \to (A \to B),$	*from box 1*
	4	$\vdots$	
Box 1	5		*Show* $a_2 : A \to B$
	6	$a_2 : A$	Assumption
	7	$a_2 : A \to B$	from Box 2
Box 2	8		*Show* $a_3 : B$
	9	$a_3 : A$	Assumption
	10	$a_2 a_3 : B$	*we use the restart rule, since we have* A *twice,*
	11		*once with* a_3 *and once with* a_2
exit	12	$a_2 : A \to B$	
exit	13	$\varnothing : A \to (A \to B)$	
	14	$a_1 : A$	*modus ponens*
exit	15	$\varnothing : ((A \to (A \to B)) \to A) \to A$	

Figure 9.23:

Box a_1	1		Show A
	2	$a_1 : A \to (A \to \ldots (A \to B) \ldots)$	Assumption
	3	We try and get A in this box.	
	4	Suppose at some inner box we want to show B	
	5		Show B
	6	show $\alpha_1 : A, \alpha_2 : A \ldots \alpha_n : A,$	
	7	i.e. show A n times and do modus	
	8	ponens with $a_1 : A \to (\ldots A \to B)$	
	9	and get $a_1 \alpha_1 \ldots \alpha_n : B$	
	10	continue to argue and get $a_1 \alpha : A$	
exit	11	$\alpha : A \to (A \to \ldots (A \to B) \to A$	

Figure 9.24:

Chapter 10

Proof Theory for Resource Logics

10.1 Concatenation logic and linear logic

This section will present in detail one basic system which we call concatenation logic **CL**. It is intimately connected with associative concatenation and is as basic to the family of various resource logical systems, as concatenation is basic to functions and categories. The best way to present it is in the traditional manner, as a Hilbert system and study its properties again in a conservative way. We have the following objectives for **CL**:

- present a basic logic for resource which is weaker than linear logic and the well known Lambek calculus from categorical grammar.
- illustrate on this logic our ideas of *LDS*, metalangauge features, and control derivation properties.
- give semantics for this logic and extend this semantics to linear logic.
- give semantical meaning to some of the puzzles of linear logic.
- show that *CD*-logics involve three orthogonal features which can be mixed. These are:
    1. Control derivation and resource considerations.
    2. Hilbert simulation (via connectives) of meta-level features.
    3. Symmetrical simulation of control features (parallel to the proof nets of linear logic).

Definition 10.1.1 (Hilbert formulation of CL and LL) *The Hilbert system* **CL** *is defined by the following:*
Axioms

1. $A \to A$

2. $(A \to B) \to ((C \to A) \to (C \to B))$

Rules

$$\frac{A, A \to B}{B}\, MP$$

$$\frac{A \to B}{(B \to C) \to (A \to C)}\, RT$$

Notice that this logic is very weak. We cannot prove (as will be seen from the semantics) that:

3. $(A \to (B \to C)) \to (B \to (A \to C))$,
   Adding axiom (3) gives us the Hilbert system **LL** of linear implication. This system is still relatively weak. It cannot prove (4) nor (5) below:

4. $A \to ((A \to B) \to B)$, nor

5. $(A \to B) \to ((B \to C) \to (A \to C))$

Remark 10.1.2 *In categorical terms axiom 2 is Geach's principle:*

$$(A, B) \Rightarrow ((C, A), (C, B)).$$

Lambek calculus allows for 3., eg Lambek allows both:

$$A, (A, B) \Rightarrow B$$

$$(A, B), A \Rightarrow B$$

while we allow only the second

$$\mathbf{CL} \vdash (A \to B) \to (A \to B)$$

while

$$\mathbf{CL} \nvdash A \to ((A \to B) \to B)$$

Definition 10.1.3 (Proof from assumptions in the Hilbert System CL) *1. A database Δ is a sequence of formulas $A_1, \ldots A_n$. A database can be presented either as a seqeunce $(A_1, \ldots, A_n)$ or as a set of labelled formulas of the form $t_1 : A_1, \ldots t_n : A_n$, with the additional configuration $t_1 < \ldots < t_n$. $t_1 \ldots t_n$ are atomic labels.*

2. *We say $\vdash_{\mathbf{CL}} B$ iff there is an annotated sequence of formulas of the form $\alpha_1, \ldots, \alpha_m = B$ (the annotation may not be explicitly exhibited) such that each element is annotated either as an axiom (in which case it is indeed an instance of an axiom) or is annotated as (and is indeed) obtained from previous elements using MP or RT.*

3. *Let $\Delta = \{t_i : A_i\}, t_1 <, \ldots, < t_n$ be a database. We say $\Delta \vdash_{\mathbf{CL}} B$ iff there exists an annotated sequence $\beta_0, C_0, \ldots, \beta_m : C_m$ with $C_m = B$ such that each element of the sequence is either in Δ or is a theorem (ie $\vdash_{\mathbf{CL}} \beta_i$) or is obtained from previous elements using modus ponens.*

4. *If C_j is obtained from C_i and $C_k = C_i \to C_j$ by modus ponens then $max(\beta_k) < min(\beta_i)$ and $\beta_j = \beta_k\beta_i$, where '$\alpha\beta$' denotes concetenation of α then β.*

5. *If $\beta_i = \varnothing$ then $\mathbf{CL} \vdash B_i$.*

Remark 10.1.4 *1. Note that in the previous definition the label β_m of B must be*

$$\beta_m = (t_1, \ldots, t_n)$$

2. *If $t_1 : A \to A$ appars in the database, then $\beta : A \to A$ can appear in the proof either as an assumption, with $\beta = t_1$ or as an instance of a theorem, in which case $\beta = \varnothing$.*

*Later, we shall prove the deduction theorem for **CL**. The above distinction then becomes important. We do not have $A \to ((A \to A) \to A)$ as a **CL** theorem. By the deduction theorem, we need to show for $t_1 < t_2$*

$$t_1 : A, t_2 : A \to A \vdash A$$

and $A \to A$ has to be used in the proof. However, since $t_2 > t_1$, we canot apply modus ponens, as the next example discusses.

Example 10.1.5 *Consider the two databases:*
Δ_1 *is:*

1. $A \to B$
2. A

Δ_2 *is:*

1. A
2. $A \to B$

$\Delta_1 \vdash B$ because we have $A \to B, A, B$ as proof. Notice the assumptions appear in the proof in the same order as they appear in the database. Each assumption appears once exactly and tickets appear to the left of minor premisses. The above proof will not do for $\Delta_2 \vdash B$ since the order of the assumptions in the proof is not the same as in Δ_2.

Example 10.1.6 *Consider Δ_3:*

1. $A \to (A \to B)$
2. A

Consider the "proof" $A \to (A \to B), A, A \to B, B$. This "proof" is not acceptable, because to get B we use MP with A as minor which is to the left of the ticket. If we add $A \to (A \to B), A, A \to B, B$ we have used A twice. The proof is valid for the database Δ_4:

1. $A \to (A \to B)$
2. A
3. A

Notice that if we do not require that the minor comes after the ticket in MP then we need to count uses of assumptions in MP, if we want to control how many times an assumption is used.

The notion $\Delta \vdash B$ was defined using the notion of $\vdash_{\mathbf{CL}} A$. In the proof of B from Δ, we "landed" theorems. Can we suggest a direct discipline for proving Δ to B using only Δ?

The answer is 'yes' within the framework of labelled deducitve systems.

Definition 10.1.7 (LDS Discipline for CL) *We assume our assumptions are labelled as a sequence $a_1 : A_1, \ldots, a_n : A_n, a, < \ldots < a_n$. Our proof procedures use metaboxes. We use the following rules:*

1. **Definition of Metabox Level 0**
   A proof is a sequence of lines. Each line contains a labelled formula and a justification. The following three cases can be used:
   (a) *The line contains an assumption $a_i : A_i$ and the justification says "assumption".*
   (b) *The line contains $\alpha\beta : B$ and the justification says "from two previous lines by MP" and there are two previous lines, one with $\alpha : A \to B$ and one with $\beta : A$, and A is not justified as a "reiteration".*
   Notice the order of concatenation of the labels.
   We have further condition that $max(\alpha) < min(\beta)$.
   (c) *The line is $\gamma : A$ and it is justified as "reiteration".*

2. **Example**

a_1 :	$C \to (A \to B)$	*assumption*
a_2 :	C	*assumption*
a_3 :	$D \to A$	*assumption*
a_4 :	D	*assumption*
a_1a_2 :	$A \to B$	*MP*
a_3a_4 :	A	*MP*
$a_1a_2a_3a_4$:	B	*MP*

3. **Metabox of Level** $n+1$
A proof is a sequence of lines. Each line contains a labelled formula and a justification. The following four cases can be used:

   (a) The line contains an assumption $a_i : A_i$ and the justification says "assumption".

   (b) The line contains $\alpha\beta : B$ and the justification says "from two previous lines by MP" and there are two previous lines one with $\alpha : A \to B$ and one with $\beta : A$ and the following holds:

      i. A is not justified as "reiteration".

      ii. $max(\alpha) < \ min(\beta)$

   Notice the order of concatenation of the labels. The reader should note that (i) above follows from (ii).

   (c) The line is $\gamma : A$ and it is justified as "reiteration".

   (d) The line is $\gamma : A \to B$ and the justification is "from the metabox of level $m \leq n$" where line 1 of the metabox is $a : A$ where a is a new atomic label and its ordering is stipulated to be bigger than any atomic label in any assumption or reiteration in previous lines of the metabox of level $n+1$. The reiterations of the metabox of level m are all implicational formulas proved in previous lines of the metabox of level $n+1$. These can be enumerated as the next lines 2, 3, 4, ... of the metabox of level m. (In practice one can leave pointers or bring them in as needed.) The last line of the metabox of level m is $\gamma a : B$ and appropriate justification is present.

Example 10.1.8 *Show $\varnothing \vdash (A \to B) \to ((C \to A) \to (C \to B))$.*

Proof. Line 1 : $\varnothing : (A \to B) \to ((C \to A) \to (C \to B))$ "from box a", see Fig 10.1.
In practice, one does not bring in reiterations nor does one use excessive indices. Thus one can write the previous proof of Figure 10.1 as in Figure 10.2. ■

Example 10.1.9 *Another example is in Figure 10.3.*

Definition 10.1.10 *1. $a_1 : A_1, \ldots, a_n : A_n, a_1 < \ldots < a_n \Vdash_m B$ if and only if (def) there exists a metabox proof of level $\leq m$ with assumptions exactly $a_1 : A_1, \ldots, a_n : A_n$ with the last line with $a_1 \ldots a_n : B$.*

2. $\Delta \Vdash_{\mathbf{CL}} B$ iff $\Delta \Vdash_m B$ for some m

Remark 10.1.11 *The previous definition actually gave a natural deduction formulation for* **CL**. *The two rules are $\to$ Introduction and $\to$ Elimination. The complete definition is as follows:*
Data structure for assumptions
The assumptions are ordered in a sequence $a_1 : A_1, \ldots, a_n : A_n$ with $a_1 < a_2 < \ldots < a_n$.
$\to$ **Elimination Rule**

$$\frac{\begin{array}{l}\alpha : A \\ \beta : A \to B\end{array}}{\beta\alpha : B}$$

Box a	1		
	2	$a : A \to B$	Assumption
	3	$a : (C \to A) \to (C \to B)$	From Box b
Box b	4		Show $C \to B$
	5	$b : C \to A$	Assumption
	6	$a : A \to B$	Reiteration
	7	$ab : C \to B$	From Box c
Box c	8		Show B
	9	$c : C$	Assumption
	10	$b : C \to A$	Reiteration
	11	$a : A \to B$	Reiteration
	12	$bc : A$	From 9, 10
	13	$abc : B$	From 11, 12
	14		
	15		

Figure 10.1:

	1	Show $(A \to B) \to [(C \to A) \to (C \to B)]$	
Box a	2		Show $(C \to A) \to (C \to B)$
	3	$a : A \to B$	Assumption
	4	$a : (C \to A) \to (C \to B)$	From Box b
Box b	5		Show $C \to B$
	6	$b : C \to A$	Assumption
	7	$ab : C \to B$	From Box c
Box c	8		Show B
	9	$c : C$	Assumption
	10	$bc : A$	
	11	$abc : B$	
	12		
	13		

Figure 10.2:

1	Show $\varnothing : (C \to (A \to B)) \to (C \to ((D \to A) \to (D \to B)))$	
2	$a_1 : C \to (A \to B)$	Assumption
3	$a_1 : C \to ((D \to A) \to (D \to B))$	from Box
4	$a_2 : C$	Assumption
5	$a_1 : C \to (A \to B)$	Reiteration
6	$a_1 a_2 : A \to B$	from 4 and 5
7	$a_1 a_2 : (D \to A) \to (D \to B)$	from Box
8	$a_3 : D \to A$	Assumption
9	$a_1 a_2 a_3 : D \to B$	from Box
10	$a_4 : D$	Assumption
11	$a_3 a_4 : A$	from 8, 10
12	$a_1 a_2 : A \to B$	Reiteration
13	$a_1 a_2 a_3 a_4 : B$	11, 12
14		
15		
16		
17		

Figure 10.3:

$max(\beta) < min(\alpha)$.
$\rightarrow$ **Introduction Rule**
To show $\gamma : A \rightarrow B$, open a new box and get a new label a and temporarily append $a : A$ to the current sequence of assumptions. Proceed to prove B with label exactly γa. When successful exit the box and continue the main proof (with $a : A$ deleted from the current sequence of assumptions).

We now want to prove a deduction theorem for $\Delta \vdash_{\mathbf{CL}} B$.

Theorem 10.1.12 (Deduction theorem for CL)

$$(A_1, \ldots, A_n) \vdash_{\mathbf{CL}} B \textit{ iff } \vdash_{\mathbf{CL}} A_1 \rightarrow (A_2 \rightarrow \ldots \rightarrow (A_n \rightarrow B) \ldots).$$

Proof.

1. Assume $\vdash_{\mathbf{CL}} A_1 \rightarrow \ldots \rightarrow (A_n \rightarrow B) \ldots)$.
   We show $(A_1, \ldots, A_n) \vdash_{\mathbf{CL}} B$. The following is a proof:

$A_1 \rightarrow (\ldots \rightarrow (A_n \rightarrow B) \ldots)$	theorem
A_1	assumption
$A_2 \rightarrow (A_3 \rightarrow \ldots (A_n \rightarrow B) \ldots)$	MP
A_2	assumption
$\vdots$	
$A_n \rightarrow B$	MP
A_n	assumption
B	MP

   Note that our conditions about the support for each line of the proof are satisfied.

2. Assume $(A_1, \ldots, A_n) \vdash_{\mathbf{CL}} B$. We want to show that

   $$\vdash_{\mathbf{CL}} A_1 \rightarrow (A_2 \rightarrow \ldots \rightarrow (A_n \rightarrow B) \ldots)$$

   We prove this by induction on the length k of the proof of $t_1, \ldots, t_n : B$ from $t_1 : A_1; \ldots; t_n : A_n; t_1 < \ldots, < t_n$. Let $\beta_1 : C_1, \ldots, \beta_k : C_k$ be the proof with $C_k = B$.

   Case $k = 1$
   Subcase 1 We have $\vdash_{\mathbf{CL}} B$ and $(A_1, \ldots, A_n) = \varnothing$, the empty sequence. Then clearly the theorem holds.

   Subcase 2 We have $B = A_1, n = 1$ and clearly $\vdash_{\mathbf{CL}} A_1 \rightarrow B$ is a theorem.

   Case $k + 1$
   Subcase 1 B is a theorem. Since $\beta_k = \varnothing$ again $(t_1, \ldots, t_n) = \varnothing$ and we have here a direct proof of B from some other theorems.

   Subcase 2 B is an assumption. Then $A_1 = B$ as $\beta_k = (t_1, \ldots, t_n)$ and $\vdash_{\mathbf{CL}} A_1 \rightarrow B$.

   Subcase 3 B is obtained from two previous lines using modus ponens. Let the two lines be $\beta_i : C_i$ and $\beta_j : C_i \rightarrow B$. Then we have max$\beta_i <$ minβ_i and $\beta_j \beta_i = t_1 \ldots t_n$. These are the conditions of a successful proof.

   So for some $1 \leq m < n$
   $\beta_j = t_1 \ldots t_n$
   $\beta_i = t_{m+1} \ldots t_n$

   This implies by definition that

   $$(A_1, \ldots, A_m) \vdash_{\mathbf{CL}} A \rightarrow B$$

   and

   $$(A_{m+1}, \ldots, A_n) \vdash_{\mathbf{CL}} A.$$

By the induction hypothesis:

$$\vdash_{\mathbf{CL}} A_1 \to \ldots \to (A_m \to (A \to B) \ldots)$$

$$\vdash_{\mathbf{CL}} A_{m+1} \to (\ldots \to (A_n \to A) \ldots).$$

We need to show that

$$\vdash_{\mathbf{CL}} A_1 \to (A_2 \to \ldots (A_m \to \ldots \to (A_n \to B) \ldots)$$

This follows from lemma 10.1.13 below.

■

Lemma 10.1.13 (Cut for CL) *If* **L** *is an extension of* **CL** *then (1) and (2) below imply (3) below:*

1. $\vdash_{\mathbf{L}} (A_1 \to \ldots \to (A_n \to (A \to B)) \ldots)$
2. $\vdash_{\mathbf{L}} D_1 \to \ldots \to (D_m \to A) \ldots)$
3. $\vdash_{\mathbf{L}} A_1 \to \ldots \to (A_n \to (D_1 \to \ldots \to (D_m \to B) \ldots)$

Proof. By induction on m and n.
Case $m = n = 1$
We have to show that (1) and (2) imply (3):

1. $\vdash_{\mathbf{L}} A_1 \to (A \to B)$
2. $\vdash_{\mathbf{L}} (D_1 \to A)$
3. $\vdash_{\mathbf{L}} A_1 \to (D_1 \to B)$

From the RT-rule, we get using (2) that

4. $\vdash_{\mathbf{L}} (A \to B) \to (D_1 \to B)$
and using left transitivity we get

5. $(A_1 \to (A \to B)) \to (A_1 \to (D_1 \to B))$
we get (3) by modus ponens of (1) and (5).

Case $m = 1, n$ **arbitrary**
From $\vdash_{\mathbf{L}} D_1 \to A$ we get $\vdash_{\mathbf{L}} (A \to B) \to (D_1 \to B)$ and by induction on n we prove:

$$\vdash_{\mathbf{L}} (A_1 \to \ldots \to (A_n \to (A \to B) \ldots) \to (A_1 \to \ldots \to A_n \to (D_1 \to B) \ldots)$$

For $n = 1$ this follows from the left transitivity axiom. Assume the above for n and get it for $n + 1$ by another application of the left transitivity axiom.

Case $m + 1, n$ **arbitrary**
We have

$$\vdash_{\mathbf{L}} (A \to B) \to ((D_{m+1} \to A) \to (D_{m+1} \to B))$$

hence

$$\vdash_{\mathbf{L}} A_1 \to \ldots \to (A_n \to (A \to B)) \ldots) \to ((A_1 \to \ldots \to$$
$$(A_n \to ((D_{m+1} \to A) \to (D_{m+1} \to B)) \ldots)$$

and therefore

$$\vdash_{\mathbf{L}} A_1 \to \ldots \to (A_n \to (D_{m+1} \to A) \to (D_{m+1} \to B) \ldots)$$

on the other hand, we have

$$\vdash_{\mathbf{L}} D_1 \to \ldots (D_m \to (D_{m+1} \to A) \ldots)$$

hence by the induction hypothesis we get

$$\vdash_{\mathbf{L}} A_1 \to \ldots \to (A_n \to (D_1 \to \ldots (D_m \to (D_{m+1} \to B) \ldots)$$

This completes the proof of lemma 10.1.13.

■

Remark 10.1.14 *Note that in* **CL** *the rule of lemma 10.1.13 is equivalent to the left transitivity axiom.*

Let $X = A \to B, Y_1 = C \to A, Y_2 = C, U = A, V = B$, then

$$\vdash X \to (U \to V)$$

is

$$\vdash (A \to B) \to (A \to B)$$

and

$$\vdash Y_1 \to (Y_2 \to U)$$

is

$$\vdash (C \to A) \to (C \to A)$$

by the rule we get

$$\vdash X \to (Y_1 \to (Y_2 \to V))$$

which is

$$\vdash (A \to B) \to ((C \to A) \to (C \to B)).$$

Theorem 10.1.15 (Cut Rule for $\vdash_{\mathbf{CL}}$) $(A_1, \ldots, A_n, B, C_1, \ldots, C_m) \vdash_{\mathbf{CL}} X$ *and* $(B_1, \ldots, B_k \vdash B$ *imply*

$$(A_1, \ldots, A_n, B_1, \ldots, B_k, C_1, \ldots, C_m) \vdash_{\mathbf{CL}} X$$

Proof. Follows from lemma 10.1.13, because by theorem 10.1.12 we have:

$$\vdash_{\mathbf{CL}} A_1 \to \ldots \to (A_n \to (B \to (C_1 \to \ldots \to (C_m \to X) \ldots)$$

and

$$\vdash_{\mathbf{CL}} B_1 \to \ldots \to (B_k \to B) \ldots)$$

■

Theorem 10.1.16 (Cut rule for $\Vdash_{\mathbf{CL}}$) *(1) and (2) imply (3).*

1. $a_1 : A_1, \ldots, a_n : A_n, b : B \Vdash_{\mathbf{CL}} a_1 \ldots a_n b : X$
   $a_1 < a_2 < \ldots < a_n < b$
   a_i, b *atomic.*

2. $b_1 : B_1, \ldots, b_k : B_k \Vdash_{\mathbf{CL}} b_1 \ldots b_k : B$
   $b_1 < b_2 < \ldots < b_n, b_i$ *atomic.*

3. $a_1 : A_1, \ldots, a_n : A_n, b_1 : B_1, \ldots, b_k : B_k \Vdash_{\mathbf{CL}} a_1 \ldots a_n b_1 \ldots b_k : X$
   $a_1 < a_2 < \ldots < a_n < b_1 < b_2 < \ldots < b_k$.

Proof. Assume (1) and (2) hold with metaboxes M_1 of level m_1 and M_2 of level m_2. We would like to exhibit a metabox M which proves (3).

The assumptions of this new metabox are

$$\begin{array}{c} a_1 : A_1 \\ \vdots \\ a_n : A_n \\ b_1 : B_1 \\ \vdots \\ b_k : B_k \end{array}$$

We can continue the proof of the new metabox by bringing in metabox M_2. This metabox will prove B with label $b_1 \dots b_k$. We now have the following situation:

$$\begin{array}{ll} a_1 : A_1 & \text{assumption} \\ \vdots & \\ b_k : B_k & \text{assumption} \\ \boxed{\text{lines as in } M_2} & \\ b_1 \dots b_k : B & \end{array}$$

We would like now to continue with Metabox M_1 using the assumptions $A_1 \dots A_n$ and the B we have just proved. The problem may be that metabox M_1 is using B with label $b : B, a_1 < a_2 < \dots < a_n < b$ while the available B in our new metabox has label $b_1 \dots b_k$. Suppose we call the sequence $b^* = b_1 \dots b_k$. Certainly $a_n <$ min$(b^*) <$ max(b^*). The metabox operations on labels are concatenations of b (or b^*) with other labels, in which case we do not care if b or b^* is used and also, when we use modus ponens

$\alpha : C$
$\beta : C \to D$
we require
max$\beta <$ minα. When b is replaced by b^*, we get α^*, β^*. Since both b^* and b are greater than a_n, we have that
max$\gamma =$ maxγ^*
min$\gamma =$ minγ^*

Let M_1^* be the box which is like M_1 except that B is labelled b^*.

The steps in M_1^* will still be correct. We can thus continue and get $a_1 \dots a_n b^* : X$ in box M_1^*.

The complete proof of $a_1 \dots a_n b_1 \dots b_k : X$ is therefore:

$$\begin{array}{c} a_1 : A_1 \\ \vdots \\ a_n : A_n \\ b_1 : B_1 \\ \vdots \\ b_k : B_k \\ M_2 \\ M_1^* \end{array}$$

■

Theorem 10.1.17 (Equivalence of Hilbert and LDS formulations) $\vdash_{\mathbf{CL}} B$ *iff* $\varnothing \Vdash_{\mathbf{CL}} B$.

Proof.

1. We show that $\vdash_{\mathbf{CL}} B$ implies $\varnothing \vdash_{\mathbf{CL}} B$.

First we show that the **CL** Hilbert Axioms are Box provable, ie that $\varnothing, \Vdash_{\mathbf{CL}} A \to A$ and $\varnothing \Vdash_{\mathbf{CL}} (A \to B) \to ((C \to A) \to (C \to B))$. These are done in the examples. We have to check the RT rule, ie

$$\frac{\varnothing \Vdash_{\mathbf{CL}} A \to B}{\varnothing \Vdash_{\mathbf{CL}} (B \to C) \to (A \to C)}.$$

Assume we have a metabox M for $\varnothing \Vdash_{\mathbf{CL}} A \to B$:

$$M = \begin{array}{|c|}\hline b : A \\ \vdots \\ b : B \\ \hline \end{array}$$

We show a metabox for $\varnothing \Vdash_{\mathbf{CL}} (B \to C) \to (A \to C)$ (figure 10-4).

	1	$a : B \to C$	Assumption
	2	$a : A \to C$	From Box
	3	$b : A$	Assumption
Box M	4		
	5	Proof lines	
	6	of Box M	
	7	$b : B$	
	8	$a : B \to C$	Reiteration
	9	$ab : C$	MP
exit	10	$a : A \to C$	

Figure 10.4:

Second we show that $\Vdash$ is closed under modus ponens, namely that $\varnothing \Vdash_{\mathbf{CL}} A \to B$ and $\varnothing \Vdash_{\mathbf{CL}} A$ imply $\varnothing \Vdash_{\mathbf{CL}} B$. This we get by the cut rule on $\Vdash_{\mathbf{CL}}$. This completes the proof of (1).

2. We now show that if $\varnothing \Vdash_{\mathbf{CL}} B$ then $\vdash_{\mathbf{CL}} B$.

   This we prove by the following lemma.

Lemma 10.1.18 *Let M be a **CL** metabox proof of $b : B$ from $a_1 : A_1, \ldots, a_n : A_n, a_1 < a_2 < \ldots < a_n$. Here we assume that each new assumption is labelled by a unique new atomic label.*

Then for each line of the form $d_1 \ldots d_k : D$ in the box M where d_i are atomic, we have

$$\vdash_{\mathbf{CL}} D_1 \to (D_2 \to \ldots \to (D_k \to D) \ldots)$$

where D_i is the formula whose label is d_i.

Proof. Each line appears uniquely in the nested box in M and has a line number. We use induction. Each line in the metabox M is one of the following cases:

1. $d : D$ assumption, in which case $\vdash_{\mathbf{CL}} D \to D$,

2. $d_1 \ldots d_k, e_1 \ldots e_m : D$
   obtained from $d_1 \ldots d_k : E \to D$ and $e_1 \ldots e_m : E$ by modus ponens and $\max(d_i) < \min(e_i)$. By the induction hypothesis on earlier lines we have

$$\vdash_{\mathbf{CL}} E_1 \to \ldots \to (E_m \to E) \ldots)$$

and

$$\vdash_{\mathbf{CL}} D_1 \to \ldots \to (D_k \to (E \to D) \ldots).$$

Hence by a previous lemma

$$\vdash D_1 \to \ldots \to (D_k \to (E_1 \to \ldots (E_m \to D) \ldots)$$

3. $d_1 \ldots d_k : E \to D$ is obtained from a box (figure 10.5)

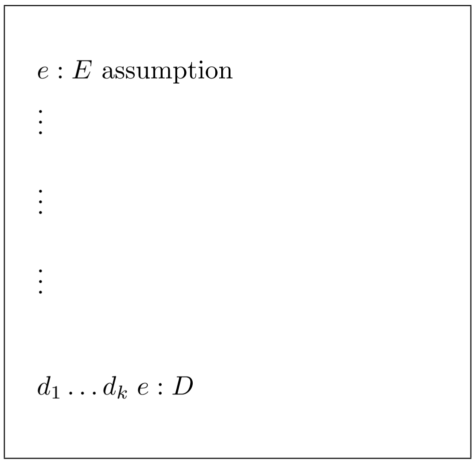


Figure 10.5:

Since $\vdash_{\mathbf{CL}} E \to E$ and all reiterations inside the box satisfy the lemma, by the induction hypothesis, we get

$$\vdash_{\mathbf{CL}} D_1 \to (D_2 \to \ldots (D_k \to (E \to D) \ldots).$$

■

This completes the proof of Theorem 10.1.17. ■

Theorem 10.1.19

$$\Delta \vdash_{\mathbf{CL}} B \textit{ iff } \Delta \Vdash_{\mathbf{CL}} B.$$

Proof. Assume $\Delta = (a_1 : A_1, \ldots, a_n : A_n)$, then
$(A_1, \ldots A_n) \vdash_{\mathbf{CL}} B$ iff

$$\vdash_{\mathbf{CL}} A_1 \to (A_2 \to \ldots \to (A_n \to B) \ldots)$$

iff $\varnothing \Vdash_{\mathbf{CL}} A_1 \to (A_2 \to \ldots \to (A_n \to B) \ldots)$
iff $\Delta \Vdash_{\mathbf{CL}} B$. ■

Lemma 10.1.20 *In* **CL**, *(1) is equivalent to (2), where $\vdash$ is as in 10.1.3*

1. $\vdash (A \to B) \to (D_1 \to (D_2 \to \ldots \to D_n) \ldots)$

2. *There exists a k such that* $\vdash D_1 \to (\ldots \to (D_k \to A) \ldots)$
$\vdash B \to (D_{k+1} \to (\ldots \to D_n) \ldots)$.

Proof. Assume (1), then by 10.1.12

$$t : A \to B, d_1 : D_1, \ldots, d_{n-1} : D_{n-1} \vdash td_1 \ldots d_{n-1} : D_n.$$

$$t < d_1 < d_2 < \ldots < d_{n-1}$$

By the **CL** proof discipline, there is a sequence

$$\alpha_1 : E_1, \ldots, \alpha_m : E_m,$$

where each $\alpha_k : E_k$ is either a theorem of the logic with $\alpha_i = \varnothing$ or is obtained from previous two lines by modus ponens, namely we have:

$$\alpha_i : E_i, \alpha_j : E_i \to E_k$$

with $j < i,\ \max(\alpha_j) <\ \min(\alpha_i)$ and $\alpha_k = \alpha_j \alpha_i$.

$A \to B$ can be used only as a ticket because it is the first in the sequence. Thus for some $\alpha_i : A$ and $\alpha_j : B$, we have $\alpha_j = t\alpha_i$ and $\min(\alpha_i) > t$. We claim that for some $k, \alpha_i = d_1 \ldots d_k$. This claim follows from Remark 10.1.4 because supports cannot have gaps in them. It now follows that there is a proof of $td_1, \ldots, d_{n-1} : D_n$ from the data

$$b : B, d_{k+1} : D_{k+1}, \ldots, d_{n-1} : D_{n-1}$$

with $b < d_{k+1} < \ldots < d_{n-1}$, where we have renamed $td_1 \ldots d_k = b$. Similarly, there is a proof of A from the data $d_1 : D_1, \ldots, , d_k : D_k$. Thus we have shown

$$\vdash D_1 \to \ldots \to (D_k \to A) \ldots)$$

and

$$\vdash B \to (D_{k+1} \to \ldots \to D_n) \ldots).$$

Assume (2), show (1). Again using the deduction theorem assume

$$\begin{array}{l} t : A \to B \\ d_i : D_i, i = 1, \ldots n-1 \end{array}$$

we want to show $td_1 \ldots d_{n-1} : D_n$.

From the first assumption we get $d_1 \ldots d_k : A$, then using $t : A \to B$ we get $td_1 \ldots d_k : B$, then by modus ponens we get $td_1 \ldots d_k : D_{k+1} \to (\ldots \to D_n) \ldots)$. We continue to get $td_1 \ldots d_{n-1} : D_n$. ■

Definition 10.1.21 *In* **CL**, *let* $(A_1, \ldots, A_n), (B_1, \ldots, B_m)$ *be two sequences of formulas, with* $m \leq n$. *Define* $A_1, \ldots, A_n) \vdash_{\mathbf{CL}} (B_1, \ldots, B_m)$ *iff when we label the data* $a_1 : A_1, \ldots, a_n : A_n$, *with* $a_1 < a_2 < \ldots < a_n$ *there exist* $1 \leq k_1 < k_2 < \ldots < k_m \leq n$ *such that for each* $1 \leq i \leq m$

$$\begin{array}{l} a_1 : A_1, \ldots, a_{k_1-1} : A_{k_1-1} \vdash_{\mathbf{CL}} a_1 \ldots a_{k_1-1} : B_1 \\ \vdots \\ a_{k_m+1} : A_{k_m+1}, \ldots, a_n : A_n \vdash_{\mathbf{CL}} a_{k_m+1} \ldots a_n : B_m \end{array}$$

Example 10.1.22 *We have*

$$(A \to B, A, C \to D, C) \vdash_{\mathbf{CL}} (B, D).$$

10.2 Concatenation logic with additional connectives

We will progressively add in this section various connectives to the basic **CL** implication.

Definition 10.2.1 *Let* **L** *be any extension of* **CL***. We define several extensions of* **L** *with new connectives and axioms as follows:*

1. *Let* $\mathbf{L}(\otimes)$ *be the extension of* **L** *with the binary connective* $\otimes$*. Add the following axioms to those of* **L** *of definition 10.1.1*
   (a) $A \to (B \to (A \otimes B))$
   (b) $((A \to (B \to C)) \to ((A \otimes B) \to C))$
2. *Let* $\mathbf{L}(\neg)$ *be the extension of* **L** *with a unary connective* $\neg$ *and the following axioms.*
   (a) $\neg\neg A \leftrightarrow A$
   (b) $(\neg(A \to \neg B) \to C] \leftrightarrow (A \to (B \to C))$
   (c) *The rule* $\dfrac{\vdash A \to B}{\vdash \neg B \to \neg A}$
3. *Let* $\mathbf{L}(\uplus\ )$ *be the extension with the additional binary* $\uplus$ *and the axioms:*
   (a) $((A \to B)\uplus C) \leftrightarrow (A \to (B\uplus C))$
   (b) $\dfrac{\vdash A \to A', \vdash B \to B'}{\vdash A\uplus B \to A'\uplus B'}$
4. *Let* **L***(!) be the system with the additional modality ! and the axioms:*
   (a) $!A \to A$
   (b) $!A \to (B \to !A)$
   (c) $\dfrac{\vdash C \to (q \to A)}{\vdash C \to A}$ *, q atomic not in C or A*

Remark 10.2.2 1. *Note that* $\vdash ((A \otimes B) \to C) \to (A \to (B \to C))$ *holds in* $\mathbf{L}(\otimes)$*.*

2. *Also notice that* $\otimes$ *is associative. Using axiom (a) we get*

$$\vdash A \to ((B \otimes C) \to (A \otimes (B \otimes C)))$$

hence

$$\vdash A \to (B \to (C \to A \otimes (B \otimes C)))$$

hence

$$\vdash (A \otimes B) \to (C \to A \otimes (B \otimes C))$$

hence

$$\vdash ((A \otimes B) \otimes C) \to (A \otimes (B \otimes C))$$

3. *Note that in* $\mathbf{CL}(\neg)$*, the following is provable:*

$$\vdash A \to (B \to \neg(A \to \neg B))$$

4. *Note that in* $\mathbf{LL}(\neg)$ *the following is provable:*

$$\vdash (A \to B) \to (\neg B \to \neg A)$$

To show that begin with

$$\vdash_{\mathbf{LL}(\neg)} A \to ((A \to B) \to B)$$

hence

$$\vdash_{\mathbf{LL}(\neg)} \neg(A \to \neg(A \to B)) \to B$$

hence

$$\vdash_{\mathbf{LL}(\neg)} \neg B \to (A \to \neg(A \to B))$$

hence

$$\vdash_{\mathbf{LL}(\neg)} \neg(\neg B \to \neg A) \to \neg(A \to B)$$

hence

$$\vdash_{\mathbf{LL}(\neg)} (A \to B) \to (\neg B \to \neg A).$$

5. *Note that in* **LL**$(\neg)$ *the following holds:*

$$\vdash_{\mathbf{LL}(\neg)} \neg B \leftrightarrow (B \to \neg(B \to B))$$

To show that observe that

$$\vdash (B \to B) \to (B \to B)$$

hence

$$\vdash B \to ((B \to B) \to B)$$

hence

$$\vdash \neg(B \to \neg(B \to B)) \to B$$

hence

$$\vdash \neg B \to (B \to \neg(B \to B)).$$

Further observe that

$$\vdash B \to ((B \to \neg(B \to B)) \to \neg(B \to B))$$

hence

$$\vdash \neg((B \to \neg(B \to B)) \to \neg(B \to B)) \to \neg B$$

hence

$$\vdash (B \to \neg(B \to B)) \to ((B \to B) \to \neg B)$$

hence

$$\vdash (B \to B) \to (B \to \neg(B \to B) \to \neg B)$$

hence

$$\vdash (B \to \neg(B \to B)) \to \neg B.$$

Theorem 10.2.3 *In* **CL**$(\otimes)$ *and in* **LL**$(\otimes)$ *the following holds:*

$$\vdash A \otimes B \textit{ iff } \vdash A \textit{ and } \vdash B.$$

Proof.

1. Assume $\vdash A$ and $\vdash B$. Since by axiom (a) $\vdash A \to (B \to A \otimes B)$, we get $\vdash A \otimes B$.

2. Assume $\vdash A \otimes B$. Let $A_1, \dots, A_n = A \otimes B$ be a proof sequence of $A \otimes B$. We show that there exists another proof of $A \otimes B$ of length $n' \leq n$, of the form $A'_1, \dots, A'_{n'}$ with $A'_{n'} = A \otimes B$ such that for some $j, k < n', A'_j = A$ and $A'_k = B$ and such that every substitution instance of an axiom in the proof is already in $A_1, \dots, A_n$. The proof is by induction on the length n of the proof of $A \otimes B$.

Length 1:
$A \otimes B$ is a substitution instance of an axiom. This case does not arise, as axioms do not have this form.
Length $n + 1$
$A \otimes B$ is obtained from previous elements of the sequence using MP or RT. The use of RT does not arise because it does not yield the form $A \otimes B$. Thus $A \otimes B$ is obtained from two previous elements in the sequence, $A_r = C_1$ and $A_s = C_1 \to A \otimes B$ by modus ponens. Let m be the maximal number such that there are $C_m, \ldots, C_1$ and $C_m \to (C_{m-1} \to \ldots \to C_1 \to (A \otimes B) \ldots)$ earlier in the proof such that $A \otimes B$ is obtained from them by modus ponens. We examine the cases $m = 1, m = 2$ and $m \geq 3$.
Case $m = 1$
Consider the ticket $A_s = C_1 \to A \otimes B$.

1. It is either an instance of an axiom, in which case it must be an instance of the identity axiom $X \to X$. In this case $C_1 = A \otimes B$ and since C_1 is an earlier element A_r of the sequence, by the induction hypothesis, we have for some $j, k < r', A'_j = A$ and $A'_k = B$.

2. It is obtained from earlier elements using RT. Again $C_1 \to A \otimes B$ is the wrong form for a result of RT, so this case does not arise.

Case $m = 2$
$C_1 \to (A \otimes B)$ is obtained by modus ponens from two earlier elements $A_i = C_2$ and $A_t = C_2 \to (C_1 \to A \otimes B)$. Again there are several possiblilities.

1. $C_2 \to (C_1 \to A \otimes B)$ is an instance of an axiom $X \to X$, in which case $C_2 = C_1 \to (A \otimes B)$ and we have a shorter proof of $A \otimes B$ and we use the induction hypothesis.

2. $C_2 \to (C_1 \to A \otimes B)$ is an instance of the axiom

$$(\alpha \to (\beta \to \gamma)) \to (\alpha \otimes \beta \to \gamma).$$

In this case

$$\begin{aligned} C_1 &= \alpha \otimes \beta \\ \gamma &= A \otimes B \\ C_2 &= \alpha \to (\beta \to \gamma) \end{aligned}$$

and we get that we have:

$$\begin{array}{l} C_1 = (\alpha \otimes \beta) = A_r \\ C_2 = (\alpha \to (\beta \to \gamma)) = A_i \\ (\alpha \to (\beta \to A \otimes B)) \to ((\alpha \otimes \beta) \to (A \otimes B)) = A_t \end{array}$$

The current proof contains the steps

$A_r = C_1 = \alpha \otimes \beta$
$A_t = (\alpha \to (\beta \to A \otimes B)) \to ((\alpha \otimes \beta) \to (A \otimes B))$, an instance of an axiom
$A_i = C_2 = \alpha \to (\beta \to A \otimes B)$
$A_s = \alpha \otimes \beta \to (A \otimes B)$ by modus ponens from A_i and A_t.
$A_n = A \otimes B$ by modus ponens from A_r and A_s.

By the induction hypothesis on r, there exist a proof $A'_1, \ldots, A'_{r'} = \alpha \otimes \beta$ with $r' \leq r$ and there exist $j, k < r'$ such that $A'_j = \alpha$ and $A'_k = \beta$. Further every substitution instance of an axiom in the A'_r proof already appears in the A proof. We can thus produce a shorter proof of $A \otimes B$ by taking out lines A_r, A_t and A_s and leaving the following:

$$A_i = C_2 = \alpha \to (\beta \to A \otimes B)$$

$s : \beta \to (A \otimes B)$ modus ponens of A_i with A_j
$A_m = A \otimes B$ modus ponens with previous line and A_k.

By renumbering we get a shorter proof of $A \otimes B$. We replaced the original proof containing $A_r A_t A_i A_s A_m$ with a proof containing A_i, line s and A_m. Note that the new proof is built up from the old lines of the old proof except for one new line $\beta \to A \otimes B$.

Now we can use the induction hypothesis on the shorter proof and conclude that there exists another proof $A''_1, \ldots, A''_{n''} = A \otimes B$ such that for some earlier lines in the proof $A''_{k''}$ and $A''_{m''}$ are equal to A and B respectively.
The A'' proof uses some and no more of the substitution instances of axioms which the original proof uses.

3. $C_2 \to (C_1 \to (A \otimes B))$ is an instance of

$$\vdash \alpha \to (\beta \to (\alpha \otimes \beta)).$$

In this case $C_1 = B, C_2 = A$ or in the case of $\mathbf{LL}(\otimes)$ we may also have $C_1 = A, C_2 = B$ and obviously our condition holds.

4. Now assume that we used the rule RT, in which case for some y, $C_1 \to y$ is an earlier element and $C_2 = y \to A \otimes B$ is an earlier element. In this case we can cut out the case of RT and obtain $A \otimes B$ directly by MP with $C_1, C_1 \to y$ and C_2. Again we use the induction hypothesis.

Case $m \geq 3$:
We now examine the general case. $A \otimes B$ is obtained by modus ponens from earlier $C_m, C_{m-1}, \ldots, C_1$ and

$$C = ((C_m \to \ldots \to (C_1 \to (A \otimes B)) \ldots).$$

Assume m was maximal for the use of modus ponens.
We now examine the possibilities for how C was obtained.

1. C is an instance of an axiom of the form
   (a) $X \to X$
   (b) $(X \to Y) \to [(Z \to X) \to (Z \to Y)]$
   (c) $((X \to (Y \to Z)) \to ((X \otimes Y) \to Z))$
   (d) $(X \to (Y \to (X \otimes Y)))$
   In the case of $\mathbf{LL}(\otimes)$ also
   (d′) $Y \to (X \to (X \otimes Y))$
2. C is obtained by using RT.
3. The case that C was obtained by modus ponens does not arise because we assumed m is maximal.

We examine each case.
Case 1a
In this case $C_m = C_{m-1} \to (\ldots C_1 \to (A \otimes B) \ldots)$. We can cut C out of the proof and get $A \otimes B$ with a shorter proof.
Case 1b
In this case

$$\begin{aligned} &C_m = X \to Y \\ &C_{m-1} = Z \to X \\ &C_{m-2} = Z \\ &C_{m-3} \to \ldots \to (C_1 \to A \otimes B) \ldots) = Y \end{aligned}$$

The current proof has in it the following lines:

r_1: $C_m = X \to (C_{m-3} \to (\ldots (C_1 \to (A \otimes B) \ldots).$

r_2: $C_{m-1} = C_{m-2} \to X$

r_3: C_{m-2}

r_4: $C = C_m \to (C_{m-1} \to C_{m-2} \to (C_{m-3} \ldots \to (C_1 \to (A \otimes B) \ldots)$

r_5: $C_{m-1} \to (C_{m-2} \to \ldots \to (C_1 \to (A \otimes B) \ldots)$

r_6: $C_{m-2} \to (\ldots \to (C_1 \to (A \otimes B) \ldots)$

r_7: $C_{m-3} \to (\ldots \to (C_1 \to (A \otimes B) \ldots)$

Lines $r_1 r_2 r_3$ are obtained in an earlier part of the proof. Line r_4 is an instance of the axiom and $r_5 r_6$ and r_7 are obtained by MP.

We can shorten the proof by letting r_2 and r_3 give us X by MP and using X we get r_7 from r_1. Thus we cut out r_4 and shorten the proof by one substitution instance of an axiom.

Case 1c

In this case

$$\begin{aligned}
&C = (\alpha \to (\beta \to \gamma)) \to ((\alpha \otimes \beta) \to \gamma)\\
&C_m = (\alpha \to (\beta \to \gamma))\\
&C_{m-1} = \alpha \otimes \beta\\
&C_{m-2} \to \ldots \to (C_1 \to (A \otimes B)) \ldots) = \gamma
\end{aligned}$$

This case is the same as case $m = 2$ subcase (3). By the induction hypothesis, $\vdash \alpha$ and $\vdash \beta$ with shorter proofs. Thus we can get γ using C_m directly.

Case 1d

In this case

$$\begin{aligned}
&C_m = X\\
&C_{m-1} = Y\\
&(C_{m-2} \to (C_{m-3} \to \ldots \to (C_1 \to (A \otimes B)) \ldots) = X \otimes Y.
\end{aligned}$$

We must have in this case $m = 2, X = A$ and $Y = B$ and the desired result follows.

Case 1d′

Similar to case 1d.

Case 2

In this case C is obtained by using RT. This means that for some X, Y, Z

$$\vdash X \to Y$$

and

$$C = (Y \to Z) \to (X \to Z)$$

hence

$$\begin{aligned}
&C_m = (Y \to Z)\\
&C_{m-1} = X\\
&C_{m-2} \to (\ldots \to (C_1 \to (A \otimes B) \ldots) = Z
\end{aligned}$$

we thus have:

$$\begin{aligned}
&\vdash C_{m-1} \to Y\\
&\vdash C_{m-1}\\
&\vdash Y \to Z\\
&C = (Y \to Z) \to (C_{m-1} \to Z)
\end{aligned}$$

The existing proof, using C is

$\vdash C_{m-1} \to Y$	existing subproof
$\vdash C = (Y \to Z) \to (C_{m-1} \to Z)$	use RT
$\vdash C_m = Y \to Z$	existing subproof
$\vdash C_{m-1}$	existing subproof
$\vdash C_{m-1} \to Z$	MP
$\vdash Z$	MP

We get Z using RT and two MP.

We can simplify by cutting out RT as follows:

$\vdash C_{m-1} \to Y$	existing
$\vdash C_{m-1}$	existing
$\vdash Y$	MP
$\vdash C_m = Y \to Z$	existing
$\vdash Z$	MP

By the induction hypothesis for the overall shorter eventual proof of $A \otimes B$ we get the theorem.

Case 3

C is obtained by modus ponens. This case cannot arise because we assumed m was maximal.

This completes the proof of the theorem. ■

The language of **CL** and that of **CL**$(\otimes)$ contains only atomic propositions and the implication or concatenation symbols $\to$ and $\otimes$. We want to enrich the language with a unary propositional function $\odot$, associating with each atomic q another atom $q^{\odot}$. We shall eventually stipulate that $q^{\odot\odot} = q$, but at this stage this is not needed.

Example 10.2.4 *We are now looking for a labelled discipline for* **CL**$(\otimes)$*. We have some problems. Suppose we use the discipline of* **CL** *of 10.1.3. Since* $\vdash (A \to (B \to C)) \leftrightarrow ((A \otimes B) \to C)$ *we must have the following successful derivations:*

1. $t_1 : A \to (B \to C), t_2 : A \otimes B \vdash t_1t_2 : C, t_1 < t_2$
2. $a_1 : (A \otimes B) \to C, a_2 : A, a_3 : B \vdash a_1a_2a_3 : C, a_1 < a_2 < a_3$
3. $t_1 : A \to (B \to C), t_2 : D \to (A \otimes B), t_3 : D \vdash t_1t_2t_3 : C, t_1 < t_2 < t_3.$

The present **CL** *discipline for* $\vdash$ *cannot achieve the above. The problem can be seen immediately when we consider modus ponens. From* $t_1 : C \to (A \otimes B)$ *and* $t_2 : C$ *we get* $t_1t_2 : A \otimes B$ *but we have no labels* $a_1 : A$ *and* $a_2 : B$ *to continue the deduction and use with* $A \to (B \to D)$ *to get* D*, even though* $A \to (B \to D)$ *is equivalent to* $(A \otimes B) \to D$*. We must therefore be able to split the label* t_1t_2 *of* $(A \otimes B)$ *into two labels* a_1 *and* a_2*, one for* A_1 *and one for* B*. Can we present a consistent discipline for doing that? The answer is yes.*

To develop a labelling discipline for **CL**$(\otimes)$, we get our ideas from the problems discussed in the previous example and from the exact nature of the labelling propagation of **CL**, as studied in Remark 10.1.4. **CL** labels (of 10.1.3) are atomic points. A database is a linearly ordered set of formulas, whose labels are linearly ordered.

For example

$$a_1 : A_1, \ldots, a_n : A_n, a_1 < a_2 <, \ldots, < a_n$$

is a database. We can think of a_i geometrically as points on the rational line, ie as rational numbers increasing in order.

When we proceed forward using modus ponens, proving conclusions from the data, as described in 10.1.3 and Remark 10.1.4, the support of any formula $\alpha : B$ proved, ie its label α, is a continuous interval $[a_i \ldots a_k]$, with no gaps.

Our idea is to label the assumption by disjoint intervals on the rationals and regard α as the end extension union.

Thus our labels are

$$[_{a_1}] : A_1, \ldots, [_{a_n}] : A_n$$

with $a_1 < a_2 < \ldots < a_n$ on the rational line. An interval is smaller than another iff all of its points come before the other. Thus any $\alpha : B$, where $\alpha = a_i, \ldots, a_k$ proved from the database, becomes the interval $[a_i \ldots a_k] : B$.

It is now easy to split any $[t_1 \ldots t_k] : A \otimes B$ into two labels $a_1 \otimes a_2$. We regard a_1, a_2 as two intervals and stipulate

$$a_1 : A, a_2 : B \quad a_1 \otimes a_2 = [t_1 \ldots t_k].$$

Mathematically the operation is simple and obvious, but the interval view gives it an easy geometrical meaning.

We have to allow for the *creation* of intervals (labels) during proofs and for carrying along systems of equalities of interval unions. Before we give the formal definitions, we consider example 10.2.5.

Example 10.2.5 *1. Prove from the database*

$$t_1 : A \to (B \to C), t_2 : A \otimes B, t_1 < t_2$$

The conclusion

$$t_1 t_2 : C$$

Here is the proof:

	Proof Line	Configuration
1	$t_2 : A \otimes B$ *assumption*	$t_1 < t_2$
2	create $a_1 : A, a_2 : B$ $a_1 < a_2, t_2 = a_1 a_2$	$t_1 < a_1 < a_2$ $a_1 a_2 = t_2$
3	$t_1 a_1 : B \to C$ *modus ponens*	*same*
4	$t_1 a_1 a_2 : C$ *modus ponens*	*same*

Since the configuration logically proves the rational interval (in the algebra $t_1 a_1 a_2 = t_1 t_2$), we succeeded in our proof.

2. Prove from the database

$$a_1 : (A \otimes B) \to C, a_2 : A, a_3 : B, a_1 < a_2 < a_3$$

the conclusion

$$a_1 a_2 a_3 : C.$$

Here is the proof

	Proof Line	Configuration
1	$a_2 : A$ *assumption*	$a_1 < a_2 < a_3$
2	$a_3 : B$ *assumption*	*same*
3	*Since the labels of lines (1) and (2) are* adjacent *in the configuration and $a_1 < a_2$, we get: $a_2 a_3 : A \otimes B$*	*same*
4	$a_1 a_2 a_3 : C$ *modus ponens*	

Notice we used the rule

$$\frac{x : A, y : B, x < y}{xy : A \otimes B} \quad \textit{provided the configuration proves that } x, y \textit{ are adjacent.}$$

Definition 10.2.6 *1. A* **CL**$(\otimes)$ *database is a sequence of formulas of the form $a_1 : A_1, \ldots, a_n : A_n$ and a configuration $a_1 < a_2 < \ldots < a_n$.*

2. *A proof from the database is a sequence of lines, each line comprises of a labelled formula $\alpha : B$, a justification and a configuration. The initial configuration is the original database configuration, namely $a_1 <, \ldots, < a_n$. The justification for the line $\alpha : B$ can be either that B is a theorem of* **CL**$(\otimes)$*, in which case $\alpha = \varnothing$, and there is no change in configuration or that B is an assumption $a_i : A_i$ from the database, in which case $\alpha = (a_i)$ and there is no change in configuration or that $\alpha : B$ is obtained from previous lines by a rule, in which case α is the label givn by the rule and the configuration changes according to the stipulation of the rule. The rule can be one of the following:*

   (a) **Modus Ponens**

   $$\begin{array}{l} \alpha : A \\ \beta : A \to B \\ \underline{max(\beta) < min\ (\alpha)} \\ \beta\alpha : B, \textit{ no change in configuration.} \end{array}$$

   (b) $\otimes$ **Elimination Rule**

   $$\frac{a_i, \ldots, a_k : A \otimes B}{x : A, y : B,}$$

   where x, y are new labels. We stipulate in the configuration that $xy = a_i, \ldots, a_k$ and that $a_1 < \ldots < a_{i-1} < x < y < a_{k-1} < \ldots a_n$, and that x is adjacent to y. The new configuration is obtained from the old one by adding the above stipulations.

   (c) $\otimes$ **Introduction rule**

   $$\frac{x : A, y : B, x < y, x \textit{ adjacent to } y}{x \otimes y : A \otimes B}$$

3. *We say $a_1 : A_1, \ldots, a_n : A_n \vdash \alpha : B$*
   $a_1 < \ldots < a_n$
   iff there exists a proof with last line $\beta : B$ and the configuration associated with the last line can prove $\beta = \alpha$.

10.3 Axiomatisation of W

The previous chapter on *Resource*-logics discussed *LDS* conditions for resource restrictions. The concatenation logic **CL** was introduced and studied in some depth, including a Hilbert axiomatisation for it. This chapter will study the connections between metabox disciplines, Hilbert axiomatisations and semantical interpretation for various resource logics in a systematic and detailed way. The metabox and labelling discipline is much richer in its ability of characterising consequence relations than the Hilbert type system discipline, which generates only theorems. The reader may thus wonder why we should bother with Hilbert type axiomatisations at all. They are about the crudest tool for characterising consequence one can have. One reason is that Hilbert axiom systems are good for comparison. Here is one system; add an axiom and you get another; take out an axiom and you get a third. Whereas in any other algorithmic proof system, where the rules are richer, it is more difficult to digest the meaning of relative changes. Hilbert systems are simple and therefore differences between systems are crude and obvious, they take the form of axioms.

There is another reason for studying Hilbert formulations. Hilbert systems are *not* crude. We shall see in the metalevel chapter that a Hilbert system can be used as a metalanguage for *LDS*. Thus with additional connective $\sharp$, the pair $\alpha : A$ can be represented as $\sharp(\alpha, A)$, a binary connective formula, and Hilbert type axioms can be put forward on $\sharp$ which forces upon it the meaning of *α is a label for A*, and further axioms can be tailored to relfect any labelling discipline. (See the metaleval chapter for details). It is therefore important for *LDS* to study and understand equivalent Hilbert type formulation of *LDS* systems.

We saw in the previous chapter how one can axiomatise Hilbert style, all wffs A of the *LDS* system **CL** which can be proved with the empty label, ie $\Vdash_{\mathbf{CL}} \varnothing : A$. We now systematically generalise the method to neighbouring logics and do the logic **W** in detail.

The logic **W**, initially presented in 7.2.1, extends the logic **CL** by adopting a weaker restriction on the $\rightarrow$ Elimination rule of **CL**, as given in 10.1.3. Assumptions $a_1 : A_1, \ldots, a_n : A_n$ are ordered in a sequence. $a_1 < a_2 < \ldots a_n$, just like in the case of **CL**.

The modus ponens

$$\frac{\begin{array}{l}\alpha : A\\ \beta : A \rightarrow B\end{array}}{\beta\alpha : B}$$

is allowed when $\max(\beta) < \max(\alpha)$. Recall that in the case of **CL** we had $\max(\beta) < \min(\alpha)$. The $\rightarrow$ introduction rule is the same.

The above restriction allows us to prove in **W** the right transitivity axiom.

$$\vdash (A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$$

but we cannot prove in **W** the commutativity axiom:

$$(A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C)).$$

Let us give a formal definition.

Definition 10.3.1 *1.* **Hilbert Formulation of W.**

Axioms

(a) $A \rightarrow A$

(b) $(A \rightarrow B) \rightarrow ((C \rightarrow A) \rightarrow (C \rightarrow B))$

(c) $A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$

Rules

$$MP.\ \frac{A, A \rightarrow B}{B}$$

2. **Proof from assumptions for W.**

To define the notion $\Delta \vdash_{\mathbf{W}} B$ we use a variant of definition 10.1.3. For the case of **W**, *we replace in condition (4) of definition 10.1.3 by the condition $max(\beta_k) < max(\beta_i)$.*

Definition 10.3.2 (LDS Discipline for W) *We adopt a similar definition to 10.1.7, with the following change:*

- *Replace the condition $max(\alpha) < min(\beta)$ appearing in the definitions of metabox of level 0 and of level $n+1$ by the condition $max(\alpha) < max(\beta)$.*
- *The notion of $\Delta \Vdash_{\mathbf{W}} B$ can now be defined as in 10.1.10.*

We just note that to obtain the Hilbert system corresponding to **W**, we extend **CL** with the right transitivity axiom.

We now proceed to prove the equivalence of the metabox discipline for **W** and the Hilbert axiomatisation. We begin with an example.

Example 10.3.3 *Consider the following*

Assumptions

$a_1 : A \rightarrow B$

$a_2 : C \rightarrow A$

Show

$C \rightarrow B$

This can be done by a box (fig 10.6):

Box a_3	1		Show B
	2	$a_3 : C$	Assumption
	3	$a_2 : C \to A$	reiteration
	4	$a_2 a_3 : A$	MP
	5	$a_1 : A \to B$	reiteration
	6	$a_1 a_2 a_3 : B$	MP
exit	7	$a_1 a_2 : C \to B$	

Figure 10.6:

We proved $C \to B$ from the assumptions using the box device.

Suppose we have further assumptions. Can we prove $(C \to B)$ by using modus ponens only, without a box? That depends on the assumptions. Let us take the following:

$$\begin{array}{ll} b_1 : & C \to C \\ b_2 : & (C \to C) \to ((C \to A) \to (C \to A)) \\ b_3 : & (C \to A) \to ((A \to B) \to (C \to B)). \end{array}$$

We can now prove $(C \to B)$ without the box. Using MP only:

$$\begin{array}{ll} b_1 b_2 : & (C \to A) \to (C \to A) \\ b_1 b_2 a_2 : & C \to A \\ b_1 b_2 a_2 b_3 : & (A \to B) \to (C \to B) \\ b_1 b_2 a_2 b_3 a_1 : & C \to B \end{array}$$

Notice that b_1 is a valid (successful) box deduction and both b_2 and b_3 are instances of the successful schema

$$(A \to B) \to ((B \to C) \to (A \to C))$$

discussed in example 9.2.4.

The following theorem is essentially in Anderson and Belnap's book [Anderson and Belnap, 1975]. We give the details here because we want to use the method for several other new logics.

Theorem 10.3.4 *Let Δ be a set of labelled assumptions and Q a goal. Assume that $\Delta \Vdash_{\mathbf{W}} \alpha : Q$ (Q can be proved from Δ with label α using box deduction of definition 10.3.2), then $\Delta \vdash_{\mathbf{W}} Q$ in the sense of definition 10.3.1. More specifically, we show that there are instances*

$$\begin{array}{c} \varnothing : A_1 \\ \vdots \\ \varnothing : A_k \end{array}$$

of the schemas

1. $C \to C$
2. $(A \to B) \to ((B \to C) \to (A \to C))$
3. $(B \to C) \to ((A \to B) \to (A \to C))$

such that $\Delta \cup \{\varnothing : A_i\} \Vdash_{MP1} \alpha : Q$

where $\Vdash_{MP1}$ denotes a proof using linear modus ponens only (no use of boxes), and this will mean of course that $\Delta \vdash_{\mathbf{W}} Q$.

Proof. By induction on the complexity of the boxes involved in the deduction of $\Delta \vdash_{\mathbf{W}} Q$.

Case n=0

Here only modus ponens is used, so there is nothing to prove.

Case $n > 0$ **Discussion of strategy**

The outer box may include nested inner boxes. Consider an innermost box of complexity 0. Let us take a typical line which justifies (proves) $A \to B$ using a box of complexity 0. We will eliminate this box and thus reduce the complexity of the outermost box. We will have to provide a new justification for $A \to B$ with the same label. We will assume a metabox of complexity $n > 0$. We assume that the inner boxes are all differently named and that $\leq$ is the subordination relation (see 3.4.2). In fact, since we start with a box named a_1 to show an implication of the form $A \to B$, we can identify the name of the box with the label of $a_1 : A$. We can recognise from the label $\alpha : C$, exactly in which outer box C was obtained; this box is in fact $max\alpha$. We know that C is a reiteration from the box $max\alpha$ if $max\alpha$ is less than the name of the current box.

The existing proof lines for the box for $A \to B$ look as follows (see fig 10.7):

line l	1	$\gamma : (A \to B)$, from Box a_1	
Box a_1	2		Show B
line $(l, 1)$	3	$a_1 : A$	Assumption
line (l, k)	4	$\beta_k : X$	
line $(l, k+1)$	5	$\beta_{k_1} : X \to Y$	
line $(l, k+2)$	6	$\beta_{k_2} = \beta_k \cup \beta_{k_1} : Y$	
line (l, k')	7	$\alpha : B$	
exit	8	$\gamma : A \to B$ with $\gamma = \alpha - \{a_1\}$	

Figure 10.7:

line l in the external proof is justified by Box a_1.

Inside the box the proof proceeds by MP1 only. A typical instance (use) of the rule is:

$$\begin{array}{lll} (l,k): & \beta_k : & X \\ (l,k_1): & \beta_{k_1} & X \to Y \\ \hline (l,k_2): & \beta_{k_2} = \beta_k \cup \beta_{k_1} : & Y \end{array}$$

The idea is to eliminate the box. Let us consider the following sequence of lines, which we refer to as *external sequence.*

$E(l, 1) = A \to A$, label $\varnothing$; corresponding to $a_1 : A$ in the box.

...

$E(l, k) = A^m \to X$, label $\beta_k - \{a_1^m\}$; corresponding to $\beta_k : X$ in the box and where m is the number of times $a_1 \in \beta_k$.

...

$E(l, k') = A \to B$, label $\alpha - \{a_1\}$, corresponding to the last line in the box.

Note that $m = 1$ or 0 depending whether X used $a_1 : A$ or not. If $m = 0, A^0 \to X$ is X. If $m = 1$ then we have $A \to X$.

The above sequence is not a proof. For each line $E(l, r)$ with label $\beta_r - \{a_1\} : C_r$ we add new assumptions and proof links to the box with name $max(\beta_r - \{a_1\})$ so that $\beta_r - \{a_1\} : C_r$ is derivable in the box $max(\beta_r - \{a_1\})$.

In particular the last line, namely

$$E(l, k') = \alpha - \{a_1\} : A \to B$$

will be derivable in box $max(\alpha - \{a_1\})$. Thus the justification for line l of the external box namely:

$$line\ l : \alpha - \{a_1\} : A \to B$$

can be taken as "reiteration from box $max(\alpha - \{a_1\})$" and this justification will replace the earlier justification "from Box a_1" and Box a_1 can be deleted.

Let us now proceed to modify all the boxes step by step, driven by the sequence $E(l, r)$. To do that we examine the possibilities arising in this sequence: There are several possibilities for a MP1 sequence in Box a_1. These have the form

$$\begin{array}{ll} \beta_k : & X \\ \beta_{k_1} : & X \to Y \\ \hline \beta_{k_2} : & Y \end{array}$$

These are the following $(m, n \geq 1)$ (see figure 10.8):

$$\begin{array}{lcl} \textit{case (0, 0)}: & X & a_1 \notin \beta_k \\ & X \to Y & a_1 \notin \beta_{k_1} \\ \cline{2-2} & Y & \end{array}$$

$$\begin{array}{lcl} \textit{case (1,0)}: & A^m \to X & a_1 \in \beta_k, m \text{ times} \\ & X \to Y & a_1 \notin \beta_{k_1} \\ \cline{2-2} & A^m \to Y & \end{array}$$

$$\begin{array}{lcl} \textit{case(0,1)}: & X & a_1 \notin \beta_k \\ & A^m \to (X \to Y) & a_1 \in \beta_{k_1}, m \text{ times} \\ \cline{2-2} & A^m \to Y & \end{array}$$

$$\begin{array}{lcl} \textit{case(1,1)}: & A^m \to X & a_1 \in \beta_k, m \text{ times} \\ & A^n \to (X \to Y) & a_1 \in \beta_{k_1}, n \text{ times} \\ \cline{2-2} & A^{m+n} \to Y & \end{array}$$

Figure 10.8: Case Diagram

For our particular logic, the following cases arise:

case (0,0): no problem. Both X and $X \to Y$ are reiterations. Note that we do not allow $\beta_k = \varnothing$. This case does not arise anyway because the reiteration X cannot be used as a minor premiss. Neither can X be proved within the box because $\beta_k \neq \varnothing$. To understand this possibility, consider $((A \to A) \to A) \to A$. We can justify this formula from Box a below (see fig 10.9):

Here line 2 is $\varnothing : X$ and line 1 is $a : X \to Y$, the modus ponens is not allowed.

case (1,0): can arise with $m = 1.X \to Y$ is a reiteration used as a ticket.

case (0,1): This case cannot happen since $\beta_k \neq \varnothing$. X is a reiteration and cannot be used as a minor premise.

case(1,1): This case cannot arise because both X and $X \to Y$ use A and so modus ponens is not allowed.

We therefore have to add assumptions and labels to justify the deduction

$$\begin{array}{c} E(l,k) = A \to X \\ E(l,k_1) = X \to Y \\ \hline E(l,k_2) = A \to Y \end{array}$$

and we have to justify the first line, namely $E(l, 1) = A \to A$.

Boxa	1		Show A
	2	$a : (A \to A) \to A$	Assumption
	3	$A \to A$	From Box b :
Boxb	4		Show A
	5	$b : A$	Assumption
	6	A	
exit	7	$\varnothing : A \to A$	
	8	$a : A$	$MP1$

Figure 10.9:

We want to take the instances of $A \to A$ and of $(A \to X) \to ((X \to Y) \to (A \to Y))$ and of $(X \to Y) \to ((A \to X) \to (A \to Y))$ and use the them to prove the theorem for the case of box deduction of complexity $n > 0$.

We have two axioms we can use (1) or (2):

1. $(A \to X) \to ((X \to Y) \to (A \to Y))$
2. $(X \to Y) \to ((A \to X) \to (A \to Y))$

Which one do we use? That depends on $\beta_k - \{a_1\}$ and β_{k_1}. β_k is the label of X, it contains a_1. This means that X was proved using $a_1 : A$ and some reiterations, say $Z_i \to U_i$. $\beta_{k_1} : X \to Y$ is also a reiteration. Which of the two sets of reiterations is from a more outer box?

If $X \to Y$ is outermost to some of the $Z_i \to U_i$, ie if $max\beta_{k_1} \leq max(\beta_k - \{a_1\})$, then we use (2). Otherwise we use (1). The rule is to use the instance of the axiom beginning with the outermost reiteration. It has to be done this way because of the following (see fig 10.10):

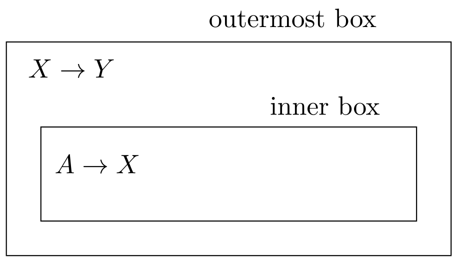


Figure 10.10:

$X \to Y$ is the outermost reiteration. If we choose (1), then $(X \to Y) \to (A \to Y)$ will appear in the inner box. $X \to Y$ will be reiterated but will not be allowed to be used as a minor premiss and thus we will not be able to get $A \to Y$ in the inner box. If, however we choose (2), then $(A \to X) \to (A \to Y)$ will be obtained in the outer box and reiterated into the inner box.

Case $n > 0$, proof replacement for the box

Consider $E(l, r) : \beta_r - \{a_1\} : C_r$ in the external sequence.

1. *Subcase n1*
   for $r = 1, C = A \to A$
   Add $\varnothing : A \to A$ as an initial assumption to the outermost box.

2. *Subcase n2*
   Assume for each $k < r$ that new assumptions were added to the boxes so that C_k is derived at box *max* $(\beta_k - \{a_1\})$ with label $\beta_k - \{a_1\}$. Consider case r. If $a_1 \notin \beta_r$, then C_r is a reiteration $X \to Y$ and the condition is satisfied.

   If C_r is $A \to Y$, and $a_1 \in \beta_r$, then we have for some earlier lines

   $E(l,k) : \beta_k - \{a_1\} : A \to X$
   $E(l,k_1) : \beta_{k_1} : X \to Y$
   $E(l,r) : \beta_r - \{a_1\} : A \to Y$
   with $\beta_r = \beta_k \cup \beta_{k_1}$

   We first consider $E(l,k)$.
   $A \to X$ is derivable in the outer box $\mathbf{a} = max\ (\beta_k - \{a_1\})$. We distinguish two cases here. The case of $k = 1$ and the other cases.

   If $k = 1$, then $X = A$ and $\beta_k - \{a_1\} = \varnothing$. In this case $A \to A$ is obtained from the outermost box because it was added there at the very first step. Box **a** will be $\varnothing$, the outermost box. Case 2 below will be applicable. We add the axiom $\varnothing : (A \to A) \to [(A \to X) \to (A \to X)]$ and by modus ponens get $(A \to X) \to (A \to X)$. Note that in fact we could have added $(A \to X) \to (A \to X)$ directly as this is an axiom too.

   We now consider $E(l,k_1)$:
   $X \to Y$ is reiterated from the outer box $\mathbf{b} = max\ \beta_{k_1}$

   Note that **a** and **b** are new boxes already modified by earlier stages of the process. We distinguish two cases.

1. subcase n2.1
   $\mathbf{b} \leq \mathbf{a}$
   In this case add a new instance of the axiom

   $$\varnothing : (X \to Y) \to ((A \to X) \to (A \to Y))$$

   to the outermost box. In box **a** we have a line
   Line n: $\beta_{k_1} : X \to Y$
   extend box **a** by inserting two more lines
   line (n,1): $\varnothing : (X \to Y) \to ((A \to X) \to (A \to Y))$
   reiteration from outer box
   Line (n, 2): $\beta_{k_1} : (A \to X) \to (A \to Y)$
   Box **a** is more inner than box **b**. In box **a** we have the line
   line n': $\beta_k - \{a_1\} : A \to X$
   we add the lines:
   line $(n', 1)$: $\beta_{k_1} : (A \to X) \to (A \to Y)$
   reiteration from box **b** (or repetition if **a**= **b**)
   line $(n', 2)$: $\beta_{k_1} \cup \beta_k - \{a_1\} : A \to Y$
   by modus ponens.

   Now with the new boxes $\mathbf{a}', \mathbf{b}'$ and the new outer box our inductive assertion holds.

2. subcase n2.2
   $\mathbf{a} < \mathbf{b}$
   In this case $\beta_k : A \to X$ is in the more outer box **b**. In this case we add $\varnothing : (A \to X) \to ((X \to Y) \to (A \to Y))$ to the outer box. We add the following two lines to box **a**.

   line $(n', 1)$: $\varnothing : (A \to X) \to ((X \to Y) \to (A \to Y))$
   reiteration from outer box **b**.
   line $(n', 2)$: $\beta_k - \{a_1\} : (X \to Y) \to (A \to Y)$
   modus ponens.

   In box **b** we add the following two lines after line $n :\ \beta_{k_1} : X \to Y$

line $(n, 1)$: $\beta_k - \{a_1\} : (X \to Y) \to (A \to Y)$
reiteration from box **a**.
line $(n, 2)$: $\beta_{k_1} \cup \beta_k - \{a_1\} : A \to Y$
by modus ponens.

Again our inductive assertion is satisfied by the new boxes.

For the case of $E(l, k')$, the last line of the sequence, we get a justification of $\alpha - \{a_1\} : A \to B$ from the new box system. We thus do not need box a_1 which originally justified $\alpha - \{a_1\} : A \to B$ because the line is now justified by reiteration. The outer box, of course, now has many instances of the axioms.

This completes the induction step for case $n > 0$.

The induction step completes the proof.

Note that the new additional assumptions were needed to be used only once! ■

Corollary 10.3.5 *The logic obtained from the notion $\varnothing \Vdash_{\mathbf{W}} A$ can be axiomatised using modus ponens and the following three schemas:*

1. $C \to C$
2. $(A \to B) \to ((B \to C) \to (A \to C))$.
3. $(B \to C) \to ((A \to B) \to (A \to C))$

Proof. Follows from 10.3.4.

1. Assume $\varnothing \Vdash_{\mathbf{W}} A$. Then by 10.3.4 there exist instances $A_1, \ldots, A_k$ of the axioms of **W** such that $\varnothing : A_1, \ldots, \varnothing : A_k \Vdash_{MP1} A$. Since no boxes are used in this proof, it is an acceptable proof of $\vdash_{\mathbf{W}} A$.

2. Assume $\vdash_{\mathbf{W}} A$, we want to show that $\varnothing \Vdash_{\mathbf{W}} A$. We prove this by induction on the proof steps in $\vdash_{\mathbf{W}}$. First show that $\varnothing \Vdash_{\mathbf{W}} A$, for any instance of an axiom of **W**. This has been done in the various examples. Then show that $\varnothing \Vdash_{\mathbf{W}} A$ and $\varnothing \Vdash_{\mathbf{W}} A \to B$ imply $\varnothing \Vdash_{\mathbf{W}} B$. This we show now. Clearly neither A nor B can be atomic (since all theorems of **W** are classical tautologies). We therefore have $A = A_1 \to A_2, B = B_1 \to B_2$ for suitable formulas.

   Since $\varnothing \Vdash_{\mathbf{W}} A_1 \to A_2$, there exists a box $\mathbf{M}_1$ of the form Figure 10.11:

Box M_1	1		Show A_2
	2	$t_1 : A_1$	Assumption
	3	$\vdots$	
	4	$t_1 : A_2$	
exit	5	$\varnothing : A_1 \to A_2$	

Figure 10.11:

   Since $\varnothing \Vdash_{\mathbf{W}} (A_1 \to A_2) \to (B_1 \to B_2)$ there exists a box M_2 of the form (see fig 10.12)

   The following is a box for $\varnothing \Vdash_{\mathbf{W}} B_1 \to B_2$, fig 10.13

   where box M_2^* is like box M_2 except that we substitute $t = \varnothing$

■

Lemma 10.3.6 *In the logic* $\mathbf{W}, \vdash_{\mathbf{W}} (1)$ *and* $\vdash_{\mathbf{W}} (2)$ *imply* $\vdash_{\mathbf{W}} (3)$*, where:*

Box M_2	1		Show $t : B_1 \to B_2$
	2	$t : A_1 \to A_2$	Assumption
	3	$t : B_1 \to B_2$	from Box
	4		Show B_2
	5	$s : B_1$	Assumption
	6	$ts : B_2$	
exit	7	$t : B_1 \to B_2$	
exit	8	$\varnothing : (A_1 \to A_2) \to (B_1 \to B_2)$	

Figure 10.12:

1. $A_1 \to \ldots \to (A_n \to (A \to B) \ldots)$
2. $D_1 \to \ldots \to (D_m \to (D \to A)) \ldots)$
3. $C_1 \to \ldots \to (C_{m+n} \to (D \to B) \ldots)$

where $C_1, \ldots, C_{m+n}$ *is any order preserving merge of the sequences* $A_1, \ldots, A_n$ *and* $D_1, \ldots, D_m$.

Proof. We need some notation. We shall use the previous theorem to show that, under the assumptions (1) and (2) we have that

$$\varnothing \Vdash (3)$$

Let a_i be an atomic name for A_i and d_j be an atomic name for D_j, $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Let t_k be an atomic name for $C_k, k = 1, \ldots, m+n$. Then by definition $\{t_1, \ldots, t_{m+n}\} = \{a_1, \ldots, a_n, d_1, \ldots, d_m\}$, and if $C_k = D_j$ then $t_k = d_j$ and if $C_k = A_i$ then $t_k = a_i$.

The $\{t_k\}$ are therefore just another enumeration of the $\{a_i, b_j\}$.

For each k, let $r_k : E_k$ be the labelled formula:

$$r_k : E_k = a_1 \ldots a_i : A_{i+1} \to \ldots \to (A_n \to (A \to B)) \text{ if } C_k = A_i$$

or

$$d_1 \ldots d_j : D_{j+1} \to \ldots \to (D_m \to (D \to A) \ldots) \text{ if } C_k = D_j.$$

We now produce a box (see fig 10.14) showing

$$\Vdash_{\mathbf{W}} C_1 \to \ldots \to (C_{m+n} \to (D \to B)) \ldots).$$

■

The example below (figure 10.15) presents a particular case for illustration:

Example 10.3.7

$$\varnothing : A_1 \to (A_2 \to (A \to B) \ldots)$$

$$\varnothing : D_1 \to (D_2 \to (D_3 \to (D \to A) \ldots)$$

Show

$$\varnothing : A_1 \to (D_1 \to (D_2 \to (A_2 \to (D_3 \to (D \to B) \ldots)$$

Example 10.3.8 *The statement of lemma 10.3.6 was not symmetrical. If we are given*

$$\vdash_{\mathbf{W}} X_1 \to \ldots \to (X_n \to (A \to B)\ldots)$$

$$\vdash_{\mathbf{W}} X_1 \to \ldots (Y_m \to A)\ldots)$$

Then we get

$$\vdash_{\mathbf{W}} Z_1 \to \ldots \to (Z_{m+n} \to B)\ldots)$$

only when $Z_1, \ldots, Z_m$ is a right *order preserving merge of $X_1, \ldots, X_m$ and $Y_1, \ldots, Y_m$. By* right merge *we mean the additional requirement $Z_{m+n} = Y_m$, ie the last element of the emerged resulting sequence is the last element of the second sequence being merged.*

The following shows why we need this restriction for **W**.

Assume

$$\vdash_{\mathbf{W}} X_1 \to (X_2 \to (A \to B)\ldots)$$

$$\vdash_{\mathbf{W}} Y \to A$$

We cannot show that

$$\vdash_{\mathbf{W}} Y \to (X_1 \to (X_2 \to B))$$

because the obvious box (figure 10.16)

Of course the above only shows that the obvious attempt to prove the assertion does not work. We shall see that the assertion is false when we give merge semantics for **W**. *Note however that using the assertion we get that*

$$\vdash (A \to (B \to C)) \to (A \to (B \to C))$$

$$\vdash B \to B$$

imply

$$\vdash (A \to (B \to C)) \to (B \to (A \to C))$$

ie we get the commutativity axiom of linear logic.

Theorem 10.3.9 *The logic obtained from the notion $\varnothing \Vdash_{\mathbf{T}_{\mathbf{E}}} A$ of 10.2.6 can be axiomatised by the schemas (1)-(4) of ticket entailment of 10.1.2, namely:.*

1. $C \to C$
2. $(A \to B) \to ((C \to A) \to (C \to B))$
3. $(A \to B) \to ((B \to C) \to (A \to C))$
4. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ *and*
   $(A \to B) \to ((A \to (B \to C)) \to (A \to C))$

Proof. We consider the proof given for the case of **W**, in theorem 10.3.4. The case analysis, for this logic allows in the innermost box also for the case of

$$\begin{gathered}\beta_{k1} - \{a_1\} : A \to X \\ \beta_{k2} - \{a_1\} : A \to (X \to Y) \\ (\beta_{k1} - \{a_1\}) \cup (\beta_{k2} - \{a_1\}) : A \to Y\end{gathered}$$

We have to argue for this case separately. By the induction hypothesis $A \to X$ is proved in Box **a**. $\mathbf{a} = max(\beta_{k_1} - \{a_1\})$ and $A \to (X \to Y)$ is proved in box **b**, $\mathbf{b}= max(\beta_{k_2} - \{a_1\})$. We have two axioms to add to the outermost box. These are:

$$(A \to X) \to ((A \to (X \to Y) \to (A \to Y))$$

and

$$(A \to (X \to Y)) \to ((A \to X) \to (A \to Y))$$

If $\mathbf{a} \leq \mathbf{b}$ we use the first axiom. Otherwise we use the second axiom. ■

Theorem 10.3.10 *Linear logic* **L**, *with the metabox discipline as defined in 9.2.6 can be axiomatised as a Hilbert system by adding the following axiom to the axioms of* **W**:

$$(A \to (B \to C)) \to (B \to (A \to C)).$$

Proof. We consider the proof given for the case of **W**, in theorem 10.3.4. The case analysis for linear logic also allows for case (0, 1) of figure 10.8, to occur in the innermost box. Namely we have

$$\begin{array}{c} X \\ A \to (A \to Y) \\ \hline \\ A \to Y \end{array}$$

In this case we use the instance of the axiom

$$\varnothing : (A \to (X \to Y)) \to (X \to (A \to Y))$$

and use modus ponens twice. ■

10.4 The logic of H-relevance

This section is purely technical. It shows how a new logic can be created by allowing formulas to label themselves and by using a known logic (intutionistic) on the labels together with the labelling discipline of relevance logic on formulas. This logic is then axiomatised. The main message of this section is that *LDS* can *generate* new logics. This example, thought, does have some intutive meaning. If A is proved from the part of the database denoted by Δ_0, then Δ_0 is the label of A. Intutionistic logic is used on the labels to decide which other parts of the database are "relevant" to Δ_0, namely all Δ' such that $\Delta_0 \vdash \Delta'$.

Definition 10.4.1 *The defining conditions on the metaboxes of this logic are as follows:*

1. *Formulas are labelled by themselves. It is convenient to give atomic labels $a : A$ and use a labelling function $h(a) = A$.*
2. *To show $A \to B$, start a box. Assume $a : A$ and show $\alpha : B$. Use modus ponens (unrestricted). The box is successful if $a \in \alpha$. Exit the box with $\alpha - \{a\} : A \to B$.*
3. *Reiterations are allowed only either through condition 3 of 9.2.1 (ie the reiteration condition of* **W**) *or through the following additional condition:*
   $\gamma : C$ is allowed into the box provided $h(A) \nvdash_{\mathbf{H}} h(\gamma)$ where **H** *is a labelling logic on the formulas.*
   We further assume that within the box we do not use modus ponens on reiterations. This can be done externally ie we do not use

   $$\begin{array}{ll} \beta_k : & X \\ \beta_{k1} : & X \to Y \\ \hline \\ \beta_{k2} & Y \end{array}$$

   where both X and $X \to Y$ come from an external box. The modus ponens can be done externally and then the Y reiterated. Formally the restriction within a box with assumption $a : A$ to show B is that either $a \in \beta_k$ or $a \in \beta_{k1}$.

Example 10.4.2 *(a) Let* **H** *be intuitionistic logic. To show $A \to (B \to A)$ (see fig 10.17)*

This box is not successful because the label of A does not contain b.

(b) Show $A \to (X \to B) \to (X \to (A \to B))$ for $A \nvdash X$. (See fig 10.18)

Example 10.4.3 *Let* **H** *be intuitionistic logic.*
Show $(B \to A) \to ((A \to B) \to (A \to B))$ *(see fig 10.19)*

Theorem 10.4.4 *The set of wffs such that* $\varnothing \vdash_{\mathbf{H}\text{-}Relevance} A$ *can be axiomatised by:*

1. $A \to A$

2. $(A \to B) \to ((B \to C) \to (A \to C))$

3. $(A \to B) \to ((C \to A) \to (C \to B))$

4a. $(A \to B) \to ((A \to (B \to C)) \to (A \to C))$

4b. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$

7. $(A \to (B \to C)) \to (B \to (A \to C))$*, provided* $A \not\vdash_{\mathbf{H}} B$*.*

Proof. We proceed like in the proof of 10.3.4 and we eliminate all uses of boxes. Let there be given a box for $A \to B$. We prefix with A rather with A^m, ie we ignore multiple uses of A. If we check the case studies, two more cases are possible.

$$\begin{array}{l} A \to (X \to Y) \\ A \to X \\ \hline A \to Y \end{array}$$

which can be taken care of by axiom 4 and

$$\begin{array}{l} A \to (X \to Y) \\ X \\ \hline A \to Y \end{array}$$

In this case X is a reiteration and therefore $A \not\vdash_{\mathbf{H}} X$. Hence we can use axiom 7 to get $X \to (A \to Y)$ and then by modus ponens we get $A \to Y$.

We have to address the possibility that X may not be a reiteration itself but be provable from reiterations. Thus we have

$$\begin{array}{l} X_1 \\ X_1 \to X \\ \hline X \end{array}$$

This possibility does not arise because of our restrictions on reiterations. ■

Theorem 10.4.5 *The following table (fig 10.20) gives the correspondences between axioms and metabox conditions in the sense of 10.3.4.*

Proof. Along the lines of previous proofs. ■

10.5 Introducing deductive relevance

This section introduces a new logic, the logic of deductive relevance. We start by motivating it. Consider a database with some information in it. Assume that from time to time the database is updated. The reason for the updating is that there seems to be a change of mind of whether certain formula should be in the database. For example A could be "John paid the invoice" and at different times there is a change of view of whether A is true or not. We can play it safe and insert in the database all the updates, with the appropriate labels indicating when they were received. Thus the formulas A and $\sim A$ may both appear in the database with different time labels

$$\begin{array}{ll} t_i : A & i = 1 \dots k \\ s_i : \sim A & i = 1 \dots m \end{array}$$

with t_i, s_i different time labels.

In general, a database of this sort will have the form $\Delta = \{t_i : A_i\}$ where t_i are labels and A_i are formulas. This database looks just like any other labelled database we have considered so far in this paper. When we prove a formula B from the database, we want to keep track of exactly what labelled formulas were used. If we do that, we can know which version of the data was used in the derivation. The labelled deduction process is the same as in the case of relevance logic. In relevance logic we are interested in resource considerations, we want to make sure that all assumptions are used. In our case we are interested in making sure the most recent updates were used. Consider the example below:

Example 10.5.1 Data:
$t_1 : A$
$t_2 : \sim A$
$s_1 : A \to B$
$s_2 : B \to \sim A$
$s_3 : \sim A \to (C \to B)$

Query:
$C \to B$

We can understand the above data as t-labelled updates of the data and s-labelled updates of the rule.

We assume $t_1 < t_2$ and $s_1 < s_2 < s_3$. We would like a derivation of $C \to B$ using the most recent data. We use the box below (see fig 10.21):

This proof is acceptable because it used the most recent data.

We can pretend in 10.5.1 that we are operating in relevant logic, if we agree that we are dealing with relevance classes of data, namely $\{t_1, t_2\}$ and $\{s_1, s_2, s_3\}$. Whenever a label x is used in a class, this "use" of the label is considered as using all lower ("past") labels. Thus in the above proof, $C \to B$ can be considered as a relevant derivation, because all labels were used. t_2 and s_3 were directly used, while t_1, s_1 and s_2 were "used" because a more recent update of that class was consulted.

The above example suggests that we can divide the labels into classes according to some agreed "Labelling Logic", $\mathcal{LL}$. $\mathcal{LL}$ helps us organise the classes of labels. When we prove any $\alpha : B$ from the database with label α, the logic $\mathcal{LL}$ will tell us whether α is considered as a relevant use of the database labels. In fact, $\mathcal{LL}$ may come as part of a package deal together with a labelling scheme. We label the data in some agreed manner, compatible with the application area (Δ a knowledge representation database for the application area) and $\mathcal{LL}$ is a suitable logic for the labels of the application area.

The next example shows how the labels can be used as degrees of certainty.

Example 10.5.2 *Consider the assumptions:*

1. $a \wedge b \to d$

2. $a \to d'$

3. $a \wedge b \wedge c \to d'$

4. a

5. b

6. c

Let $\mathbf{P}(n)$ *give the degree of certainty we attach to clause* n*.* $0 \leq \mathbf{P}(n) \leq 1$.

To deduce d' *we can follows two paths: we can use clauses 3,4,5,6 and get* $d'_{\{3,4,5,6\}}$ *or we can use 4 and 2 and get* $d'_{\{2,4\}}$*. In each case we can get the degree of certainty using* $\mathbf{P}$*.*

The label indicates which clauses were used. In the second case we can use the relevance deduction theorem and get

$$\{clauses 1,3,4,5,6\} \vdash (a \to d') \to d'_{\{4\}}$$

but the proof of $d'_{\{2,4\}}$ does not justify:

$$\{clauses 1,2,3,4,5\} \vdash c \to d'.$$

The latter does hold, however, because of the existence of the other proof of $d'_{\{3,4,5,6\}}$.

Note therefore that we do *not* require that the assumption (for the use of the deduction theorem) be used in *all* proofs but only in *at least one*.

In each case the degree of certainty is obtained from the labels. Obviously what we need is a general system of deductive relevance which depends generally on a labelling scheme and on a labelling logic $\mathcal{LL}$, which can be specialised to the various familiar systems (such as relevance logic, fuzzy logic etc) by a suitable choice of the labels and the logic $\mathcal{LL}$. The system in its full generality need not necessarily have an intrinsic intuitive meaning, beyond that of having some logic on the labels. Each choice of labels and logic $\mathcal{LL}$ will, however, model some known system and intuitions.

As a first step, let us examine what logic $\mathcal{LL}$ would correspond to relevance logic itself. Let us recall the essential principle of the relevance labelling.

The relevance labelling follows the following principles:

1. All assumptions are named (labelled) using different atomic names. The *same* wff A may be put in with more than one name: e.g.

$$n_1 : A$$

$$n_2 : A$$

   This can arise, for example, when trying to prove $\vdash A \to (A \to A)$, which fails.

2. If A is labelled α and $A \to B$ is labelled β then using modus ponens we emerge with B labelled $\alpha \cup \beta$ (because B uses in its proof both A and $A \to B$).

3. If we want to show $A \to B$, we assume A with name α (usually $\alpha = \{a\}$, with "a" a new name). We prove forward B, which ends up, at the end of the proof with label β. Then if $\alpha \subseteq \beta$, we can conclude $A \to B$ with label $\beta - \alpha$:

Thus in diagram

$$\{a\} : A$$
$$\ldots$$
$$\ldots$$
$$\beta = \{a, b, c, \ldots\} : B$$

allows us to deduce $A \to B$ with label $\{b, c \ldots\} = \beta - \{a\}$.

We now make the following observation.

We can regard $a \in \alpha$ as $\alpha \vdash a$ in the boolean logic of the atomic labels $a, b, c, \ldots$. Since all labels involved are atoms, or sets of atoms, the set theoretic relation $\alpha \subseteq \beta$ is identical with the logical relation $\beta \vdash \alpha$, in any logic, eg classical logic.

We can generalise the relevance logic procedure by allowing a *logic* $\mathcal{LL}$ (Labelled Logic) on the labels of the formulas. Imagine each formula assumption gets an label a. When we use modus ponens we get:

$$\begin{array}{ll} \alpha : A & \\ \beta : A \to B & \\ \hline \alpha \wedge \beta : B & \text{in the logic } \mathcal{LL}. \end{array}$$

To prove $A \to B$ with label γ we assume:

$$\begin{array}{l} \alpha : A \\ \dots \\ \dots \\ \beta : B \\ \hline \gamma : A \to B \end{array}$$

We argue from A to B and want to exit with $A \to B$ with label γ. For A to be used in the proof of B we need

$$\beta \vdash_{\mathcal{LL}} \alpha.$$

γ would then be

$$\gamma = \wedge \, \{\text{names of assumptions } x \mid \beta \vdash_{\mathcal{LL}} x \text{ and } \alpha \not\vdash_{\mathcal{LL}} x\}$$

Thus to be able to get γ effectively we need $\mathcal{LL}$ to be computable, and of course that the number of assumptions finite.

In the case of ordinary relevance logic, the general definition reduces to the old definition. To see this note that if:

$\alpha = \{a_1, ..., a_n\}$	old definition
$\alpha^* = a_1 \wedge \ldots \wedge a_n$	new definition.

Thus

$$(\alpha \cup \beta)^* = \alpha^* \wedge \beta^*$$

Since a_i, b_j are atoms then if $\alpha \subseteq \beta$

$$\beta - \alpha = \{a \mid \beta \vdash a \text{ and } \alpha \not\vdash a\}$$

in the logic $\mathcal{LL}$ of classical conjunctions and classical truth tables.

Let us call the logic obtained from labelling on $\mathcal{LL}$ by "**DR(LL)**". In words: "Deductive relevance *based on* the labelling logic $\mathcal{LL}$".

It may be convenient to name a formula A by the symbol "**a**". If A is syntacticaly different from B then "**a**" and "**b**" are considered different atomic names. If A is put in the database twice, then we obviously have to use different atomic names and not just put "**a**" in twice. Eg "$\mathbf{a}_1$" and "$\mathbf{a}_2$".

Important Remark

Note that there is no reason why the labelling logic $\mathcal{LL}$ should be monotonic. It can be any non-monotonic system which allows for conjunctions, ie allows us to form

$$\frac{\alpha : A, \beta : A \to B}{\alpha \wedge \beta : B}$$

and which is decidable. We need decidability in order to form γ as shown:

$\alpha : A$

$\dots \qquad \beta \mid\!\sim \alpha$

...
$\beta : B$

$\gamma : A \to B$, $\gamma = \wedge\{x \mid \beta \mid\!\sim x$ and $\alpha \not\mid\!\sim x\}$
$\mid\!\sim$ is the non monotonic $\mathcal{LL}$

Example 10.5.3 *We recall 2.2.1 of Chapter 2.*
Show that $\vdash (B \to A) \to ((A \to B) \to (A \to B))$ *in relevance logic.*

To show that, we show that the assumptions
a_1*:*$B \to A$
a_2*:*$A \to B$
a_3*:*A
prove B *with label* (a_1, a_2, a_3)

This was done in 2.2.1. To illustrate our labelling logic ideas, let us use relevance logic *itself as the labelling logic. Thus* R *is relevance logic and* $D = \mathbf{DR}(R)$ *is the resulting logic when relevance logic is used on the labels. We use the formulas themselves as their names: We thus want to show:*

$$B \to A, A \to B, A \vdash_D B \textit{ with label } (B \to A) \wedge (A \to B) \wedge A.$$

Since $(B \to A) \wedge (A \to B) \wedge A \vdash_R A$*, we use the dedution theorem in* $D = DR(R)$ *and get:*

$$B \to A, A \to B \quad \vdash_D A \to B \qquad \textit{with label } \gamma, \textit{where:}$$

$$\gamma = \{x \mid x \textit{ is a name and } (B \to A) \wedge (A \to B) \wedge A \vdash_R x \textit{ and } A \not\vdash_R x\}$$

We have 3 names. $A \not\vdash_R A \to B$ *and* $A \not\vdash_R B \to A$*, hence* $\gamma = (A \to B) \wedge (B \to A)$*. We can use the deduction theorem again and get:*

$$B \to A \vdash_D (A \to B) \to (A \to B) \textit{ with label } B \to A$$

and again

$$\vdash_D (B \to A) \to ((A \to B) \to (A \to B)).$$

Example 10.5.4 $\mathbf{DR}(R)$ *is not the same as* R*. Consider the formula:*

$$(A \to A) \to ((A \to A) \to (A \to A))$$

This is a theorem of R*, because* $A \to A, A \to A, A \vdash_R A$ *with label* $(A \to A, A \to A, A)$*. (We use modus ponens twice). We also have by the deduction theorem*

$$A \to A, A \to A \vdash_R A \to A \textit{ with label } (A \to A, A \to A)$$

$$A \to A \vdash_R (A \to A) \to (A \to A) \textit{ with label } (A \to A)$$

However in $D = \mathbf{DR}(R)$ *the exit label is computed differently and hence we get*

$$A \to A \vdash_D (A \to A) \to (A \to A) \textit{ with label } \varnothing$$

In fact, any theorem of R *of the form* $\vdash_R A \to (B \to C)$ *such that* $\vdash_R B \to A$ *will not be a theorem of* D*.*

Example 10.5.5 *Now let us use intuitionistic logic* Int *as the labelling logic in 10.5.3, and abbreviate* $D = \mathbf{DR}(Int)$ *We get:*

$$B \to A, A \to B, A \vdash_D B \textit{ with label } (B \to A) \wedge A \wedge (A \to B)$$

Use the deduction theorem

$B \to A$, $A \to B \quad \vdash_D A \to B$ with label γ.
$\gamma = \{x \mid (B \to A) \wedge A \wedge (A \to B) \vdash_{Int} x$ *but* $A \not\vdash_{Int} x\}$

Clearly $\gamma = \{A \to B\}$. In intuitionistic logic $A \vdash_{Int} B \to A$, which is not true in relevance logic. Thus we get
$B \to A \vdash_D (A \to B) \to (A \to B)$ *with label* γ
$\gamma = \wedge\{x \mid A \to B \vdash_{Int} x$ *and* $A \to B \not\vdash_{Int} x\} = \mathbf{truth}$.

We cannot continue and use the deduction theorem any further. Thus for the logic **DR**(*Int*)*, where the labelling logic is intuitionistic logic, we do* not *have:*

1. $\vdash_{\mathbf{DR}(Int)} (B \to A) \to ((A \to B) \to (A \to B))$.
   On the other hand we do *have*

2. $\vdash_{\mathbf{DR}(Int)} A \to ((A' \to A') \to A)$

(1) is a theorem of relevance logic while (2) is not. We can see that we are getting something new. We have not proved yet that what we get is a logic. In fact we must show that for any monotonic logic $\mathcal{LL}$, the logic $\mathbf{DR}(\mathcal{LL})$ is indeed a consequence relation. At the moment I have proof only for the case of **DR**(*Int*)*. The difficult part to prove being the cut:*

$$\Delta \vdash_{\mathbf{DR}(Int)} A \text{ and } \Delta, A \vdash_{\mathbf{DR}(Int)} B \Rightarrow \Delta \vdash_{\mathbf{DR}(Int)} B.$$

10.6 The cut rule for deductive relevance

To investigate the relationship between a logic **X** and its deductive relevance counterpart **DR(X)** in general and **DR**(**int**) in particular we need to proceed to do through a series of definitions and lemmas.

Definition 10.6.1 *1. Define for a Hilbert system* **H***, the notion $\vdash_{\mathbf{H}} A$ as follows:*
$\vdash_{\mathbf{H}} A$ iff there exists a sequence $B_1, B_2, \ldots B_n = A$ such that each member of the sequence is either an instance of an axiom or is obtained from the two previous formulas of the sequence by modus ponens.

2. Define $A_1, \ldots, A_n \vdash_{\mathbf{H}} B$ as

$$\vdash_{\mathbf{H}} A_1 \to (A_2 \to \ldots \to (A_n \to B) \ldots).$$

Note that this definition is independent of the order of A_i because of the axiom

$$\vdash (A \to (B \to C)) \to (B \to (A \to C)).$$

Also note that $\vdash_{\mathbf{H}}$ is non monotonic, ie $A \vdash_{\mathbf{H}} A$ may hold, but $A, B \vdash_{\mathbf{H}} A$ may not hold because its meaning is $\vdash_{\mathbf{H}} A \to (B \to A)$.

3. To define a monotonic consequence relation based on **H***, let $A_1, \ldots, A_n, \Vdash_{\mathbf{H}} B$ be defined to hold iff there exists a sequence of formulas $B_1, \ldots, B_n = B$ such that each member B_i is either an assumption A_j or a theroem, $\vdash_{\mathbf{H}} B_i$, or is obtained from two previous elements of the sequence by modus ponens.*
Note that in this case we will have a monotonic consequence relation $A, B \Vdash_{\mathbf{H}} A$ will hold. We do have however:

$$\varnothing \Vdash_{\mathbf{H}} B \text{ iff } \varnothing \vdash_{\mathbf{H}} B \text{ iff } \vdash_{\mathbf{H}} B.$$

Definition 10.6.2 *(a) A set of assumptions has the form $\Delta = \{a_i : A_i\}$, where A_i are formulas and a_i are all different atomic labels.*

(b) We define by induction the notion of a proof tree of $\Delta \vdash t : B$, where t is a label.

(1) $\tau = \{\Delta \vdash t : B\}$ is a one node tree if $t : B \in \Delta$. This node is the bottom node of the tree.

(2) If τ_1 is a proof tree with bottom node $\Delta \vdash t : A$ and τ_2 is a proof tree with bottom node $\Delta \vdash s : A \to B$ then

is a tree with bottom node $\Delta \vdash st : B$.

(3) If τ is a tree with bottom node $\Delta \cup \{a : A\} \vdash t : B$ and a appears in the string t then:

is a tree with bottom node $\Delta \vdash t - a : A \to B$.

(c) We say $A_1 \dots A_n \vdash B$ if for some distinct atomic labelling $a_i : A_i$ we have a proof tree for $\{a_i : A_i\} \vdash a_1 \dots a_n : B$

Theorem 10.6.3 ([Anderson and Belnap, 1975]) *$A_1, \dots, A_n \vdash B$ in the sense of (c) of 10.6.2 iff $A_1, \dots, A_n \vdash_{\mathbf{H}} B$, for $\mathbf{H}$ being relevance logic in the sense of 10.6.1.*

Definition 10.6.4 *1. Let $\vdash$ be a non monotonic consequence relation. $\vdash$ is said to be a* labelling logic *if it satisfies the following conditions:*

(a) $\dfrac{\Delta \vdash B \,; \Delta' \vdash B \to C}{\Delta, \Delta' \vdash C}$

(b) $\dfrac{\Delta, \Delta' \vdash D_2 \,; D_1 \vdash \wedge\Delta'}{\Delta, D_1 \vdash D_2}$

(c) $\dfrac{\Delta, D_1 \vdash D_2}{\Delta \vdash D_1 \to D_2}$

2. A Hilbert System I is said to be a labelling logic *if $\vdash_I$ is a labelling logic. A labelling logic I (or $\vdash$) is said to be* monotonic *if $\vdash_I$ (resp. $\vdash$) is monotonic.*
Note that for $\mathbf{H}$= Relevance logic whose axioms are mentioned in 2.2.1, $\vdash_{\mathbf{H}}$ is a labelling logic. $\Vdash_{\mathbf{H}}$ is a monotonic labelling logic.

3. A database *is a set of pairs of the form:*

$$\begin{array}{ccc} a_1 & : & A_1 \\ \dots & & \\ \dots & & \\ a_n & : & A_n \end{array}$$

where a_i are all different atomic labels.

4. Let I be a labelling logic, on the same language as the data. A function h on $\{a_i \mid a_i : A_i \in \Delta\}$ is a logical support function *if $h(a_i)$ is a set of wff and $h(a_i) \vdash_I A_i$.*

Definition 10.6.5 *Let Δ be a database and h a support function and B a wff. We define the provability $\vdash^h_{n\mathbf{DR}(I)}$ by induction of n. The basic notion we are defining is*

$$\Delta \vdash^h_n B, \alpha, \pi$$

where Δ is a database with labelled wffs. B is the proved wff. α is a set of labels used in the proof and π is a set of wff of I, the logical support of B; $n = 0, 1, 2, \dots$.
Case n = 0
$\Delta \vdash^h_0 B, \alpha, \pi$ iff there exists a sequence $(B_i, \alpha_i, \pi_i) i = 1, \dots, k$ with $(B_k, \alpha_k, \pi_k) = (B, \alpha, \pi)$ and each element in the sequence satisfies the following:
Condition 1:

$$a_i : B_i \in \Delta \text{ and } \pi_i = h(a_i) \text{ and } \alpha_i = \{a_i\}.$$

Condition 2:
There exist $B_j, B_m, j, m, < i$ such that $B_m = B_j \to B_i$ and $\alpha_i = \alpha_m \cup \alpha_j$ and $\pi_i = \pi_m \cup \pi_j$.

Case n+1
We now define $\Delta \vdash^h_{n+1} B, \alpha, \pi$.

The above holds if there is a sequence satisfying any of the above conditions (1) and (2) or the following additional Condition (3):
Condition 3:
$B_k = D_1 \to D_2$ *and*
$\Delta' = \Delta \cup \{d : D_1 \mid d \text{ a new label } \} \cup \{b_j : B_j \mid b_j \text{ new labels and } j < k\}$,
$h' = h \cup \{(d, D_1)\} \cup \{(b_j, \pi_j)\}$
and
$\Delta' \vdash^{h'}_m D_2, \gamma, \pi_0$, *for some* $m \leq n$ *and* $\pi_0 \vdash_I D_1$ *and* $\alpha_k = \gamma - \{d\}$ *and*
$\pi_k = \{x \in \pi_0 \mid D_1 \not\vdash_I x\}$.
Case n=∞:
Let $\Delta \vdash^h B, \alpha, \pi$ *if for some* $n, \Delta \vdash^h_n B, \alpha, \pi$.

Lemma 10.6.6 *Let Δ be a labelled database and let h be a support function. Let $U = U(\Delta, h)$ be $U = \cup_{(a:A)\in\Delta} h(a)$. For any α let $U_\alpha = \cup_{a\in\alpha} h(a)$. Then if $\Delta \vdash^h_n B, \alpha, \pi$ we have $\pi \subseteq U$ and $\pi \vdash_I B$. (In fact $\pi \subseteq \cup_{a\in\alpha} h(a) = U_\alpha$).*

Proof. The proof is by induction on the definition of $\vdash^h_n$.
Case n=0:
Assume $\Delta \vdash^h_0 B, \alpha, \pi$. Then there is a sequence as in the definition. If $a_k : B_k \in \Delta$ then $\alpha_k = \{a_k\}$ and $\pi_k = h(a_k)$ and $\pi_k \vdash_I B_k$ and $\pi_k \subseteq U_{\alpha_k}$. If $B_m = B_j \to B_k$ then $\alpha_k = \alpha_m \cup \alpha_j$ and $\pi_k = \pi_m \cup \pi_j$ and since by the induction hypothesis $U_{\alpha_m} \supseteq \pi_m \vdash_I B_m$ and $U_{\alpha_j} \supseteq \pi_j \vdash_I B_j$ we get $U_{\alpha_k} \supseteq \pi_k \vdash_I B_k$.
Case n+1:
Assume $\Delta \vdash^h_{n+1} B, \alpha, \pi$. We check the case of $B = B_k = D_1 \to D_2$. We have $\Delta' = \Delta \cup \{(d : D_1)\} \cup \{(b_j : B_j) \mid j < k\}, h' = h \cup \{(d, D_1)\} \cup \{(b_j, \pi_j) \mid j < k\}$ and $\Delta' \vdash^{h'}_m D_2, \gamma, \pi_0$ for $m \leq n$, and $U_\gamma \supseteq \pi_0 \vdash_I D_1$ and $\alpha_k = \gamma - \{d\}$ and $\pi_k = \{x \in \pi_0 \mid D_1 \not\vdash_I x\}$ and $U_{\alpha_j} \supseteq \pi_j \vdash_I B_j$, for $j < k$. We have to show that $U_{\alpha_k} \supseteq \pi$ and $\pi \vdash_I D_1 \to D_2$.

Consider all wffs in π_0. Classify them into two sets $\{x_i\}, \{y_j\}$, where $D_1 \vdash_I y_j$ and $D_1 \not\vdash x_i$. hence $\pi = \{x_i\}$. Clearly $\pi \subseteq U_{\alpha_k}$ since $D_1 \notin \pi$. We know that by the induction hypothesis we have:

$$\pi_0 = \wedge_i x_i \wedge \wedge_j y_j \vdash_I D_2.$$

Also since $D_1 \vdash_I \wedge y_j$ we get $\wedge x_i \wedge D_1 \vdash_I D_2$ and hence $\wedge x_i \vdash_I (D_1 \to D_2)$. ■

Lemma 10.6.7 *Assume I is a monotonic labelling logic.*
Let $\underline{h} \supseteq h$ mean that for any label $a, \underline{h}(a) \supseteq h(a)$. Assume $\Delta \vdash^h_n B, \alpha, \pi$ then $\Delta \vdash^h_n B, \alpha, \underline{\pi}$, where $\underline{\pi}$ satisfies $(\pi \cup U^*) \supseteq \underline{\pi} \supseteq \pi$ and $U = \cup_a \underline{h}(a)$ and $U^* = \cup_a(\underline{h}(a) - h(a))$.

Proof. By induction on the proof of $\Delta \vdash^h_n B, \alpha_\pi$.
Case n=0:
$\Delta \vdash^h_n B, \alpha, \pi_i$. Then there exists a sequence as in the definition. Replace h by $\underline{h}$ and use the same sequence. We get a sequence with $\underline{\pi}$ satisfying the lemma.
Case n+1:
To show the lemma for $\Delta \vdash^h_{n+1} B, \alpha, \pi_i$ we use the second half of the definition. We need to consider the case $B = B + k = D_1 \to D_2$, with

$$\Delta' = \Delta \cup \{(d : D_1)\} \cup \{(b_j : B_j) \mid j < k\},$$

$$h' = h \cup \{(d, D_1)\} \cup \{(b_j, \pi_j) \mid j < k\}.$$

and the following holds:

$$\Delta' \vdash^{h'}_m D_2, \gamma, \pi_0,$$

$$\pi_0 \vdash_I D_1, \alpha_k = \gamma - \{d\},$$

$$\pi = \{x \in \pi_0 \mid D_1 \not\vdash_I x\}.$$

By a previous lemma $\pi_o \vdash_I D_2$. By the induction hypothesis the same proof sequence will give:

$$\underline{h}' = \underline{h} \cup \{(d, D_1)\} \cup \{(b_j, \underline{\pi}_j) \mid j < k\}.$$

Since B_j has a shorter proof we get:

$$\pi_j \cup U^* \supseteq \underline{\pi}_j \supseteq \pi_j$$

Again by the induction hypothesis:

$$\Delta' \vdash^{h'}_m D_2, \gamma, \underline{\pi}_0$$

$$(\pi_0 \cup U^*) \supseteq \pi_0 \supseteq \pi_).$$

Since $\pi_0 \vdash_I D_1$ we get $\underline{\pi}_0 \vdash_I D_1$. Since by the previous lemma $\pi_0 \vdash_I D_2$, we get $\underline{\pi}_0 \vdash_I D_2$. Define $\underline{\pi} = \{x \in \underline{\pi}_0 \mid D_1 \not\vdash_I x\}$. Clearly then $\Delta \vdash^h_{n+1} B, \gamma, \underline{\pi}$. Clearly $(\pi \cup U^*) \supseteq \underline{\pi} \supseteq \pi$. ■

Lemma 10.6.8 *For a monotonic labelling logic, the following holds:*
If $\Delta \vdash^h A$ and $\Delta, A \vdash^h B$ then for some $h1 \supseteq h$ $\Delta \vdash^{h1} B$.

Proof. Assume $\Delta^h_n \vdash A, \alpha, \pi_1$ and $\Delta, A \vdash^h_m B, \beta, \pi_2$. Let $h1$ be defined by $h1(x) = h(x) \cup \pi_1$. By the previous lemma, since $U^* \subseteq \pi_1$ we get $\Delta \vdash^{h1}_n A, \alpha, \underline{\pi}_1$ with $\pi_1 \subseteq \underline{\pi}_1 \subseteq (\pi_1 \cup U^*) \subseteq \pi \cup \pi_1$. We also have $\Delta, A \vdash^{h1}_m B, \beta, \underline{\pi}_2$.

We can now string the two proofs together:

Δ
...
  proof as in $\Delta \vdash^{h1} A$
...
A
...
  proof as in $\Delta, A \vdash^{h1} B$.
...
B

The crucial reason that we can indeed string the proofs together is that the label $\underline{\pi}_1$ of A at the end of the proof of $\Delta \vdash^{h1} A$ is the same as the label $h1$ gives to A as an item of data in the proof of $\Delta, A \vdash^{h1} B$. The construction of $h1$ from h by adding $\underline{\pi}_1$ to all labels was designed to ensure this. ■

Lemma 10.6.9 *1. If $a : A \in \Delta$ then $\Delta \vdash^h A$.*

2. If $\Delta \vdash^h A$ then $\Delta, B \vdash^h A$.

Proof. By definition. ■

Theorem 10.6.10 *Let I be a monotonic logic and $\Delta \subseteq J$ be a set of wffs. Let $\Delta \vdash^J_{\mathbf{DR}(I)} A$ iff for some h (with $U(h) \subseteq J$)$\Delta \vdash^h_{\mathbf{DR}(I)} A$. Then $\vdash^J_{\mathbf{DR}(I)}$ is a consequence relation.*

Proof. From previous lemmas. ■

Example 10.6.11 Relevance Logic
Let $\Delta = \{a_i : A_i\}$ with a_i new and different atoms. Let $h_R(a_i) = a_i \wedge (\wedge_j A_j)$. then, provided $h(a_i) \vdash h(a_j)$ iff $i = j$, we get:

Lemma 10.6.12 *$\vdash^h_R$ is relevance logic.*

Proof. Show by induction that $\Delta \vdash^h_R A, \alpha, \pi$ iff $\pi = \wedge\alpha \wedge \Delta$. ■

Theorem 10.6.13 *Let I = Relevance logic and let $h(a : A) = A$. Then $\mathbf{DR}(R) = R$.*

Proof. Show that if $\Delta \vdash^h A, \alpha, \pi$ then $\pi = \{A \mid a \in \alpha$ and $a : A \in \Delta\}$. ■

10.7 Negation

This section studies properties of negation in concatenation and linear logics. We begin with the idea of adding an arbitrary unary connective $\odot$ to the language and studying its properties. We begin with $\mathbf{CL}(\otimes)$. The language of $\mathbf{CL}(\otimes)$ contains only atomic propositions and the implication and concatenation symbols $\to$ and $\otimes$. We want to enrich the language with a unary propositional function $\odot$, associating with each atomic q another atom $q^{\odot}$. We shall eventually stipulate that $q^{\odot\odot} = q$, but at this stage this is not needed. For a general theroy of negation see Gabbay-Wansing [12].

Definition 10.7.1 *1. Let $\mathbf{CL}(\otimes, \odot)$ be the extension of $\mathbf{CL}(\otimes)$ with a unary propositional function symbol $\odot$. Extend the notion of a* formula *and a* literal *to to be the following:*

(a) Any atom q is a formula and a literal.

(b) If A and B are formulas so are $A \to B$ and $A \otimes B$.

(c) If A is a literal (a formula without $\to$ and $\otimes$) then $A^{\odot}$ is a formula and a literal.

Essentially $\odot$ is not a connective but a function symbol generating more literals (atoms) from literals (atoms). We shall later deal with function symbols $\odot$ satisfying $q^{\odot\odot} = q$. Such functions are indexed by "2".

Thus when we write $\mathbf{CL}_2(\otimes, \odot)$, we mean that $\odot$ satisfies $q^{\odot\odot} = q$.

2. We can, of course, extend the language of $\mathbf{CL}(\otimes)$ with a new proper unary connective ρ to obtain $\mathbf{CL}(\otimes, \rho)$. For example in 3.1 we added to $\mathbf{CL}(\otimes)$ the negation symbol $\neg$ and obtained $\mathbf{CL}(\otimes, \neg)$. The formulas of $\mathbf{CL}(\otimes, \rho)$ are defined as follows:

(a) Any atomic q is a formula and a literal.

(b) If A and B are formulas so are $A \to B$, $A \otimes B$ and ρA.

(c) If A is a literal so is ρA.

The difference between $\mathbf{CL}(\otimes, \odot)$ and $\mathbf{CL}(\otimes, \rho)$ is that $\odot$ is allowed to apply only to literals. We are not allowed to form $\odot(A \to B)$.

The proof theory of $\mathbf{CL}(\otimes, \odot))$ is the same as that of $\mathbf{CL}(\otimes)$. For all purposes $\mathbf{CL}(\otimes, \odot)$ is just $\mathbf{CL}(\otimes)$ with several types of atoms. We have the following

$$\mathbf{CL}(\otimes, \odot) \vdash A \text{ iff } \mathbf{CL}(\otimes) \vdash A'$$

where A' is obtained from A by replacing each atom $q^{\odot \cdots \odot}$ (with n times $\odot$) by a completely new atom q_n.

Even the case of $\mathbf{CL}(\otimes, \odot)$ is no different. We may think that perhaps an additional axiom $q^{\odot\odot} = q$ is needed but this is not the case. Take $\mathbf{CL}(\otimes)$ and divide its atoms into pairs, q and $q^{\odot}$. Then any formula A of $\mathbf{CL}(\otimes, \odot)$ can be translated into a formula A' of $\mathbf{CL}(\otimes)$ by letting $(q^{\odot \cdots \odot})' = q$ iff the number of $\odot$ is even, $q^{\odot}$ if the number is odd.

We have

$$\mathbf{CL}(\otimes, \odot) \vdash A \text{ iff } \mathbf{CL}(\otimes) \vdash A'$$

We can take the above as the definition of provability in $\mathbf{CL}(\otimes, \odot)$.

We can do the above translation only because $q^{\odot}$ is *completely independent* of q.

Definition 10.7.2 *1. Let $\mathbf{CL}(\otimes, \rho)$ be the extension of $\mathbf{CL}(\otimes)$ with an additional connective ρ. We say that ρ is a* rewrite extension *of $\mathbf{CL}(\otimes)$, provided it satisfies axioms of the form:*

- $\rho(A \to B) \leftrightarrow \Psi_{\to}(A, \rho A, B, \rho B)$
- $\rho(A \otimes B) \leftrightarrow \Psi_{\otimes}(A, \rho A, B, \rho B)$

where $\psi_{\to}, \Psi_{\otimes}$ are formulas of $\mathbf{CL}(\otimes, \rho)$ *of* lower complexity *than $\rho(A \to B)$ and $\rho(A \otimes B)$ respectively, according to some well defined complexity measure.*

2. *Let* $\mathbf{CL}(\otimes, \odot)$ *be the extension of* $\mathbf{CL}(\otimes)$ *with the function symbol $\odot$. Let $\Psi(\odot, q, \bot_1, \ldots, \bot_n)$ be a formula with one atom q, the symbol $\odot$, and n atoms $\bot_1, \ldots, \bot_n$ considered as parameters. Let* $\mathbf{CL}_\Psi(\otimes, \odot)$ *be the extension of* $\mathbf{CL}(\otimes, \odot)$ *with the additional axiom:*

$$\Psi(q, \bot_1, \ldots, \bot_n).$$

The above axiom puts down conditions on the functionality of $\odot$ on the atoms. For example, we may have

$$\odot q \leftrightarrow (q \to \bot)$$

for a fixed atom $\bot$. This says that the function $\odot q$ is definable in terms of $\to$ and $\bot$. Another possibility is

$$\odot \odot q \leftrightarrow q.$$

This says $\odot$ is idempotent.

Example 10.7.3 *Let "ρ" be "$\neg$", a negation symbol and consider the rewrite*

$$\begin{array}{l}\neg(A \to B) \leftrightarrow (A \otimes \neg B)\\ \neg(A \otimes B) \leftrightarrow (A \to \neg B)\\ \neg\neg A \leftrightarrow A\end{array}$$

Then we obtain the extension $\mathbf{CL}(\otimes, \neg)$.

Our purpose is to explain the negation in $\mathbf{LL}(\otimes, \neg)$ in terms of rewrite extensions and functionality of a unary connective. In fact, we are going to show that $\neg$ is the rewrite of example 5.3 which has the additional functionality of

$$\neg A \leftrightarrow (A \to \bot)$$

for a fixed $\bot$.

The next sequence of definitions and lemmas will establish these claims.

Lemma 10.7.4 *Let $\bot$ be an arbitrary atom in* $\mathbf{LL}$, *then the following holds:*

1. $\vdash_{\mathbf{LL}} ((A \to \bot) \to \bot) \leftrightarrow ((((A \to \bot) \to \bot) \to \bot) \to \bot)$

2. *Let* $\mathbf{LL}^{\bot}$ *be the fragment of* $\mathbf{LL}$ *built up using $\to$ inductively as follows:*

   (a) *$\bot$ is a wff of* $\mathbf{LL}^{\bot}$. *If q is any atom then $(q \to \bot) \to \bot$ is a wff of* $\mathbf{LL}^{\bot}$

   (b) *If A and B are wffs of* $\mathbf{LL}^{\bot}$ *so is $A \to B$.*

   Then for any A of $\mathbf{LL}^{\bot}$ *we have*

$$\vdash_{\mathbf{LL}} A \leftrightarrow ((A \to \bot) \to \bot)$$

3. *Let A, B and C be formulas of* $\mathbf{LL}^{\bot}$ *then*

$$\vdash_{\mathbf{LL}} (((A \to (B \to \bot)) \to \bot) \to C) \leftrightarrow (A \to (B \to C)).$$

Proof.

1. First observe that
$$\vdash_{\mathbf{LL}} A \to ((A \to \bot) \to \bot),$$
because
$$\vdash_{\mathbf{LL}} (A \to \bot) \to (A \to \bot).$$
Second, observe that since
$$\vdash_{\mathbf{LL}} (A \to B) \to [(B \to C) \to (A \to C)]$$
we get for $C = \bot$ that
$$\vdash (A \to B) \to [(B \to \bot) \to (A \to \bot)]$$
Hence by using our first and second observations we get since
$$\vdash_{\mathbf{LL}} ((A \to \bot) \to (((A \to \bot) \to \bot) \to \bot$$
that
$$\vdash ((((A \to \bot) \to \bot) \to \bot) \to \bot) \to ((A \to \bot) \to \bot)$$
which proves (1).

2. Let A be a wff of $\mathbf{LL}^{\bot}$. We show
$$\vdash_{\mathbf{LL}} ((A \to \bot) \to \bot) \to A.$$
We prove this by induction on the structure of A. for A of the form $(q \to \bot) \to \bot$, this is proved in (1).

   Assume
$$\vdash A \leftrightarrow ((A \to \bot) \to \bot)$$
$$\vdash B \leftrightarrow ((A \to \bot) \to \bot)$$
we shall show

   (a) $\vdash (((A \to B) \to \bot) \to \bot) \to (A \to B)$
   We can show instead

   (a′) $\vdash (((A \to B) \to \bot) \to \bot) \to (A \to ((B \to \bot) \to \bot))$
   we have
$$\vdash (A \to B) \to ((B \to \bot) \to (A \to \bot))$$
   hence

   (b) $\vdash A \to ((B \to \bot) \to ((A \to B) \to \bot))$
   since
$$\vdash (p \to (q \to r)) \to ((r \to s) \to (p \to (q \to s)))$$
   we get for $p = A$, $q = B \to \bot$, $r = (A \to B) \to \bot$ and $s = \bot$
   the formula
$$\vdash (b) \to (a')$$
   and by modus ponens we get $\vdash (a')$.

3. Since $\vdash A \to (B \to ((A \to (B \to \bot)) \to \bot))$ we get
$$\vdash (((A \to (B \to \bot)) \to \bot) \to C) \to (A \to (B \to C)).$$
For the other direction, in order to show
$$\vdash (A \to (B \to C)) \to (((A \to (B \to \bot)) \to \bot) \to C).$$

We show instead that

$$\vdash (A \to (B \to C)) \to ((A \to (B \to \bot)) \to \bot) \to ((C \to \bot) \to \bot)))$$

using the fact that

$$\vdash ((C \to \bot) \to \bot) \to C.$$

The above follows from

$$\vdash (A \to (B \to C) \to ((C \to \bot) \to (A \to (B \to \bot)))$$

which is indeed provable.

■

Lemma 10.7.5 *Let $\bot$ be an arbitrary atom. Let $\tau^{\bot}$ be a translation from* $\mathbf{LL}(\neg)$ *of definition 10.3.1 into* $\mathbf{LL}$ *defined as follows*

1. $\tau^{\bot}(q) = \mathit{def}\ (q \to \bot) \to \bot$, *for atomic q.*
2. $\tau^{\bot}(A \to B) = \tau^{\bot}(A) \to \tau^{\bot}(B)$.
3. $\tau^{\bot}(\neg A) = \tau^{\bot}(A) \to \bot$.

Then the following holds:

$$\mathbf{LL}(\neg) \vdash A \textit{ iff } \mathbf{LL} \vdash \tau^{\bot}(A).$$

Proof.

1. Assume $\mathbf{LL}(\neg) \vdash A$. Consider $\tau^{\bot}(A)$. This formula is in the fragment $\mathbf{LL}^{\bot}$ of $\mathbf{LL}$ as defined in lemma 10.7.4. By that same lemma, the $\tau^{\bot}$ translations of the axioms of $\mathbf{LL}(\neg)$ are theorems of $\mathbf{LL}$. The translation of modus ponens itself and the translation of $(A \to B) \to (\neg B \to \neg A)$ is a theorem of $\mathbf{LL}$. Therefore any $\mathbf{LL}(\neg)$ proof of A will be translated into an $\mathbf{LL}$ proof of $\tau^{\bot}(A)$ and hence $\mathbf{LL} \vdash \tau^{\bot}(A)$.

2. Assume that $\mathbf{LL}(\neg) \nvdash A$. We will show that $\mathbf{LL} \nvdash \tau^{\bot}(A)$. We show that by constructing a countermodel. We begin with $\mathbf{LL}(\neg)$. We can consider $\mathbf{LL}(\neg)$ as a version of $\mathbf{LL}$ without $\neg$, by viewing any wff of the form $\neg B$ as atomic. Thus we pretend that $\neg$ is a symbol generating new atoms. We construct the canonical $\mathbf{LL}$ structure (X, h) of this language, as in 10.8.10. In this canonical structure, the elements are all multisets of $\mathbf{LL}(\neg)$ formulas $\{A_1, \ldots, A_n\}$. For atomic q (and also for negated $\neg B$, which we consider as atomic), we have:

$$h(\{A_1, \ldots, A_n\}, q) = 1 \text{ iff } \vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to q) \ldots)$$

and

$$h(\{A_1, \ldots, A_n\}, \neg B) = 1 \text{ iff } \vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to \neg B) \ldots).$$

We can prove by induction on $\to$ that for any wff C built up from these "atoms" we have:

$$h(\{A_1, \ldots, A_n\}, C) = 1 \text{ iff } \vdash_{\mathbf{LL}(\neg)} A_1 \to (A_2 \to \ldots \to (A_n \to C) \ldots).$$

What we do not have in this structure is an inductive condition for the value $h(t, \neg B)$. Our next step is to obtain such an inductive clause.

The language $\mathbf{LL}(\neg)$ has real atoms $\{q_1, q_2, \ldots\}$. We pretended that any negated formula of the form $\neg B$ is also an "atom" and got the canonical structure (X, h) for the language with "atoms" $\{q_i\}$ and $\{\neg B\}$. We now want to create a new atom $\bot$ and reduce $\neg B$ to $B \to \bot$.

We begin by taking a new atomic letter $\bot$ and defining a new assignment h' on the atoms $\{q_i\} \cup \{\bot\}$. h' is defined using the following clauses:

- $h'(\{A_1,\ldots,A_n\},q) = h(\{A_1,\ldots,A_n\},q)$ for atomic $q \neq \bot$.
- $h'(\{A_1,\ldots,A_n\},\bot) = 1$iff $\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_{n-1} \to \neg A_n)\ldots)$.

Notice that in $\mathbf{LL}(\neg)$ the second condition is equivalent to the following:

$$\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots A_{i-1} \to (A_{i+1} \to \ldots \to (A_n \to \neg A_i)\ldots)$$

We now have a structure for the language $\mathbf{LL}^\bot$, ie $\mathbf{LL}$ with $\bot$. Consider a formula A of the language $\mathbf{LL}(\neg)$. It gets a truth value $h(\{A_1,\ldots,A_n\},A)$ at the point $\{A_1,\ldots,A_n\}$ under h by considering as "atoms" in it any real atom q_i and any subformula of the form $\neg B$. In parallel we can consider $h'(\{A_1,\ldots,A_n\},\tau^\bot(A))$. This function also gives a value because $\tau^\bot(A)$ is a formula built up from real atoms q_i and $\bot$ and h' is an assignment for this language. How are these two values related? We claim that for any A we have:

(*) $h(\{A_1,\ldots,A_n\},A) = 1$ iff $h'(\{A_1,\ldots,A_n\},\tau^\bot(A)) = 1$.
To show (*) we need to show (**) below:

(**) For any B of $\mathbf{LL}(\neg)$ we have:

$$h(\{A_1,\ldots,A_n\},\neg B) = 1 \text{ iff } h'(\{A_1,\ldots,A_n\},\tau^\bot(B) \to \bot) = 1.$$

Case of atomic $q \neq \bot$
First note that for q atomic, $\tau^\bot(q) \to \bot$ is $((q \to \bot) \to \bot)) \to \bot$ which is equivalent to $q \to \bot$ in $\mathbf{LL}$. Thus to prove (**) in the atomic case it is sufficient to show that $h(t,\neg q) = h'(t, q \to \bot)$. Using that we prove (*) namely that $h(t,q) = h'(t,(q \to \bot) \to \bot)$.

1. First we prove (**). Note that:

   (a) $h(\{A_1,\ldots,A_n\},\neg q) = 1$ iff by definition $\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to \neg q)\ldots)$.
   On the other hand we also have

   (b) $h'(\{A_1\ldots,A_n\},q \to \bot) = 1$ iff for any $\{B_1,\ldots,B_m\}$ if $h'(\{B_1,\ldots,B_m\},q) = 1$ then $h'(\{A_1,\ldots,A_n,B_1,\ldots,B_m\},\bot) = 1$. The above holds iff whenever $\vdash_{\mathbf{LL}(\neg)} B_1 \to \ldots \to (B_m \to q)\ldots)$ then $\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to (B_1 \to \ldots \to B_{m-1} \to \neg B_n)\ldots)$.

   Assume (a) and show (b):
   If $\vdash_{\mathbf{LL}(\neg)} B_1 \to \ldots \to (B_m \to q)\ldots)$ then $\vdash_{\mathbf{LL}(\neg)} B_1 \to \ldots \to (\neg q \to \neg B_m)\ldots)$ or equivalently $\vdash_{\mathbf{LL}(\neg)} \neg q \to (B_1 \to (\ldots \to (B_{m-1} \to \neg B_m)\ldots)$ and hence from (a) we get

   $$\vdash A_1 \to \ldots A_n \to (B_1 \to \ldots (B_{m-1} \to \neg B_m))\ldots)$$

   Assume (b) and show (a). From (b) we get by taking $m = 1$ and $B_m = q$ that $\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to \neg q)\ldots)$, which is (a).

2. We now prove (*):
   Assume that $h(\{A_1,\ldots,A_n\},q) = 1$. Then $\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to q)\ldots)$.

   To show $h'(\{A_1,\ldots,A_n,\tau^\bot(q)\})) = 1$, observe that since $\tau^\bot(q)$ is $(q \to \bot) \to \bot$ we need to show that $h'(\{A_1,\ldots,A_n\},(q \to \bot) \to \bot)) = 1$. Let $\{B_1,\ldots,B_m\}$ be such that $h'(\{B_1,\ldots B_m\},q \to \bot) = 1$. We want to show $h'(\{A_1,\ldots,A_n,B_1,\ldots,B_m\},\bot) = 1$

   By (**) which we have already proved for atomic q, we deduce that

   $$\vdash_{\mathbf{LL}(\neg)} B_1 \to \ldots (B_m \to \neg q)\ldots)$$

   hence

   $$\vdash_{\mathbf{LL}(\neg)} q \to (B_1 \to \ldots \to (B_{m-1} \to \neg B_m)\ldots)$$

hence

$$\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to \ldots \to (B_{m-1} \to \neg B_m) \ldots)$$

hence by definition

$$h'(\{A_1, \ldots, A_n, B_1, \ldots, B_m\}, \bot) = 1.$$

For the other direction, assume $h'(\{A_1, \ldots, A_n\}, (q \to \bot) \to \bot) = 1$. We show that $\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to q) \ldots)$.

From (**) we get that $h'(\{\neg q\}, q \to \bot) = 1$ and hence

$$h'(\{A_1, \ldots, A_n, \neg q\}, \bot) = 1$$

and hence

$$\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to q) \ldots)$$

and hence

$$h(\{A_1, \ldots, A_n\}, q) = 1.$$

This concludes the proof of both (*) and (**) for the case of atomic q. Note that we also got the following

(***) $h'(t, q) = h'(t, (q \to \bot) \to \bot)$, for q atomic.

The non atomic case
We first prove (**): Assume

$$h(\{A_1, \ldots, A_n\}, \neg B) = 1$$

Then

$$\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to \neg B) \ldots).$$

We show

$$h'(\{A_1, \ldots, A_n\}, \tau^{\bot}(B) \to \bot) = 1$$

Assume

$$h'(\{B_1, \ldots, B_m\}, \tau^{\bot}(B)) = 1$$

Then by the induction hypothesis, from (*) we get

$$\vdash_{\mathbf{LL}(\neg)} B_1 \to \ldots \to (B_m \to B)$$

or

$$\vdash_{\mathbf{LL}(\neg)} \neg B \to (B_1 \to \ldots (B_{m-1} \to \neg B_m) \ldots)$$

Therefore

$$\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to B_1 \to \ldots (B_{m-1} \to \neg B_m) \ldots)$$

and so by definition

$$h'(\{A_1, \ldots, A_n, B_1, \ldots, B_m\}, \bot) = 1$$

Assume

$$h'(\{A_1, \ldots, A_n\}, \tau^{\bot}(B) \to \bot) = 1,$$

we show

$$h(\{A_1, \ldots, A_n\}, \neg B) = 1.$$

By the induction hypothesis $h'(\{B\}, \tau^{\bot}(B)) = 1$ and so $h'(\{A_1, \ldots, A_n, B\}, \bot) = 1$ and so by definition

$$\vdash_{\mathbf{LL}(\neg)} A_1 \to \ldots \to (A_n \to \neg B) \ldots)$$

hence

$$h(\{A_1, \ldots, A_n\}, \neg B) = 1.$$

It is now possible to prove (*) for the non atomic case. For the case $= \neg B$, note that $\tau^{\perp}(\neg B)$ is $\tau^{\perp}(B) \to \perp$. For B atomic the result follows from (***) and for other B the result follows from (**). The case of $B = C \to D$ is also immediate because $\tau^{\perp}$ commutes with $\to$ and both h and h' follow the same truth table for $\to$.

We now concluded the proofs of (*) and (**) and we can resume the main line of the proof.

Let A be such that $\mathbf{LL}(\neg) \not\vdash A$. Consider the structure (X, h). We have $h(\varnothing, A) = 0$. Consider the structure (X, h'). We have $h'(\varnothing, \tau^{\perp}(A) = 0$. It therefore follows that $\mathbf{LL} \not\vdash \tau^{\perp}(A)$.

This concludes the proof of lemma 10.7.5. ■

10.8 Concatenation semantics

This section will provide Kripke like possible world semantics for some of the Resource Logics.

We continue with the system **CL**, defined in 9.2.5 and axiomatised in 10.1.1.

Our starting point is a general implicational system, with no axioms or rules at all. Thus we have a language with $\to$ only and possibly the classical connectives $\wedge$ and $\vee$. We know nothing about $\to$. So we have to give it the most general semantics available and that would be a form of neighbourhood semantics.

Definition 10.8.1 (SNS Semantics) *1. A (binary) selection neighbourhood propositional structure* **SNS** *has the form* $(\mathbf{s}, S, \mathbf{R}, h)$, *where* S *is a set of possible worlds, or states,* s *is a selection function, giving for each* $Q \subseteq S$ *a subset* $\mathbf{s}(Q)$ *the subset of preferred worlds; and* $\mathbf{R}$ *is the three place neighbourhood relation* $\mathbf{R} \subseteq S \times 2^S \times 2^S$. *We require* $\mathbf{s}$ *to be monotonic decreasing, namely* $Q_1 \subseteq Q_2 \Rightarrow \mathbf{s}(Q_2) \subseteq \mathbf{s}(Q_1)$.

h is an assignment giving truth values to each atom and world $h(t, q) \in \{0, 1\}$.

2. *The truth value of a formula* A *at a world* t *under* h *is denoted by* $\|A\|^h_t$ *and is defined as follows*

   2.1. $\|A\|^h_t = 1$ *iff* $h(t, A) = 1$ *for* A *atomic.*

   2.2. $\|A \to B\|^h_t = 1$ *iff* $(t, \|A\|, \|B\|) \in \mathbf{R}$ *where* $\|Q\| = \{t \mid \|Q\|^h_t = 1\}$.

   2.3. $\|A \wedge B\|^h_t = 1$ *iff* $\|A\|^h_t = \|B\|^h_t = 1$

   2.4. $\|A \vee B\|^h_t = 1$ *iff* $\|A\|^h_t = 1$ *or* $\|B\|^h_t = 1$.

3. *We say* A *holds in the structure if* $\|A\|^h_t = 1$ *for all* $t \in \mathbf{s}(S)$.

4. *We say* A *is valid in the semantics iff* A *holds in all structures*

5. *We say* $A \mid\!\sim B$ *iff for all structures* $\|A\| \cap \mathbf{s}(\|A\|) \subseteq \|B\|$, $\mid\!\sim$ *is a notion of consequence between wffs. In the presence of conjunction we can let* $A_1, \ldots, A_n \mid\!\sim B$ *be* $A_1 \wedge \ldots \wedge A_n \mid\!\sim B$.

We shall proceed with the pure implicational fragment or possibly with the $\{\to, \wedge\}$ fragment for a while, without disjunciton because disjunction can be problematic and requires special attention. Our interest focusses on implication. We defined a consequence relation $A \mid\!\sim B$ between formulas. In order for $\mid\!\sim$ to be a logical sysem it must satisfy

Reflexivity:	$A \mid\!\sim A$
Monotonicity:	$A_1, \ldots A_n \mid\!\sim A$
	implies $A_1, \ldots, A_n, B \mid\!\sim A$
Cut:	$A_1, \ldots, A_n, B \mid\!\sim A$
	and $B_1, \ldots, B_m \mid\!\sim B$
	imply $A_1, \ldots, A_n, B_1, \ldots, B_m \mid\!\sim A$

Cut and monotonicity do not necessarily hold. Whether or not they hold depends on the semantics. Thus we have the capability of dealing with non-monotonic systems.

For example, to satisfy the Cut Rule we need s to satisfy the following:

$$Q_1 \cap Q_2 \cap \mathbf{s}(Q_1 \cap Q_2) \subseteq Q \text{ and } Q_3 \cap \mathbf{s}(Q_3) \subseteq Q_2$$

imply

$$Q_1 \cap Q_3 \cap \mathbf{s}(Q_1 \cap Q_3) \subseteq Q$$

or better:

$$Q' \cap \mathbf{s}(Q') \subseteq Q \text{ and } Q'' \cap \mathbf{s}(Q'') \subseteq Q'$$

imply

$$Q'' \cap \mathbf{s}(Q'') \subseteq Q$$

Remark 10.8.2 (Semantical Conditions) *We can offer various natural conditions on our* **SNS** *semantics, leading to some well-known interpretations.*

1. *We can assume that for some $S_0 \subseteq S$, we have $\mathbf{s}(S) = S_0$, and that, in fact, $\mathbf{s}(Q) = S_0$ for all $Q \subseteq S$. Thus our structures now have the form $(S_0, S, \mathbf{R}, h)$.*

   Obvious choices for S_0 are $S_0 = S$ or $S_0 = \{s_0\}$ for some fixed $s_0 \in S$.

   This new semantics is neighbourhood semantics without selection function. If $S_0 = S$ is chosen, then the rule:

   $$\frac{\vdash A \leftrightarrow A', \vdash B \leftrightarrow B'}{\vdash (A \rightarrow B) \leftrightarrow (A' \rightarrow B')}$$

   is valid.

   This rule is not valid if we choose $S_0 = \{s_0\}$. This choice gives validity in the actual world as opposed to validity in the frame.

2. *We can assume that $\mathbf{R}(t, Q_1, Q_2)$ and $\mathbf{s}(Q)$ are derived from some first order formulas*

   $$\Psi_{\mathbf{R}}(t, Q_1, Q_2, P_1, \ldots, P_m), \Psi_{\mathbf{s}}(t, Q, P_1, \ldots, P_2)$$

   where P_i are some fixed predicates on S. Namely we have:

   - *$\mathbf{R}(t, Q_1, Q_2)$ iff $(S, P_1, \ldots, P_m) \vDash \Psi_{\mathbf{R}}$.*
   - *$t \in \mathbf{s}(Q)$ iff $(S, P_1, \ldots, P_m) \vDash \Psi_{\mathbf{s}}$.*

   For example we may have a preferential order $\leq$ on S and $\mathbf{s}(Q)$ is the set of minimal points relative to Q, defined using the order

   $$\Psi_{\mathbf{s}}(t, Q) = \ def\ \sim \exists s \in Q(s \lneqq t).$$

   Examples of such semantics is the Kraus Lehman Magidor preferential semantics for non monotonic consequence relations. See the section below on Cumulative Concatenation Logic **CCL**.

3. *We can alternatively assume that $\mathbf{R}(t, Q_1, Q_2)$ is derived from a pointwise ternary relation $R(x, y, z)$ through the definition:*

   - *$\mathbf{R}(t, Q_1, Q_2)$ iff $\forall y \forall z[y \in Q_1 \wedge R(t, y, z) \rightarrow z \in Q_2]$.*
     or possibly:
   - *$\mathbf{R}(t, Q_1, Q_2)$ iff $\forall y \exists z[y \in Q_1 \rightarrow R(t, y, z) \wedge z \in Q_2]$.*

   Examples of such systems are Site Logic **SL**, *(see example below) and the merge sematnics of definition 10.8.4 below.*

4. *We can further assume that R is functional, i.e. $\forall t, y \exists ! z R(t, y, z)$, we can let $z = t + y$.*

   An example of such a system is concatenation logic.

Example 10.8.3 *Consider the rule of modus ponens*

$$\frac{\vdash A \quad \vdash A \to B}{\vdash B}$$

in the seamantics with $(S_0, S, \mathbf{R}, h)$. We need a condition on $\mathbf{R}$ which will ensure that:

1. $(\forall x \in S_0)(\|A\|_x = 1)$
2. $(\forall x \in S_0)\mathbf{R}(x, \|A\|, \|B\|)$
   imply
3. $\forall x \in S_0(\|B\|_x = 1)$.

Obviously we need a condition to ensure that

$$\forall x \in S_0 \mathbf{R}(s, Q_1, Q_2)$$

and

$$S_0 \subseteq Q_1$$

imply

$$S_0 \subseteq Q_2.$$

For the semantics for which $S_0 = S$, we get the condition

(*) $\forall x \mathbf{R}(x, S, Q) \Rightarrow S = Q$

For the semantics for which $S_0 = \{s_0\}$,we get the condition

(**) $\mathbf{R}(s_0, Q_1, Q_2) \wedge s_0 \in Q_1 \Rightarrow s_0 \in Q_2$

If in addition we assume that $\mathbf{R}$ is obtained from a pointwise ternary relation R, namely:

$$\mathbf{R}(t, Q_1, Q_2) \text{ iff } \forall yz[y \in Q_1 \wedge R(t, y, z) \to z \in Q_2]$$

then the conditions become:

- $\forall x[\forall z \forall y[R(x, y, z) \to z \in Q] \to S = Q]$
  and
- $[\forall z \forall y[R(s_0, y, z) \wedge y \in Q_1 \to z \in Q_2] \to Q_2 = \{s_0\}]$

Example 10.8.4 (Site Logic) *We introduce the logic* **SL**, Site Logic*, see Barwise-Gabbay [11]. The language of* **SL** *contains atomic propositions and the binary connective $\to$. The logic is defined semantically through applicative structures of the form (S, W, o, h) where S is a set of states, $o \in S$ and W are application functions associating with each $t \in S$ a function $W(t) : S \to 2^S$. We write $W(t, s) \subseteq S$, for $W(t)(s)$ or, when convenient, $t(s) \subseteq S, h$ is an assignment, for atomic q and $t \in S, h(t, q) \in \{0, 1\}$.*

Let $(t_1, \ldots, t_n)$ be a sequence. Define $W_{(t_1,\ldots,t_n)}$ to be the function $W(t_1) \circ W(t_2) \circ \ldots \circ W(t_n)$ (or $t_1 \circ t_2 \circ \ldots \circ t_n$), where "$\circ$" is functional composition defined as follows:

$$W(t) \circ W(s)(x) = \{y \mid \exists z[z \in W(s)(x) \wedge y \in W(t)(z)]\}$$

Given a structure (S, W, o, h), extend h to h^* defined on all formulas as follows:
$h^*(t, A \to B) = 1$ iff for all s such that $h^*(s, A) = 1$ we have that for all $y \in W(t, s), h^*(y, B) = 1$.

We can assume the identity function id *is available, with* id $(t) = \{t\}$.

We let $(S, W, o, h) \vDash A$ iff def $h^*(o, A) = 1$.

We say $\vDash A$ iff A holds in all structures.

Note that in this logic, not even $A \to A$ is a theorem. If we insist that $o =$ id, then $A \to A$ is a theorem.

The following is an axiom system for **SL**.

Axioms:

none.

Inference Rules:

$$\frac{B \vdash C}{A \to B \vdash A \to C}$$

Functional application is not necessarily associative. Given t, s and u, we can either consider the set:

$$t[s(u)] = \{y \mid \exists z \in s(u)(y = t(z))\}$$

or the set

$$[t(s)](u) = \{y \mid \exists z \in t(s)(y = z(u))\}.$$

These two sets need not be equal. If we stipulate that they are, then

$$h^*(\mathrm{id}, (A \to B) \to ((C \to A) \to (C \to B))) = 1.$$

The relation $z \in x(y)$ is a three place relation $R(x, y, z)$ and shows the connection between the site sematnics and the ternary possible worlds semantics.

There is another way of looking at the functional $x(y)$. For each $z_i \in x(y)$, we can imagine a point binary function $+_i$ giving $z_i = x +_i y$, so that $x(y) = \{x +_i y \mid z_i \in x(y)\}$. This way we replace the set function by a family of binary functions. In case we have only one such function, ie $x(y) = \{z\}$ and $+$ is associative, we get concatenation logic, studied in the next section.

Definition 10.8.5 *1. A* **CL** *structure is a pair (X, h) where X is a set of atomic letters and h is a function assigning to each atomic proposition q and each finite sequence $\alpha \in X^*$ a truth value $h(\alpha, q) \in \{0, 1\}$, where X^* is the set of all finite (or empty) sequences of elements of X.*

The function h is called an assignment.

The function h can be extended to a function h^* on arbitrary formulas via the following recursive equation:

$$(*) \quad h^*(\alpha, A \to B) = 1 \textit{ iff }$$
$$\forall\beta[h^*(\beta, A) = 1 \textit{ and } (\alpha = \varnothing \vee \beta \neq \varnothing) \Rightarrow h^*(\alpha \otimes \beta, B) = 1]$$

where $\alpha \otimes \beta$ is result of concatenating the sequence β to α from the right. $+$ is assumed to be associative, ie

$$(a \otimes b) \otimes c = a \otimes (b \otimes c) = (a \otimes b \otimes c)$$

2. We say that a formula A is valid in the semantics if for all structures (X, h), we have $h^*(\varnothing, A) = 1$.

Theorem 10.8.6 *For the Hilbert system* **CL** *of 10.1.1 we have $\vdash_{\mathbf{CL}} A$ iff A is valid in the* **CL** *semantics.*

Proof.

1. It is easy to verify that the axioms and rules of **CL** are valid in the proposed semantics.
2. To show completeness we assume $\nvdash_{\mathbf{CL}} A$ and produce a counter model.

The construction is given below in 10.8.10 with the key lemma being 10.8.11. ■

We now define the semantics for logics stronger than **CL**. The logics we consider are linear logic **LL**, relevance logic **R**, and intuitionistic logic of 9.1.2 as well as for the logic **BR**1 of 9.2.8.

Definition 10.8.7 *An* **LL** *(linear) structure is a pair (X, h), where X is a nonempty set and h is an assignment giving a truth value $h(Y, q) \in \{0, 1\}$ to each atomic q and each multiset Y built out of elements of X. If we assume that h is monotonic in Y, ie*

$$Y_1 \subseteq Y_2 \Rightarrow h(Y_1, q) \geq h(Y_2, q)$$

we get **BR1** *structures. If we assume the multisets Y are sets we get* **R** *relevance structures and if we assume both that we are dealing with subsets Y of X and that $h(Y, q)$ is monotonic, we get intuitionistic structures. In each one of these cases, the function h can be extended to a function h^* on arbitrary formulas by letting*

$$h^*(Y, A \to B) = 1 \text{ iff } \forall Y'[h^*(Y', A) = 1 \Rightarrow h^*(Y \cup Y', B) = 1]$$

Theorem 10.8.8 *A is a theorem of* **BR1** *(respectively, linear, relevance,intuitionistic logic) iff for any corresponding respective structure $(X, h), h^*(\varnothing, A) = 1$.*

Proof. Soundness can be verified directly. For completeness see the construction below and Lemma 10.8.11. ■

Exercise 10.8.9 *What are the corresponding conditions for strict implication and for entailment?*

Construction 10.8.10 (Formula Sequences Canonical Structures) *Let X be the set of all formulas of the language. Let α be a sequence of formulas, $(A_1, \ldots, A_n)$. Define a canonical assignment $h_{\mathbf{L}}$ for any logic* **L** *stronger than* **CL** *by:*

$$(*)\ h_{\mathbf{L}}((A_1, \ldots, A_n), q) = 1 \text{ iff } \vdash_{\mathbf{L}} A_1 \to (A_2 \to \ldots \to (A_n \to q) \ldots).$$

We prove the following.

Lemma 10.8.11 *For any A,*

1. $h_{\mathbf{L}}((A_1, \ldots, A_n), A = 1$ *iff* $\vdash_{\mathbf{L}} A_1 \to (\ldots (A_n \to A) \ldots)$
2. $h_{\mathbf{L}}(\varnothing, A) = 1$ *iff* $\vdash_{\mathbf{L}} A$.

Proof. By induction on A

1. For A atomic the lemma hold by definition.
2. Assume $h_{\mathbf{L}}((A_1, \ldots, A_n), A \to B) = 1$ and show that $\vdash_{\mathbf{L}} A_1 \to \ldots (A_n \to (A \to B) \ldots)$. Take $\alpha = (A)$. Since $\vdash_{\mathbf{L}} A \to A$, by the induction hypothesis we get $h_{\mathbf{L}}(A, A) = 1$, and hence
$$h_{\mathbf{L}}((A_1, \ldots, A_n, A), B) = 1$$
and hence by the induction hypothesis,
$$\vdash_{\mathbf{L}} A_1 \to (A_1 \to \ldots \to (A \to B) \ldots).$$
3. Assume $\vdash_{\mathbf{L}} A_1 \to \ldots \to (A_n \to (A \to B) \ldots)$. We want to show that
$$h_{\mathbf{L}}((A_1, \ldots, A_n), A \to B) = 1.$$
To show that, let $(D_1, \ldots, D_m)$ be any point such that
$$h_{\mathbf{L}}((D_1, \ldots, D_m), A) = 1$$
and we will show that
$$h_{\mathbf{L}}((A_1, \ldots, A_n, D_1, \ldots, D_m), B) = 1.$$

By the induction hypothesis, it is sufficient to show that in **L** the following holds:
From:

$$\vdash_{\mathbf{L}} A_1 \to (A_1 \to \ldots \to (A_n \to (A \to B)\ldots))$$

and

$$\vdash_{\mathbf{L}} D_1 \to \ldots (\to (D_m \to A)\ldots)$$

it follows that

$$\vdash_{\mathbf{L}} A_1 \to \ldots (A_n \to (D_1 \to \ldots \to (D_m \to B)\ldots))$$

This follows from Lemma 10.1.13. Thus Lemma 10.8.11 is proved.

■

Lemma 10.8.12 *In the canonical structure of the previous construction, we have the following correspondence between axioms and properties..*

1. $h_{\mathbf{L}}((A_1, \ldots, A_n), B) = h_{\mathbf{L}}((A'_1, \ldots, A'_n), B)$ *provided the multisets* $\{A_1, \ldots, A_n\}$ *and* $\{A'_1, \ldots, A'_n\}$ *are equal.*

2. $h_{\mathbf{L}}((A_1, \ldots, A_n), B) = 1$ *implies* $h_{\mathbf{L}}((A_1, \ldots, A_n, A), B) = 1$.

Property (1) corresponds to the commutativity axiom

$$\vdash (A_1 \to \ldots \to (A_n \to B)\ldots) \to (A'_1 \to \ldots \to (A'_n \to B)\ldots)$$

whenever $\{A_i\} = \{A'_i\}$ *as multisets. The relevance axiom*

$$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$$

allows us to regard the multisets as sets. Similarly, Property (2) corresponds to the axiom $\vdash B \to (A \to B)$ *which entails*

$$\vdash (A_1 \to \ldots \to (A_n \to B)\ldots) \to (A_1 \ldots \to (A_n \to (A \to B)\ldots))$$

Proof. Easy. ■

We continue and give semantics for the logic **W**. We need to define several concepts.

Definition 10.8.13 1. *Let* α, β *be two sequences, we say that* γ *is a* right merge *of* α, β *if* γ *is an order preserving merge of* α, β *and the last element of* γ *is the last element of* β. *We write* $\gamma \in$ RMerge (α, β). *We can similarly define* left merge *LMerge*(α, β).

We can also define the right merge of any sequence of sequences, $\alpha_1, \ldots, \alpha_m$ *to be any sequence* γ *which is an order preserving merge of* $\alpha_1, \ldots, \alpha_m$ *and which satisfies the condition that the last element of* α_i *precedes the last element of* α_j *whenever* $i < j$.

2. *Let* (X, h) *be a model in the sense of 10.8.1. We can extend* h *to an* $h^\dagger$ *on arbitrary formulas through the following recursive equation:*

$$\begin{aligned}&(\dagger) h^\dagger(\alpha, A \to B) = 1 \textit{ iff}\\ &\forall \beta, \gamma[\gamma \in \textit{ LMerge } (\alpha, \beta) \textit{ and}\\ &h^\dagger(\beta, A) = 1 \textit{ imply } h^\dagger(\gamma, B) = 1].\end{aligned}$$

We call this semantics the left merge semantics *and say a formula* A *is valid in the semantics if for all models* (X, h) *we have* $h^\dagger(\varnothing, A) = 1$.

3. *We can similarly define the notion of* merge semantics *where in the recursive condition* ($\dagger$) *of (2) above, we require that* $\gamma \in$ Merge (α, β).

Lemma 10.8.14 *The logic* **W** *is characterised by the right merge semantics, ie we have for all* $A, \vdash_{\mathbf{W}} A$ *iff* A *is valid in the right merge semantics.*

Proof.

1. **Soundness**
   First we show that for any instance of an axiom and any $h^\dagger$, we have $h^\dagger(\varnothing, A) = 1$.

   We check the left and right transitivity axioms.

   (a) Assume $h^\dagger(\alpha, A \to B) = 1$. We want to show $h^\dagger(\alpha, (C \to A) \to (C \to B)) = 1$. Let $\gamma \in$ *RMerge* (α, β) and assume $h^\dagger(\beta, C \to A) = 1$. We want to show $h^\dagger(\gamma, C \to B)$. Let $\eta \in$ *RMerge* (γ, δ) and assume $h^\dagger(\delta, C) = 1$. We want to show $h^\dagger(\eta, B) = 1$.

   η is a right merge of α, β, δ and it therefore induces a right merge δ_1 of β and δ.

   We therefore deduce that $h^\dagger(\delta_1, A) = 1$ and since $h^\dagger(\alpha, A \to B) = 1$ again we get $h^\dagger(\eta, B) = 1$ since η is a right merge of α and δ_1.

   (b) We now assume that we have $h^\dagger(\alpha, A \to B) = 1$ and want to show that $h^\dagger(\alpha, (B \to C) \to (A \to C)) = 1$. Let $\gamma \in$ *RMerge* (α, β) and assume $h^\dagger(\beta, B \to C) = 1$. We want to show $h^\dagger(\gamma, A \to C) = 1$. Let $\eta \in$ *RMerge* (γ, δ) and assume $h^\dagger(\delta, A) = 1$. We must show $h^\dagger(\eta, C) = 1$. η induces a right merge η_1 of α and γ. Hence we get $h^\dagger(\eta_1, B) = 1$. Since $h^\dagger(\beta, B \to C) = 1$ and η is a right merge of β and η_1 we get again that $h^\dagger(\eta, C) = 1$.

   Second we show that validity in the right merge semantics is closed under modus ponens. Assume that A and $A \to B$ are valid in all models. We show that B is valid in all models. Let (X, h) be any model and consider $h^\dagger(\varnothing, B)$. Since $h^\dagger(\varnothing, A) = 1$ and $h^\dagger(\varnothing, A \to B) = 1$ and $\varnothing \in$ *RMerge* $(\varnothing, \varnothing)$ we get $h^\dagger(\varnothing, B) = 1$.

2. **Completeness**
   Assume $\not\vdash_{\mathbf{W}} A$, we want to exhibit a model (X, h) with $h^\dagger(\varnothing, A) = 0$. We take the canonical model of sequences of formulas constructed in 10.8.10. We want to prove the analog of lemma 10.8.11 for it. We need to examine case 3 of the proof of 10.8.11, namely that we have

   $$\vdash_{\mathbf{W}} A_1 \to \ldots \to (A_n \to (A \to B) \ldots)$$

   and we want to show that $h_{\mathbf{W}}((A_1, \ldots, A_n), A \to B) = 1$. To prove that we assume that $h_{\mathbf{W}}((D_1, \ldots, D_m, D), A) = 1$ and want to show that $h_{\mathbf{W}}((C_1, \ldots, C_{m+n}), B) = 1$, where $(C_1, \ldots, C_{m+n}, D)$ is any right merge of $(A_1, \ldots, A_n)$ and $(D_1, \ldots, D_m, D)$. By the induction hypothesis, we know that

   $$\vdash_{\mathbf{W}} D_1 \to \ldots \to (D_m \to (D \to A) \ldots)$$

   and we want to show that

   $$\vdash_{\mathbf{W}} C_1 \to \ldots \to (C_{m+n} \to (D \to B) \ldots).$$

   However, this is exactly what lemma 10.8.11 proved.

■

Example 10.8.15 *The previous semantics suggests a natural semantic definition of a new logic, which we call* Merge logic. *Its theorems are all wffs valid in the merge semantics. It is an extension of* **W** *and is indeed linear logic (which is complete for multiset semantics). It is complete for the following rule:*

$$\vdash A_1 \to \ldots \to (A_n \to (A \to B) \ldots)$$

and

$$\vdash D_1 \to \ldots (\to (D_m \to A) \ldots)$$

imply

$$\vdash C_1 \to \ldots \to (C_{m+n} \to B) \ldots$$

where $C_1, \ldots, C_{m+n}$ *is a merge of* $A_1, \ldots, A_n, D_1, \ldots, D_m$.

We can easily believe that the above rule can be proved from the commutativity axiom

$$(A \to (B \to C)) \to (B \to (A \to C))$$

We thus have two semantics for linear logic, the multiset semantics and the Merge semantics.

Example 10.8.16 *We can define the* destructive Merge semantics*, (the* DMerge semantics*) to be the following:*

When we merge two sequences and in the process of merging the element y is about to be put next to the element x, if $x = y$ we can destroy y and not put it in. Thus the usual merge function allows for the following non-deterministic rules:
$\text{Merge}(at, bs) \approx ab\text{Merge}(t, s)$
$\text{Merge}(at, bs) \approx ba\text{Merge}(t, s)$

In particular the usual merge function will give
$\text{Merge}(at, as) = aa\text{merge}(t, s)$

The DMerge *rewrites are the following:*
$\text{DMerge}(at, bs) \approx ab\text{DMerge}(t, s), a \neq b.$
$\text{DMerge}(at, bs) \approx ba\text{DMerge}(t, s), a \neq b.$
$\text{DMerge}(at, as) = a\text{DMerge}(t, s).$

We can show that the axiom

$$(A \to (A \to B)) \to (A \to B)$$

is validated in the DMerge *semantics. This axiom when added to linear logic characterises relevance logic.*

Definition 10.8.17 (Concatenation semantics for $\mathbf{CL}(\otimes)$) *Let (X, h) be a $\mathbf{CL}$ structure as in definition 10.8.1. Define satisfaction in the structure as in (*) of 10.8.1 for the case of $A \to B$ and add the following clause (**) for the case of $A \otimes B$:*

(**) $h^*(\alpha, A \otimes B) = 1$ *iff*
$\exists \beta, \gamma(\alpha = \beta + \gamma$ *and* $h^*(\beta, A) = 1$ *and* $h^*(\gamma, B) = 1)$.

Open Problem 10.8.18 (Completeness theorem of $\mathbf{CL}(\otimes)$ for the concatenation semantics.) *For any A, $\mathbf{CL}(\otimes) \vdash A$ iff A holds in all $\mathbf{CL}(\otimes)$ concatenation structures.*

Proof. the above theorem is open. However, we can obtain completeness if the following additional axiom is available.

$$(C \to A \otimes B) \to A \otimes (C \to B).$$

1. Soundness can easily be verified. The additional axioms of $\mathbf{CL}(\otimes)$ are indeed valid.

2. Completeness is a bit more complicated. The canonical structure of construction in 10.8.10 is not suitable for our purpose. In that canonical structure the nodes are sequences of formulas $(A_1, \ldots, A_n)$ and the canonical assignment is such that B holds at $(A_1, \ldots, A_n)$ if and only if $\vdash A_1 \to (\ldots \to (A_n \to B))$.

This canonical model works well for $\to$ but not for $\otimes$. Consider the sequences $(C \to (A \otimes B), C)$. Clearly $\vdash ((C \to (A \otimes B)) \to (C \to (A \otimes B))$ hence, if our induction is to go through, we must have that $A \otimes B$ gets value 1 at $(C \to A \otimes B, C)$. We must therefore be able to split the sequence into two sequences, one which proves A and one which proves B. This is not possible.

We therefore have to construct a new canonical structure, one which is more finely tuned for our purpose. This is done in construction 10.8.19 below, where completeness of the semantics is proved for a new proof theory $\Delta \vdash_{\text{canonical}} B$.

This allows us to conclude the completeness proof. We observe that for the empty database $\varnothing$, we have by theorem 10.2.3 that $\varnothing \vdash_{\mathbf{CL}(\otimes)} A$ iff $\varnothing \vdash_{\text{canonical}} A$.

Hence if $\mathbf{CL}(\otimes) \not\vdash A$ then $h^*(\varnothing, A) = 0$. ■

Construction 10.8.19 (Proof sequences canonical structure) *1. A labelled sequence of* $\mathbf{CL}(\otimes)$ *formulas* $\tau = (\alpha_1 : A_1, \ldots, \alpha_r : A_r)$ *is a canonical* $\mathbf{CL}(\otimes)$ *proof sequence (with repetitions) from a database* $\Delta_\tau = \{t_i : B_i\}, t_1 < \ldots < t_n$*, iff each labelled formula in the sequence satisfies one of the following conditions:*

(a) A_i *is a* $\mathbf{CL}(\otimes)$ *theorem and* $\alpha_i = \varnothing$*.*

(b) A_i *is either atomic or an implication and is obtained from two previous elements* $\alpha_j : A_k \to A_i$ *and* $\alpha_k : A_k, j, k < i$ *by modus ponens and* $\alpha_i = \alpha_j \otimes \alpha_k$ *and* $max(\alpha_j) < min(\alpha_k)$*.*

(c) A_i *is a repetition of* A_j*,* $j < i$*, and* $A_i = A_j, \alpha_i = \alpha_j$*.*

(d) A_i *is of the form* $A_j \otimes A_{i-1}$ *with* $j < i$ *and for some* $m < k < n$*, we have that* $(\alpha_1 : A_1, \ldots, \alpha_j : A_j)$ *is a canonical proof sequence from* $\{t_1 : B_1, \ldots, t_m : B_m\}$ *and* $(\alpha_{j+1} : A_{j+1}, \ldots, \alpha_{i-1} : A_{i-1})$ *is a canonical proof sequence from* $\{t_{m+1} : B_{m+1}, \ldots, t_k : B_k\}$ *and* $\alpha_i = \alpha_j \otimes \alpha_{i-1}$*.*

(e) The label t_i *of* $(t_i : A_i)$ *is atomic, corresponding to an assumption,* $t_j : B_j$ *and* $A_i = B_j$ *and* $t_i = t_j$*.*

(f) $\alpha_r = (t_1, \ldots, t_n)$

2. Let $\Delta = (A_1, \ldots, A_n)$ *be a sequence of wffs of* $\mathbf{CL}(\otimes)$*. We say* $\Delta \vdash_{canonical} B$ *if when labelling* Δ *as* $\Delta = \{t_1 : A_1, \ldots, t_n : A_n\}, t_1 < \ldots < t_n$*, then there exists a canonical proof sequence with last element* $(t_1, \ldots, t_n) : B$*.*

3. Let X *be the set of all sequences of the form* Δ*. Define an assignment* h *on* X *by*

$$h(\Delta, q) = 1 \text{ iff } \Delta \vdash_{canonical} q.$$

4. We claim the following holds in the canonical structure (X, h)*.*
$(\sharp)$: $h^*((A_1, \ldots, A_r), B) = 1$ *iff* $(A_1, \ldots, A_r) \vdash_{canonical} B$
To show $(\sharp)$ *we proceed by induction.*

(a) Case of $\to$*:*
Assume $h^*((A_1, \ldots, A_r), C \to D) = 1$*. Then for any sequence* $(C_1, \ldots, C_k)$ *in which* C *holds, we have that* D *holds at*
$(A_1, \ldots, A_r, C_1, \ldots, C_k)$*.*
Consider the database $\Delta' = \{x : C\}$ *where* x *is a new atomic label and the canoncial prof sequence* $(x : C)$ *based on* Δ'*. By the induction hypothesis, since for* $\beta = x, (x : C, x : C)$ *is also a canonical proof sequence based on* Δ'*, we have that* C *holds at* $(x : C)$*. Hence* D *holds at* $(A_1, \ldots, A_n, C)$*. Therefore by the induction hypothesis* $(A_1, \ldots, A_n, C) \vdash_{canonical} D$*. Thus there exists a label* δ *such that*

$$(\alpha_1 : A_1, \ldots, \alpha_r : A_r, x : C, \delta : D)$$

is a canonical proof sequence based on $t_1 : A_1, \ldots, t_n : A_n, x : C$ $t_1 < \ldots < t_n < x$*. By condition (2f),* $\delta = (t_1, \ldots, t_n, x)$*. We want to show that there exists a label* β *such that* $(\alpha_1 : A_1, \ldots, \alpha_r : A_r, \beta : C \to D)$ *is a canonical proof sequence based on* $t_1 : A_1, \ldots, t_n : A_n$*. This is obvious, since* $\delta = (t_1, \ldots, t_n, x)$*, we must have* $A_n = C \to D$ *and* $\alpha_r = (t_1, \ldots, t_n)$ *by the proof discipline defined in (1).*
Hence $(\alpha_1 : A_1, \ldots, \alpha_r : A_r, \alpha_r : C \to D)$ *is a canonical proof sequence for* $C \to D$*. The last element in the sequence is a repetition.*
Assume now that there exists a β *such that* $(\alpha_1 : A_1, \ldots, \alpha_r : A_r), A_r = C \to D)$ *a canonical proof sequence. We want to show that* $h^*((A_1, \ldots, A_r), C \to D) = 1$*. By the induction hypothesis we should show that whenever* $(\gamma : C_1, \ldots, \gamma_k : C_k), C_k = C$ *is a canonical proof sequence for* C *from* $\{s_1 : C_1, \ldots, s_m : C_m\}$ *then there exists a canonical proof sequence of* D *from* $(A_1, \ldots, A_n, C_1, \ldots, C_k)$*. It is easy to find such a sequence.*

In fact, $(\alpha_1 : A_1, \ldots, \alpha_r : A_r, \gamma_1 : C_1, \ldots, \gamma_k : C_k, \delta : D)$ is a canonical proof sequence from $t_1 : A_1, \ldots, t_r : A_r, \{s_1 : C_1, \ldots, s_m : C_m\}$ with $t_1 < \ldots < t_r < s_1 < \ldots < s_m$ and with $\delta = (t_1, \ldots, t_r, s_1, \ldots, s_m)$.

(b) Case of $\otimes$
Let $h^*((A_1, \ldots, A_r), C \otimes D) = 1$. Then for some m, C holds at $(A_1, \ldots, A_m)$ and D holds at $(A_{m+1}, \ldots, A_r)$. By the induction hypothesis, there are canonical proof sequences for C from $\{t_1 : A_1, \ldots, t_m : A_m\}$ and for D from $\{t_{m+1} : A_{m+1}, \ldots, t_r : A_r\}$.
Let τ, τ' be the two sequences, then

$$\tau \otimes \tau' \otimes ((t_1, \ldots, t_r) : C \otimes D)$$

is a canonical proof sequence for $C \otimes D$.
Assume there exists a canonical proof sequence

$$\alpha_1 : C_1, \ldots, \alpha_k : C_k, (t_1, \ldots, t_r) : C \otimes D$$

for $C \otimes D$ from $\{t_1 : A_1, \ldots, t_r : A_r\}$. Then by definition, there exists an m and an n such that $\alpha : C_1, \ldots, \alpha_n : C_n$ is a canonical proof sequence for $C_n = C$ from $t_1 : A_1, \ldots, t_m : A_m$ and $\alpha_{n+1} : C_{n+1}, \ldots, \alpha_k : C_k$ is a canonical proof sequence for D from $t_{m+1} : A_{m+1}, \ldots, t_r : A_r$. By the induction hypothesis C holds at $(A_1, \ldots, A_m)$ and D holds at $(A_{m+1}, \ldots, A_r)$.

This concludes the proof of ($\sharp$).
Thus the construction is concluded.

Definition 10.8.20 (Semantics for LL($\neg$)) *An* **LL**($\neg$) *structure has the form $(X, h, X_\perp)$, where (X, h) is an* **LL** *structure in the sense of definition 10.8.7 and $X_\perp \subseteq X^*$. Further assume that h satisfies the following for each atomic q.*

1. *There exists a $\perp$ such that for all $t, h(t, \perp) = 1$ iff $t \in X_\perp$.*
2. *For $\perp$ of (1), and all $q, h(t, q) = h(t, ((q \to \perp) \to \perp))$.*
3. *$h(t, \neg A) = h(t, A \to \perp)$, for any A.*

Lemma 10.8.21 **LL**($\neg$) *is sound and complete for the semantics of the previous definition.*

Proof. Follows from 10.8.19. ■

Remark 10.8.22 *We would like to find a way of constructing structures of the form $(X, h, X_\perp)$ of definition 10.8.20, so that the semantics is not defined by stipulation. At the moment what we do is take any structure (X, h') as in 10.8.10. Choose an arbitrary atom $\perp$ and let $X_\perp = \{t \mid h'(t, \perp) = 1\}$ and define a new h by letting*

$$h(t, q) = h'(t, (q \to \perp) \to \perp).$$

Then we know by previous lemmas that $(X, h, X_\perp)$ satisfies our requirements. We would like to choose X with some additional internal structure on X and use the additional internal structure to define $X_\perp$ and h and be able to prove directly that all the required properties hold.

To give an idea of what our difficulties are, consider the canonical structure of sequences of formulas $(A_1, \ldots, A_n)$. Clearly $\perp$ holds at $(A_1, \ldots, A_n)$ iff $\vdash A_1 \to (\ldots (A_{n-1} \to \neg A_n) \ldots)$.

However this condition uses the proof theory of the langauge and not the pattern list structure of $(A_1, \ldots, A_n)$. For example $\perp$ should hold at $(p \to \neg q, p, q)$ but in the canonical model this is just a sequence of formulas. We can bring the pattern out by writing $((p, \neg q), p, q)$. To be acceptable, X must allow for sequences of sequences, etc. If we follow this approach we will need the families of hereditarily finite sequences, multisets and sets.

Remark 10.8.23 (Semantics for Negation in Relevance Logic) *The semantics of the previous remark can be modified to be a sematnics for relevance logic with negation (*$\mathbf{LL}(\neg)$ *with the additional axiom* $(A \to (A \to B)) \to (A \to B)$*). All we need to do is to let* X^* *be the family of all finite subsets of* X*. This new semantics for* $\neg$ *solves an old problem for the semantics of negation of relevance logic, namely to have a semantical condition on* $\neg$ *which at the same time allows for* $\neg\neg A \leftrightarrow A$ *and* $\neg A = defA \to \bot$*, for a constant* $\bot$*.*

10.9 Semi algebraic semantics for resource logics

Definition 10.9.1 (Semi-algebraic semantics for $\mathbf{CL}(\otimes)$)

1. *Let* $(T, \leq, \otimes, 0)$ *be a partially ordered set with a binary associative operation* $\otimes$ *on it.*

   We require that it satisfies the conditions:

   (a) $0 \otimes x = x$

   (b) $(x \otimes y) \otimes z = x \otimes (y \otimes z)$.

   (c) $x \leq x'$ *and* $y \leq y' \Rightarrow x \otimes y \leq y \otimes y'$.

2. *A* $\mathbf{CL}(\otimes)$ *semi-algebraic structure has the form* $(T, \leq, \otimes, 0, h)$ *where* $(T, \leq, \otimes, 0)$ *is as in (1) and* h *is an assignment, giving for each* $t \in T$ *and atomic* q *a truth value.* h *should satisfy the following condition:*

   (a) $h(t, q) = 1$ *and* $t' \leq t$ *imply* $h(t', q) = 1$

3. *We define by induction the truth values* $\|A\|_t^h$ *and* $\|A\|_t^g$*,for any formula* A *and any* $t \in T$ *according to the following clauses*

   (a) $\|A\|_t^h = 1$ *iff* $h(t, A) = 1$*, for* A *atomic.*

   (b) $\|A \to B\|_t^h = 1$ *iff for all* $x \in T$*, if* $\|A\|_x^h = 1$ *then* $\|B\|_{t\otimes x}^h = 1$
   $\|A \otimes B\|_t^h = 1$ *iff* $\exists x, y, t \leq x \otimes y$ *and*
   $\|A\|_x^h = 1$ *and* $\|B\|_y^h = 1$

4. *We say that* A *holds in the structure iff* $\|A\|_0^h = 1$*.*

Lemma 10.9.2 *In any structure, for any* A, t *and* t' *we have* $\|A\|_t^h = 1$ *and* $t' \leq t$ *imply* $\|A\|_{t'}^h = 1$*.*

Proof. By induction on A.

1. For A atomic this is as required.

2. $\|A \otimes B\|_t^h = 1$ and $t' \leq t$ imply $\|A \otimes B\|_{t'}^h = 1$ by definition.

3. Assume $\|A \to B\|_t^h = 1$ and assume $t' \leq t$. We want to show $\|A \to B\|_{t'}^h = 1$.

Let x be any point such that $\|A\|_x^h = 1$. We show $\|B\|_{t'\otimes x}^h = 1$. We know that $\|B\|_{t\otimes x}^h = 1$ (since $\|A \to B\|_t^h = 1$), and hence by the induction hypotheis, since $t' \otimes x \leq t \otimes x$ we get $\|B\|_{t'\otimes x}^h = 1$. This concludes the proof of the lemma. ■

Theorem 10.9.3 (Completeness theorem for $\mathbf{CL}(\otimes)$.)
$\mathbf{CL}(\otimes) \vdash A$ *iff* A *holds in all semi-algebraic structures.*

Proof.

1. *Soundness*
   We show that all instances of the axioms hold and that there is a closure under the inference rules. It is easy to verify that the axioms involving $\to$ hold. We check the $\otimes$ axiom. We have to show that for any t, $\|(A \otimes B) \to C\|_t^h = \|A \to (B \to C)\|_t^h$.

Assume $\|A \to (B \to C)\|^h_t = 1$. Let x be such $\|A \otimes B\|^h_t = 1$. Thus for some $u, v, x \leq u \otimes v$ and $\|A\|^h_u = 1$ and $\|B\|^h_v = 1$. We want to show $\|C\|^h_{t \otimes x} = 1$. From the assumptions we get $\|C\|^h_{t \otimes u \otimes v}$. Since $t \otimes x \leq t \otimes u \otimes v$, we get $\|C\|^h_{t \otimes x} = 1$. Thus $\|(A \otimes B) \to C\|^h_t = 1$.

Assume $\|(A \otimes B) \to C\|^h_t = 1$. We show $\|A \to (B \to C)\|^h_t = 1$. Let x, y be such $\|A\|^h_x = 1$ and $\|B\|^h_y = 1$. Then certainly $\|A \otimes B\|^h_{x \otimes y} = 1$ and hence $\|C\|^h_{t \otimes x \otimes y} = 1$. Thus by definition since x, y were arbitrary we get $\|A \to (B \to C)\|^h_t = 1$.

2. *Completeness*
   Let T_0 be the set of all wffs of $\mathbf{CL}(\otimes)$. Let $T = T_0 \cup \{\varnothing\}$. Define

   (a) $t \leq t'$ iff $\vdash t \to t'$, for $t \neq \varnothing$.

   (b) $\varnothing \leq t'$ iff $t' = \varnothing$ or $\vdash t'$

   (c) $h(t, q) = 1$ iff $\vdash t \to q$, for $t \neq \varnothing$.

   (d) $h(\varnothing, q) = 0$.

   Let $\otimes$ be the connective $\otimes$. Clearly $\vdash t \to t', \vdash s \to s'$ imply $\vdash (t \otimes s) \to (t' \otimes s')$.

   We thus have a structure. We prove by induction on A that for any t:

$$\|A\|^h_t = 1 \text{ iff } \vdash t \to A, \text{ for } t \neq \varnothing$$

$$\|A\|^h_\varnothing = 1 \text{ iff } \vdash A$$

   (a) This certainly holds for atomic A.

   (b) Assume $\|A \otimes B\|^h_t = 1$. Then for some x, y such that $\vdash t \to (x \otimes y)$ we have by the induction hypothesis:

$$\vdash x \to A \text{ and } \vdash y \to B.$$

   Clearly then $\vdash t \to (A \otimes B)$. If $t = \varnothing$ then we have $\vdash x \otimes y$ and this implies $\vdash A \otimes B$.

   Assume $\vdash t \to (A \otimes B)$. then certainly $t \leq A \otimes B$ and by the induction hypothesis $\|A\|^h_A = 1$ and $\|B\|^h_B = 1$.

   (c) Assume $\|A \to B\|^h_t = 1$. Then for all x, if $\vdash x \to A$ then $\vdash (t \otimes x) \to B$. Let $x = A$. We get $\vdash (t \otimes A) \to B$, ie $\vdash t \to (A \to B)$. If $t \neq \varnothing$, we get $\vdash A \to B$.

   Assume $\vdash t \to (A \to B)$. then certainly for any x, if $\vdash x \to A$ then $\vdash t \otimes x \to B$. If $t = \varnothing$, we are given $\vdash A \to B$, then certainly if $\vdash x \to A$ then $\vdash x \to B$.

   To complete the proof observe that if $\nvdash A$ then by construction $\|A\|^h_\varnothing = 0$.

■

Remark 10.9.4 *First note that in view of the completeness theorem in 10.8.11, we can assume that the semialgebraic semantics has the form $(T, \otimes, 0)$, with $\leq$ taken as equality. T can be taken as the set of all finite sequences of construction 10.8.10, $\otimes$ is concatenation and 0 is $\varnothing$.*

The previous semi-algebraic semantics should be compared with the algebraic semantics given by Girard [Girard, 1987] and Avron [Avron, 1988b, pp. 174–175]. We stress that we are not giving algebraic semantics here but essentially a possible worlds semantics based on algebra. The Kripke semantics for intuitionistic logic also has order $\leq$ involved and if strong negation and some temporal operators are also present, we get some successor or other functions involved.

The difference comes out clearly when we define the semantics for the modality !

We can take the Kripke like clause

$$\|!A\|^h_t = 1 \textit{ iff } \forall x \|A\|^h_{t \otimes x} = 1$$

We do not need an algebraic operation for ! See 10.9.9 below. Compare with Avron [Avron, 1988b, pp. 181–182]. The classical conjunction and disjunction can also be given their traditional truth tables, namely

$$\|A \wedge B\|^h_t = 1 \textit{ iff } \|A\|^h_t = \|B\|^h_t = 1$$

and similarly for $\vee$. The algebraic interpretation with given $\wedge$ and $\vee$ the lattice theoretic join and meet. The distributive law will hold in our interpretation, however, without the option of not having it.

We shall give an even more interesting possible world semantics in the next section.

We can similarly give semi-algebraic semantics for $\mathbf{CL}(\neg)$ as follows:

Definition 10.9.5 *1. Let $(T, \leq, \otimes, \perp, 0)$ be a partially ordered set with a binary associative operation $\otimes$ on it and an involution $\perp$ on it satisfying the following conditions.*

(a) $0 \otimes x = x$

(b) $x \otimes (y \otimes z) = (x \otimes y) \otimes z$

(c) $x \leq x'$ *and* $y \leq y' \Rightarrow x \otimes y \leq x' \otimes y'$

(d) $x^{\perp\perp} = x$

(e) $x \leq y \Rightarrow y^{\perp} \leq x^{\perp}$.

2. CL($\neg$) structure has the form $(T, \leq, \otimes, \perp, 0, h, g)$, where $(T, \leq, \otimes, \perp, 0)$ are as in (1) and h and g are two assignments satisfying:

(a) $h(t, q) = 1$ *and* $t' \leq t$ *imply* $h(t', q) = 1$

(b) $g(t, q) = 1$ *and* $t \leq t'$ *imply* $g(t', q) = 1$.

3. We define by induction the truth values $\|A\|^h_t$ and $\|A\|^g_t$ for any wff A and any $t \in T$ as follows:

(a) For q atomic, let

$$\|q\|^h_t = h(t, q)$$
$$\|q\|^g_t = g(t, q)$$

(b) $\|A \to B\|^h_t = 1$ *iff for al $x \in T$, if $\|A\|^h_x = 1$ then $\|A\|^h_{t \otimes x} = 1$*
$\|A \to B\|^g_t = 1$ *iff* $\exists x, y(t^{\perp} \leq x \otimes y$ *and* $\|A\|^h_x = 1$ *and* $\|B\|^g_{y^{\perp}} = 1)$.

(c) $\|\neg A\|^h_t = \|A\|^g_{t^{\perp}}$
$\|\neg A\|^g_t = \|A\|^h_{t^{\perp}}$.

4. We say A holds in the structure iff $\|A\|^h_0 = 1$. We say A is valid if A holds in all structures.

Lemma 10.9.6 *For any A, t and t'*

1. $\|A\|^h_t = 1$ *and* $t' \leq t$ *imply* $\|A\|^h_{t'} = 1$

2. $\|A\|^g_t = 1$ *and* $t \leq t'$ *imply* $\|A\|^g_{t'} = 1$.

Proof. By induction on A.

1. For atomic A this holds by definition.

2. Let $\|A \to B\|^h_t = 1$ and let $\|A\|^h_x = 1$, for x arbitrary. Then $\|B\|^h_{t \otimes x} = 1$. Now assume $t' \leq t$, then $t' \otimes x \leq t \otimes x$ and hence by the induction hypothesis $\|B\|^h_{t' \otimes x} = 1$. Since x was arbitrary we get $\|A \to B\|^h_t = 1$.

3. Let $\|A \to B\|^g_t = 1$. Then for some $x, y, t^{\perp} \leq x \otimes y$ and $\|A\|^h_x = 1$ and $\|B\|^g_{y^{\perp}} = 1$. Assume $t \leq t'$, then $t'^{\perp} \leq t^{\perp}$ and so by definition $\|A \to B\|^g_{t'} = 1$ because the same x, y will do.

4. $\|\neg A\|^h_t = 1$ and $t' \leq t$ imply $\|\neg A\|^g_{t^{\perp}} = 1$ and $t^{\perp} \leq t'^{\perp}$, which imply by the induction hypothesis $\|A\|^g_{t'^{\perp}}$, which implies $\|\neg A\|^h_{t'} = 1$
$\|\neg A\|^g_t = 1$ and $t \leq t'$ imply $\|A\|^h_{t^{\perp}} = 1$ and $t'^{\perp} \leq t^{\perp}$, which imply by the induction hypothesis $\|A\|^h_{t'^{\perp}} = 1$, which implies $\|\neg A\|^g_{t'} = 1$.

This completes the proof of the lemma. ■

Theorem 10.9.7 (Completeness theorem of CL($\neg$) for the semi-algebraic semantics) CL($\neg$) $\vdash$ *A iff A holds in all semi-algebraic structures.*

Proof.
Soundness
The axioms and rules can be checked to be valid in the semantics.
Completeness
Let T_0 be the set of all formulas of the language. Let $T = T_0 \cup \{\varnothing\}$. Understand $\varnothing \to A$ to be A and $A \to \varnothing$ to be $\neg A$. Let $A \leq B$ hold iff $\vdash A \to B$ and let $A \otimes B$ be $\neg(A \to \neg B)$, and $A^{\perp}$ be $\neg A$. Clearly $\varnothing \otimes B = \neg(\varnothing \to \neg B) = \neg\neg B = B$, and from the axioms one can show that $(T, \leq, \otimes, \perp, \varnothing)$ is a semi-algebraic structure.

Define

$$\begin{aligned} h(A,q) = 1 &\text{ iff } \vdash A \to q \\ g(A,q) = 1 &\text{ iff } \vdash q \to A. \end{aligned}$$

Completeness follows from the following lemma below ■

Lemma 10.9.8 *For any B, A*

$$\begin{aligned} \|A\|^h_B = 1 &\text{ iff } \vdash B \to A \\ \|A\|^g_B &\text{ iff } \vdash A \to B. \end{aligned}$$

Proof. By induction on A.

1. For atomic A this holds by definition.

2. Assume $\|C \to D\|^h_B = 1$ then by the induction hypothesis since $\|C\|^h_C = 1$ we have, $\|D\|^h_{B\otimes C} = 1$ therefore $\vdash B \otimes C \to D$ ie $\vdash B \to (C \to D)$.

   Assume $\vdash B \to (C \to D)$. Then $\vdash (B \otimes C) \to D$, then by the induction hypothesis $\|D\|^h_{B\otimes C} = 1$. Let x be such that $\|C\|^h_x = 1$. Then by the induction hypothesis $\vdash x \to C$. Hence $\vdash (B \otimes x) \to (B \otimes C)$ and so $\|D\|^h_{B\otimes x} = 1$.

3. Assume $\|C \to D\|^g_B = 1$ then for some $x, y \vdash \neg B \to (x \otimes y)$ and $\|C\|^h_x = 1$ and $\|D\|^g_{\neg y} = 1$ iff for some x, y

   $$\begin{aligned} &\vdash (x \to \neg y) \to B \\ &\vdash x \to C \\ &\vdash D \to \neg y. \end{aligned}$$

   We need to show

   $$\vdash (C \to D) \to B$$

   however the last two assumptions imply

   $$\vdash (C \to D) \to (x \to \neg y)$$

   which yields $\vdash (C \to D) \to B)$.

   Assume $\vdash (C \to D) \to B$. Let $x = C, y = \neg D$. Then $\neg B \leq x \otimes y$ and $\vdash x \to C$ and $\vdash D \to \neg y$ which yields by the induction hypothesis

   $$\|C \to D\|^g_B = 1.$$

4. The case of $\neg$ follows from the rule

   $$\vdash A \to \neg B \text{ iff } \vdash B \to \neg A.$$

■

Theorem 10.9.9 *Let* $\mathbf{CL}(\otimes, !)$ *be the system with both* $\otimes$ *and* $!$*, with both groups of axioms for* $\otimes$ *and for* $!$*, as given in 10.2.1. Then* $\mathbf{CL}(\otimes, !)$ *is complete for the* $\mathbf{CL}(\otimes)$ *semi-algebraic structures, with the added semantic condition for ! given below*

$$\|!A\|_t^h = 1 \textit{ iff for all } x, \|A\|_{t\otimes x}^h = 1.$$

Proof. One can verify that lemma 10.9.2 still holds.
Soundness
Easy to check the axioms for ! are valid.
Completeness
Construct the canonical model as in the completeness proof of $\mathbf{CL}(\otimes)$. We show that $\|A\|_t^h = 1$ iff $\vdash t \to A$, for any t, A by induction. We have to verify the case of $!$, as follows:

Assume $\vdash t \to !A$

We want to show $\|!A\|_t^h = 1$.

We have to show $\|A\|_{t\otimes x}^h = 1$ for any x. Thus by the induction hypothesis $\vdash t \otimes x \to A$ or $\vdash t \to (x \to A)$.

However, $\vdash !A \to A$
and $\vdash !A \to (x \to !A)$
hence $\vdash !A \to (x \to A)$
and since $\vdash t \to !A$
we get $\vdash t \to (x \to A)$.

Assume $\|!A\|_t^h = 1$. Hence for all $x, \vdash t \to (x \to A)$. Especially for atomic x not appearing in t or A. We thus conclude $\vdash t \to !A$. This completes the proof. ■

10.10 Metalevel features via connectives

This section describes how one can implement metalevel features of a logical system via special additional connectives. Consider the resource logic **BR1** of 9.2.8. The system is based on $\to$ only. Its basic characterising feature as a resource logic is that assumptions cannot be used in proofs more than once. The inference rule is modus ponens. Its labelling system proposes a discipline for propagating the labels through modus ponens. Let us write the two rules:

Modus Ponens

$$\begin{array}{l} \alpha : A \\ \underline{\beta : A \to B} \\ \beta \otimes \alpha : B \end{array}$$

$\to$ Introduction

To show $t : A \to B$, assume $x : A$ and show $t \otimes x : B$.

Let us now adopt the policy that a wff is labelled (named) by other wffs. We want to do the labelling from within the logic itself. To achieve this we add a binary connective $\otimes$ to the language and understand $(A_1 \otimes A_2)$ as the formula indicating that the label of A_2 is A_1.

According to this reading the two rules above become:

Modus Ponens for $\otimes$

$$\begin{array}{l} \alpha \otimes A \\ \underline{\beta \otimes (A \to B)} \\ (\beta \otimes \alpha) \otimes B \end{array}$$

$\rightarrow$ Introduction for $\otimes$

$$[t \otimes (A \rightarrow B)] \leftrightarrow [(A \otimes A) \rightarrow ((t \otimes A) \otimes B)]$$

Here we used A to label itself.

We can define a Hilbert type logic $\mathbf{L}_{\otimes,\rightarrow}$ with the above connectives and the above two rules taken as axioms with modus ponens as the inference rule. This logic has the same standing as any other logic (eg intuitionistic logic with $\rightarrow$ and $\vee$). The difference is in the axioms we adopt for $\otimes$. It is hoped that for any labelling discipline for a logic *LDS* with $\rightarrow$, there exist the corresponding axioms for the logic $\mathbf{L}$ with $\{\rightarrow, \otimes\}$ and a translation $\mathbf{T}$ such that the following theorem holds for all A:

$$\mathit{LDS} \vdash A \text{ iff } \mathbf{L} \vdash \mathbf{T}A$$

We should make it clear that $\rightarrow$ of $\mathbf{L}$ is classical or intuitionistic. It is not the same as the $\rightarrow$ of the *LDS*, which could be relevant, linear or any other weak system. The weakening effect on the (classical or intuitionistic) $\rightarrow$ of $\mathbf{L}$ is achieved via the presence of the $\otimes$ connective. Axioms on the $\otimes$ serve to fine tune the restrictions on the $\rightarrow$ of $\mathbf{L}$.

To see an example, consider the formula $A \rightarrow A$. We know $A \rightarrow A$ is a theorem of any *LDS* system. We also know that in *LDS* a theorem must be derived with label $\varnothing$. Let us allow the constant $\varnothing$ in the language $\mathbf{L}_{\otimes,\rightarrow}$, and extend $\mathbf{L}$ with the axioms

$$((\varnothing \otimes A) \otimes B) \leftrightarrow ((A \otimes \varnothing) \otimes B) \leftrightarrow (A \otimes B)$$

Define the mapping $\mathbf{T}$ from *LDS* into $\mathbf{L}$ by

$$\mathbf{T}A = (\varnothing \otimes A)$$

This means that we expect that

$$\mathit{LDS} \vdash A \text{ iff } \mathbf{L} \vdash \varnothing \otimes A$$

We therefore try:

Example 10.10.1 *Let us check whether $\mathbf{L} \vdash (\varnothing \otimes (A \rightarrow A))$. The proof is as follows:*

1. $(A \otimes A) \rightarrow (A \otimes A)$
   (Can be assumed as an axiom of $\mathbf{L}$.)

2. $(A \otimes A) \rightarrow ((\varnothing \otimes A) \otimes A)$
   From (1) and the fact that $\vdash ((\varnothing \otimes A) \otimes A) \leftrightarrow (\varnothing \otimes A)$

3. $(\varnothing \otimes (A \rightarrow A))$
   By the rule $\rightarrow$ Introduction for $\otimes$.

Example 10.10.2 *Let us check the provability of $A \rightarrow (B \rightarrow A)$. We will try to find out what axioms to add to $\mathbf{L}$ to allow for a successful derivation of $\varnothing \otimes (A \rightarrow (B \rightarrow A))$.*

Try the axiom $\mathbf{L} \vdash (A \otimes B) \rightarrow (A \otimes X) \otimes B$

1. $(A \otimes A) \rightarrow (A \otimes B) \otimes A$ *axiom*

2. $(A \otimes A) \rightarrow [(B \otimes B) \rightarrow (A \otimes B) \otimes A]$ *from 1*

3. $A \otimes A) \rightarrow (A \otimes (B \rightarrow A))$ *from 2*

4. $\varnothing \otimes (A \rightarrow (B \rightarrow A))$ *from 3.*

Lemma 10.10.3

$$t \otimes (A_1 \to \ldots \to (A_n \to B) \ldots)$$

is equivalent in **L** *to:*

$$= (\bigwedge_{i=1}^{n} (A_i \otimes A_i) \to (\ldots (t \otimes A_1) \otimes \ldots \otimes A_n) \otimes B)$$

Proof. By induction on n.
Case $n = 1$

$$t \otimes (A \to B) \leftrightarrow [A \otimes A \to (t \otimes A) \otimes B]$$

by an axiom
Case $n \otimes 1$

$$t \otimes (A_1 \to \ldots \to (A_n \to (A_{n \otimes 1} \to B) \ldots)$$

$$\leftrightarrow \bigwedge_{i=1}^{n} (A_i \otimes A_i) \to (\ldots (t \otimes A_1) \otimes \ldots \otimes A_n) \otimes (A_n \to B))$$

Let $y = (\ldots (t \otimes A_1) \otimes \ldots \otimes A_n)$
then

$$y \otimes (A_n \to B) \leftrightarrow (A_n \otimes A_n) \to ((y \otimes A_n) \otimes B)$$

hence we get by substitution that the original formula is equivalent to

$$\bigwedge_{i=1}^{n} (A_i \otimes A_i) \to ((y \otimes A_n) \otimes B)$$

which is the desired result. ■

Example 10.10.4 *Let us check the axiom* $(A \to B) \to ((C \to A) \to (C \to B))$.
Consider:

$$\varnothing \otimes ((A \to B) \to ((C \to A) \to (C \to B))$$

by the lemma is equivalent in **L** *to:*

$$((A \to B) \otimes (A \to B)) \wedge ((C \to A) \otimes (C \to A)) \wedge (C \otimes C) \to (((A \to B) \otimes C \to A)) \otimes C) \otimes B$$

Let us prove this formula in **L**. *Assume the antecedent and show the consquent. We thus assume*

1. $(A \to B) \otimes (A \to B)$
2. $(C \to A) \otimes (C \to A)$
3. $C \otimes C$

and want to prove

$$(((A \to B) \otimes (C \to A)) \otimes C) \otimes B$$

4. *2 is equivalent to:*
   $(C \otimes C) \to ((C \to A) \otimes C) \otimes A$
5. *From 3 and 4 we get:*
   $((C \to A) \otimes C) \otimes A$
6. *From 1 and 5, by modus ponens for* $\otimes$ *we get*
   $((A \to B) \otimes ((C \to A) \otimes C)) \otimes B$

To obtain the desired result, we need an axiom of associativity, namely:

$$((A \otimes B) \otimes C) \otimes D \leftrightarrow (A \otimes (B \otimes C)) \otimes D$$

Example 10.10.5 *Try the axiom $(A \to (B \to C)) \to (B \to (A \to C))$. We claim the correspoiding $\otimes$ axiom for it is a form of semi-commutativity (mixed with associativity), namely:*

$$((A \otimes B) \otimes C) \otimes D \leftrightarrow ((A \otimes C) \otimes B) \otimes D.$$

We want to show:

$$\mathbf{L} \vdash \varnothing \otimes [(A \to (B \to C)) \to (B \to (A \to C))]$$

By Lemma 10.10.3 it is sufficient to show:

$$\mathbf{L} \vdash ((A \to (B \to C)) \otimes (A \to (B \to C)) \wedge (B \otimes B) \wedge (A \otimes A) \to$$

$$(((A \to (B \to C)) \otimes B) \otimes A) \otimes C.$$

We assume the antecedent and show the consequent:
We also have that:

$$(A \to (B \to C)) \otimes (A \to (B \to C))$$

is equivalent to:

$$(A \otimes A) \wedge (B \otimes B) \to (((A \to (B \to C)) \otimes A) \otimes B) \otimes C.$$

Since $(A \otimes A) \wedge (B \otimes B)$ are part of the antecedent which we have assumed, we get the following:

$$(((A \to (B \to C)) \otimes A) \otimes B) \otimes C.$$

By semi-commutativity we get the desired consequent:

Example 10.10.6 *Consider*

$$(A \to (A \to B)) \to (A \to B)$$

Let us check what is needed to have

$$\mathbf{L} \vdash \varnothing \otimes ((A \to (A \to B)) \to (A \to B))$$

To show that it is sufficient to assume

$$((A \to (A \to B)) \otimes (A \to (A \to B))) \wedge (A \otimes A)$$

and prove in **L***:*

$$((A \to (A \to B)) \otimes A) \otimes B$$

However,

$$(A \to (A \to B)) \otimes (A \to (A \to B))$$

is equivalent to:

$$(A \otimes A) \wedge (A \otimes A) \to (((A \to (A \to B)) \otimes A) \otimes A) \otimes B$$

We can prove the consequent provided we have the axiom

$$((A \otimes A) \otimes X) \leftrightarrow (A \otimes X)$$

To illustrate the correspondence between *LDS* and the Hilbert system with teh additional $\otimes$, we formulate and prove the following theorem:

Theorem 10.10.7 *Consider the resource concatenation logic* **CL** *with the axioms*

1. $A \to A$

2. $(A \to B) \to ((C \to A) \to (C \to B))$

and the rules of modus ponens and right transitivity, namely

$$\frac{A, A \to B}{B}$$

and

$$\frac{A \to B}{(B \to C) \to (A \to C)}$$

Consider the logic $\mathbf{L_1}$ *in the language of* $\otimes, \varnothing$, *and* $\to$ *and the following axioms for* $\otimes$, *besides all classical tautologies for* $\to$*:*

1. $((\varnothing \otimes A) \otimes B) \leftrightarrow (A \otimes B) \leftrightarrow ((A \otimes \varnothing) \otimes B)$

2. $(A \otimes D) \wedge (B \otimes (D \to C)) \to ((B \otimes A) \otimes C)$

3. $(t \otimes (A \to B)) \leftrightarrow ((A \otimes A) \to ((t \otimes A) \otimes B))$

4. $(((A \otimes B) \otimes C) \otimes D) \leftrightarrow ((A \otimes (B \otimes C)) \otimes D)$

Then the following holds for any wff A.

$$\mathbf{CL} \vdash A \textit{ iff } \mathbf{L}_1 \vdash (\varnothing \otimes A).$$

Proof.

1. Assume $\mathbf{CL} \vdash A$. We want to show $\mathbf{L}_1 \vdash \varnothing \otimes A$ we follow the ususal method of showing that the translations of the axioms and rules of **CL** are all provable in $\mathbf{L}_1$. The axioms (1) and (2) were checked in examples 10.10.1 and 10.10.4. Let us check the rule of modus ponens:

   We assume
   $$\mathbf{L}_1 \vdash \varnothing \otimes A$$
   $$\mathbf{L}_1 \vdash \varnothing \otimes (A \to B)$$
   therefore by axiom (2) we get
   $$\mathbf{L}_1 \vdash (\varnothing \otimes \varnothing) \otimes B$$
   which is the desired result since $\varnothing \otimes \varnothing \leftrightarrow \varnothing$.

   We now check the right transitivity rule. We assume

   (a) $\mathbf{L}_1 \vdash \varnothing \otimes (A \to B)$
   and show

   (b) $\mathbf{L}_1 \vdash \varnothing \otimes ((B \to C) \to (A \to C))$.

   (1) is equivalent to
   $$\mathbf{L}_1 \vdash (A \otimes A) \to (A \otimes B)$$
   and (2) is equivalent to
   $$\mathbf{L}_1 \vdash ((B \to C) \otimes (B \to C)) \to ((B \to C) \otimes (A \to C))$$
   which is equivalent to
   $$\mathbf{L}_1 \vdash ((B \to C) \otimes (B \to C)) \wedge (A \otimes A) \to ((B \to C) \otimes A) \otimes C$$
   We now show the last formula.

   Assume

   2.1. $(B \to C) \otimes (B \to C)$

2.2. $A \otimes A$
show

2.3. $((B \to C) \otimes A) \otimes C$

From (1) we get

2.4. $A \otimes B$

From axiom (2) and (2.1) and (2.4) we get the desired (2.3).

2. Assume now that **CL** $\nvdash A$, we want to show that $\mathbf{L}_1 \nvdash \varnothing \otimes A$. We use the semantical interpretation of **CL**, given in definition 10.8.5 and theorem 10.8.6. Since **CL** $\nvdash A$, there exists a model (X, h), such that $h(\varnothing, A) = 0$.

   We use this model to define an assignment g on the language $\mathbf{L}_1$ such that all instances of axioms of $\mathbf{L}_1$ get value 1, while the wff $\varnothing \otimes A$ gets value 0. The language $\mathbf{L}_1$ has the classical connectives and the additional connective $\otimes$ and $\varnothing$. $\otimes$ is not truth functional and so even if we know the truth values of A and B, we do not know the value of $A \otimes B$. We can regard anything of the form $A \otimes B$ as atomic and give it values arbitrarily. Of course if we do that we might not get all instances of the axioms of $\mathbf{L}_1$ getting value 1. We thus need a way of assigning vlaues to all wffs of the form $A \otimes B$ in such a way that all axioms of $\mathbf{L}_1$ get value 1. We can use h for that purpose, except that h relates only to a fragment of the language of $\mathbf{L}_1$ and not to all of it. This is not surprising because the $\otimes$ language is a metalanguage and can say a lot more. At this stage we have to consider the sublangauge $\mathbf{L}^*$ of the langauge of $\{\otimes, \varnothing, \to\}$ defined as the set of all formulas of the form $(A_1 \otimes \ldots \otimes A_n) \otimes B$, where A_i is a wff without $\otimes$ or $\varnothing$. Note that since we have associativity, this is a good definition. For this fragment $\mathbf{L}^*$, we can define for any $B, g((A_1 \otimes \ldots \otimes A_n) \otimes B) = 1$ iff $h((A_1, \ldots, A_n), B) = 1$ in the model (X, h). By the completeness theorem, we know that for any $B, h((A_1, \ldots, A_n), B) = 1$ iff $\mathbf{CL} \vdash A_1 \to (\ldots \to (A_n \to B) \ldots)$.

   We now have to verify that all axioms of $\mathbf{L}_1$ get value 1. To do that we use the translation into **CL**, as follows:

   (a) $g((\varnothing \otimes A) \otimes B) = h(A, B) = g(A \otimes B)$.

   (b) $g((A \otimes D) \wedge (B \otimes (D \to C)) \to ((B \otimes A) \otimes C)) = 1$ iff $h(A, D) = 1$ and $h(B, D \to C) = 1$ imply $h((B, A), C) = 1$
   iff $\vdash_{\mathbf{CL}} A \to D$ and $\vdash_{\mathbf{CL}} B \to (D \to C)$ imply $\vdash_{\mathbf{CL}} B \to (A \to C)$
   which indeed holds by lemma 10.1.13.

   (c) $g(t \otimes (A \to B)) = h(t, A \to B)$
   $g(A \otimes A) = h(A, A) = 1$
   $g((t \otimes A) \otimes B) = h((t, A), B)$
   we thus get that
   $g(t \otimes (A \to B) = 1$ iff $\vdash_{\mathbf{CL}} t \to (A \to B)$
   iff $g(t \otimes A, B) = 1$
   iff $g(A \otimes A \to (t \otimes A) \otimes B) = 1$.

   (d) $g(((A \otimes B) \otimes C \otimes D) = g((A \otimes (B \otimes C)) \otimes D)$
   by definition, as we relied on associativity and defined $g((A_1 \otimes \ldots \otimes A_n) \otimes B)$.

   We can now prove the other direction of our theorem. If **CL** $\nvdash A$, then in the canonical model $(X, h), h(\varnothing, A) = 0$ and hence $g(\varnothing \otimes A) = 0$ and hence $\mathbf{L}_1 \nvdash \varnothing \otimes A$.

   The assignment g in the proof of 10.9.7 was defined for only the fragment $\mathbf{L}^*$ of the language $\mathbf{L}_1$ with $\otimes$. It would be nice to be able to extend it to the whole language. The extension also depends on what meaning we want to give to the other wffs, the ones not in the fragment, eg formulas like $(A \otimes B) \to C$ and $A \to (B \otimes C)$. Such an extension is given in example 10.10.10 below.

The best way to approach the problem is to consider the model $(\otimes(X, h)$ we have been dealing with. This model gives values to formulas with $\to$ only. If we can give a semantic condition to

$$h((A_1, \ldots, A_n), B \otimes C),$$

then $\otimes$ will have a meaning. The only role for $B \otimes C$ we have so far is in the context of $(B \otimes C) \otimes D$, were we read it as $(B \otimes C)$ is the label of D and semantically translate it into the model (X, h) as $h((B, C), D) = 1$.

Thus $B \otimes C$ is supposed to be a label. We therefore try the following definition.

■

Definition 10.10.8 *$h((A_1, \ldots, A_n), B \otimes C) = 1$ iff def for some $1 \leq k \leq n, h((A_1, \ldots, A_k), B) = 1$ and $h((A_{k+1}, \ldots, A_n), C) = 1$.*

We thus read $B \otimes C$ as saying "I am the local label $(A_1, \ldots, A_n)$".

Thus the axiom for $\otimes$ would be the following (written informally in a "metalanguage"):

$$t \otimes (A \otimes B) \Leftrightarrow \exists s_1, s_2[s_1 \otimes s_2 = t \text{ and } (s_1 \otimes A) \text{ and } (s_2 \otimes B)]$$

Lemma 10.10.9 *For $\otimes$ defined semantically:*

$$h(t, A \otimes B) = 1 \textit{ iff def } \exists x \exists y[x \otimes y = t \textit{ and } h(x, A) = 1 \textit{ and } h(y, B) = 1]$$

The following holds:

1. $h(\varnothing, A \otimes B) \to C) = h(\varnothing, A \to (B \to C))$
2. $h(\varnothing, A \to (B \to A \otimes B)) = 1$

Proof.

1. To show (1), we proceed as follows:
   $h(\varnothing, (A \otimes B) \to C) = 1$ iff
   $\forall x[h(x, A \otimes B) = 1 \Rightarrow h(x, C) = 1$
   iff $\forall x[\exists x_1 x_2(h(x_1, A) = 1$ and $h(x_2, B) = 1$
   and $x = x_1 \otimes x_2) \Rightarrow h(x, C) = 1]$
   iff $\forall x \forall x_1 \forall x_2[h(x_1, A) = 1$ and $h(x_2, B) = 1$
   and $x = x_1 \otimes x_2 \Rightarrow h(x, C) = 1]$
   iff $\forall x_1 x_2[h(x_1, A) = 1 \Rightarrow [h(x_2, B) = 1 \Rightarrow h(x_1 \otimes x_2, C) = 1]]$
   iff $\forall x_1[h(x_1, A) = 1 \Rightarrow \forall x_2[h(x_2, B) = 1 \Rightarrow h(x_2 \otimes x_2, C) = 1]]$
   iff $\forall x_1[h(x_1, A) = 1 \Rightarrow h(x_1, B \to C) = 1]$
   iff $h(\varnothing, A \to (B \to C)) = 1$

2. To show (2) we check
   $h(\varnothing, A \to (B \to (A \otimes B))) = 1$
   iff $\forall x \forall y[h(x, A) = 1$ and $h(y, B) = 1 \Rightarrow h(x \otimes y, A \otimes B) = 1$
   iff $\exists x \exists y[h(x, A) = 1$ and $h(y, B) = 1 \Rightarrow h(x \otimes y, A \otimes B) = 1$
   iff *truth* because of the definition of $h(t, A \otimes B)$.

■

We are now in a position to semantically define a logic **CL**$\otimes$, which is the extension of the logic **CL** to the language with $\to$ and $\otimes$. The truth table for $\otimes$ is given as in the preceding lemma. The theorems of **CL**$\otimes$ are all wffs A such that for all models $(X, h), h(\varnothing, A) = 1$. Clearly **CL**$\otimes$ is a *conservative* extension of **CL**, since they are both complete for the same semantics.

Example 10.10.10 *We are now ready to extend the assignment g of the metalanguage $\mathbf{L}_{\{\varnothing,\otimes,\to\}}$ from the fragment suggested in the proof of 10.9.7 to the entire language.*

$$g^*(A \otimes B) = 1 \textit{ iff } \mathbf{CL}\otimes \vdash A \to B.$$

For wffs inthe fragement, namely of the form $(A_1 \otimes \ldots \otimes A_n) \otimes B$, where A_i, B contain no $\otimes$, we have
$\mathbf{CL}\otimes \vdash (A_1 \otimes \ldots \otimes A_n) \to B$
iff $\mathbf{CL}\otimes \vdash A_1 \to \ldots (A_n \to B) \ldots)$
iff $h((A_1, \ldots, A_n), B) = 1$
iff $g(A \otimes B) = 1$.

We still do not have a Hilbert axiomatisation for **CL**$\otimes$. This is our next task.

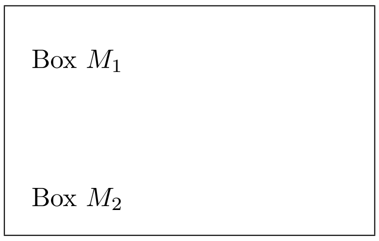
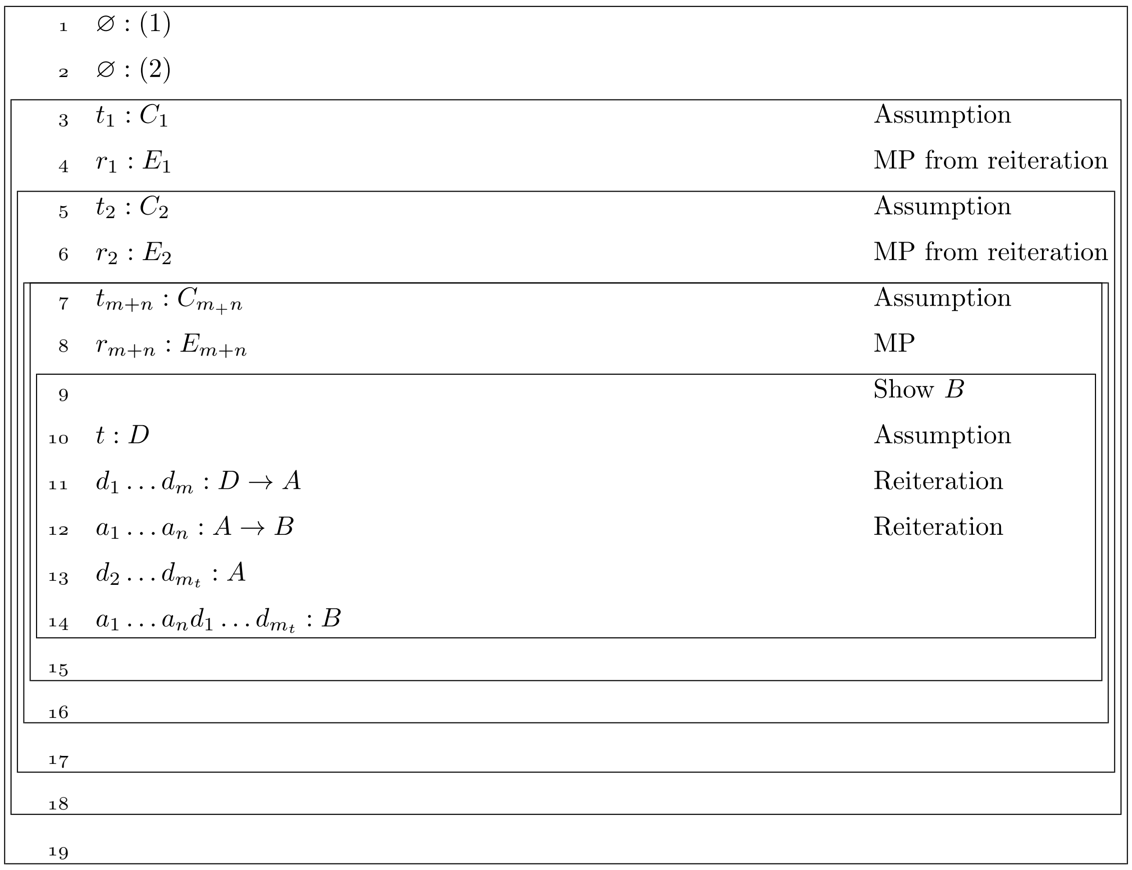


Figure 10.13:



1	$\varnothing : (1)$	
2	$\varnothing : (2)$	
3	$t_1 : C_1$	Assumption
4	$r_1 : E_1$	MP from reiteration
5	$t_2 : C_2$	Assumption
6	$r_2 : E_2$	MP from reiteration
7	$t_{m+n} : C_{m+n}$	Assumption
8	$r_{m+n} : E_{m+n}$	MP
9		Show B
10	$t : D$	Assumption
11	$d_1 \dots d_m : D \to A$	Reiteration
12	$a_1 \dots a_n : A \to B$	Reiteration
13	$d_2 \dots d_{m_t} : A$	
14	$a_1 \dots a_n d_1 \dots d_{m_t} : B$	
15		
16		
17		
18		
19		

Figure 10.14:

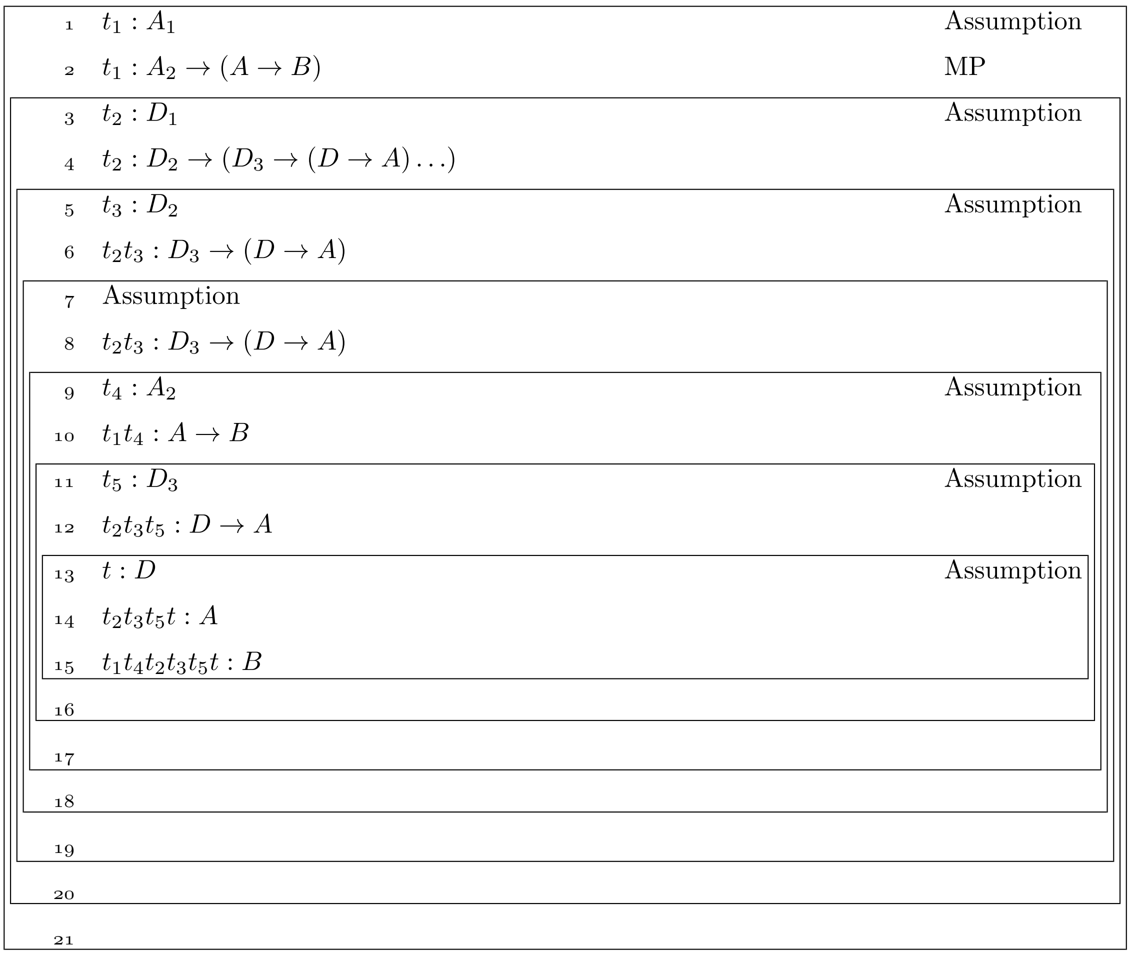


Figure 10.15:

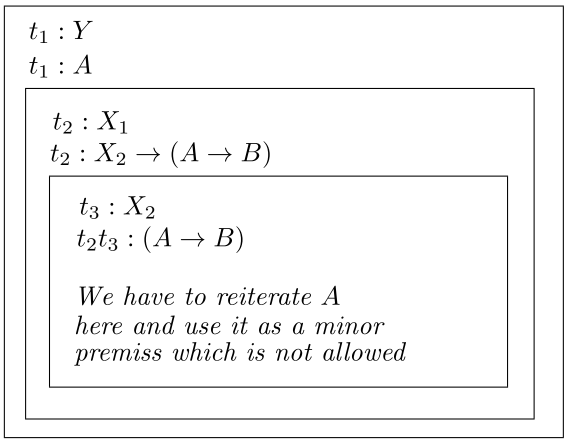

$t_1 : Y$
$t_1 : A$
$t_2 : X_1$
$t_2 : X_2 \to (A \to B)$
$t_3 : X_2$
$t_2t_3 : (A \to B)$
We have to reiterate A here and use it as a minor premiss which is not allowed

Figure 10.16:

1		show $B \to A$
2	$a : A$	Assumption
3	$B \to A$	from Box
4		show A
5	$b : B$	Assumption
6	$a : A$	reiteration
7		

Figure 10.17:

	1		show $X \to (A \to B)$
	2	$a : A \to (X \to B)$	Assumption
	3	$a : X \to (A \to B)$	from Box
	4		show $A \to B$
	5	$b : X$	Assumption
	6	$ab : A \to B$	from Box
	7		Show B
	8	$c : A$	Assumption
	9	$b : X$	reiteration
	10	$a : A \to (X \to B)$	reiteration
	11	$ac : X \to B$	MP
	12	$acb : B$	MP
exit	13	$ab : A \to B$	
exit	14	$a : X \to (A \to B)$	
exit	15	$\varnothing : (A \to (X \to B)) \to (X \to (A \to B))$	

Figure 10.18:

	1		show $(A \to B) \to (A \to B)$
	2	$a : B \to A$	Assumption
	3	$a : (A \to B) \to (A \to B)$	from Box
	4		show $A \to B$
	5	$b : A \to B$	Assumption
	6	$ab : A \to B$	from Box
	7		show B
	8	$c : A$	Assumption
	9	$b : A \to B$	reiteration
	10	$bc : B$	MP
	11	$B \to A$ cannot be reiterated	
	12	*because in* $H, A \nvdash B \to A$	
exit	13	$b : A \to B$	
exit	14	$\varnothing : (A \to B) \to (A \to B)$	

Figure 10.19:

Axioms	**Condition**
W	as in theorem 10.3.4
(4)	allow the use of assumptions more than once. (Replace MP1 by MP)
C(axiom 8)	allow Restart
6β	allow for a box for $\beta \to B$ of the form Show B $a : \beta$ Assumption $\alpha : B$ exit $\beta \to B$ even though the label a is not in α provided β satisfies the corresponding condition of 6β
7γ	allow to reiterate a γ provided it satisfies condition 7γ
4m,n for $m + n \leq$k together with 5k,1	use any assumption at most k times (ie MPk) $\dfrac{\begin{array}{c}\alpha : A \\ \beta : A \to B\end{array}}{\alpha \cup \beta : B}$ provided $\alpha \cup \beta$ does not contain any $X, \ldots, X$, k times, α, β multisets

Figure 10.20:

	1		Show B
	2	$a : C$	assumption
	3	$t_2 s_3 : C \to B$	
	4	$t_2 s_3 a : B$	
exit	5	$t_2 s_3 : C \to B$	

Figure 10.21:

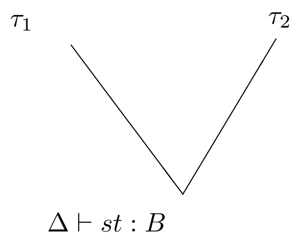


Figure 10.22:

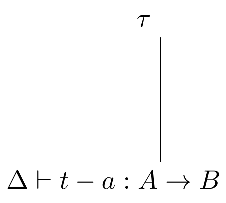


Figure 10.23:

Chapter 11

Many Valued Logics

11.1 Introduction

This Chapter shows that many valued logics fit comfortably within the Labelled Deductive Systems framework. The basic idea of *LDS* is that formulas come labelled in the form $t : A$. A database is a set of such labelled formulas and the basic consequence relation has the form:

$$\{t_i : A_i\} \vdash s : B$$

In the implicational fragment, to show

$$t : A \to B$$

we open a box, assume $x : A$, prove using our metabox discipline $y : B$ and we are able to exit with the conclusion $t : A \to B$ provided t, x, y satisfy a suitable relationship ψ characteristic to the logic in question.

If we write the box condition in a semi formalised form, it would be the following:

$A \to B$ can be derived with label t iff for all labels x ranging over a certain set L_1 there exists a label y such that $x : A$ can prove $y : B$ and t, x, y satisfy the required conditions.

In symbols

$$t : A \to B \text{ iff } \forall x \in L_i \exists y (x : A \text{ “proves” } y : B \text{ and } \varphi(t, x, y) \text{ holds})$$

What is important here is the combination of quantifiers “$\forall x \exists y$”.

In the case of resource logics $\forall x$ ranged over new atomic labels and y was requried to satisfy $y = t + x$. In practice we skolemised and chose an arbitrary label x for $x : A$ and showed $t + x : B$.

Thus we can see now the familiar form from previous chapters:

In modal logic we had the same universal quantifier on labels. To show $t : \Box A$ we show $s : A$ for every s such that tRs. The precise definition depends on the exact labelling discipline. For

	1	$x : A$	Assumption
	2	$\vdots$	x a new label
	3	$\vdots$	
	4	$\vdots$	
	5	$tx : B$	
exit	6	$t : A \to B$	

Figure 11.1:

modal **K** to show $\Box A$ in the middle of a proof one could take an arbitrary label x, assume tRx, open a box, bring into the box the appropriate set of reiterations, namely $\{x : B \mid t : \Box B$ has already been proved$\}$, and show $x : A$.

It turns out that the quantifier combination for showing $A \to B$ in many valued logics is exactly the same as in the case of resource or modal logics. In many valued logics atomic formulas get truth values from some set. The more complex formulas get their value according to a prescribed truth table. Some values are considered designated and the theorems of logic are those formulas which get designated values under all assignments of values to the atoms.

Example 11.1.1 (Łukasiewicz Infinite Valued Logic Ł$_\infty$) *The truth values are all rationals (or equivalently one can take all reals) in the interval* $[0, 1]$. *1 is the designated value. The truth table for* $\neg$ *and* $\to$ *are as follows:*

A	B	$\neg A$	$A \to B$
x	y	$1 - x$	$\min(1, 1 - x + y)$

Łukasiewicz $n + 1$ valued logic $\mathbf{Ł_n}$ has values $\{0, \frac{1}{n}, \frac{2}{n}, \dots, 1\}$ with the same truth table as the infinte case.

The following formulas are many valued tautologies in the logic $\mathbf{Ł}_\infty$ (ie they get the value 1 under all assignments of values to the atoms).

L_0	$A \to A$
L_1	$A \to (B \to A)$
$L_{2.1}$	$(A \to B) \to ((B \to C) \to (A \to C))$
$L_{2.2}$	$(A \to B) \to ((C \to A) \to (C \to B))$
L_3	$(A \to (B \to C) \to (B \to (A \to C))$
L_4	$((A \to B) \to B) \to ((B \to A) \to A)$
L_5	$((A \to B) \to (B \to A)) \to (B \to A)$
L_6	$(\neg B \to \neg A) \to (A \to B)$

The rule of modus ponens is also valid.

If A is a tautology and $A \to B$ ia a tautology, then B is a tautology.

Further new tautologies arise when we look at Łukasiewicz n-valued logics. For example:

$$M_2 : ((((A \to B) \to A) \to A) \to (B \to C)) \to (B \to C)$$

is a tautology of Ł$_2$.

It is known that axioms $L_0 - L_6$ together with modus ponens axiomatise all the tautologies of Ł$_\infty$ (Rose-Rosser 1957). In fact, the subset of $\{L_1, L_{2.1}, L_4, L_6\}$ proves the other axioms. Axiom M_2 was put forward by A Avron 88 as an addition to the axioms of Ł$_\infty$ to axiomatise Ł$_2$. Ł$_2$ was originally axiomatised by M Wajsberg 1931. A Rose 1956a put foward the following axiom R_n to be added to the axioms of Ł$_\infty$ to axiomatise Ł$_n$.

$$R_n : ((A^n \to B) \to A) \to A$$

The above example will be used to show how may valued logics can be perceived as a labelled deductive system. The label t of $t : A \to B$ will be the truth value of $A \to B$.

Thus we want to have:
Show $t : A \to B$ if box

We must therefore propose a proof discipline for the box.

The idea of regarding truth values as labels is similar to an idea of Scott 1974, also occurring to Urquhart 1974, where truth values were converted into valuations. Our approach is a bit different as will be seen from the sequel.

We are now faced with the problem of how to characterise many valued logics in general and Łukasiewicz $\mathbf{Ł}_\infty$ and $\mathbf{Ł_n}$ in particular as a metabox system. We have the advantage that the set of tautologies is already axiomatised and we have a (many valued) semantics for the logic which we

	1	$x : A$	Assumption
	2	$\vdots$	
	3	proof steps	
	4	$\vdots$	
	5	$y : B$	provided a condition essentially saying
	6		$t = \ \min(1, 1 - x + y)$
exit	7	$t : A \to B$	

Figure 11.2:

can use. The disadvantage is that our labelling systems for $\to$ are all based on some form of the dedution theorem (to show $t : A \to B$ assume $x : A$ and prove $y : B$) while in Łukasiewicz logics the deduction theorem is known to fail!

My answer is that the deduction theorem does not fail. In fact everything fits beautifuly. We will give the intuitive explanations in this section and give the mathematics in the next section.

Let us observe the axiom system for the implicational fragment of $\mathbf{Ł}_\infty$ in 11.1.1. Axioms $L_0 - L_3$ are nothing but the axioms of the resource logic **BR1**.

Recall from an earlier Chapter that its labelling discipline is known:

1. Labels are multisets. Assumptions get different atomic labels.

2. Modus ponens propagates labels as follows:

$$\begin{array}{ll} \alpha : A & \\ \beta : A \to B & \\ \hline \alpha \cup \beta : B & \text{provided } \alpha \cap \beta = \varnothing \end{array}$$

   The condition $\alpha \cap \beta = \varnothing$ stops one from using the same assumption more than once.

3. To show $\gamma : A \to B$ assume $x : A, x$ a new label and show $\beta : B$ provided $\gamma = \beta - \{x\}$. We do *not* require $x \in \beta$, ie there is no need to use the assumption A.

Another way of looking at **BR1** is that it is like linear logic, where assumptions can be used at most once but where we do *not* require that *all* assumptions be used.

Further recall that the system **BR1** is complete for the semantics of the form (S, h) where S is a multiset and h is an assignment giving truth values $h(X, q) \in \{0, 1\}$ for atoms q and multisets $X \subseteq S$. We assume $X \subseteq Y \to h(X, q) \leq h(Y, q)$. The truth condition for $\to$ is:

$$h(X, A \to B) = 1 \text{ iff } \forall Y (h(Y, A)) = 1 \to h(X \cup Y, B) = 1$$

we have $\mathbf{BR1} \vdash A$ iff $\forall h(h(\varnothing, A) = 1)$.

To get the implicational Łukasiewicz logic Ł$_\infty$ we need to add to **BR1** axioms L_4 and L_5. In fact, Rose has shown that the axiom $R_n : ((A^n \to B) \to A) \to A$ the implicational fragment of $\mathbf{Ł_{n-1}}$ when added to $L_0, \dots, L_5$.

This was proved by Rose, [Rose, 1956b, Rose, 1956a]. It was not known at the time whether L_5 is redundant and whether it can be proved from the other axioms in the pure implicational fragment (ie without the use of L_6). J Barkley Rosser [Barkley-Rosser, 1960] reports a communication from A R Turquette that L_5 is independent. I have not seen a published proof, however, the following example shows that L_5 does not follow from the other implicational axioms even in Łukasiewicz 3 valued logic.

Example 11.1.2 *Consider the following matrix for $\to$ with values $\{1, X, a, b\}$ with 1 designated as follows (see figure 11.3):*

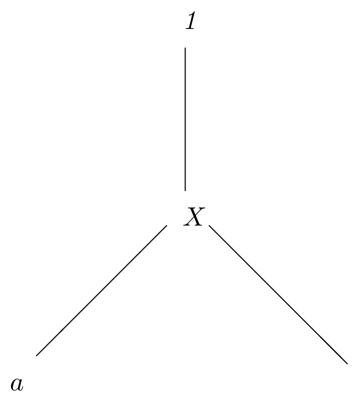


Figure 11.3:

	Table for $\to$:	
A	B	$A \to B$
1	*1*	*1*
X	*1*	*1*
a	*1*	*1*
b	*1*	*1*
1	X	X
X	X	*1*
a	X	*1*
b	X	*1*
1	a	a
X	a	X
a	a	*1*
b	a	X
1	b	b
X	b	X
a	b	X
b	b	*1*

Axioms L_0 to L_4 and R_n always get value 1 in this table. Also if A always gets 1 and $A \to B$ always gets value 1 so does B.

Axiom L_5 gets value X if A is assigned a and B is assigned b.

These matrices, called non-linear MV-matrices, are studies in 11.3 below:

We thus have to figure out the metabox discipline conditions which correspond to axioms L_5 and L_6. These are not difficult to find. The formula $(A \to B) \to B$ corresponds in classical logic to disjunction $A \vee B$. Thus axiom L_5 says $(A \vee B) \to (B \vee A)$. Suppose we treat $(A \to B) \to B$ as disjunction in our metabox discipline, would that be equivalent to adding axiom L_5 to the corresponding Hilbert system?

The answer is yes.

The rule for disjunction is well known.

1. $A \to C$

2. $B \to C$
3. $\dfrac{A \vee B}{C}$

Thus our metabox rule would be:
To show $\alpha : C$: we show (for some *wisely chosen* A and B) the following:

1. $\alpha_1 : A \to C$
2. $\alpha_2 : A \to C$
3. $\alpha_3 : (A \to B) \to B$
   (or $\alpha_3 : (B \to A) \to A$)

and conclude $\alpha : C$ provided

$$\alpha = \alpha_1 \cup \alpha_2 \cup \alpha_3.$$

A more practical form of the rule is: To show $\alpha : A \to B$
Show instead both $\alpha_1 : A \to (A \to B)$ and $\alpha_2 : (A \to B) \to B$ provided $\alpha = \alpha_1 \cup \alpha_2$.

Example 11.1.3 *Show* $\varnothing : ((A \to B) \to B) \to ((B \to A) \to A)$ *(fig 11.4)*

	1		Show $a_1 : (B \to A) \to A$
	2	$a_1 : (A \to B) \to B$	assumption
	3	$a_1 : (B \to A) \to A$	from Box
	4		Show $a_1a_2 : A$
	5	$a_2 : B \to A$	assumption
	6	use disjunction rule	
	7	$\varnothing : A \to A$	
	8	$a_2 : B \to A$	
	9	$a_1 : (A \to B) \to B$	
	10	$a_1a_2 : A$	
exit	11	$a_1 : (B \to A) \to A$	
exit	12	$((A \to B) \to B) \to ((B \to A) \to A)$	

Figure 11.4:

Negation can be added the usual way, by adding a symbol $\perp$ with the understanding that $\perp$ implies anything, ie $\varnothing : \perp \to A$ is always available.

Example 11.1.4 *To show* $(\neg A \to \neg B) \to (B \to A)$, *(fig 11.5)*

The reader may wonder about our notion of $A_1 \ldots, A_n \vdash B$ and what does it mean in terms of the many valued truth tables.

In Scott 1974, the truth value meaning of $A_1, \ldots A_n \vdash_{\text{Scott}} B$ for Łukasiewicz logic is "For any many valued assignments h if all $h(A_i) = 1$ then also $h(B) = 1$."

Our metabox notion of $\vdash_{\text{LDS}}$ corresponds to the following:
"For any assignment h we have $\Sigma_i h(A_i) \leq h(B)$."

Thus we can have the deduction theorem holding in some form. It also makes sense in terms of Ł$_\infty$ being a resource logic because the "numerical resource weights" $h(A_i)$ we give to the assumption A_i are added and they should not "exceed" the "allocation" $h(B)$ given to the proposed conclusion B.

	1		show $a_1 : B \to A$
	2	$a_1 : (A \to \bot) \to (B \to \bot)$	assumption
	3	$a_1 : B \to A$	from Box
	4		Show $a_1 a_2 : A$
	5	$a_2 : B$	
	6	$\varnothing : ((A \to \bot) \to (B \to \bot)) \to (B \to ((A \to \bot) \to \bot)$	
	7	$a_1 : B \to ((A \to \bot) \to \bot)$	
	8	$a_1 a_2 : (A \to \bot) \to \bot$	
	9	$\varnothing : ((A \to \bot) \to \bot) \to ((\bot \to A) \to A)$	
	10	$a_1 a_2 : (\bot \to A) \to A$	
	11	$\varnothing : \bot \to A$	
	12	$a_1 a_2 : A$	
exit	13	$a_1 : B \to A$	
exit	14	$\varnothing : ((A \to \bot) \to (B \to \bot)) \to (B \to A)$	

Figure 11.5:

11.2 Introducing Łukasiewicz many valued logics

Our starting point is the basic resource logic **W** of Chapter 3. Its axioms are:
$A \to A$
$(A \to B) \to ((B \to C) \to (A \to C))$
$(B \to C) \to ((A \to B) \to (A \to C))$

Given any logic **L**, we are interested in self reflection for this logic. Let $\varphi(x, q_i)$ be an implicational formula with propositional variables x and q_i (q_i are regarded as parameters). Let us assume that $\varphi(x_1, q_i)$ is monotonic in x in classical logic, either monotonic increasing or monotonic decreasing. This means that in classical loigc we have either

1. $\vdash (x \to y) \to (\varphi(x, q) \to \varphi(y, q))$ or we have

2. $\vdash (x \to y) \to (\varphi(y, q) \to \varphi(x, q))$

We are interested in the smallest extension $\mathbf{L}_\varphi$ of **L** such that the above monotonicity condition (increasing or decreasing) holds in $\mathbf{L}_\varphi$. Depending on what φ is, $\mathbf{L}_\varphi$ may be stronger or equal to **L**.

The above equation makes sense if we understand $A \to B$ functionally, as a type taking objects of type A into objects of type B. For example, the Curry-Howard functional interpretation for intuitionistic logic reads $\to$ in this way. We get $\vdash A \to B$ iff the type $A \to B$ is non empty. A monotonic φ can then be understood as another functional from types to types and the axiom

$$\vdash (A \to B) \to (\varphi(A) \to \varphi(B))$$

can be understood as the requirement of the commutativity of the diagram below: for increasing φ (see fig 11.6)

for decreasing φ we have the diagram fig 11.7

The axioms of prefixing and suffixing can be understood as φ extensions $\mathbf{L}_\varphi$ of the logical system **L** with reflexivity ($\vdash A \to A$) as the only axiom where φ is any one of the formulas:

$$\begin{tikzcd}\varphi(A) \arrow[r] & \varphi(B) \\ A \arrow[u] \arrow[r] & B \arrow[u]\end{tikzcd}$$

Figure 11.6:

$$\begin{tikzcd}\varphi(A) & \varphi(B) \arrow[l] \\ A \arrow[u] \arrow[r] & B \arrow[u]\end{tikzcd}$$

Figure 11.7:

$$\begin{aligned}\varphi_1(A,q) &= A \to q \\ \varphi_2(A,q) &= q \to A\end{aligned}$$

$\mathbf{L}_{\varphi_1}$ yields the axiom

$$(A \to B) \to ((B \to q) \to (A \to q))$$

and $\mathbf{L}_{\varphi_2}$ yields the axiom

$$(A \to B) \to ((q \to A) \to (q \to B))$$

We now turn to **BR1** and consider the formula $\varphi(A,q) = (q \to A) \to A$ and consider $\mathbf{L}_\varphi$. We have the following axiom for $\mathbf{L}_\varphi$

$$\vdash (A \to B) \to (((q \to A) \to A) \to ((q \to B) \to B))$$

hence by commutativity

$$\vdash (q \to B) \to (((q \to A) \to A) \to ((A \to B) \to B))$$

substitute $q = B$ and get

$$\vdash ((B \to A) \to A) \to ((A \to B) \to B)$$

which is axiom L_4 of $\mathbf{Ł}_\infty$.

Conversely, from L_4 we can get $\mathbf{L}_\varphi$ as follows:

$$\vdash (A \to B) \to ((B \to q) \to ((A \to q)))$$

$$\vdash ((B \to q) \to ((A \to q)) \to ((A \to q) \to q) \to ((B \to q) \to q))$$

hence

$$\vdash (A \to B) \to (((A \to q) \to q) \to ((B \to q) \to q))$$

and from L_4 we get

$$\vdash (A \to B) \to (((q \to A) \to A) \to ((q \to B) \to B)).$$

To be continued

11.3 Non linear Łukasiewicz many valued logics

11.4 Finite many valued logics

Chapter 12

Extending the Curry-Howard Interpretation to LDS Systems

12.1 Introduction

This chapter extends the Curry-Howard interpretation to logics weaker and stronger than intuitionistic logic. The idea of the Curry-Howard interpretation is very simple. A formula of the implicational fragment of logic is regarded as a type. The atoms are the basic types and any formula $A \to B$ is read as the type of functions from A to B (ie B^A). If we allow ourselves the machinery of λ-calculus in defining terms via λ-abstraction, then some types will have definable terms in them; for example, the identity function $\lambda x^A.x^A$ is a definable element of the type $A \to A$. It turns out that the set of formulas A for which it can be demonstrated that there exists a definable function of type A is exactly the set of all intuitionistic theorems. The set of definable terms is closed under modus ponens because if $t^A \in A$ and $t^{A\to B}(t^A) \in B$. Clearly, if we label every formula A by a term t^A of type A, ie write $t^A : A$, then the usual modus ponens propagation of labels namely

$$\begin{array}{c}\alpha : A \\ \beta : A \to B \\ \hline \beta + \alpha : B\end{array}$$

corresponds to the natural application of the functional terms.

Conversely, the label of $A \to B$ is t iff $\forall x$[if x is the label of A then there is a derivation of B with label $t + x$].

If we are regarding A as a type and x as a term of that type and a derivation propagates terms via functional applications, then $t + x$ would be $t(x)$, where t is a functional built up during the proof of B from A. t is definable now as $t = \lambda x t(x)$.

Thus we see that exiting a metabox in *LDS* terminology corresponds to λ-abstraction in the Curry Howard methodology. The next step to take is to realise that since restrictions on exiting a box and on the derivations within the box give rise to weaker logics, such as linear or relevance logics or other resource logics, we must also be able to obtain weaker logics via restricting λ-abstraction. We can thus hope to give a Curry-Howard like interpretation to other resource logics, weaker than intuitionistic logic.

What are our options in restricting λ-abstraction? The restrictions must be natural. The λ-calculus is a powerful language and can be used to simulate or characterise any recursive logic. We do not want to hack a λ-term for any given logic and artifically restrict λ-abstraction by means of that term, to yield that logic. We want natural conditions. so let us see what we have when we write $\lambda x t(x_1, \ldots, x_n)$

1. x may not be free in t. Do we allow abstraction in this case?
2. x may have more than one occurrence in t. eg $t(x, x)$. Do we abstract $\lambda x t(x, x)$ (ie get rid of all occurrences or do we abstract one?).

One occurrence only, ie rename and abstract

$$\lambda x' t(x', x)$$

Another dimension is the order of abstraction. The varaibles x_i are typed. Some are typed higher than others. Do we say abstract the lower types first and then the higher types? What is the order we use?

The reader will note that these are natural questions to ask and therefore give rise to natural restrictions.

We shall see that such natural restrictions give rise to the same resource logics of previous chapters. Furthermore, we should see a correspondence between λ-restrictions and axioms.

The above discussion motivated restrictions on λ abstraction, *restricting* the supply of definable terms and thus getting *less* types A demonstrably non-empty. Less non-empty types means less theorems which means a weaker logic. How about stronger logics, intermediate logics between intutionistic and classical logic?

What mechanism do we have for characterising them? Obviously we need to increase the stock of definable (or existing) terms so that more types can be shown non-empty, ie more theorems are available. We do not want to just throw in (stipulate) existence of functionals, in an ad-hoc manner just to obtain the intermediate logic we want. We should put forward some reasonable principles, the kind natural to λ-calculus functional environment, and show that adopting them yields corresponding logics.

We chose one simple principle, that of completing a functional diagram. If φ is a *monotonic* increasing functional on types, then one can ask the question can we complete the following diagram

$$\begin{tikzcd}
\varphi(A) \arrow[r, "?"] & \varphi(B) \\
A \arrow[u] \arrow[r] & B \arrow[u]
\end{tikzcd}$$

Figure 12.1:

being able to complete such diagrams requires existence of terms. Our mechanism for getting more terms (in order to get more logics) is to postulate the completions of diagrams.

Take for example

$$\varphi(x) = (C \to x) \to x.$$

C is a parameter. Postulating the existence functionals completing the diagram above yields classical logic.

The two methods may be combined. We can restrict λ-abstraction and at the same time postulate the completion of some diagrams. This will give us new logics, which are neither weaker nor stronger than intuitionistic logic. The following figure summarises the situation (se fig 12.2):

12.2 Formulas as types

The idea of reading a formula as a type originates with Curry [Curry, 1934] and is used to give a λ-calculus interpretation of an intuitionistic theorem. A formula of intuitionistic implicational logic is a theorem if and only if, when read as a type, it can be shown to be non-empty using the rules of term-construction, namely *abstraction* and *application.* By varying the natural abstraction principles available in the λ-calculus, we are able to extend the point of view of formulae-as-types to some weak systems of implication (relevance, linear, etc.) as well as to systems which are stronger than intuitionistic.

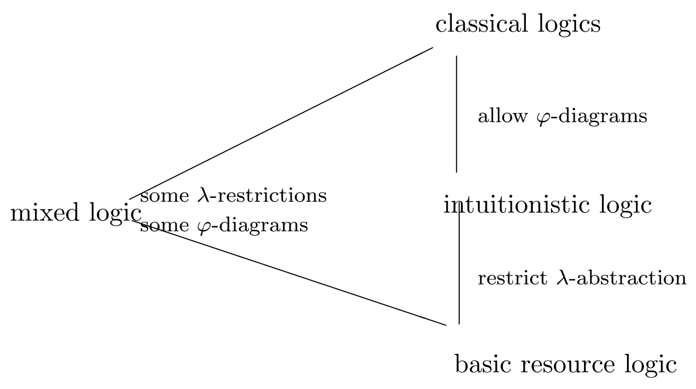


Figure 12.2:

Let us then take a standard definition of the terms and operators needed to obtain a λ-calculus, and let us examine the abstraction rule more closely.

Definition 12.2.1 *1.* Lambda terms *are words over the following alphabet:*

$v_0, v_1, \ldots$	*variables,*
λ	*abstractor,*
$(,)$	*parentheses.*

2. The set of λ-terms is defined inductively as follows:

(a) $x \in \Lambda$*;*

(b) $M \in \Lambda \to (\lambda x.M) \in \Lambda$*;*

(c) $M, N \in \Lambda \to (MN) \in \Lambda$*;*

where x in (a) or (b) is an arbitrary variable.

Note that there are many hidden assumptions in the case (2) of the definition of λ-terms, in particular λ-abstraction terms, eg:

1. M may have *no* free occurrence of x:
   (a) M is an open term, but contains no free occurrence of x;
   (b) M is a closed term, thus contains no free variable at all;
2. M may have *one* free occurrence of x:
   (a) M may be of the form '(Tx)' (or '`APPLY`(T, x)');
   (b) M may be of the form '(xT)' (or '`APPLY`(x, T)');
3. M may have *more than one* free occurrence of x:
   (a) the λ-*abstraction* may cancel *exactly one* of the free occurrences of x;
   (b) the λ-*abstraction* may cancel *all* free occurrences of x;

Moreover, in (3), where *application* is being defined (which can be done by juxtaposition as in '(MN)', or by an explicit non-canonical operator '`APPLY`(M, N)' in the terminology used here in this paper), 'M' is assumed to be of 'higher' level than 'N': 'M' is supposed to be the 'course-of-value' of a function, while 'N' is assumed to be the argument.

Now, by working with some of these hidden assumptions one can use the simple typed λ-calculus together with the Curry-Howard-Tait interpretation to formalise a number of systems of implication, as we shall demonstrate below.

We read the first rule of $\to$-*introduction*, namely:

$$\dfrac{\begin{array}{c}[x \in A]\\ b(x) \in B\end{array}}{\lambda x.b(x) \in A \to B}$$

as follows: having made the assumption '$x \in A$', and arriving at the conclusion '$b(x) \in B$' by means of one of the rules available, then we can discharge the assumption by making a λ-abstraction of the assumption-term ('x') over the conclusion-term ('$b(x)$'). In other words, when constructing a proof-tree one can discharge an assumption if there is at least one proof step between the assumption and the conclusion where the assumption is discharged. So, in the construction of a proof of '$\lambda x.x \in A \to A$', as we shall see below, we need at least *reflexivity* in order to arrive at a conclusion of the form '$x \in A$' from the assumption '$[x \in A]$'.

Example 12.2.2 $A \to A$ *(reflexivity)*

$$\dfrac{\dfrac{\dfrac{[x \in A]}{x = x \in A}}{x \in A}}{\lambda x.x \in A \to A}$$

and we have the 'identity' construction, which corresponds to combinator 'I*'* $\equiv \lambda x.x$ *[Curry and Feys, 1958, Hindley and Seldin, 1986].*

Example 12.2.3 $(A \to B) \to ((C \to A) \to (C \to B))$ *(left transitivity)*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[z \in C] \qquad [y \in C \to A]}{\texttt{APPLY}(y,z) \in A} \qquad [x \in A \to B]}{\texttt{APPLY}(x,\texttt{APPLY}(y,z)) \in B}}{\lambda z.\texttt{APPLY}(x,\texttt{APPLY}(y,z)) \in C \to B}}{\lambda y.\lambda z.\texttt{APPLY}(x,\texttt{APPLY}(y,z)) \in (C \to A) \to (C \to B)}}{\lambda x.\lambda y.\lambda z.\texttt{APPLY}(x,\texttt{APPLY}(y,z)) \in (A \to B) \to ((C \to A) \to (C \to B))}$$

whose resulting closed term corresponds to combinator 'B*'* $\equiv \lambda x.\lambda y.\lambda z.\texttt{APPLY}(x,\texttt{APPLY}(y,z))$ *[Curry and Feys, 1958, Hindley and Seldin, 1986].*

Example 12.2.4 $(A \to B) \to ((B \to C) \to (A \to C))$ *(right transitivity)*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[z \in A] \qquad [x \in A \to B]}{\texttt{APPLY}(x,z) \in B} \qquad [y \in B \to C]}{\texttt{APPLY}(y,\texttt{APPLY}(x,z)) \in C}}{\lambda z.\texttt{APPLY}(y,\texttt{APPLY}(x,z)) \in A \to C}}{\lambda y.\lambda z.\texttt{APPLY}(y,\texttt{APPLY}(x,z)) \in (B \to C) \to (A \to C)}}{\lambda x.\lambda y.\lambda z.\texttt{APPLY}(y,\texttt{APPLY}(x,z)) \in (A \to B) \to ((B \to C) \to (A \to C))}$$

whose resulting closed term corresponds to a combinator which results from applying combinator 'C*' to combinator '*B*', or what Curry has called combinator '*B'*' in [Curry and Feys, 1958], and in [Curry, 1963]: '*B'*'* $\equiv \lambda x.\lambda y.\lambda z.\texttt{APPLY}(y,\texttt{APPLY}(x,z))$ *('*CB*' in [Hindley and Seldin, 1986]).*

In order to invalidate the derivation above one would have to impose the restriction on the λ-abstraction rule such that the abstractions have to occur in the order 'from higher to lower subterms'.

Example 12.2.5 $(A \to (B \to C)) \to (B \to (A \to C))$ *(permutation)*

$$\dfrac{\dfrac{\dfrac{\dfrac{[y \in B] \qquad \dfrac{[z \in A] \qquad [x \in A \to (B \to C)]}{\texttt{APPLY}(x,z) \in B \to C}}{\texttt{APPLY}(\texttt{APPLY}(x,z),y) \in C}}{\lambda z.\texttt{APPLY}(\texttt{APPLY}(x,z),y) \in A \to C}}{\lambda y.\lambda z.\texttt{APPLY}(\texttt{APPLY}(x,z),y) \in B \to (A \to C)}}{\lambda x.\lambda y.\lambda z.\texttt{APPLY}(\texttt{APPLY}(x,z),y) \in (A \to (B \to C)) \to (B \to (A \to C))}$$

whose resulting closed term corresponds to combinator 'C*'* $\equiv \lambda x.\lambda y.\lambda z.\mathtt{APPLY}(\mathtt{APPLY}(x,z),y)$ *([Curry and Feys, 1958, Hindley and Seldin, 1986]).*

In order to invalidate the derivation above one would have to impose the restriction on the λ-abstraction *rule such that the abstractions have to occur in the order 'from higher to lower subterms' within the order 'from inner to outer subterms'.*

Example 12.2.6 $(A \to (A \to B)) \to (A \to B)$ *(*contraction*)*

$$\dfrac{\dfrac{\dfrac{\boxed{[y \in A]} \qquad \dfrac{\boxed{[y \in A]} \qquad [x \in A \to (A \to B)]}{\mathtt{APPLY}(x,y) \in A \to B}}{\mathtt{APPLY}(\mathtt{APPLY}(x,y),y) \in B}}{\boxed{\lambda y.}\mathtt{APPLY}(\mathtt{APPLY}(x,y),y) \in A \to B}}{\lambda x.\lambda y.\mathtt{APPLY}(\mathtt{APPLY}(x,y),y) \in (A \to (A \to B)) \to (A \to B)}$$

whose resulting closed term corresponds to combinator 'W*'* $\equiv \lambda x.\lambda y.\mathtt{APPLY}(\mathtt{APPLY}(x,y),y)$ *([Curry and Feys, 1958, Hindley and Seldin, 1986]).*

The assumption '$\boxed{[y \in A]}$*' is used twice and in a nested way. So, the restriction one has to impose here is rather obvious: a* λ*-abstraction will cancel one free occurrence of the variable at a time.*

Example 12.2.7 $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ *(*distribution*)*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\boxed{[z \in A]} \qquad [x \in A \to (B \to C)]}{\mathtt{APPLY}(x,z) \in B \to C} \qquad \dfrac{\boxed{[z \in A]} \qquad [y \in A \to B]}{\mathtt{APPLY}(y,z) \in B}}{\mathtt{APPLY}(\mathtt{APPLY}(x,z),\mathtt{APPLY}(y,z)) \in C}}{\boxed{\lambda z.}\mathtt{APPLY}(\mathtt{APPLY}(x,z),\mathtt{APPLY}(y,z)) \in A \to C}}{\lambda y.\lambda z.\mathtt{APPLY}(\mathtt{APPLY}(x,z),\mathtt{APPLY}(y,z)) \in (A \to B) \to (A \to C)}}{\lambda x.\lambda y.\lambda z.\mathtt{APPLY}(\mathtt{APPLY}(x,z),\mathtt{APPLY}(y,z)) \in (A \to (B \to C)) \to ((A \to B) \to (A \to C))}$$

whose resulting closed term corresponds to combinator 'S*'* $\equiv \lambda x.\lambda y.\lambda z.\mathtt{APPLY}(\mathtt{APPLY}(x,z),\mathtt{APPLY}(y,z))$ *([Curry and Feys, 1958, Hindley and Seldin, 1986]).*

Note that the assumption '$\boxed{z \in A}$*' is used twice, and both occurrences are discharged in one single abstraction '*$\boxed{\lambda z.}$*'. To obtain a linear implication one has to restrict the discharging abstraction to one occurrence of the assumption only. In other words, each discharge affects only one (linear) path in the proof-tree, instead of affecting all branching occurrences like in the proof above.*

Example 12.2.8 $(A \to B) \to ((A \to (B \to C)) \to (A \to C))$ *(*distribution on the minor*)*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\boxed{[z \in A]} \qquad [x \in A \to B]}{\mathtt{APPLY}(x,z) \in B} \qquad \dfrac{\boxed{[z \in A]} \qquad [y \in A \to (B \to C)]}{\mathtt{APPLY}(y,z) \in B \to C}}{\mathtt{APPLY}(\mathtt{APPLY}(y,z),\mathtt{APPLY}(x,z)) \in C}}{\boxed{\lambda z.}\mathtt{APPLY}(\mathtt{APPLY}(y,z),\mathtt{APPLY}(x,z)) \in A \to C}}{\lambda y.\lambda z.\mathtt{APPLY}(\mathtt{APPLY}(y,z),\mathtt{APPLY}(x,z)) \in (A \to (B \to C)) \to (A \to C)}}{\lambda x.\lambda y.\lambda z.\mathtt{APPLY}(\mathtt{APPLY}(y,z),\mathtt{APPLY}(x,z)) \in (A \to B) \to ((A \to (B \to C)) \to (A \to C))}$$

whose resulting closed term corresponds to a variation of the 'S*', precisely:*
'SC*'* $\equiv \lambda x.\lambda y.\lambda z.\mathtt{APPLY}(\mathtt{APPLY}(y,z),\mathtt{APPLY}(x,z))$*.*

Observe that the same remarks as to the 'non-linearity' of the discharge/abstraction made for the previous case also applies for the case here.

Example 12.2.9 $A \to (B \to A)$

Now we want to build a derivation of the above axiom and show where it can be invalidated by the appropriate side condition. By imposing the condition that the abstraction can only be made when there is indeed at least one free occurrence of the variable being abstracted from the expression ('$b(x)$' in the case below), one can obtain a 'relevant' abstraction:
$\to$-introduction

$$\dfrac{\begin{array}{c}[x \in A]\\ b(x) \in B\end{array}}{\lambda x.b(x) \in A \to B}$$

The proof-tree is constructed as follows:

$$\dfrac{\dfrac{\dfrac{\boxed{[y \in B]}}{[x \in A]}}{\boxed{\lambda y.x} \in B \to A}}{\lambda x.\lambda y.x \in A \to (B \to A)}$$

whose resulting closed term corresponds to combinator 'K' $\equiv \lambda x.\lambda y.x$ (Curry & Feys 1958, p. 153; Hindley & Seldin 1986, p. 191).

Note that the discharge/abstraction of the assumption '$\boxed{[y \in B]}$' is made over the expression 'x' in '$\boxed{\lambda y.x}$', which prevents it from being considered 'relevant', given that the expression 'x' does not contain any free occurrence of 'y'. Such a 'non-relevant' discharge/abstraction is called 'vacuous discharge' in [Hindley and Seldin, 1986].

So, the restricted λ-abstraction to be adopted in order to invalidate the derivation above is exactly the relevant abstraction, i.e., there must be at least one free occurrence of the variable in the term on which the abstraction is operating. In logical terms, what the relevant abstraction does is to allow us to ensure that within the Curry-Howard-Tait interpretation we can have a "systematic handle on relevance in the sense of logical dependence" ([Anderson and Belnap, 1975]). The idea is that for 'A' to be relevant to 'B' it must be *necessary* to use 'A' in the deduction of 'B' from 'A'. At first sight it may look as though the restriction on λ-abstraction is not by itself sufficient, given that a proof of '$(A \to B) \to ((B \to A) \to (A \to B))$' can be constructed using relevant λ-abstractions only, as in eg:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[z \in A] \qquad [x \in A \to B]}{\mathtt{APPLY}(x,z) \in B} \qquad [y \in B \to A]}{\mathtt{APPLY}(y,\mathtt{APPLY}(x,z)) \in A}\,(*) \qquad [x \in A \to B]}{\mathtt{APPLY}(x,\mathtt{APPLY}(y,\mathtt{APPLY}(x,z))) \in B}\,(*)}{\lambda z.\mathtt{APPLY}(x,\mathtt{APPLY}(y,\mathtt{APPLY}(x,z))) \in A \to B}}{\lambda y.\lambda z.\mathtt{APPLY}(x,\mathtt{APPLY}(y,\mathtt{APPLY}(x,z))) \in (B \to A) \to (A \to B)}}{\lambda x.\lambda y.\lambda z.\mathtt{APPLY}(x,\mathtt{APPLY}(y,\mathtt{APPLY}(x,z))) \in (A \to B) \to ((B \to A) \to (A \to B))}$$

However, it is easy to show that the steps marked with '$(*)$' were unnecessary and could be eliminated, and the '$\lambda y.$'-abstraction would have to be made over the term obtained prior to those steps, namely '$\mathtt{APPLY}(x,z)$'. A 'minimal' proof of '$(A \to B) \to ((B \to A) \to (A \to B))$' would have to involve at least one non-relevant abstraction, such as e.g. in:

$$\dfrac{\dfrac{\dfrac{[z \in A] \qquad \dfrac{[y \in \mathbf{B} \to A]}{[x \in A \to B]}}{\mathtt{APPLY}(x,z) \in B}}{\lambda z.\mathtt{APPLY}(x,z) \in A \to B}}{\dfrac{\lambda y.\lambda z.\mathtt{APPLY}(x,z) \in (B \to A) \to (A \to B)}{\lambda x.\lambda y.\lambda z.\mathtt{APPLY}(x,z) \in (A \to B) \to ((B \to A) \to (A \to B))}}$$

where the $\lambda y.$-abstraction is a non-relevant abstraction.

As pointed out in Lambek [Lambek, 1989, p. 234], in his *The Calculi of Lambda-Conversion* Church already distinguished the relevant from the non-relevant λ-*abstraction*. See, e.g.:

"If M does not contain the variable x (as a free variable), then $(\lambda x M)$ might be used to denote a function whose value is constant and equal to (the thing denoted by) M, and whose range of arguments consists of all things. This usage is contemplated below in connection with the calculi of λ-K-conversion, but is excluded from the calculi of λ-conversion and λ-δ-conversion – for technical reasons which will appear."

(Church 1941, pp. 6–7.)

12.3 Correspondence between terms and proofs

12.4 Curry Howard interpretation for general LDS systems

Part IV

Labelled Modal and Temporal Systems

Chapter 13

Labelled Modal and Temporal Logics

13.1 Introduction

In resource logics, the labelling was essentially for the purpose of resource management. To prove $\alpha : A$ meant that A was obtained or assumed using the assumptions named in α. This chapter deals with logics where the label has a different meaning. To prove $\alpha : A$ means A is true or is associated with the possible world α. A database is, as before, a set of labelled formulas. A database Δ can prove a labelled formula $t : A$. To understand the intuitive meaning behind these concepts take the following example:

Example 13.1.1 *There is signal trouble in the Camden Town underground station. Liverpool Street underground station is flooded. Is this going to disrupt the schedule at Kings Cross station? Here the database is labelled:*
Data
c: Signal Trouble
l: Flooded
Query
k: Delays in schedule
Technically this is no different from:
Data

$a :$	A
$b :$	$A \to B$

Query

$ab :$	B

In our case the labels signify location, not resource management.

We need to present a view of modal and temporal logics, in which the labels make intrinsic sense. The reaser may try and guess that since modal and temporal logics have a natural possible worlds semantics, we are viewing the worlds as labels and writing $\alpha : A$ to mean A holds in the world α. This is technically correct, but has the conceptual standing of a mere mathematical device, desgined to conform modal and temporal logics into our *LDS* framework. I should like to claim more. I should like to put forward that the primary conceputal framework for modal (and temporal) logics is *LDS*. The possible world semantics for these logics just happens to be compatible with the *LDS* approach, through the device of taking the worlds as labels. To be successful in putting forward this point of view, we have to motivate and identify modal and temporal logic from within the *LDS* framework, using considerations that are intrinsically natural to the *LDS* point of view. This we now proceed to achieve.

Our starting point is the notion of a logical system presented in Chapter 1. Recall that a logical system to us is a pair, $(\vdash, \mathbf{S}_\vdash)$, where $\vdash$ is a consequence relation and $\mathbf{S}_\vdash$ is an algorithmic system (recursively enumerable) for generating the pairs (Δ, Q) such that $\Delta \vdash Q$ holds, where Δ is a finite set of wffs and Q is a wff. We refer to Δ as the database, Q as the query and $\mathbf{S}_\vdash$ is referred to as the algorithmic system for $\vdash$. Thus for example according to our view, the classical logic consequence relation $\vdash$ with $\mathbf{S}_1$ being a resolution based automated system and $\mathbf{S}_2$ being a Gentzen or a Hilbert system give rise to two different logics, $(\vdash, \mathbf{S}_1)$ and $(\vdash, \mathbf{S}_2)$.

$\vdash$ can be presented mathematically in some manner. It is just a set of pairs $\{(\Delta, Q)\}$ satisfying the axioms of consequence relation, namely, Revlexivity, Monotonicity and Cut.

We regard the problem of whether $\Delta \vdash Q$, denoted by $\Delta \vdash ?Q$, as a problem of querying a set of data. Suppose we are given a set of assumptions Δ written in some formal language $\mathbf{L}$. This set represents our data concerning some aspect of an application. We may have queries from the data, represented in (probably the same) language $\mathbf{L}'$. The basic (non-temporal) logical problem associated with this situation is:
Does the query follow from the data,
or in symbols:
$\Delta \vdash ?Q$.
A logical system $(\vdash, \mathbf{S}_\vdash)$ gives us the pairs (Δ, Q) such that the answer is yes. There are two mainstream types of logics involved. The monotonic ones and the non-monotonic ones. The difference has to do with the properties of the consequence relation $\vdash$. A monotonic logic would satisfy the three rules of consequence relation given above. The non-monotonic logics do not necessarily satisfy monotonicity or cut.

The conceptual difference between the two is that in the monotonic case $\Delta \vdash A$ means that A gets an answer yes from a part of Δ and it does not matter what the rest of Δ is.

Thus $\Delta \vdash A$ iff for some minimal $\Delta_0 \subseteq \Delta, \Delta_0 \vdash A$ and so we also have $\Delta \cup \Delta' \vdash A$, because the answer really depends only on Δ_0.

In the non-monotonic case the answer depends on the entire set Δ. Equivalently in the non monotonic case the answer may be obtained from some $\Delta_0 \subseteq \Delta$, but it *does* matter what else is in Δ besides Δ_0. Thus if more data is added eg we have a new set, $\Delta \cup \Delta'$, the answer may not continue to be yes.

Example 13.1.2 *1. Consider the database Δ*

$$(a \to b) \to a$$

$$c$$

and the query $?a$
Whether or not $\Delta \vdash a$ depends on the underlying logic. If it is classical logic the answer is yes. For intuitionistic logic the answer is no.

2. Consider the same database with the query $?\neg b$. In classsical and intuitionistic logic we do not get the answer yes. If the logic is non-monotonic with the closed world assumption the answer would be yes! (The closed world assumption is the understanding that any atom which is not a head of a clause is negated by the database.)

Note that so far we have no temporal dimension involved. We have only a database Δ and a logic (a system for getting answers from Δ). The logic may be one of a great variety available in the literature. There is classical logic, intuitionistic logic, relevance logic, inheritance systems, circumscription, Horn Clause logic, etc. We have listed a mixed variety of systems because any one of them will do for getting answers from data.

Each logic has two features, the first is the representation and query language, and the second is the deductive system (possibly non-monotonic) for getting answers from data. In the above list, classical, intuitionistic and relevant logics are all based on the same language but represent different answering mechanisms. Circumscription is still based on the same language but yields answers in a totally different way and an inheritance system is completely different even in its representation.

We now proceed to widen our horizon and consider several related databases. Denote them by $\{(\Delta_t, \vdash_t) \mid t \in T\}$. Each database Δ_t has its own answering logical system $\vdash_t$. For simplicity let us assume that all the Δ_t are based on the same language and employ the same logic $\vdash_t = \vdash$. We can thus present the system as $\{\Delta_t \mid t \in T\}$ the $\vdash$ being implicit. The area of Labelled (eg modal and temporal) logics deals with logical connections between such databases. The queries we want to ask are of the form:

"Which databases give answer yes to Q_i and how are they related."

Thus the next step is pluralistic. We move from one single set of assumptions and logic to a whole related system of them. The added dimension is the study of connections between the databases. Let us consider for example the simplest connection, namely:

If the answer to Q is yes from *one* database, the answer is yes from *all* of them.

We are not able to express such a relationship without a richer possibly external language. The language and logic of Δ_t involves Δ_t alone and is not able to deal with the above relation which is a meta-relation to $\{\Delta_t \mid t \in T\}$. The problem can be further complicated when T, the label set, has its own structure (eg it may be partially ordered by $<$ or may have a binary function $\otimes$ on it e.g. whenever t, s are labels so is $t \otimes s$).

Thus indexed (modal and temporal) logics as an area of research deals with systems of databases $\{\Delta_t \mid t \in T\}$ where the index T has a structure of its own. (We symbolise this fact by writing $(T, <, \otimes)$ as a typical example with one relation symbol and one function symbol.)

The queries either relate to any particular databases $\Delta_t, t \in T$ or to some patterns of relationships between them.

Example 13.1.3 *The following are examples of a system of databases (see fig 13.1)*

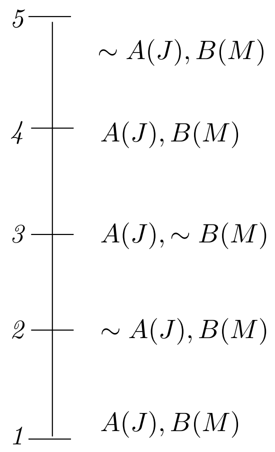


Figure 13.1:

We see a pattern here that whenever $A(J)$ and $B(M)$ are true, $\sim A(J)$ is true next. This we cannot express without special additional language capability.

The reader may still think our presentation and point of view so far is traditional. However, we have already departed from the traditional view and laid the ground for further generalisations, which can address the variety of systems available in the area of logic and computation.

First, under the present conceptual framework, the structure (eg $(T, <, \otimes)$) of labels receives more prominence and is allowed to have (logical) life of its own.

Second, we are under no obligation to disallow a system $(\Delta_t, \vdash_t)$ where $\vdash_t$ are different logics (though it makes some sense to ask that they be based on the same language). This corresponds naturally to metabox disciplines where different metaboxes allow for different derivation rules. The internal logic of a new box is one of the options of our discipline, as discussed in Chapter 3.

Third, the logics $\vdash_t$ need not be monotonic.

Fourth, in the system $(\Delta_t, \vdash_t)$ we may have relationships not only among queries but also among $\vdash_t$. We may have that the logics relate to each other in conformity with the labelling structure $(T, <, \otimes)$. For example, if t represents time, $\vdash_t$ may be a logic evolving with time.

Note that in practical computer science problems which use logic we have occasions, and need to call upon, many combinations of the above features of an indexed system. We defined the notion of a logical system to be a pair $(\vdash, \mathbf{S}_\vdash)$. We therefore must specify how we see the system $\{\Delta_t, \vdash_t\}$ as a logical system. There are many ways of defining the composite system. One simple way is to take the formulas of the composite system to be of the form $t : A$, where $A \in \Delta_t$. The relation $\{t_i : A_i\} \vdash s : B$ is defined somehow using $\{\vdash_t\}$ and the algorithmic system $\mathbf{S}_\vdash$ for the composite consequence relation $\vdash$ is also defined in some way using $\{\mathbf{S}_{\vdash_t}\}$. Thus we do obtain a composite $(\vdash, \mathbf{S}_\vdash)$.

Having described the scope of possibilities with systems of databases $(\Delta_t, \vdash_t)$ we yet have to ask what exactly are our options are with regard to a language describing the relationships between the Δ_t's.

Here there are two extreme possibilities. The external and the internal ones, and of course, many combinations of them. The external view is to have a metalanguage $\mathbf{M}$ which can describe the notion of $\Delta_t \vdash_t A$. $\mathbf{M}$ is a metalanguage, allowing explicit reference to t and to $(T, <, \otimes)$, having names for the elements of Δ_t and, by using its own (ie $\mathbf{M}$'s) connectives and logical means, to describe relationships among the Δ_t's. This approach is studied in its general setting in the metalevel chapter.

The second approach is to enrich the language of each Δ_t with special additional connectives and operators allowing it to relate to its "neighbours" in the structure $(T, <, \otimes)$. This involves a slight change of point of view. Δ_t is no longer a database of data isolated on its own but it will now contain data (in the enriched language) of how it is related to neighbouring $\Delta_s, s \neq t$. So for example, $\Box A \in \Delta_t$ may put in the database at t the information that all other neighbouring databases answer the query $?A$ as yes. A side effect of this change in view is that we cannot any longer take any old family of databases and put them together to form a system.

The experienced modal logician will notice that the system $(T, <, \otimes, \Delta_t, \vdash_t, t \in T)$ is a mixture of syntax and semantics. $(T, <)$ can be regarded as a possible world structure, where at each $t \in T$, we do not have the traditional classical model but a theory Δ_t in the logic $\vdash_t$. This is in itself an innovation, in the context of possible world semantics. It is especially suited for a computational point of view. To find the value of an atom q at a possible world t, we do not consult a mathematical assignment $h(t, q)$ as in the traditional Kripke seantics, but submit the query $?q$ to an algorithmic proof procedure $\Delta_t \vdash ?q$.

The above considerations hsow the place of modal and temporal logics within the framework of *LDS*. It is the study of systems of databases adn their relationship. The primary intrinsic meaning of a lable $\alpha : A$ is a database α and wff A. The modal connectives talk about how queries from different α_i relate to each other. Thus we enrich our language and develop systems of modal and/or temporal logics. How fortunate we are that the natural semantics for these logics is the possible world one which is compatible with the original reasons for introducing the systems.

13.2 H-modal logic

This seciton develops a modal system on a set of labels which is itself a logic. In terms of the discussion of the previous section, we have a labelled system of databases $(\Delta_\alpha, \vdash_\alpha)$, where α are some wffs of some logic $\mathbf{H}$.

This is the most general framework one can consider. $\mathbf{H}$ need not be a logt can be a labelling algebra. The temporal and modal connectives give us the connections between the systems $(\Delta_\alpha, \vdash_\alpha$).

Definition 13.2.1 *Labelled Classical* $\mathbf{H}$*-Modal Logic.*

(a) A labelling logic is any logical system with wffs in some language. We denote these Wffs by α, β, γ

(b) *Let* **H** *be a labelling logic. We define the notions of the* **H***-modal (or temporal) logic* **M***(***H***) by defining its Wffs and terms. We initially consider the connectives* $\neg, \wedge, \vee, \rightarrow, \Box, \Diamond$ *and the quantifiers* $\forall$ *and* $\exists$*. Other possibilities are* $\Rightarrow$ *(strict implication), P, F (the temporal Past, Future connectives) or intuitionistic implication.*

For the case of $\Box, \Diamond$ we adopt the following clauses.

1. Let Q be an n-place predicate and $x_1 \dots x_n$ be variables then $Q(x_1, \dots, x_n)$ is an atomic formula of **M**(**H**) with the free variables $x_1 \dots x_n$.

2. If $A(x_i), B(y_i)$ are Wffs with the indicated free variables so are $A \wedge B, \neg A, A \rightarrow B, A \vee B, \Diamond A, \Box A$, with the corresponding free variables.

3. If $A(y, x_i)$ is a Wff with y, x_i free then $\forall y A, \exists y A$ is a Wff with x_i free.

4. If α is a label and c is a symbol for a constant, then $c(\alpha)$ is a constant term. The intuition behind $c(\alpha)$ is that c is introduced at label α. If α labels a possible world then c was first introduced at world α. $c(\alpha)$ may also appear in another world β.

5. If $A(y, x_1, ..., x_n)$ is a formula with x_i, y free and α is a label then $\eta y(\alpha, A(y, x_i))$ is a term with x_i free and α a parameter. η is an ε-function or a Skolem function. Its meaning is that if $\alpha : \exists y A(y, x_1, \dots, x_n)$ holds at label α, then there is a constant $c(\alpha, x_1, \dots, x_n)$ such that $\alpha : A(c(\alpha, x_1, \dots, x_n), x_1, \dots, x_n)$ holds. The constant c depends of course on $x_1, \dots x_n$ and is introduced at α, hence dependence on α. We use the η-function to denote c.

6. If A is a wff then "A" is a label (ie atomic Wff of the labelling logic). The intuition behind the name "A" of A is the following: There will be many occasions where we need to introduce a new label because of logical considerations having to do with a Wff A. In that case we want "A", the name of A, also to be involved. The simplest example is when $\alpha : \Diamond A$ is true. This means that there exists a possible world β in which $\beta : A$ is true. We might write $\beta = \alpha \wedge$ "A"

Definition 13.2.2 *A database is a set of labelled wffs or terms of the form*

$$\alpha : A$$

$$\alpha : t$$

where A is a Wff and t is a term.

The next definition defines the notion of a logical system for the language of **M***(***H***). The idea is to separate the object language (*$\Box, \Diamond$*) from the metalanguage (***H** *and the labels). Logical rules will give the connection. The labels* α, β *are the possible worlds for* $\Box$ *and* $\Diamond$*. The ordering and general behaviour of the possible worlds is governed by the logic* **H**. **H** *says which world is relatively possible to which world. We write* $\alpha <_{\mathbf{H}} \beta$ *to be* $\vdash_{\mathbf{H}} \beta \rightarrow \alpha$*. Of course this forces* $<_{\mathbf{H}}$ *to be transitive and reflexive. This is not a limitation. The very idea of having a logic "controlling"* $<$ *is a far reaching generalisation. There is scope for* **H** *to interact with the logic* **M***(***H***) itself (***H**= **M***(***H***)?) in various ways. We shall see in the sequel.*

Definition 13.2.3 *The logic* **M***(***H***)*
We need the following conventions:

- **H** *is a labelling logic*
- $\alpha : t_i$ *means the term* t_i *exists in the label* α*. The existence has to be indicated or inferred.*
- $\alpha < \beta$ *is the possible world relation on the labels. It is defined in some way using* **H***, the labelling logic. For example we can have*

$$\alpha < \beta = \textit{ definition } \vdash_{\mathbf{H}} \beta \rightarrow \alpha$$

- *There are problems in reading $\alpha : \Diamond A(t); \Box A(t)$. Does t have to exist at α? Similarly $?\alpha : \Box A(t)$, does that mean t exists in any possible world and $A(t)$ is true there or does it mean $A(t)$ is true in all possible worlds in which t exists?*

We adopt the following axioms and rules for the logic $\mathbf{M}(\mathbf{H})$.

1. $\dfrac{\alpha : \Box A, \alpha < \beta}{\beta : A}$

2. $\dfrac{\beta : A(t_i), \alpha : t_i, \alpha < \beta}{\alpha : \Diamond A(t_i)}$

3a. $\dfrac{\alpha : \exists y A(y, x_i)}{\alpha : A(\eta y(\alpha, A(y, x_i)), x_i)}$

3b. $\dfrac{\alpha : \exists y A(y, x_1, \ldots, x_n)}{\alpha : \eta y(\alpha, A)}$

This axiom says the η Skolem constant exists at α.

4. $\dfrac{\alpha : t, \alpha < \beta}{\beta : t}$

The domains of the possible worlds increase.

5. $\dfrac{\alpha : \forall x A(x), \alpha : t}{\alpha : A(t)}$

6. $\dfrac{\alpha : \Diamond A}{\xi(\alpha, A) : A}$

where $\xi(\alpha, A)$ is a label which depends on α and on A, for example $\alpha\wedge$ "A". Here we adopt the view that if $\Diamond A$ is true at a world α, then A is true in a possible world $\xi(\alpha, A)$. The value of this world depends on the logic $\mathbf{M}(\mathbf{H})$ and is indicated by the definition of ξ.

7a. $\dfrac{\varnothing}{\alpha : A}$

for A a truth functional tautology.

7b. $\dfrac{\alpha : A, \alpha : A \to B}{\alpha : B}$

8. $\dfrac{\alpha : \Box A; \alpha : \Box(A \to B)}{\alpha : \Box B}$

If the language contains strict implication $\Rightarrow$ then we can have the rule

9. $\dfrac{\alpha : A \Rightarrow B, \vdash_{\mathbf{H}} \beta \to \alpha, \beta : A}{\beta : B}$

Strict implication can be defined as $\Box(A \to B)$

Example 13.2.4 *Let $\mathbf{H}$ be a classical logic and let $\{a_i\}$ be atomic labels. Consider a propostional modal language with $\Box$ and $\Diamond$. According to 13.2.1, each a_i is a label, and assume that if A is a formula then "A" is an atomic label. We let $\xi(\alpha, A)$ be the label $\alpha\wedge$ "A".*

Let us reconsider 2.2.3. We show that the data:

1. $\alpha : \Box\Box B$

2. $\beta : \Diamond(B \to C)$

3. $\alpha <_{\mathbf{H}} \beta$

imply the goal

$\alpha : \Diamond\Diamond C$

The following are the proof steps:

4. *From (2) we get:*
   $\beta\wedge$ "$B \to C$" $: B \to C$

5. *From (1) and (3) and the fact that* $\vdash_{\mathbf{H}} \beta\wedge$ "$B \to C$" $\to \beta$ *we get that*
   $\beta\wedge$ "$B \to C$" $: C$.

6. *From 5 we get*
   $\alpha : \Diamond\Diamond C$

The proof is the same as the one in 2.2.4 but it illustrates the notation.

Remark 13.2.5 *There are several important points to remember about 13.2.1 - 13.2.3.*

a *We defined* $\alpha <_{\mathbf{H}} \beta$ *as* $\vdash_{\mathbf{H}} \beta \to \alpha$. *This makes* $<_{\mathbf{H}}$ *transitive and reflexive. For a temporal logic, the transitivity is acceptable, but the reflexivity is not. It is no problem to define a new ordering* $\alpha < \beta$ *iff* $\vdash_{\mathbf{H}} \beta \to \alpha$ *and* $\not\vdash_{\mathbf{H}} \alpha \to \beta$. *This definition will work because in the rule*

$$\frac{\alpha : \Diamond A}{\alpha \wedge \text{"}A\text{"} : A}$$

We do have $\alpha < \alpha\wedge$ "A"

b *Furthermore, notice that we are not committed to* **H** *being a monotonic logic. Thus a variety of* **H** *can be used. This aspect of our formalism is new and interesting because it means that the set of possible worlds is a* logical space *subject to some logical considerations.*

c *We have more freedom with the choice of* $\xi(\alpha, A)$. *If we make* ξ *dependent only on* A *only we get the logic* **S5**, *for example.*

We also have freedom with the naming function $A \mapsto$ "A". *We assumed* "A" *was atomic in the previous example.* "A" *can be taken to be as* A *itself. If we do that the logic* **H** *will make some worlds equal and will thus affect* **M**(**H**).

Example 13.2.6 *Consider a positive intuitionistic predicate language with* $\wedge, \to, \bot, \forall$. *Let* **H** *be intuitionistic logic. Let* "A" *be* A *and let* $\xi(\alpha, A)$ *be* $\alpha \wedge A$. *The rules are:*

1. $\dfrac{\alpha : A \to B, \beta : A, \vdash \beta \to \alpha}{\beta : B}$

2. $\dfrac{\alpha : A, \vdash \beta \to \alpha}{\beta : A}$

3. $\dfrac{\alpha : A \wedge B}{\alpha : A}$

   and

   $\dfrac{\alpha : A, \alpha : B}{\alpha : A \wedge B}$

Consider a configuration ρ *with* $\alpha : A \in \rho$ *iff* $\vdash_{\mathbf{H}} \alpha \to A$, *for* A *atomic. One can show, using the rules of intutionistic logic for the labels* **H** *and the rules (1)-(4) above, that we get for any* A *and* α:

$$\vdash_{\mathbf{H}} \alpha \to A \textit{ iff } \rho \vdash_{\mathbf{M(H)}} \alpha : A$$

4. $\dfrac{\alpha : \forall x A(x)}{\alpha : A(y)}$

and

$$\frac{\alpha : A(x),\ \textit{for } x \textit{ not free in } \alpha}{\alpha : \forall x A(x)}$$

Example 13.2.7 *Consider the logic* **CL** *of 9.1.2 in a language with* $\rightarrow$ *and* $\forall$ *only. Let* **H** *be* **CL** *itself and let the labels be finite sequences of wffs of* **CL**. *Let * be concatenation of sequences. Let "A" be A and let $\xi(\alpha, A)$ be $\alpha * (A)$.*

Consider the following rules:

1. $\dfrac{\alpha : A \rightarrow B, \beta : A}{\alpha * \beta : B}$

2. $\dfrac{\alpha : \forall x A(x)}{\alpha : A(y)}$

and

$$\frac{\alpha : A(x), x \textit{ not free in } \alpha}{\alpha : \forall x A(x)}$$

Consider a configuration of ρ with $\alpha : A \in \rho$ iff

$$\vdash_{\mathbf{CL}} \alpha_1 \rightarrow (\alpha_2 \rightarrow \ldots \rightarrow (\alpha_n \rightarrow A) \ldots)$$

where A is atomic and where $\alpha = (\alpha_1, \ldots, \alpha_n)$.

Then one can show using (1) and (2) above that for any wff A of **CL** *we have*

$$\vdash_{\mathbf{CL}} \alpha_1 \rightarrow \ldots \rightarrow (\alpha_n \rightarrow A) \ldots) \textit{ iff } \rho \vdash_{\mathbf{M(H)}} \alpha : A$$

Remark 13.2.8 *Note that Relevance Implication is complete for the semantics of possible worlds $\{\alpha\}$ with operation $\otimes$, where*

$$\alpha \vDash A \rightarrow B \textit{ iff } \forall \beta (\beta \vDash A \Rightarrow \alpha \otimes \beta \vDash B)$$

where α is a subset of a set X and $\otimes$ is union.

If we take α a multiset subset of X and $\otimes$ multiset union, we get Linear Logic **L**.

If we take α any finite sequence of elements from X and $\otimes$ concatenation, we get the logic **CL** *above. See section 10.1.*

13.3 Temporal labelled deduction systems

This section will present the labelled deduction methodology which serves as a framework for developing temporal proof theory. We begin by asking ourselves what is a temporal database? Intuitively, looking at existing real temporal problems, we can say that we have information about things happening in different times and some connections between them. Figure 13.2 is an example of *A Temporal Configuration*

The figure shows a finite set of points of time and some labelled formulas. These are supposed to hold at the times indicated by the labels. Notice that we have not only labelled assertions but also Horn clauses showing dependencies across times. Thus at time t it may be true that B will be true. We represent that as $t : FB$. The language we are using has F and P as connectives. It is possible to have more connectives and still remain within the Horn clause framework. Most sueful among them are "$t : F^s A$" and "$t : P^s A$" reading "$t < s$ and $s : A$" and "$s < t$ and $s : A$", in words: " A will be true at time $s > t$".

The temporal configuration comprises of two components.

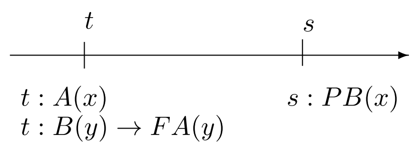


Figure 13.2:

1. A (finite) graph $(\rho, <)$ of time points and the temporal relationships between them.
2. With each point of the graph we associate a (finite) set of clauses and assertions, representing what is true at that point.

In Horn clause computational logic, there is an agreement that if a formula of the form $A(x) \to B(x)$ appears in the database with x free then it is understood that x is universally quantified. Thus we assume $\forall x(A(x) \to B(x))$ is in the database. The variable x is then called universal (or type 1). In the case of modal and temporal logics, we need another type of variable, called type 2 or a Skolem variable. To explain the reason, consider the item of data

"$t : FB(x)$".

This reads, according to our agreement,

"$\forall x FB(x)$ true at t."

For example it might be the sentence: t: "Everyone will leave."

The time in the future in which $B(x)$ is true depends on x. In our example, the time of leaving depends on the person x. Thus, for a given unknown (uninstantiated) u, that is for a given person u which we are not yet specifying, we know there must be a point t_1 of time (t_1 is dependent on u) with $t_1 : B(u)$. This is the time in which u leaves.

This u is by agreement not a type 1 variable. It is a u to be chosen later. Really u is a Skolem constant and we do not want to and cannot read it as $t_1 : \forall u B(u)$. Thus we need two types of variables. The other alternative is to make the dependency of t_1 on u explicit and to write

$$t_1(x) : B(x)$$

with x a universal type 1 variable, but then the object language variable x appears in the world indices as well. The world indices, ie the t's, are external to the formal clausal temporal language, and it is simpler not to mix the t's and the x's. We chose the two type of variable approach. Notice that when we ask for a goal $?G(u)$, u is a variable to be instantiated, ie a type 2 variable. So we have these variables anyway, and we prefer to develop a systematic way of dealing with them.

To explain the rôle of the two types of variables, consider the following classical Horn clause database and query:

$$\begin{array}{ll} A(x,y) \to B(x,y) & ?B(u,u) \\ A(a,a) & \end{array}$$

This means "Find an instantiation u_0 of u such that $\forall x, y[A(x,y) \to B(x,y)] \wedge A(a,a) \vdash B(u_0, u_0)$. there is no reason why we cannot allow for the following

$$\begin{array}{ll} A(u,y) \to B(x,u) & ?B(u,u) \\ A(a,a) & \end{array}$$

In this case we want to find a u_0 such that

$$\forall x, y[A(u_0, y) \to B(x, u_0)] \wedge A(a,a) \vdash B(u_0, u_0)$$

or to show

$$\vdash \exists u\{[\forall x, y[A(u,y) \to B(x,u)] \wedge A(a,a) \to B(u,u)]\}$$

u is called type 2 (Skolem) variable and x, y are universal type 1 variables. Given a database and a query of the form $\Delta(x,y,u)?Q(u)$, success means $\vdash \exists u[\forall x, y\Delta(x,y,u) \to Q(u)]$.

$$\begin{tikzcd}
 & x_1 \\
t \arrow[ur] \arrow[rr] & & x_2
\end{tikzcd}$$

Figure 13.3:

The next sequence of definitions will develop the syntax of the Temporal Horn Clause fragment. A lot depends on the flow of time. We will give a general definition, (definition 13.3.3 below), which includes the following connectives:

$\Box$ Always

F It will be the case

P It was the case

G It will always be the case (not including now)

H It has always been the case (up to now and not including now)

$\mathcal{O}$ Next moment of time (in particular it implies that such a moment of time exists)

$\otimes$ Previous moment of time (in particular it implies that such a moment of time exists).

Later on we will also deal with S (Since) and U (Until).
The flows of time involved are mainly three.

- General partial orders $(T, <)$.
- Linear orders
- The integers or the natural numbers.

The logic and theorem provers involved, even for the same connectives, are different for different partial orders. Thus the reader should be careful to note in which flow of time we are operating. Usually the connectives $\mathcal{O}$ and $\otimes$ assume we are working in the flow of time of integers.

Having fixed a flow of time $(T, <)$, the Temporal Fragment will generate finite configurations of points of time according to the information available to it. These are denoted by $(\rho, <)$. We are supposed to have $\rho \subseteq T$ (more precisely ρ will be homomorphic into T), and the ordering on ρ be the same as the ordering on T. The situation gets a bit complicated if we have a new point s and we do not know where it is supposed to be in relation to known points. We will need to consider all possibilities. Which possibilities do arise depend on $(T, <)$, the background flow of time we are working with. Again we should watch for variations in the sequel.

Definition 13.3.1 *Let $(\rho, <)$ be a finite partial order. Let $t \in \rho$ and let s be a new point. Let $\rho' = \rho \cup \{s\}$, and let $<'$ be a partial order on ρ'. Then $(\rho', <, t)$ is said to be a (one new point)* future *(respectively* past*) configuration of $(\rho, <, t)$ iff $t <' s$ (respectively $s <' t$) and $\forall xy \in \rho(x < y \leftrightarrow x <' y)$.*

Example 13.3.2 *Consider a general partial flow $(T, <)$ and consider the subflow $(\rho, <)$: The possible future configurations (relative to $T, <)$) of one additional point s are in figure 13.4:*

For a finite $(\rho, <)$ there is a finite number of future and past non-isomorphic configurations. This finite number is exponential in the size of ρ. So in the general case without simplifying assumptions we will have an intractable exponential computation. A configuration gives all possibilities of putting a point in the future or past.

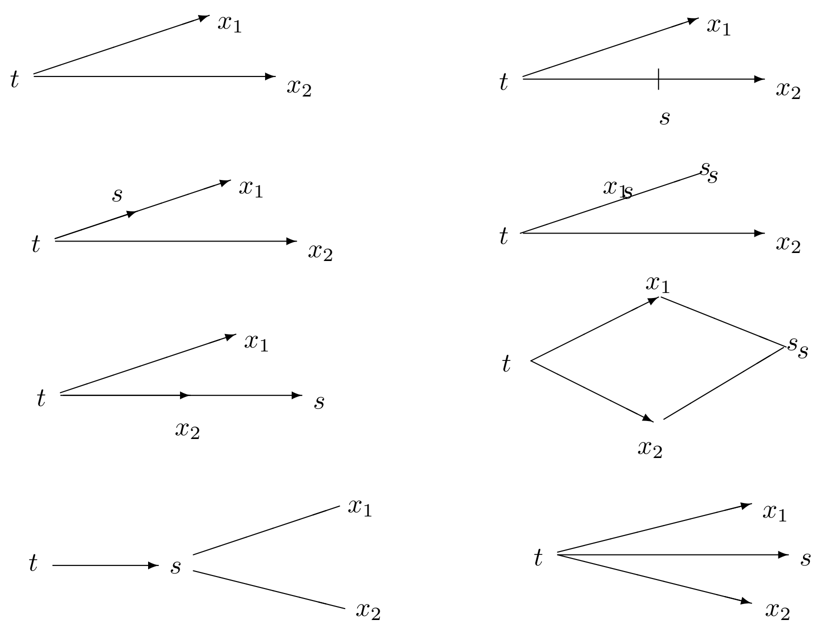


Figure 13.4:

In case of an ordering in which a next element or a previous element exists (like $t+1$ and $t-1$ in the integers) the possiblilities for configurations are different. In this case we must assume that we know the exact *distance between the elements of $(\rho, <)$.*

For example in the configuration $\{t < x_1, t < x_2\}$ of figure 13.3above we may have the further following information (see Fig 13.4) as part of the configuration:

$$t = \otimes^{18} x_1$$

$$t = \otimes^{6} x_2$$

so that we have only a finite number of possiblities for putting s in.

Note that although $\otimes$ operates on propositions, it can also be used to operate on points of time, denoting the predecessor function.

Definition 13.3.3 *Consider a Temporal Fragment with the following connectives and predicates:*

1. *Atomic predicates.*
2. *Function symbols and constants*
3. *Two types of variables:*
   Universal variables (Type 1) $V = \{x_1, y_1, z_1, x_2, y_2, z_2, \ldots\}$
   and Skolem variables (Type 2) $U = \{u_1, v_1, u_2, v_2, \ldots\}$.
4. *The connectives $\wedge, \rightarrow, \vee, F, P, \mathcal{O}, \otimes, \Box$ and $\neg$.*

FA reads:	*it will be the case that A.*
PA reads:	*it was the case that A.*
$\mathcal{O}A$ reads:	*A is true tomorrow (if a tomorrow exists, if tomorrow does not exist then it is false).*
$\otimes A$ reads:	*A was true yesterday (if yesterday does not exist then it is false).*
$\neg$:	*represents negation by failure.*

We define now the notions of an Ordinary Clause*, an* Always Clause*, a* Body*, a* Head *and a* Goal*.*

1. *A* Clause *is either an always clause or an ordinary clause.*
2. *An* Always Clause *has the form $\Box A$, where A is an ordinary clause.*
3. *An* Ordinary Clause *is a Head or an $A \to H$, where A is a body and H is a Head.*
4. *A* Head *is either an atomic formula or an FA or a PA or an $\mathcal{O}A$ or an $\otimes A$ where A is a finite conjunction of ordinary clauses.*
5. *A* Body *is either an atomic formula or an FA or a PA or an $\mathcal{O}A$ or an $\otimes A$ or $\neg A$ or a conjunction of bodies where A is a body.*
6. *A* Goal *is a body whose variables are* all *Skolem variables.*
7. *A disjunction of goals is also a goal.*

Remark 13.3.4 *Definition 13.3.3 included all possible temporal connectives. In practice different systems may contain only some of these connectives. For example a modal system may contain only $\Diamond$ (corresponding to F) and $\Box$. A future discrete system may contain only $\mathcal{O}$ and F etc.*

Depending on the system and the flow of time, the dependencies between the connectives change. For example we have the equivalence

$$\Box(a \to \mathcal{O}b) \text{ and } \Box(\otimes a \to b)$$

whenever both $\otimes a$ and $\mathcal{O}b$ are meaningful.

Definition 13.3.5 *Let $(T, <)$ be a flow of time. Let $(\rho, <)$ be a finite partial order. A* labelled temporal database *is a set of labelled ordinary clauses of the form $(t_i : A_i), t \in \rho$ and always clauses of the form $\Box A_i, A_i$ a clause. A labelled goal has the form $t : G$, where G is a goal.*

Δ is said to be a labelled temporal database over $(T, <)$ if $(\rho, <)$ is homomorphic into $(T, <)$.

Definition 13.3.6 *We now define the computation procedure for the temporal prolog for the language of definitions 13.3.3 and 13.3.5. We assume a flow of time $(T, <)$. $\rho \subseteq T$ is a finite set of points of time involved so far in the computation. The exact computation steps depend on the flow of time. It is different for branching, discrete linear etc. We will give the definition for linear time, though not necessarily discrete. Thus the meaning of $\mathcal{O}A$ in this logic is that there exists a next moment and A is true at this next moment. Similarly for $\otimes A$. $\otimes A$ reads there exists a previous moment and A was true at that previous moment.*

We define the success predicate $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0, \Theta)$ where $t \in \rho, (\rho, <)$ is a finite partial order and Δ is a set of labelled clauses $(t : A), t \in \rho$.

$\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0, \Theta)$ reads: The labelled goal $t : G$ succeeds from Δ under the substitution Θ to all the type 2 variables of G and Δ in the computation with starting labelled goal $t_0 : G_0$.

When Θ is known, we write $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0)$ only.

We define the simultaneous success and the failure of a set $\mathbf{\Pi}$ of metapredicates of the form $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0)$ under a substitution Θ to type 2 variables. To explain intuitive the meaning of success or failure, assume first that Θ is a substitution which grounds all the Skolem type 2 variables. In this case $(\mathbf{\Pi}, \Theta)$ succeeds if by definition all $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0, \Theta) \in \mathbf{\Pi}$ succeed and $(\mathbf{\Pi}, \Theta)$ fails if at least one of $\mathbf{S} \in \mathbf{\Pi}$ fails. The success or failure of $\mathbf{S}$ for a Θ as above has to be defined recursively. For a general $\Theta, (\mathbf{\Pi}, \Theta)$ succeeds, if for some Θ' such that $\Theta\Theta'$ grounds all type 2 variables $(\mathbf{\Pi}, \Theta\Theta')$ succeeds. $(\mathbf{\Pi}, \Theta)$ fails if for all Θ' such that $\Theta\Theta'$ grounds all type 2 variables we have that $(\mathbf{\Pi}, \Theta\Theta')$ fails. We need to give recursive procedures for the computation of the success and failure of $(\mathbf{\Pi}, \Theta)$. In the case of the recursion, a given $(\mathbf{\Pi}, \Theta)$ will be changed to a $(\mathbf{\Pi}', \Theta')$ by taking $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0) \in \mathbf{\Pi}$ and replacing it by $\mathbf{S}(\rho', <', \Delta', G', t', G_0, t_0)$. We *will have several such changes and thus get several $\mathbf{\Pi}'$ by replacing several $\mathbf{S}$'s in Π. We write the several possibilities as $(\mathbf{\Pi}'_i, \Theta'_i)$. If we write $(\mathbf{\Pi}, \Theta)$ to mean $(\mathbf{\Pi}, \Theta)$ succeeds and $\sim (\mathbf{\Pi}, \Theta)$ to read $(\mathbf{\Pi}, \Theta)$ fails, then our recursive computation rules have the form: $(\mathbf{\Pi}, \Theta)$ succeeds (or fails) if some boolean combination of $(\mathbf{\Pi}'_i, \Theta'_i)$ succeed (or fail). The rules allow us to pick an element in $\mathbf{\Pi}$, eg $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0)$ and replace it with one or more elements to obtain the different $(\mathbf{\Pi}'_i, \Theta'_i)$, where Θ'_i is obtained from Θ. In case of failure we require that Θ grounds all type 2 variables. We do not define failure for a nongrounding Θ.*

To summarise the general structure of the rules is:
$(\mathbf{\Pi}, \Theta)$ succeeds (or fails) if some boolean combination of the successes and failures of some $(\mathbf{\Pi}'_i, \Theta'_i)$ holds and $(\mathbf{\Pi}, \Theta)$ and $(\mathbf{\Pi}'_i, \Theta'_i)$ are related according to one of the following cases:

I If $\mathbf{\Pi} = \varnothing$ then $(\mathbf{\Pi}, \Theta)$ succeeds, (ie the boolean combination of $(\mathbf{\Pi}_i, \Theta_i)$ is truth.

II $(\mathbf{\Pi}, \Theta)$ fails if for some $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0)$ in $\mathbf{\Pi}$ we have G is atomic and for all $\Box(A \to H) \in \Delta$ and for all $(t : A \to H) \in \Delta, H\Theta$ does not *unify with $G\Theta$. Further, for all Ω and s such that $t = \Omega s$ and for all $s : A \to \Omega H$ and all $\Box(A \to \Omega H)$ we have $H\Theta$ does not unify with $G\Theta$, where Ω is a sequence of $\mathcal{O}$ and $\otimes$.*

Remark 13.3.7 *We must qualify the conditions of the notion of failure. If we have a goal $t : G$, with G atomic, we know for sure that $t : G$ finitely fails under a substitution Θ, if $G\Theta$ cannot unify with any head of a clause. This is what the condition above says. What are the candidates for unification? These are either clauses of the form $t : A \to H$, with H atomic or $\Box(A \to H)$, with H atomic.*

Do we have to consider the case where H is not atomic? The answer depends on the flow of time and on the configuration $(\rho, <)$ we are dealing with. If we have eg $t : A \to FG$ then if $A \to FG$ is true at t, G would be true (if at all) in some s, $t < s$. This s is irrelevant to our query $?t : G$. Even if we have $t' < t$ and $t' : A \to FG$ and A true at t', we still can ignore this clause because we are not assured that any s such that $t' < s$ and G true at s would be the desired t (ie $t = s$).

The only case we have to worry about is when the flow of time and the configuration are such that we have for example $t' : A \to \mathcal{O}^5 G$ and $t = \mathcal{O}^5 t'$.

In this case we must add the following clause to the notion of failure:

For every s such that $t = \mathcal{O}^n s$ and every $s : A \to \mathcal{O}^n H$, $G\Theta$ and $H\Theta$ do not unify.

We also have to check what happens in the case of always clauses.

Consider an integer flow of time and the clause $\Box(A \to \mathcal{O}^5 \otimes^{27} H)$, This true at the point $s = \otimes^5 \mathcal{O}^{27} t$ and hence for failure we need that $G\Theta$ does not unify with $H\Theta$.

The above explains the additional condition on failure.

The following conditions (1) - (10), (12) - (13) relate to the success of $(\mathbf{\Pi}, \Theta)$ if $(\mathbf{\Pi}'_i, \Theta'_i)$ succeed. Condition (11) uses the notion of failure, to give the success of negation by failure. Conditions (1)-(10), (12)-(13) give certain alternatives for success. They give failure if each one of these alternatives end up in failure.

1. **Success rule for atomic query:**
   $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0) \in \mathbf{\Pi}$ and G is atomic and for some head H, $(t : H) \in \Delta$ and for some substitutions Θ_1 to the universal variables of H and Θ_2 to the existential variables of H and G we have $H\Theta_1\Theta\Theta_2 = G\Theta\Theta_2$ and $\mathbf{\Pi}' = \mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_o)\}$ and $\Theta' = \Theta\Theta_2$.

2. **Computation rule for atomic query:**
   $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_o) \in \mathbf{\Pi}$ and G is atomic and for some $(t : A \to H) \in \Delta$ or for some $\Box(A \to H) \in \Delta$ and for some Θ_1, Θ_2, we have $H\Theta_1\Theta\Theta_2 = G\Theta\Theta_2$ and $\mathbf{\Pi}' = (\mathbf{\Pi} - \{\{\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_o)\}) \cup \{\mathbf{S}(\rho, <, \Delta, A\Theta_1, t, G_0, t_o)\}$ and $\Theta' = \Theta\Theta_2$.

The above rules dealt with the atomic case. Rules (3), (4) and (4*) deal with the case the goal is FG. The meaning of (3), (4) and (4*) is the following:
we ask FG at t. How can we be sure that FG is true at t? There are two possibilities, (a) and (b)

(a) We have $t < s$ and at $s : G$ succeeds. This is rule (3).

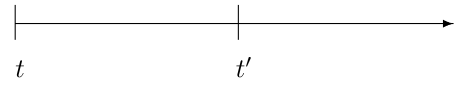


Figure 13.5:

(b) Assume that we have the fact that $A \to FB$ is true at t. We ask for A and succeed and hence FB is true at t. Thus there should exist a point s' in the future of t where B is true. Where can s' be? We don't know where s' is in the future of t. So we consider all future configurations for s'. This gives us all future possibilities where s' can be. We assume for each of these possibilities that B is true at s' and check whether either G follows at s' or FG follows at s'. If we find that for all future constellations of where s' can be $G \vee FG$ succeeds in s' from B, then FG holds at t. Here we use the transitivity of $<$. (4a) gives the possibilities where s' is an old point s in the future of t. (4b) gives the possibilities where s' is a new point forming a new configuration. Success is needed from *all* possibilities.

3. **Immediate rule for F:**
$\mathbf{S}(\rho, <, \Delta, FG, t, G_0, t_o) \in \mathbf{\Pi}$ and for some $s \in \rho$ such that $t < s$ we have $\mathbf{\Pi}' = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, FG, t, G_0, t_o)\}) \cup \{\mathbf{S}(\rho, <, \Delta, G, s, G_0, t_o)\}$ and $\Theta' = \Theta$.

4. **First configuration rule for F:**
$\mathbf{S}(\rho, <, \Delta, FG, t, G_0, t_o) \in \mathbf{\Pi}$ and some $(t : A \to F \wedge_j B_j) \in \Delta$ and some Θ_1, Θ_2 we have both (a) and (b) below are true. A may not appear in which case we pretend $A =$ truth.

   (a) For *all* $s \in \rho$, such that $t < s$ we have that
   $\mathbf{\Pi}'_s = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, FG, t, G_0, t_0)\}) \cup \{\mathbf{S}(\rho, <, \Delta, E\Theta_1, t, G_0, t_0)\} \cup \{\mathbf{S}(\rho, <, \Delta \cup \{(s : B_j\Theta_1) \mid j = 1, 2, \ldots\}, D, s, G_0, t_0\}$ succeeds with $\Theta'_s = \Theta\Theta_2$ and $D = G \vee FG$ and $E = A$.

   (b) For all future configurations of $(\rho, <, t)$ with a new letter s, denoted by the form $(\rho_s, <_s)$ we have that
   $\mathbf{\Pi}'_s = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, FG, t, G_0, t_o\}) \cup \{\mathbf{S}(\rho, <, \Delta, E\Theta_1, t, G_0, t_o)\} \cup \{\mathbf{S}(\rho_s, <_s, \Delta \cup \{(s : B_j) \mid j = 1, 2, \ldots\}, D, s, G_0, t_0)\}$ succeeds with $\Theta'_s = \Theta\Theta_2$ and $D = G \vee FG$ and $E = A$.

The reader should note that conditions (3), (4a) and (4b) are needed only when the flow of time has some special properties. To explain by example, assume we have the configuration
and $\Delta = \{t : A \to FB, t' : C\}$ as data and our query is $?t : FG$.

Then according to rules (3), (4) we have to check and succeed in all the following cases:

1. From rule (3) we check $\{t' : C, t : A \to FB\}?t' : G$

2. From rule (4a) we check$\{t' : C, t : A \to FB, t' : B\}?t' : G$

3. From rule (4b) we check $\{t' : C, t : A \to FB, s : B\}?s : G$

for the three configurations below

If time is linear configuration 3.3 does not arise and we are essentially checking 3.1, 3.2 and the case 4a corresponding to $t' = s$.

If we do not have any special properties of time, success in case 3.2 is required. Since we must succeed in all cases and 3.2 is the case with *least* assumptions, it is enough to check 3.2 alone.

Thus for the case of no special properties of the flow of time, case 4 can be replaced by case 4 general below:

4 *general.* $\mathbf{S}(\rho, <, \Delta, FG, t, G_0, t_0) \in \mathbf{\Pi}$ and for the future configuration $(\rho_1, <_1)$ defined as $\rho_1 = \rho \cup \{s\}$ and $<_1 = < \cup \{t < s\}, s$ a new letter we have that: $\mathbf{\Pi}'s = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, FG, t, G_0, t_0)\} \cup \{\mathbf{S}(\rho, <, \Delta, E\Theta_1, t, G_0, t_0)\} \cup \{\mathbf{S}(\rho_1, <_1, \Delta \cup \{(s : B_j) \mid j = 1, 2, \ldots\}, D, s, G_0, t_0)\}$ succeeds with $\Theta'_s = \Theta\Theta_1$ and $D = G \vee FG$ and $E = A$.

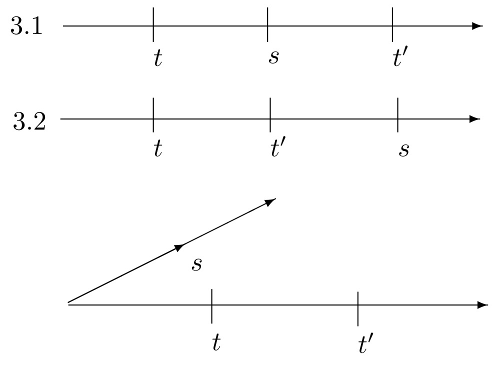


Figure 13.6:

4*. **2nd Configuration rule for F:**
For some $\mathbf{S}(\rho, <, \Delta, FG, t, G_0, t_0)$ and some $\Box(A \to F \wedge_j B_j) \in \Delta$ and some $\Theta_1\Theta_2$ we have both cases 4a and 4b above true with $E = A \vee FA$ and $D = G \vee FG$.

4**general* Similar to (4 *general*) for the case of general flow.

This is the the mirror image of (3) with "PG" replacing "FG" and "$s < t$" replacing "$t < s$".

6; 6* This is the mirror image of (4) and (4*) with "PG" replacing "FG", "$s < t$" replacing "$t < s$" and "past configuration" replacing "future configuration".

6 *general* This is the image of (4 *general*).

We give now the computation rules (7)-(10) for $\mathcal{O}$ and $\otimes$ for orderings in which a next point and/or previous points exist. If $t \in T$ has a next point we denote this point by $s = \mathcal{O}t$. If it has a previous point we denote it by $s = \otimes t$. For example, if $(T, <)$ is the integers then $\mathcal{O}t = t + 1$ and $+t = t - 1$. If $(T, <)$ is a tree then $\otimes t$ always exists, except at the root, but $\mathcal{O}t$ may or may not exist. For the sake of simplicity we must assume that if we have $\mathcal{O}$ or $\otimes$ in the language then $\mathcal{O}t$ or $\otimes t$ always exist. Otherwise we can sneak negation in by putting $(t : \mathcal{O}A) \in \Delta$ when $\mathcal{O}t$ does not exist!

7. **Immediate rule for $\mathcal{O}$:**
$\mathbf{S}(\rho, <, \Delta, \mathcal{O}G, t, G_0, t_o) \in \mathbf{\Pi}$ and $\mathcal{O}t$ exists and $\mathcal{O}t \in \rho$ and $\Theta' = \Theta$ and $\mathbf{\Pi}' = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, \mathcal{O}G, t, G_0, t_o)\} \cup \{\mathbf{S}(\rho, <, \Delta, G, \mathcal{O}t, G_0, t_o)\}$

8. **Configuration rule for $\mathcal{O}$:**
$\mathbf{S}(\rho, <, \Delta, \mathcal{O}G, t, G_0, t_o) \in \mathbf{\Pi}$ and some Θ_1, Θ_2 some $(t : A \to \mathcal{O} \wedge_j B_j) \in \Delta$ and $\mathbf{\Pi}' = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, \mathcal{O}G, t, G_0, t_o)\} \cup \{\mathbf{S}(\rho, <, \Delta, A\Theta_1, t, G_0, t_o)\} \cup \cup\{\mathbf{S}(\rho \cup \{\mathcal{O}t\}, <', \Delta \cup \{(\mathcal{O}t : B_j)\}, G, \mathcal{O}t, G_0, t_0)\}$ succeeds with $\Theta' = \Theta\Theta_2$, and $<'$ is the appropriate ordering closure of $< \cup\{(t, \mathcal{O}t)\}$

   Notice that case 8 is parallel to case 4. We do not need 8a and 8b because of $\mathcal{O}t \in \rho$ then what would be case (8b) becomes (7).

9. The mirror image of (7) with "$\otimes$" replacing "$\mathcal{O}$".

10. The mirror image of (8) with "$\otimes$" replacing "$\mathcal{O}$".

11. **Negation as failure rule:**
$\mathbf{S}(\rho, <, \Delta, \neg G, t, G_0, t_0) \in \mathbf{\Pi}$ and Θ grounds every type 2 variable and the computation for success of $\mathbf{S}(\rho, <, \Delta, G, t, \Theta)$ ends up in failure.

12. **Disjunction rule:**
$\mathbf{S}(\rho, <, \Delta, G_1 \vee G_2, t, G_0, t_o) \in \mathbf{\Pi}$ and $\mathbf{\Pi}' = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, G_1 \vee G_2, t, G_0, t_o)\}) \cup \{\mathbf{S}(\rho, <, \Delta, G_i, t, G_0, t_o)\}$ and $\Theta' = \Theta$ and $i \in \{1, 2\}$.

13. **Conjunction rule:**
$\mathbf{S}(\rho, <, \Delta, G_1 \wedge G_2, t, G_0, t_o) \in \mathbf{\Pi}$ and $\mathbf{\Pi}' = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, G_1 \wedge G_2, t, G_0, t_o)\}) \cup \{\mathbf{S}(\rho, <, \Delta, G_i, t, G_0, t_o) \mid i \in \{1, 2\}\}$

14. **Restart Rule**
$\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_o) \in \mathbf{\Pi}$ and $\mathbf{\Pi}' = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_o)\}) \cup \{\mathbf{S}(\rho, <, \Delta, G_1, t_0, G_0, t_0)\}$ where G_1 is obtained from G_0 by substituting completely new type 2 variables u'_i for the type 2 variables u_i of G_0, and where Θ' extends Θ by giving $\Theta'(u'_i) = u'_i$ for the new variables u'_i.

15. **To Start the Computation:**
Given Δ and $t_0 : G_0$ and a flow $(T, <)$, we start the computation with $\mathbf{\Pi} = \{\mathbf{S}(\rho, <, \Delta, G_0, t_0, G_0, t_0)\}$, where $(\rho, <)$ is the configuratin associated with Δ, over $(T, <)$ (definition D3).

Let us check some examples

Example 13.3.8
Data:

1. $t : a \rightarrow Fb$

2. $\Box(b \rightarrow Fc)$

3. $t : a$

Query: $?t : Fc$
Configuration: $\{t\}$
Using rule (4) we create a future s with $t < s$ and ask the two queries: (the notation $A?B$ means that we add A to the data (1), (2), (3) and ask $?B$).*

4. $?t : \mathcal{C} \vee Fb$
and

5. $s : c?s : c \vee Fc$

(5) succeeds and (4) splits into two queries by rule (4).

6. $?t : a$
and

7. $s' : b?s' : b.$

The above computation is the same as the one in figure 3 of part 1 of this paper. The only difference is that since our data are labelled we do not have to keep copying the database. More significant differences will show in the next example.

Example 13.3.9
Data:

(1) $t : FA$

(2) $t : FB$

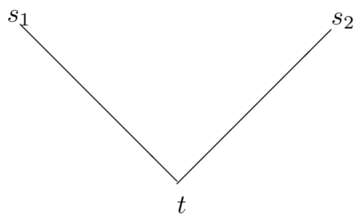


Figure 13.7:

Query: *$t : F\varphi$ where $\varphi = (A \wedge B) \vee (A \wedge FB) \vee (B \wedge FA)$.*

The query will fail in any flow of time in which the future is not linear. The purpose of this example is to examine what happens when time is linear. Using (1) we introduce a point s, with $s : A$ and query from s the following:

$$?s : \varphi \vee F\varphi$$

If we do not use the restart rule, the query will fail. Now that we are at a point s there is no way to go back to t. We therefore cannot reason that we also have a point $s' : B$ and $t < s$ and $t < s'$ and that because of linearity either $s = s'$ or $s < s'$ or $s' < s$. However, if we are allowed to restart, we can continue and ask $t : F\varphi$ and now use the clause $t : FB$ to introduce s'. We now reason using linearity in rule (4) that the configurations are

either $t < s < s'$
or $t < s' < s$
or $t < s = s'$.

and φ succeeds at t for each configuration.

The reader should note the reason for the need to use the restart rule. When time is just a partial order, the two assumptions $t : FA$ and $t : FB$ do not interact. Thus when asking $t : FC$, we know that there are two points $s_1 : A$ and $s_2 : B$, see figure 13.7.

and C can be true in either one of them. $s_1 : A$ has no influence on $s_2 : B$. When conditions on time (such as linearity) are introduced, s_1 does influence s_2 and hence we must introduce both at the same time. When one does forward deduction one can introduce both s_1 and s_2 going forward. The backward rules do not allow for that. That is why we need the restart rule. When we restart, we keep all that has been done (with, for example s_1) and have the opportunity to restart with s_2. The restart rule can be used to solve the linearity problem for classical logic only. Its side effect is that it turns intuitionistic logic into classical logic, see my paper on N-Prolog. In theorem proving based on intuitionistic logic where disjunctions are allowed, forward reasoning cannot be avoided. See next example.

It is instructive to translate the above into Horn Clause logic and see what happens there.

Example 13.3.10 *1. $t : FA$ translates into $(\exists s_1 > t)A^*(s_1)$*

2. $t : FB$ translates into $(\exists s_2 > t)A^*(s_2)$.
The query translates into the formula $\psi(t)$:
$\psi = \exists s > t[A^*(s) \wedge B^*(s)] \vee \exists s_1 > t[A^*(s_1) \wedge \exists s_2 > s_1 B^*(s_2)] \vee \exists s_2 > t[B^*(s_2) \wedge \wedge \exists s_1 > s_2 A^*(s_2)]$
which is equivalent to the disjunction of

(a) $[t < s \wedge A^*(s) \wedge B^*(s)]$

(b) $t < s_1 \wedge s_1 < s_2 \wedge A^*(s_1) \wedge B^*(s_2)$

(c) $t < s_1 \wedge s_2 < s_1 \wedge A^*(s_1) \wedge B^*(s_2)$.

all of (a), (b), (c) fail from the data, unless we add to the data the disjunction

$$\forall xy(x < y \vee x = y \vee y < x)$$

Since this is not a Horn Clause, we are unable to express it in the database.

This means that we can handle properties of time which are not necessarily expressible by an object language formula of the logic. In some cases (finiteness of time) because they are not first order, in other cases (irreflexivity) because there is no corresponding formula (axiom) and in other cases because of syntactical restrictions (linearity).

We can now make clear our classical versus intuitionistic distinction. If the underlying logic is classical then we are checking whether $\Delta \vdash \psi$ in classical logic. If our underlying logic is intuitionistic, then we are checking whether $\Delta \vdash \psi$ in intuitionistic logic where Δ and ψ are defined below:

Δ is the translation of the data together with the axioms for linear ordering namely, the conjunction of:

1. $\exists s_1 > tA^*(s_1)$
2. $\exists s_2 > tB^*(s_2)$
3. $\forall xy(x < y \vee x = y \vee y < x)$
4. $\forall x \exists y(x < y)$
5. $\forall x \exists y(y < x)$
6. $\forall xyz(x < y \wedge y < z \to x < z)$
7. $\forall x \neg(x < x)$.

ψ is the translation of the query as given above.

The computation of example 13.3.9, using restart, answers the question $\Delta \vdash ?\psi$ in classical logic. To answer the question $\Delta \vdash ?\psi$ in intuitionistic logic we cannot use restart, we must use forward rules as well.

Example 13.3.11 *Example 13.3.9 for the case that the underlying logic is intuitionistic.* Data *and* Query *as in Example 13.3.9.*

Going forward, we get:

(3) $s : A$ *from (1)*

(4) $s' : B$ *from (2)*

By linearity, either

$$t < s < s'$$

$$\text{or } t < s' < s$$

$$\text{or } t < s = s'$$

ψ *will succeed for each case.*

Our language does not allow us to ask queries of the form $\Box G(x)$, where x are all universal variables (ie $\forall x \Box G(x)$). However such queries can be computed from a database Δ. The *only* way to get *always* information out of Δ for a general flow of time is via the always clauses in the database. Always clauses are true everywhere, so if we want to know what else is true everywhere, we ask it from the always clauses: Thus to ask:

$$?\Box G(x), x \text{ a universal variable.}$$

we first skolemise and then ask

$$\{X, \Box X \mid \Box X \in \Delta\}?G(c)$$

where c is a Skolem constant.

We can add a new rule to definition 13.3.6:

(15) **Always Rule**

$\mathbf{S}(\rho, <, \Delta, \Box G, t, G_0, t_0) \in \mathbf{\Pi}$ and $\mathbf{\Pi}' = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, \Box G, t, G_0, t_0)\}) \cup \{\mathbf{S}(\{s\}, \varnothing, \Delta', G', s, g', s)\}$

where s is a completely new point and G' is obtained from G by substituting new Skolem constants for all the universal variables of G and

$$\Delta' = \{B, \Box B \mid \Box B \in \Delta\}.$$

We can use (15) to add another clause to the computation of Definition D4, namely

(16) $\mathbf{S}(\rho<, \Delta, F(A \wedge B), t, G_0, t_0) \in \mathbf{\Pi}$ and $\mathbf{\Pi}' = (\mathbf{\Pi} - \{\mathbf{S}(\rho, <, \Delta, F(A \wedge B), t, G_0, t_0)\}) \cup \{\mathbf{S}(\rho, <, \Delta, FA, t, G_0, t_0), \mathbf{S}(\rho, <, \Delta, \Box B, t, G_0, t_0)\}$

Example 13.3.12

Data	Query	Configuration
$\Box a$	$t : F(a \wedge b)$	$\{t\}$
$t : Fb$		

First Computation.

Create $s, t < s$ and get

Data	Query	Configuration
$\Box a$	$s : a \wedge b$	$t < s$
$t : Fb$		
$s : b$		

$s : b$ succeeds from the data. $s : a$ succeeds by rule 2, definition 13.3.6.

Second Computation

Use rule 16. Since $?\Box a$ succeeds ask for Fb and proceed as in the first computation.

13.4 Different flows of time

We now check the effect of different flows of time on our logical deduction (computation). We consider a typical example:

Example 13.4.1

Data	Query	Configuration
$t : FFA$	$?t : FA$	$\{t\}$

The possible world flow is a general binary relation.

We create by rule (4b) of definition 13.3.6, a future configuration $t < s$ and add to the database $s : FA$

we get:

Data	Query	Configuration
$t : FFA$	$?t : FA$	$t < s$
$s : FA$		

Again we apply rule (4a) of definition 13.3.6 and get the new configuration with $s < s'$ and the new item of data $s' : A$. We get

Data	Query	Configuration
$t : FFA$	$?t : FA$	$t < s$
$s : FA$		$s < s'$
$s' : A$		

whether or not we can proceed from here depends on the flow of time. If $<$ is transitive, then $t < s'$ holds and we can get $t : FA$ in the data by rule 3.

Actually by rule (4) we could have proceeded along the following sequence of deduction. Rule (4*) is especially geared for transitivity.*

Data	Query	Configuration
$t : FFA$	$t : FA$	t

using rule (4) we get*

Data	Query	Configuration
$t : FFA$	$s : FA \vee FFA$	$t < s$
$s : FA$		

The first disjunct of the query succeeds.

If $<$ is not transitive, rule 3 does not apply, since $t < s'$ does not hold.

Supose our query were $?t : FFFA$.

If $<$ is reflexive then we can succeed with $?t : FFFA$ because $t < t$.

If $<$ is dense (ie $\forall xy(x < y \to \exists z(x < z \wedge z < y))$ we should also succeed because we can create a point z with $t < z < s$.

$z : FFA$ will succeed and hence $t : FFFA$ will also succeed.

Here we encounter a new rule (density rule), whereby points can always be "landed" between existing points in a configuration.

We now address the flow of time of the type natural numbers, $\{1, 2, 3, 4, \ldots\}$. This has the special property that it is generated by a function symbol **s**.

$$\{1, \mathbf{s}(1), \mathbf{ss}(1)\ldots\}.$$

Example 13.4.2

Data	**Query**	**Configuration**
$\Box(q \to \mathcal{O}q)$	$1 : F(p \wedge q)$	$\{1\}$
$1 : \mathcal{O}q$		
$1 : Fp$		

If time is the natural numbers, the query should succeed from the data. If time is not the natural numbers but, for example $\{1, 2, 3, \ldots w, w+1, w+2, \ldots\}$ then the query should fail.

How do we represent the fact that time is the natural numbers in our computation rule? What is needed is the ability to do some induction. We can use rule (4b) and introduce a point t with $1 < t$ into the configuration and even say that $t = n$, for some n. We thus get:

Data	**Query**	**Configuration**
$\Box(q \to \mathcal{O}q)$	$1 : F(p \wedge q)$	$1 < n$
$1 : \mathcal{O}q$		
$1 : Fp$		
$n : p$		

Somehow we want to derive $n : q$ from the first two assumptions. The key reason for the success of $F(p \wedge q)$ is the success of $\Box q$ from the first two assumptions. We need an induction axiom on the flow of time.

To get a clue as to what to do, let us see what Horn Clause logic would do with the translations of the data and goal.

Translated Data

$\forall t[1 \leq t \wedge Q^*(t) \to Q^*(t+1)]$
$Q^*(1)$
$\exists t P^*(t)$

Translated Query

$\exists t(P^*(t) \wedge Q^*(t))$

After we skolemise, the database becomes:

1. $1 \leq t \wedge Q^*(t) \to Q^*(t+1)$
2. $Q^*(1)$
3. $P^*(c)$

and the query is

$$P^*(s) \wedge Q^*(s)$$

We proceed by letting $s = c$. We ask $Q^*(c)$ and have to ask after a slightly generalised form of unification $?1 \leq c \wedge Q^*(c-1)$.

Obviously this will lead nowhere without an induction axiom. The induction axiom should be that for any *predicate $PRED$*

$PRED(1) \wedge \forall x[1 \leq x \wedge PRED(x) \to PRED(x+1)] \to \forall x PRED(x)$

written in Horn clause form this becomes:

$$\exists x \forall y[PRED(1) \wedge [1 \leq x \wedge PRED(x) \rightarrow PRED(x+1)] \rightarrow PRED(y)].$$

Skolemising gives us

4. $PRED(1) \wedge (1 \leq d \wedge PRED(d) \rightarrow PRED(d+1)) \rightarrow PRED(y)$

d is a skolem constant.

Let us now ask the query $P^*(s) \wedge Q^*(s)$ from the database with (1)-(4). We unify with clause (3) and ask $Q^*(c)$. We unify with clause (4) and ask $Q^*(1)$ which succeeds and ask for the implication

$$?1 \leq d \wedge Q^*(d) \rightarrow Q^*(d+1)$$

This should succeed since it is a special case of clause (1) for $t = d$.

The above shows that we need to add an induction axiom of the form

$$\mathcal{O}x \wedge \Box(x \rightarrow \mathcal{O}x) \rightarrow \Box x$$

Imagine that we are at time t, and assume $t' < t$. If A is true at t' and $\Box(A \rightarrow \mathcal{O}A)$ is true, then A is true at t.

We thus need the following rule:

(18) **Induction Rule:**

$t : F(A \wedge B)$ succeeds from Δ at a certain configuration if the following conditions all hold.

1. *$t : FB$ suceed.*
2. *For some $s < t, s : A$ succeeds.*
3. *$m : \mathcal{O}A$ succeeds from the database Δ', where $\Delta' = \{X, \Box X \mid \Box X \in \Delta\} \cup \{A\}$ and m is a completely new time point and the new configuration is $\{m\}$.*

The above shows how to compute when time is the natural numbers. This is not the best way of doing it. In fact, the characteristic feature involved here is that the ordering of the flow of time is a Herbrand universe generated by a finite set of function symbols. FA is read as "A is true at a point generated by the function symbols". This property requires a special study. See our Metatem paper, Barringer 1989.

13.5 A theorem prover for modal and temporal logics

This section will briefly indicate how our temporal Horn clause computation can be extended to be an automated deduction system for full modal and temporal logic. We present computation rules for propositional temporal logic with $F, P, \mathcal{O}, \otimes, \wedge \rightarrow$ and $\bot$. We approach predicate logic in a subsequent paper as it is relatively complex. The presentation will be intuitive:

Definition 13.5.1 *We define the notions of a* Full Clause*, a* Body *and a* Head.

(a) *A* Full Clause *is either an atom q or $\bot$ or $B \rightarrow H$, or H where B is a body and H is a head.*

(b) *A* Body *is a conjunction of full clauses.*

(c) *a* Head *is either an atom q or $\bot$ or FH or PH or $\mathcal{O}H$ or $\otimes H$, where H is a body.*

Notice that negation by failure is not allowed. We used the connectives $\wedge, \rightarrow, \bot$. The other connectives, $\vee$ and $\sim$ are definable in the usual way $\sim A = A \rightarrow \bot$ and $A \vee B = (A \rightarrow \bot) \rightarrow B$. The reader can show that every formula of the language with the connectives $\{\sim, \wedge, \vee, F, G, P, H\}$is equivalent to a conjunction of full clauses. We use the following equivalences:

$$A \rightarrow (B \wedge C) = (A \rightarrow B) \wedge (A \rightarrow C)$$

$$A \to (B \to C) = A \wedge B \to C$$

$$GA = F(A \to \bot) \to \bot$$

$$HA = P(A \to \bot) \to \bot$$

Definition 13.5.2 *A database is a set of labelled full clauses of the form $(\Delta, \rho, <)$, where $\rho = \{t \mid t : A \in \Delta,$ for some $A\}$. A query is a labelled full clause.*

Definition 13.5.3 *The following is a definition of the predicate $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0)$, which reads:*
The labelled goal $t : G$ succeeds from $(\Delta, \rho, <)$ with parameter (initial goal) t_0, G_0.

1(a) $\mathbf{S}(\rho, <, \Delta, q, t, G_0, t_0)$ *for q atomic or $\bot$ if for some $t : A \to q$,* $\mathbf{S}(\rho, <, \Delta, A, t, G_0, t_0)$.

(b) *If $t : q \in \Delta$ or $s : \bot \in \Delta$ then* $\mathbf{S}(\rho, <, \Delta, q, t, G_0, t_0)$.

(c) $\mathbf{S}(\rho, <, \Delta, \bot, t, G_0, t_0)$ *if* $\mathbf{S}(\rho, <, \Delta, \bot, s, G_0, t_0)$.
This rule says that we can get a contradiction from any label, it would be considered a contradiction of the whole system.

2. $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0)$ *if for some $s : A \to \bot$,* $\mathbf{S}(\rho, <, \Delta, A, s, G_0, t_0)$.

3. $\mathbf{S}(\rho, <, \Delta, t, FG, G_0, t_0)$ *if for some $s \in \rho, t < s$ and* $\mathbf{S}(\rho, <, \Delta, G, s, G_0, t_0)$.

4. $\mathbf{S}(\rho, <, \Delta, FG, t, G_0, t_0)$ *if for some $t : A \to FB \in \Delta$ we have both (a) and (b) below hold true:*

(a) for all $s \in \rho$ such that $t < s$ we have $\mathbf{S}(\rho, <, \Delta^*, s, D, G_0, t_0)$ and $\mathbf{S}(\rho, <, \Delta, E, t, G_0, t_0)$ hold, where $\Delta^* = \Delta \cup \{s : B\}$ and $D \in \{G, FG\}$ and $E \in \{A, FA\}$.
Note: *The choice of D and E is made here for the case of transitive time. In modal logic, where $<$ is not necessarily transitive, we take $D = G, E = A$. Other conditions on $<$ correspond to different choices of D and E.*

(b) For all future configurations of $(\rho, <, t)$ with a new letter s, denoted by $(\rho_s, <_s)$ we have $\mathbf{S}(\rho_s, <_s, \Delta^*, s, D, G_0, t_0)$ and $\mathbf{S}(\rho_s, <_s, \Delta,\ E, t, G_0, t_0)$ hold, where Δ^*, E, D are as in (a).

5. This is the mirror image of (3).

6. This is the mirror image of (4).

7(a) $\mathbf{S}(\rho, <, \Delta, A_1 \wedge A_2, t, G_0, t_0)$ *if both* $\mathbf{S}(\rho, <, \Delta, A_i, t, G_0, t_0)$ *hold for $i = 1, 2$*

(b) $\mathbf{S}(\rho, <, \Delta, A \to B, t, G_0, t_0)$ *if* $\mathbf{S}(\rho, <, \Delta \cup \{t : A\}, B, t, G_0, t_0)$.

8. **Restart Rule**
$\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0)$ *if* $\mathbf{S}(\rho, <, \Delta, G_0, t_0, G_0, t_0))$.

If the language contains $\mathcal{O}$ and $\otimes$ then the following are the relevant rules.

9. $\mathbf{S}(\rho, <, \Delta, \mathcal{O}G, t, G_0, t_0)$ *if $\mathcal{O}t$ exists and $\mathcal{O}t \in \rho$ and*
$\mathbf{S}(\rho, <, \Delta, G, \mathcal{O}t, G_0, t_0)$.

10. $\mathbf{S}(\rho, <, \Delta, \mathcal{O}G, t, G_0, t_0)$ *if for some $t : A \to \mathcal{O}B \in \Delta$ both*
$\mathbf{S}(\rho, <, \Delta, A, t, G_0, t_0)$ *and* $\mathbf{S}(\rho \cup \{\mathcal{O}t\}, <', \Delta \cup \{\mathcal{O}t : B\}, G, \mathcal{O}t, G_0, t_0)$ *hold where $<'$ is the appropriate ordering closure of $< \cup \{t < \mathcal{O}t\}$*

11. This is the mirror image of (9) for $\otimes$.

12. This is the mirror image of (10) for $\otimes$.

Example 13.5.4 *(Here $\Box$ can be either G or H.)*

	Data	Query	Configuration
1.	$t : \Box a$	$?t : \Box b$	$\{t\}$
2.	$t : \Box(a \to b)$		*t is a constant*

Translation:

	Data	Query	Configuration
1.	$t : F(a \to \bot) \to \bot$	$t : F(b \to \bot) \to \bot$	$\{t\}$
2.	$t : F((a \to b) \to \bot) \to \bot$		

Computation:

The problem becomes

	Additional Data	Current Query	Configuration
3.	$t : F(b \to \bot)$	$?t : \bot$	$\{t\}$
from 2		$?t_0 : F((a \to b) \to \bot)$	

*From (3) using ** create a new point s:*

	Additional Data	Current Query	Configuration
4.	$s : b \to \bot$	$?s : (a \to b) \to \bot$	$t < s$
	add $s : a \to b$ to	*the database and*	*ask*
5.	$s : (a \to b)$	$?s : \bot$	

from (4) and (5) we ask:

$$?s : a$$

From computation rule (2) and clause 1 of the data we ask

$$?t : F(a \to \bot)$$

From computation rule (2) we ask

$$?s : a \to \bot$$

We add $s : a$ to the data and ask:

	Additional Data	Current Query	Configuration
6.	$s : a$	$?s : \bot$	$t < s$

The query succeeds.

13.6 Modal and temporal Herbrand universes

This section deals with the soundness of our computation rules. In conjunction with soundness it is useful to clarify the notion of modal and temporal Herbrand models. For simplicty we deal with temporal logic with P, F only and transitive irreflexive time or with modal logic with one modality $\Diamond$ and a general binary accessiblitiy relation $<$. We get our clues from some examples:

Example 13.6.1 *Consider the database*

1. $t : a \to \Diamond b$
2. $\Box(b \to c)$
3. $t : a$

The constellation is $\{t\}$.

If we translate the clauses into predicate logic we get:

1. $a^*(t) \to \exists s > t b^*(s)$
2. $\forall x[b^*(x) \to c^*(x)]$

3. $a^*(t)$

Translated into Horn clauses we get after skolemising:

1.1 $a^*(t) \to b^*(s)$

1.2 $a^*(t) \to t < s$

2 $b^*(x) \to c^*(x)$

3 $a^*(t)$

t, s *are Skolem constants.*

From this program, the queries

$$a^*(t), \neg b^*(t), \neg c^*(t), \neg a(s), b(s), c^*(s)$$

all succeed. $\neg$ *is negation by failure.*

It is easy to recognise that $\neg a^*(s)$ succeeds because there is no head which unifies with $a^*(s)$. The meaning of the query $\neg a^*(s)$ in terms of modalities is the query $\Diamond\neg a$.

The question is how do we recognise syntactically what fails in the modal language? The heads of clauses can be whole databases and there is no immediate way of syntactically recognising which atoms are not heads of clauses.

Example 13.6.2 *We consider a more complex example:*

1. $t : a \to \Diamond b$

2. $\Box(b \to c)$

3. $t : a$

4. $t : a \to \Diamond d$

We have added clause (4) to the database in the previous example. The translation of the first three clauses will proceed as before. We will get

1.1 $a^*(t) \to c^*(s)$

1.2 $a^*(t) \to t < s$

2 $b^*(x) \to c^*(x)$

3 $a^*(t)$

We are now ready to translate clause 4. This should be translated like clause (1) into

4.1 $a^*(t) \to d^*(r)$

4.2 $a^*(t) \to t < r.$

The above translation is correct if the set of possible world is just an ordering. Suppose we know further that in our modal logic the set of possible worlds is linearly ordered. Since $t < s \wedge t < r \to s = r \vee s < r \vee r < s$*, this fact must be reflected in the Horn clause database. The only way to do it is to add it as an integrity constraint.*

Thus our temporal program translates into a Horn-clause program with integrity constraints.

This will be true in the general case. Whether we need integrity constraints or not will depend on the flow of time.

Let us begin by translating from the modal and temporal language into Horn clauses. The labelled wff $t : A$ *will be translated into a set of formulas of predicte logic denoted by* Horn (t, A). Horn(t, A) *is supposed to be logically equivalent to* A*. The basic translation of a labelled atomic predicate formula* $t : A(x_1 \dots x_n)$ *is* $A^*(t, x_1 \dots x_n)$*.* A^* *is a formula of a two sorted predicate logic where the first sort ranges over labels and the second sort over domain elements (of the world* t*).*

Definition 13.6.3 *Consider a temporal predicate language with connectives P and F, and $\neg$ for negation by failure.*

Consider the notion of labelled temporal clauses, as defined in definition 13.3.3.

Let Horn(t, A) *be a translation function associating with each labelled clause or goal a set of Horn clauses in the two sorted language described above. The letters t, s which appear in the translation are Skolem constants. They are assumed to be* all different.

We assume that we are dealing with a general transitive flow of time. This is to simplify the translation. If time has extra conditions, ie linearity, additional integrity constraints may need to be added. If time is characterised by non-first order conditions, (eg finiteness) then an adequate translation into Horn clause logic may not be possible.

The following are the translation clauses:

1. *Horn* $(t, A(x_1 \dots x_n)) = A^*(t, x_1 \dots x_n)$, for A atomic.
2. *Horn* $(t, FA) = \{t < s\} \cup \mathit{Horn}(s, A)$
   Horn $(t, PA) = \{s < t\} \cup \mathit{Horn}(s, A)$
3. *Horn* $(t, A \wedge B) = \mathit{Horn}(t, A) \cup \mathit{Horn}(t, B)$
4. *Horn* $(t, \neg A) = \neg \bigwedge \mathit{Horn}(t, A)$
5. *Horn* $(t, A \to F \bigwedge B_j) = \{\bigwedge \mathit{Horn}(t, A) \to t < s\} \cup \bigcup_{B_j} \{\bigwedge \mathit{Horn}(s, A) \wedge C \to D \mid (C \to D) \in \mathit{Horn}(s, B_j)\}$
6. *Horn*$(t, A \to P \bigwedge B_j) = \{\bigwedge \mathit{Horn}(t, A) \to s < t\} \cup \bigcup_{B_j} \{\bigwedge \mathit{Horn}(s, A) \wedge C \to D \mid (C \to D) \in \mathit{Horn}(s, B_j)\}$
7. *Horn*$(t, \Box A) = \mathit{Horn}(x, A)$ where x is a universal variable.

Example 13.6.4 *To explain the translation of $t : A \to F(B_1 \wedge (B_2 \to B_3))$, let us write it in predicate logic. $A \to F(B_1 \wedge B_2 \to B_3))$ is true at t if A true at t implies $F(B_1 \wedge (B_2 \to B_3))$ is true at t. $F(B_1 \wedge (B_2 \to B_3))$ true at t if for some s, $t < s$ and $B_1 \wedge (B_2 \to B_3)$ are true at s.*

Thus we have the translation

$$A^*(t) \to \exists s(t < s \wedge B_1^*(s) \wedge (B_2^*(s) \to B_3^*(s)))$$

Skolemising on s and writing it in Horn clauses we get the conjunction

$A^*(t) \to t < s$
$A^*(t) \to B_1^*(s)$
$A^*(t) \wedge B_2^*(s) \to B_3^*(s)$.

Let us see what the translation Horn *does:*

Horn $(t, A \to F(B_1 \wedge (B_2 \to B_3))) = \{\bigwedge \text{Horn}(t, A) \to t < s\} \cup \{\bigwedge \mathit{Horn}(t, A) \to \mathit{Horn}(s, B_2\} \cup \{\bigwedge \text{Horn}(t, A) \wedge \bigwedge \text{Horn}(s, B_2) \to \bigwedge \text{Horn}(s, B_3)\} =$
$\{A^*(t) \to t < s, A^*(t) \to B_2^*(s), A^*(t) \wedge B_2^*(s) \to B_3^*(s)\}$

We prove soundness of the computation of definition 13.3.6, relative to the Horn clause computation for Horn database in classical logic. In other words if the tranlation *Horn*(t, A) is accepted as intuitively should and the computation of $\mathbf{S}(\rho, <, \Delta, G, t, G_0, t_0, \Theta)$ can be translated isomorphically into a Horn clause computation of the form *Horn*(t, Δ)?*Horn*(t, G) then the soundness of the classical Horn clause computation would imply the soundness of our computation.

This method of translation will also relate our temporal computation to that of ordinary Horn clause computation.

The basic unit of our temporal computation is $\mathbf{S}(\rho, < \Delta, G, t, G_0, t_0, \Theta)$. $t : G$ is the curent labelled goal and $t_0 : G_0$ is the original goal. $(\rho, <, \Delta)$ is the database and Θ is the current substitution. $t_0 : G_0$ is used in the Restart rule. For temporal flow of time which is ordinary transitive $<$, we do not need the Restart Rule. Thus we have to translate $(\rho, <, \Delta)$ to classical logic and translate $t : G$ and Θ to classical logic and see what each computation step of $\mathbf{S}$ of the source translate into the classical logic target.

Definition 13.6.5 *Let $(\rho, <)$ be a constellation and let Δ be a labelled database such that*

$$\rho = \{t \mid \textit{ for some } A, t : A \in \Delta\}$$

Let Horn $((\rho, <), \Delta) = \{t < s \mid t, s \in \rho$ and $t < s\} \cup \cup_{t:A\in\Delta} Horn(t, A)$.

Theorem 13.6.6 (Soundness) $\mathbf{S}(\rho, <, \Delta, G, t, \Theta)$ *succeeds in temporal logic if and only if in the sorted classical logic* Horn $((\rho, <), \Delta)$?Horn(t, G) *succeeds with* Θ.

Proof. The proof is by induction on the complexity of the computation tree of $\mathbf{S}(\rho, <, \Delta, G, t, \Theta)$.

We follow the inductive steps of Definition 13.3.5. The translation of $(\mathbf{\Pi}, \Theta)$ is a conjunction of Horn clause queries, all required to succeed under the same subsitution Θ.

I The empty goal succeeds in both cases.

II $(\mathbf{\Pi}, \Theta)$ fails if for some $\mathbf{S}(\rho, <, \Delta, G, t)$, we have G is atomic and for all $\Box(A \to H) \in \Delta$ and all $t : A \to H \in \Delta, G\Theta$ and $H\Theta$ do not unify. The reason they do not unify is because of what Θ substitutes to the variables u_i.

The corresponding Horn clause predicate programs are
$\bigwedge Horn(x, A) \to H^*(x)$
and
$\bigwedge Horn(t, A) \to H^*(t)$
and the goal is $?G^*(t)$

Clearly since x is a general universal variable, the success of the two sorted unification depends on the other variables and Θ. Thus unification does *not* succeed in the classical predicate case iff it does not succeed in the temporal case.

Rules (1) and (2) deal with the atomic case: The query is $G^*(t)$ and in the database among the data are

$$\bigwedge Horn(t, A) \to H^*(t) \text{ and } \bigwedge Horn(x, A) \to H^*(x)$$

for the cases of $t : A \to H$ and $\Box(A \to H)$ respectively.

For the Horn clause program to succeed $G^*(t)$ must unify with $H^*(t)$. This will hold if and only if the substitution for the domain variables allows unification, which is exactly the condition of definition 13.3.5.

Rules (3), (4g) and (4*g) deal with the case of a goal of the form $?t : FG$. The translation of the goal is $t < u \wedge \bigwedge Horn(u, G)$ where u is an existential variable.

Rule (3) gives success when for some $s, t < s \in \Delta$ and $?s : G$ succeeds. In this case let $u = s$, then $t < u$ succeeds and $\bigwedge Horn(s, G)$ succeeds by the induction hypothesis.

We now turn to the general rules (4g) and (4*g). These rules yield success when for some clause of the form

$$t : A \to F \bigwedge B_j$$

or

$$\Box(A \to F \bigwedge B_j)$$

$\Delta ?t : A$ succeeds and $\Delta \cup \{(s : B_j)\} ?s : G \vee FG$ both succeed. s is a new point.

The translation $\bigwedge Horn(t, A)$ succeeds by the induction hypothesis.

The translation of

$$t : A \to F \bigwedge B_j$$

or

$$\Box(A \to F \bigwedge B_j)$$

contains the following database:

1. $\bigwedge Horn(t, A) \to t < s$
2. For every B_j and every $C \to D$ in $Horn(s, B_j)$ the clause
$\bigwedge Horn(s, A) \wedge C \to D$.

Since $\bigwedge Horn(t, A)$ succeeds we can assume we have in our database

1* $t < s$

2* $C \to D$, for $C \to D \in Horn(s, B_j)$ for some j.

(1*), (2*) were obtained by substituting *truth* in (1) and (2) for $\wedge Horn(t, A)$.

The goal to show is $t < u \wedge \bigwedge Horn(u, G)$.

Again for $u = s, t < u$ succeeds from (1*) and by the induction hypothesis, since $\Delta \cup \{s : B_j\}?s : G \vee FG$ is successful, we get $\bigcup_j Horn(s, B_j)? \bigwedge Horn(s, G) \vee (s < u' \wedge \bigwedge Horn(u', G)$ should succeed, with u' an existential variable.

However, (2*) is exactly $\bigcup_j\ Horn(s, \bigwedge B_j)$. Therefore we have shown that rules (4g) and (4*g) are sound.

Rules (6g) and (6*g) are sound because they are the mirror images of (4g) and (4*g).

The next relevant rules for our soundness cases are (11) - (13). These follow immediately since the rules for $\wedge, \vee, \neg$ are the same in both computations.

Rule 14, the Restart Rule, is definitely sound. If we try to show in general that $\Delta \vdash A$ then since in classical logic $\sim A \to A$ is the same as A ($\sim$ is classical negation) it is equivalent to show $\Delta, \sim A \vdash A$.

If $\sim A$ is now in the data, we can *at any time* try to show A instead of the current goal G. This will give us A (shown) and $\sim A$ (in Data) which is a contradiction and this yield *any goal* including the current goal G.

We have thus completed the soundness proof. ■

13.7 Tractability and persistence

We defined a temporal database Δ essentially as a finite piece of information telling us which temporal formulas are true at what times. In the most general case, for a general flow of time $(T, <)$, all a database can do is to provide a set of the form $\{t_i : A_i\}$, meaning that A_i is true at time t_i and a configuration $(\{t_i\}, <)$, giving the temporal relationships among $\{t_i\}$. A query would be of the form $?t : Q$, where t is one of the t_i. The computation of the query from the data is in the general case exponential, as we found in Section 2, from the case analysis of clause 4 of Definition 13.3.6 and from example 13.3.2. We must therefore analyse the reasons for the complexity and see whether there are simplifying natural assumptions, which will make the computational problem more tractable.

There are three main components which contribute to complexity:

1. The complexity of the temporal formulas allowed in the data and in the query. We allow $t : A$ into the database, with A having temporal operators. So for example, $t : FA$ is allowed and also $t : \mathcal{O}A$. $t : FA$ makes life more difficult because it has in it a hidden Skolem function. It really means $\exists s[t < s$ and $(s : A)]$. This gives rise to case analysis, as we do not know in general where s is. See examples 13.3.10, 13.6.1 and 13.6.2. In this respect $t : \mathcal{O}A$ is a relatively simple item. It says $(t + 1) : A$. In fact any temporal operator which specifies the time is relatively less complex. In practice, we do need to allow data of the form $t : FA$. Sometimes we know an event will take place in the future but we do not know when. The mere fact that A is going to be true can affect our present actions. A concrete example where such a case may arise is when some one accepts a new appointment beginning next year, but has not yet resigned from his old position. We know he is going to resign but we do not know when.

2. The flow of time itself gives rise to complexity. The flow of time may be non-Horn clause (eg linear time, which is defined by a disjunctive axiom.

$$\forall xy[x < y \vee y < x \vee x = y]$$

   This complicates the case analysis of (1) above.

3. Complexity arises from the behaviour. If atomic predicates get truth values at random moments of time, the database can be complex to describe. A very natural simplifying assumption in the case of temporal logic is *persistence*. If atomic statements and their negations remain true for a while then they give rise to less complexity. Such examples are abundant. For example, people usually stay at their residences and jobs for a while. So for example, any payroll or local tax system can benefit from persistence as a simplifying assumption. Thus in databases where there is a great deal of persistence, we can use this fact to simplify our representation and querying. In fact, we shall see that a completely different approach to temporal representation can be adopted when one can make use of persistence.

   Another simplifying assumption is *recurrence*. Saturdays, for example, recur every week, so are paydays. This simplifies the representation and querying. Again, a payroll system would benefit from that.

We said at the beginning that a database Δ is a finitely generated piece of temporal information stating what is true and when. If we do not have any simplifying assumptions, we have to represent Δ in the form $\Delta = \{t_i : A_i\}$ and end up needing the computation rules of Section 2 to answer queries.

Suppose now that we adopt all three simplifying assumptions for our database. We assume that the A_i are only atoms and their negations, we further assume that each A_i is either persistent or recurrent, and let us assume, to be realistic, that the flow of time is linear. Linearity does not make the computation more complicated in this particular case, because we are not allowing data of the form $t : FA$, and so complicated case analysis does not arise. In fact, together with persistence and recurrence, linearity becomes an additional simplifying assumption!

Our aim is to check what form should our temporal logic programming machine take in view of our chosen simplifying assumptions.

First note that the most nautral units of data are no longer of the form:

$$t : A;$$

reading A is true at t but either of the form:

$$[t, s] : A, [t < s];$$

reading A is true in the closed interval $[t, s]$ or the form:

$$t \| d : A;$$

reading A is true at t and recurrently at $t + d$, $t + 2d, \ldots$
that is, every d moments of time.

A is assumed to be a literal (atom or a negation of an atom) and $[t, s]$ is supposed to be a maximal interval where A is true. In $t\|d$, d is supposed to be the minimal cycle for A to recur. The reasons for adopting the notation $[t, s] : A$ and $t\|d : A$ are not mathematical but simply intuitive and practical. This is the way we think about temporal atomic data when persistence or recurrence are present. In the literature there has been a great debate whether to evaluate temporal statements at points or intervals. Some researchers were so committed to intervals that they tended, unfortunately, to disregard any system which uses points. Our position here is clear and intuitive. First perform all the computations using intervals Evaluation at points is possible and trivial. To evaluate $t : A$, ie to ask $?t : A$ as a query from a database, compute the (maximal) intervals at which A is true and see whather t is there. To evaluate $[t, s] : A$ do the same, and check whether $[t, s]$ is a subset.

The query language is left in its full generality. ie we can ask queries of the form $t : A$ where A is unrestricted (eg $A = FB$, etc). It makes sense also to allow queries fo the form $[t, s] : A$, though how exactly we are going to find the answer remains to be seen. The reader should be aware that the data representation language and the query language are no longer the same. This is an important factor. There has been a lot of confusion, especially among the AI community, in connection with these matters. We shall see later that as far as computational tractability is

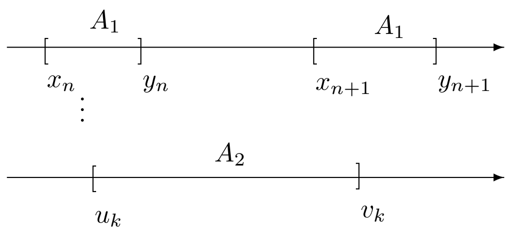


Figure 13.8:

concerned, the restriction to persistent data allows one to strengthen the query language to full predicate quantification over time points.

At this stage we might consider allowing recurrence within an interval, ie we allow something like

"A is true every d days in the interval $[t, s]$."

We can denote this by:

$$[t \| d, s] : A;$$

meaning A is true at $t, t + d, t + 2d$, as long as $t + nd \leq s, n = 1, 2, 3, \ldots$.

We may as well equally have recurrent intervals. An example of that would be taking a two week holiday every year. This we denote by:

$$[t, s] \| d : A, \quad t < s, \quad (s - t) < d;$$

reading A is true at the intervals $[t, s], [t + d, s + d], [t + 2d, s + 2d]$, etc.

The reader should note that adopting this notation takes us outside the realm of first order logic. Consider the integer flow of time. We can easily say that q is true at all even numbers by writing $[0, 0] \| 1$ as a truth set for q and $[1, 1] \| 1$ as a truth set for $\sim q$. (ie q is true at 0 and recurs every 1 unit and $\sim q$ is true at 1 and recurs every 1 unit).

The above seem to be the most natural options to consider. We can already see that it makes no sense any more to check how the computation rules of definition 13.3.6 simplify our case. Our case is so specialised that we may as well devise computation rules directly for it. This should not surprise us. It happens in mathematics all the time. The theory of abelian groups for example, is completely different from the theory of semigroups, although abelian groups are a special case of semigroups. The case of abelian groups is so special that it does not relate any more to the general case.

Let us go back to the question of how to answer a query from our newly defined simplified databases. We start with an even more simple case, assuming only persistence and assuming that the flow of time is the integers. This simple assumption will allow us to present our point of view of how to evaluate a formula at a point or at an interval. It will also ensure we are still within what is expressible in first order logic.

Assume that he atom q is true at the maximal intervals $[x_n, y_n], x_n \leq y_n < x_{n+1}$. Then $\sim q$ is true at the intervals $[y_n + 1, x_{n+1} - 1]$, a sequence of the same form, ie $y_n + 1 \leq x_{n+1} - 1$ and $x_{n+1} - 1 < y_{n+1} + 1$.

It is easy to compute the intervals corresponding to the truth values of conjunctions, we take intersection.

If $I_j = \bigcup_n [x_n^j, y_n^j]$ then $I_1 \cap I_2 = \bigcup_n [x_n, y_n]$ and the points x_n, y_n can be effectively linearly computed. Also, if I_j is the interval set for A_j, the interval set for $U(A_1, A_2)$ can be effectively computed.

In the diagram, $U(A_1, A_2)$ is true at $[u_k, y_n - 1], [u_k, y_{n+1} - 1]$ which simplifies to the maximal $[u_k, y_{n+1} - 1]$.

The importance of the above is that we can regard a query formula of the full language with Until and Since as an operator on the model (database) to give a new database. If the database Δ

gives for each atom or its negation the set of intervals where it is true, then a formula A operators on Δ to give the new set of intervals Δ_A thus to answer $\Delta?t : A$ the question we ask is $t \in \Delta_A$. The new notion is that the query operates on the model.

This approach was adopted by I Torsun and K Manning [Torsun and Manning, 1990] when implementing the query language **USF**. The complexity of computation is polynomial (n^2). Note that although we have restricted the database formulas to atoms, we discovered that at no additional cost we can increase the query language to include the connectives Since and Until. It is well known that in the case of integers the expressive power of Since and Until is equivalent to quantification over time points, Kamp [Kamp, 1968].

To give the reader another glimpse of what is to come, note that intuitively we have several options:

1. We can assume persistence of atoms and negation of atoms. In this case we can express temporal models in first order logic. The query language can be full Since and Until logic. This option does not allow for recurrence. In practical terms this means that we cannot generate or control easily recurrent events. Note that the database does not need to contain Horn clauses as data. Clauses of the form $\Box$(present wff$_1 \rightarrow$ present wff$_2$) are redundant and can be eliminated (this has to be properly proved!).

   Clauses of the form $\Box$ (Past wff$_1 \rightarrow$ Present wff$_2$) are not allowed as they correspond to recurrence.

2. This option wants to have recurrence, is not interested in first order expressibility. How do we generate recurrence?

   The language **USF** (which was introduced for completely different reasons) allows one to generate the database using rules of the form $\Box$ (Past formula $\rightarrow$ Present or Future formula).

The above rules together with some initial items of data of the form $t : A$, A a literal, can generate persistent and recurrent models.

Part V

LDS Metatheory

Chapter 14

Labelled Analytic Deduction

14.1 Introduction

his chapter develops tableaux methodology for *Labelled Deductive Systems*. It is based on a paper of D'Agostino and Gabbay.

We generalize the notion of classical analytic deduction (i.e. deduction via elimination rules) by combining the methodology of Labelled Deductive Systems with the classical system **KE** [D'Agostino and Mondadori, 1994]. *LDS* is a unifying framework for the study of logics and of their interactions. In the LDS approach the basic units of logical derivation are not just formulae but *labelled formulae*, where the labels belong to a given "labelling algebra". The derivation rules act on the labels as well as on the formulae, according to certain fixed rules of propagation. By virtue of the extra power of the labelling algebras, standard (classical or intuitionistic) proof systems can be extended to cover a much wider territory without modifying their structure. The system **KE** is a new tree method for classical analytic deduction based on "analytic cut". **KE** is a refutation system, like analytic tableaux and resolution, but it is essentially more efficient than tableaux and, unlike resolution, does not require any reduction to normal form.

We start our investigation with the family of substructural logics. These are logical systems (such as Lambek's calculus, Anderson and Belnap's relevance logic and Girard's linear logic) which arise from disallowing some or all of the usual structural properties of the notion of logical consequence. This extension of traditional logic yields a subtle analysis of the logical operators which is more in tune with the needs of applications. In section 2 of this Chapter we generalize the classical **KE** system via the *LDS* methodology to provide a uniform refutation system for the family of substructural logics.

The main features of this generalized method are the following: (a) each logic in the family is associated with a "labelling algebra"; (b) the tree-expansion rules (for labelled formulae) are the same for all the logics in the family; (c) the difference between one logic and the other is captured by the conditions under which a branch is declared closed; (d) such conditions depend only on the labelling algebra associated with each logic; (e) classical and intuitionistic negations are characterized uniformly, by means of the same tree-expansion rules, and their difference is reduced to a difference in the labelling algebra used in closing a branch. In this first part we lay the theoretical foundations of our method. In the second part we shall continue our investigation of substructural logics and discuss the algorithmic aspects of our approach.

In Section 2 we start our investigation of substructural logics. We deal with the so-called "intensional" (or "multiplicative") operators and lay down the foundations of our system of Labelled Analytic Deduction. In the second part (forthcoming) we shall:

- Analyse the "extensional" (or "additive") operators.
- Discuss the *general decision problem*: given a database Γ, a query $?A$ and a labelling algebra $\mathcal{A}$, determine whether there exists a tree for $\Gamma ?A$ in which all the branches are closed according to the labelling algebra $\mathcal{A}$. In its full generality the problem is not algorithmically solvable.

However, there are interesting and still quite general subproblems which are solvable and whose solution stems naturally from our approach.

14.2 Tableaux for algebraic LDS

Before embarking on the general study to tableaux in *LDS*, we develop tableaux for algebraic *LDS* more specifically for implicational logics.

Our starting point is the general semantics for implicational logics of Definition 4.1.2. To simplify even further, we disregard φ and $\uplus$ and concentrate only on f. Thus the $\to$ elimination rule is:

$$\frac{\begin{array}{r} x : A \\ y : A \to B \end{array}}{f(y,x) : B}$$

where x, y are elements of an algebra of labels, f is a function (associated with modus ponens) giving the new label after modus ponens has been performed. Different f's or different labelling algebras yield different rules of modus ponens and, possibly, different logics. For example, if we take an arbitrary semigroup as the algebra of the labels, with $\circ$ as multiplication and $f(x,y) = x \circ y$, we can have

$$\frac{\begin{array}{r} x : A \\ y : A \to B \end{array}}{y \circ x : B}$$

The rule of $\to$-introduction is the inverse of $\to$-elimination.

$$\frac{\begin{array}{r} t : A \\ f(y,t) : B \end{array}}{y : A \to B} \text{ with } t \text{ atomic.}$$

To show $y : A \to B$ we assume $t : A$ (with a new atomic label t) and we must prove $z : B$, for a z such that $z = f(y,t)$. In the semigroup case $z = y \circ t$. Clearly, the above rules are a generalization of the traditional rules of natural deduction.

Can labels be used in a similar way to generalize classical analytic deduction? The notion of analytic deduction in classical logic is associated with the idea that all deductive arguments can be modelled by a system of *elimination rules* which "analyse" complex formulae by specifying the *consequences* of their truth and falsity in terms of the truth and falsity of their subformulae. It is mantained that this notion is adequately formalized by the method of analytic tableaux which is, in turn, a direct descendant of Gentzen's (cut-free) sequent calculus. However, it has been shown elsewhere [D'Agostino and Mondadori, 1994] that the theory of classical analytic deduction based on cut-free systems is flawed by several anomalies, both conceptual and technical[1], which are responsible for severe inefficiency, as well as lack of elegance and transparence. It turns out that a better theory of analytic deduction can be built "around" the cut rule. The subformula principle — which is what counts both for the theory and the applications — does not require the *elimination* of cuts, but only their *restriction* to "analytic" applications, i.e. applications involving exclusively subformulae of the theorem. The extra inferential power generated by analytic cuts can then be exploited to simplify the elimination rules. In [D'Agostino and Mondadori, 1994] these ideas lead to the formulation of a new system of analytic deduction for classical logic, the system **KE**, which is proved to be essentially more efficient than the tableau method[2]. The rules of the classical **KE** system are listed in Table 14.1. Notice that all the operational rules have a *linear* format and the only branching rule is a rule expressing the classical Principle of Bivalence. Hence,

[1]This anomalous behaviour culminates in the disconcerting fact (shown in [D'Agostino, 1992]) that cut-free systems cannot polynomially simulate the truth-tables.

[2]The analytic **KE** system can linearly simulate the tableau method, but the tableau method cannot p-simulate the analytic **KE** system. Notice that these results apply also to the case in which the the analytic cut rule is restricted to "mechanical" applications which are generated by a systematic refutation procedure. In fact it can be shown that any (deterministic) proof-procedure based on the tableau method can be turned into an essentially more efficient one based on the analytic **KE** system. On this point see also [D'Agostino, 1994].

Disjunction Rules

$$\frac{\begin{array}{l}TA \vee B\\ FA\end{array}}{TB}\,\mathrm{E}T\vee 1 \qquad \frac{\begin{array}{l}TA \vee B\\ FB\end{array}}{TA}\,\mathrm{E}T\vee 2 \qquad \frac{FA \vee B}{\begin{array}{l}FA\\ FB\end{array}}\,\mathrm{E}F\vee$$

Conjunction Rules

$$\frac{\begin{array}{l}FA \wedge B\\ TA\end{array}}{FB}\,\mathrm{E}F\wedge 1 \qquad \frac{\begin{array}{l}FA \wedge B\\ TB\end{array}}{FA}\,\mathrm{E}F\wedge 2 \qquad \frac{TA \wedge B}{\begin{array}{l}TA\\ TB\end{array}}\,\mathrm{E}T\wedge$$

Implication Rules

$$\frac{\begin{array}{l}TA \to B\\ TA\end{array}}{TB}\,\mathrm{E}T \to 1 \qquad \frac{\begin{array}{l}TA \to B\\ FB\end{array}}{FA}\,\mathrm{E}T \to 2 \qquad \frac{FA \to B}{\begin{array}{l}TA\\ FB\end{array}}\,\mathrm{E}F \to$$

Negation Rules

$$\frac{T\neg A}{FA}\,\mathrm{E}T\neg \qquad \frac{F\neg A}{TA}\,\mathrm{E}F\neg$$

Principle of Bivalence

$$\frac{}{T(A) \mid F(A)}\,\mathrm{PB}$$

Table 14.1: The rules of the classical system **KE**. The applications of the only branching rule can be restricted to subformulae and further controlled by systematic procedures.

all the branches are mutually exclusive, so that the amount of branching is reduced to a minimum. As with the tableau method, derivations are trees of signed formulae and a formula A is proved by a closed tree for FA. For the details see [D'Agostino and Mondadori, 1994].

Can we use the expressive power of this classical system by applying it to a labelled deductive system? Consider the labelled implication rules given above. These followed the "natural deduction" pattern of eliminations and corresponding introductions. The paradigm of analytic deduction, on the other hand, allows for elimination rules only, which specify the consequences of the truth (or falsity) of a complex sentence in terms of the truth (or falsity) of their subformulae. Let us say that a labelled formula $x : A$ *holds*, if it is provable by means of the above implication rules (plus, possibly, other auxiliary rules; the reader is referred to Chapter 4 for the details). We write $TA : x$ for "$x : A$ holds" and $FA : x$ for "$x : A$ does not hold". Given this interpretation of the signs T and F, we can easily obtain "analytic rules" for implication which appear as generalizations of the classical **KE** rules given in Table 14.1:

$$\frac{\begin{array}{l}TA \to B : x\\ TA : y\end{array}}{TB : f(x, y)} \qquad\qquad \frac{FA \to B : y}{\begin{array}{l}TA : t\\ FB : f(y, t)\end{array}}$$

where t is a *new atomic* label.

The general notion of Labelled Analytic Deduction[3] will be formulated and investigated in a later section, but in this section, we concentrate on implicational logics in order to provide

[3]Labels have been used occasionally by several researchers in the context of analytic tableaux. Interesting examples are the method of prefixed tableaux for modal logics in [Fitting, 1983] and the tableau method for many-

a clear illustration of our methodology. These are proof-theoretically defined logics (including Lambek's calculus, Girard's linear logic and Anderson and Belnap's relevance logic[4]) which still lack a reasonable algorithmic proof theory. However, they yield a subtle analysis of the logical operators and are receiving considerable attention in various fields, such as linguistics, philosophy and theoretical computer science, as well as in several application areas. In the sequel we shall show how a *uniform* and *transparent* system of analytic deduction for all these logics can be obtained via a generalization of the classical **KE** system via the LDS methodology. There is no claim that our approach should be the most appealing to everybody's intuition or the most suitable for all needs. However, it will probably be convenient for the fast-growing community of researchers who are involved in the development and implementation of algorithmic proof systems for substructural logics. Moreover, the ideas presented here are not specific to substructural logics. We consider the results in this section as a first step in the development of a general theory of Labelled Analytic Deduction which will be presented in a later chapter.

In our this chpater an algebraic *LDS deduction problem* is defined by three parameters: a *structured database* Δ (for implicational logics we only need one-dimensional structures), a *query* $?A$ and a *labelling algebra* $\mathcal{A}$. The problem asks whether or not A follows from Δ given the notion of information specified by the labelling algebra $\mathcal{A}$ (corresponding to the allowed structural rules). Our system of Labelled Analytic Deduction provides a uniform method for dealing with such deduction problems, where:

a. the tree-expansion rules are *the same* for all logical systems in the family;

b. the difference between logical systems is captured by the conditions under which a branch is declared closed;

c. such conditions depend only on the labelling algebra associated with each logical system;

d. classical and intuitionistic negations are characterized uniformly, by means of *the same* tree-expansion rules, and their difference is reduced to a difference in the labelling algebra used in closing a branch.

In this paper we start our investigation of substructural logics. We deal with the so-called "intensional" (or "multiplicative") operators and lay down the foundations of our system of Labelled Analytic Deduction. In the second part (forthcoming) we shall:

- Analyse the "extensional" (or "additive") operators.

- Discuss the *general decision problem*: given a database Γ, a query $?A$ and a labelling algebra $\mathcal{A}$, determine whether there exists a tree for $\Gamma?A$ in which all the branches are closed according to the labelling algebra $\mathcal{A}$. In its full generality the problem is not algorithmically solvable. However, there are interesting and still quite general subproblems which are solvable and whose solution stems naturally from our approach.

14.2.1 Substructural consequence relations

According to the traditional definition a consequence relation is a relation $\vdash$ between a *set* of formulae Γ and a formula A. The set Γ represents the data, and A a proposition which follows from the data. In many applications of logic, however, situations arise in which the data is not best represented as a set of formulae, but as a richer structure, e.g. a multiset, a sequence, a diagram, etc. These considerations have led to the study of *structured consequence relations* as a new area

valued logics in [Hähnle, 1992]. In his [Wallen, 1990] Wallen also uses labels to capture modal logics in the context of his extension of Bibel's connection method which he relates to analytic tableaux. See also [Maslov, 1969] for a very early example of the use of labels in a sequent calculus to capture intuitionistic logic. For the use of labels as a book-keeping mechanism in the context of natural deduction systems for relevance logic, the main reference is to [Anderson and Belnap Jr, 1975].

[4]Related work on tableau-like methods for relevance logics is contained in [Dunn, 1976, McRobbie and Belnap, 1979, Thistlewaite *et al.*, 1988].

of logical research motivated by computer science needs. In this paper we focus on the case in which the data-structures are *mono-dimensional.*

We take a *consequence relation* as a relation $\vdash$ between *sequences* of formulae and formulae, satisfying:

Identity $$A \vdash A$$

Surgical Cut $$\frac{\Gamma \vdash A \qquad \Delta, A, \Lambda \vdash B}{\Delta, \Gamma, \Lambda \vdash B}$$

Since the Γ, Δ and Λ range over sequences, an application of the cut rule replaces an *occurrence* of the formula A with the sequence Γ exactly in the position of A.

Besides the general axioms of Identity and Surgical Cut given above, the additional axioms which characterize a specific consequence relation can be divided in two groups:

- axioms describing the general properties of $\vdash$, called *structural conditions or rules*;
- axioms describing properties of logical operators, called *operational conditions or rules.*

Structural rules. Consequence relations may or may not obey any of the following conditions describing their structural properties:

Exchange	$\dfrac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$	Contraction	$\dfrac{\Gamma, A, A, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}$
Expansion	$\dfrac{\Gamma, A, \Delta \vdash C}{\Gamma, A, A, \Delta \vdash C}$	Weakening	$\dfrac{\Gamma, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}$

We can think of a sequent $\Gamma \vdash A$ as stating that the piece of information expressed by A is "contained" in the data structure expressed by Γ. Then different combinations of the above structural rules can be seen as different ways of defining the properties of the "information flow" expressed by the turnstile, depending on the structure of the data and on the allowed ways of manipulating it. For instance, if we disallow the Weakening rule, the consequence relation becomes sensitive to the *relevance* of the data to the conclusion: *all* the data has to be used in the derivation process. If also Expansion is disallowed, this notion of relevance extends to the single occurrences of the formulae in the data (as in Anderson and Belnap's system of Relevance Logic, each *occurrence* of a formula in the data has to be used). If contraction is disallowed, each item of data can be used only *once* and has to be replicated if it is used more than once (as in Girard's Linear Logic and its satellites). In this way the process of deriving a formula becomes more similar to a physical process and the consequence relation becomes sensitive to the resources employed. Finally, if Exchange is disallowed, the "chronology" of this process – i.e. the order in which formulae occur in the data – becomes significant (as in the Lambek calculus).

Notice that if Exchange is allowed, the antecedent of a sequent can be regarded as a *multiset*[5].If Contraction and Expansion are also allowed (of course the Weakening rule implies the Expansion rule) the antecedent of a sequent can be regarded as a *set* of formulae, as with the standard Gentzen systems, i.e. the number of occurrences of formulae does not count, nor does the order in which they occur.

The operator $\otimes$. In the classical and intuitionistic sequent calculus the comma occurring in the left-hand side[6] of a sequent is associated with classical conjunction. This operator represents a particular way of combining pieces of information. Its inferential role depends on two components:

[5]For multiset-based consequence relations, the reader is referred to [Avron, 1991].

[6]For the time being we are considering only single-conclusion sequents. Later on we shall consider also multi-conclusion sequents in the context of logical systems with classical (involutive) negation.

the operational rules defining its meaning, and the structural rules defining the meaning of the turnstile. In a general setting, in which the only condition which are required to hold are Identity and Surgical cut, the comma is no longer equivalent to classical conjunction. However, these new consequence relations are still closed under the standard conditions defining classical conjunction. Such condition characterizes a new type of "substructural" conjunction, denoted by $\otimes$ which is no longer classical (because of the failure of some or all of the structural rules), and yet is defined in the same way as its classical version and therefore, in a sense, has the same "meaning", namely:

$$\text{C}_{\otimes} \qquad \Gamma, A\otimes B, \Delta \vdash C \Longleftrightarrow \Gamma, A, B, \Delta \vdash C$$

Clearly, a sequent $\Gamma \vdash A$ becomes equivalent to $\otimes\Gamma \vdash A$ where $\otimes\Gamma$ denotes the $\otimes$-concatenation of the formulae in Γ.

Given Identity and Surgical cut, $\text{C}_{\otimes}$ can be easily shown to be equivalent to the pair of sequent rules:

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A\otimes B, \Delta \vdash C} \tag{14.1}$$

It is not difficult to see that $\text{C}_{\otimes}$ together with Identity and Surgical Cut, implies:

$$\frac{A \vdash B \quad C \vdash D}{A \otimes C \vdash B \otimes D} \tag{14.2}$$

Moreover, let us consider the following restricted form of cut:

$$\text{Transitivity} \qquad \frac{A \vdash B \quad B \vdash C}{A \vdash C}$$

Then, it is easy to see that Transitivity, together with Identity, $\text{C}_{\otimes}$ and (14.2), imply Surgical Cut.

The connective $\rightarrow$. Implication is usually characterized by the following condition:

$$\text{C}_{\rightarrow} \qquad \Gamma \vdash A \rightarrow B \Longleftrightarrow \Gamma, A \vdash B$$

which, under the assumption of Identity and Surgical cut is equivalent to the pair of rules:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \qquad \frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Delta, A \rightarrow B, \Gamma \vdash C} \tag{14.3}$$

The operator $\neg$. Negation can be defined *á la* Johansson, in terms of implication and a constant $\bot$. Then the condition $\text{C}_{\rightarrow}$ above immediately yields:

$$(\text{C}_{\neg}) \qquad \Gamma, A \vdash \bot \Longleftrightarrow \Gamma \vdash \neg A.$$

Again, it is easy to see that, given Identity and Surgical cut, the condition $\text{C}_{\neg}$ is equivalent to the following pair of sequent rules:

$$\frac{\Gamma, A \vdash \bot,}{\Gamma \vdash \neg A} \qquad \frac{\Gamma \vdash A}{\neg A, \Gamma \vdash \bot.} \tag{14.4}$$

Classical negation and the operator $\uplus$. Classical negation is characterized by the additional condition expressed by the "double negation law":

$$\neg\neg A \vdash A \tag{14.5}$$

which, given $C_\neg$ — or equivalently the sequent rules in (14.4) —, Identity and Surgical Cut, is equivalent to the rule:

$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A} \tag{14.6}$$

Let us now define a binary operator $\uplus$ as follows:

$$A \uplus B =_{\text{def}} \neg A \to B \tag{14.7}$$

If (14.5) holds *and* Exchange is allowed, $\uplus$ is commutative and associative. Moreover the following equivalences also hold:

$$\Gamma, A \vdash B \Longleftrightarrow \Gamma \vdash \neg A \uplus B \qquad\qquad \Gamma, \neg A \vdash B \Longleftrightarrow \Gamma \vdash A \uplus B \tag{14.8}$$

Thus, in the logics satisfying the Exchange property and the double negation law (14.5) we can naturally introduce *multi-conclusion* sequents of the form

$$\Gamma \vdash \Delta$$

where Δ is, like Γ, a list of formulae, with the intended meaning $\otimes\Gamma \vdash \uplus\Delta$ (by $\otimes\Gamma$ and $\uplus\Delta$ we denote, respectively, the $\otimes$-concatenation of the formulae in Γ and the $\uplus$-concatenation of the formulae in Δ). The properties of the (classical) negation operator allow us to translate back and forth from the single-conclusion formulation to the multi-conclusion one. The operator $\uplus$ corresponds to the comma in the right-hand side of a multi-conclusion sequent, just as the operator $\otimes$ corresponds to the comma in the left-hand side. Under these circumstances the following equivalence can also be easily shown:

$$A \otimes B \dashv\vdash \neg(A \to \neg B) \tag{14.9}$$

When all the usual structural rules are allowed, $\uplus$ is clearly equivalent to classical disjunction.

Notice that, since in the logics satisfying (14.5) and Exchange both $\otimes$ and $\uplus$ are commutative, the antecedent and the succedent of a sequent can be regarded as *multisets* rather than just sequences, and Surgical Cut can be replaced by the more familiar (though not necessarely more natural):

$$\frac{\Gamma \vdash A, \Delta \qquad \Lambda, A \vdash \Pi}{\Gamma, \Lambda \vdash \Delta, \Pi}$$

The reader can easily derive multi-conclusion versions of $C_\otimes$, $C_\to$ and $C_\neg$, and the corresponding sequent rules. Clearly $\uplus$ can be defined directly by the following condition:

$C_\uplus$ $$\Gamma \vdash \Delta, A, B, \Lambda \Longleftrightarrow \Gamma \vdash \Delta, A \uplus B, \Lambda$$

which, under the given assumptions, is in turn equivalent to the following pair of rules:

$$\frac{\Gamma, A \vdash \Delta \quad B, \Lambda \vdash \Pi}{\Gamma, A \uplus B, \Lambda \vdash \Delta, \Pi} \qquad\qquad \frac{\Gamma \vdash \Delta, A, B, \Lambda}{\Gamma \vdash \Delta, A \uplus B, \Lambda} \tag{14.10}$$

14.2.2 Information frames

As suggested above a theory of labelled analytic deduction could be developed entirely in proof-theoretical terms. Let us consider, for instance, the analytic rules for implication we obtained from the labelled natural deduction rules. The labels may be interpreted as lists of formulae, the binary operation $\circ$ as list concatenation, and the signed expressions $TA : x$ and $FA : x$ as, respectively, "A is provable from x" and "A is not provable from x". So the use of signs does not need to involve any "semantic" consideration. In general the labelling algebra can encode any type of information which we may want to propagate via the derivation rules. Thus, in the general case, signed formulae can be considered as simply connecting formulae from a given logical language and labels from a given labelling algebra. The latter may come from any source, even a specific implementation in some application area. We do not need to commit *a priori* to any fixed interpretation of the labels.

On the other hand, if we formulate a set of rules for labelled *signed* formulae, it is often possible to read them as *semantic* rules for some Kripke-style semantics: the labels are now interpreted as points of a space to which the notion of truth and falsity are relativized. The same labelled signed formulae $TA : x$ and $FA : x$ can be read as "A is true at x" and "A is false at x" respectively. The labelling algebra describes the structure of this valuation space. So, the traditional distinction between "syntax" and "semantics" becomes blurred: what ultimately counts is the algebra of the labels.

We begin by giving an informal description of the structure of our labelling algebras. We shall sometimes use a "semantic" terminology to bring out its relation with other familiar structures which are traditionally studied as part of semantics. However a purely "syntactic" reading is always possible. We stress that our aim is *not* to provide "another" semantics[7] for substructural logics, but to develop a system of labelled analytic deduction. "Semantics", here, is a by-product of algorithmic problems[8].

Information tokens. We begin with the notion of resource propositional *LDS* as defined in Definition 4.2.1. We assume the labels are *information tokens.* Information tokens may *verify* sentences. Let us write $x \Vdash A$ for "x verifies A" or "A is true at x". We shall also say that "A is false at x" if $x \nVdash A$ (our semantics is bivalent and "false" is the same as "non-true"). We write $x \sqsubseteq y$ whenever the information token y verifies all the sentences that are verified by x, so that y contains at least the same information as x (possibly more). Hence:

$$x \Vdash A \text{ and } x \sqsubseteq y \Longrightarrow y \Vdash A \tag{14.11}$$

The relation $\sqsubseteq$ is a preorder, i.e. it is reflexive and transitive. We assume that two "indiscernible" information tokens, which verify exactly the same sentences, are identical. Hence $\sqsubseteq$ is a partial order.

Moreover, it is natural to assume that verification is closed under deducibility, that is:

$$x \Vdash A \text{ and } A \vdash B \Longrightarrow x \Vdash B \tag{14.12}$$

An information token can be considered equivalent to the set of the sentences that it verifies (each set will then be closed under $\vdash$). Then the partial ordering $\sqsubseteq$ would be set-inclusion, and $x \Vdash A$ would just mean that $A \in x$. We shall call this interpretation *the canonical interpretation.* It is important to realize that in a context in which the contraction rule may be disallowed, (i.e. our logic is "resource-sensitive") an information token may verify each formula in a set Individually), but not *all* the formulae in the set (collectively). The information token is lost once it has been "spent" to verify a formula, and we need a "fresh" copy of it to verify another formula. This is no longer the case when the contraction rule is allowed and, so to speak, our resources can be multiplied indefinitely. Thus in terms of Definition 4.2.1 '$\leq$' is '$\sqsubseteq$', φ is always true, the labelling function $f(x, y)$ will turn out to be $x \circ y$ defined below and $x \uplus y$ will turn out to be $x \circ y$ when used. See also Definition 4.1.2.

Composition of information tokens. Given any two information tokens x and y we can combine them to form a new information token $x \circ y$. The operation $\circ$ is associative but, in

[7]Algebraic semantics for intuitionistic substructural logics is given by Dôsen in [Dôsen, 1988]. A related semantics for linear logic is given in [Avron, 1988a]. These semantics are ultimately more or less straightforward adaptations of the Lindenbaum-Tarski method to the new consequence relations. In [Dôsen, 1989] Dôsen also gives relational (possibile-world) semantics for subsystems of intuitionistic logic, in the style of the Kripke semantics. A Kripke-style semantics for linear logic is also given in [Allwein and Dunn, 1993]. These Kripke-style semantics ultimately stem from Urquhart's semantics for relevant implication described in [Urquhart, 1972]. For related semantic investigations into substructural logics see also [Ono, 1993] and [Sambin, 1993]. Under certain circumstances, semantics for arbitrary consequence relations can be generated in a systematic way. On this point see [Gabbay, 1993b, Gabbay and Ohlbach, 1993a, Gabbay and Ohlbach, 1993b, Gabbay, 1993c].

[8]We do not exclude, of course, that some readers may find the "semantic" interpretation of the rules more familiar or heuristically useful. From this point of view, we stress that our "semantics" covers all substructural logics weaker than *classical* logics: classical and intuitionistic negation are characterized in a uniform way, and their difference is reduced to the algebra of the labels (or, if you wish, to the structure of the valuation space).

general, it is neither commutative nor idempotent. Again, a composition $x_1 \circ \cdots \circ x_n$ of atomic information tokens verifies sentences in a particularly strict, resource-sensitive way: the atomic information tokens have *all* to be used to verify the sentence *in the order* in which they occur.

Thus, in general,

$$\begin{array}{rcl} x \circ y & \not\sqsubseteq & y \circ x \\ x & \not\sqsubseteq & x \circ x \\ x \circ x & \not\sqsubseteq & x \\ x & \not\sqsubseteq & x \circ y \end{array}$$

Information frames. So far we have assumed that information tokens form a partially ordered set and are closed under a composition operation $\circ$. We now make two further assumptions which are both very natural under the canonical interpretation suggested above. First we assume that the partial ordering induces a *complete lattice* (under the canonical interpretation information tokens form a complete lattice of sets). Secondly, we assume that the operation $\circ$ is *continuous* in both arguments. This means that the operation preserves limits:

$$\bigsqcup\{a \circ x | x \in S\} = a \circ \bigsqcup S \quad \text{and} \quad \bigsqcup\{x \circ a | x \in S\} = \bigsqcup S \circ a \tag{14.13}$$

for all *directed*[9] sets S of information tokens. The continuity of $\circ$ is obvious under the canonical interpretation, if we define $x \circ y =_{\text{def}} \{A \otimes B | A \in x, B \in y\}$.

Notice that the continuity of $\circ$ implies that the composition operation $\circ$ is *order-preserving*:

$$x_1 \sqsubseteq y_1 \quad \text{and} \quad x_2 \sqsubseteq y_2 \quad \text{imply} \quad x_1 \circ x_2 \sqsubseteq y_1 \circ y_2 \tag{14.14}$$

A distinguished information token is the *neutral* information token denoted by 1. This acts like an identity element (thus $\circ$ is a monoid operation[10]), so that its combination with an arbitrary token x is equal to x itself. Notice that this identity element 1 does not coincide, in general, with the bottom element of the lattice, but it is the top element of a distinguished subset of information tokens: the tokens which verify the theorems of the given logical system. (Under the canonical interpretation 1 is the set of all theorems (as opposed to $\varnothing$).)

The structure described so far will be called an *information frame*. The following definition summarizes our previous presentation:

Definition 14.2.1 *An* information frame *is a structure $\mathcal{L} = (P, \circ, 1, \sqsubseteq)$ such that:*

1. *P is a non-empty set of elements called* information tokens*;*
2. *P is a complete lattice under $\sqsubseteq$;*
3. *$\circ$ is a binary operation on P which is*
   (a) *associative: $x \circ (y \circ z) = (x \circ y) \circ z$;*
   (b) *continuous in both arguments: for every directed family $\{z_i\}$, $\bigsqcup\{z_i \circ x\} = \bigsqcup\{z_i\} \circ x$ and $\bigsqcup\{x \circ z_i\} = x \circ \bigsqcup\{z_i\}$;*
4. *$1 \in P$ and for every $x \in P$, $x \circ 1 = 1 \circ x = x$;*

Recall that the continuity of $\circ$ implies that this operation is order-preserving: $x_1 \sqsubseteq x_2$ implies $x_1 \circ y \sqsubseteq x_2 \circ y$ and $y \circ x_1 \sqsubseteq y \circ x_2$. In the sequel we shall often identify an information frame with the set of its elements and use $\mathcal{L}$ to refer ambiguously to both. The context will ensure that no confusion arises.

We can define classes of information frames which satisfy additional conditions on the ordering $\sqsubseteq$:

[9]Recall that a set S is *directed* when every finite subset of S has un upper bound in S.

[10]Recall that a *monoid* is a structure $(P, \circ, 1)$, where P is a non-empty set, $\circ$ is an associative binary operation on $\circ$, and 1 is an identity element, i.e. a constant such that $1 \circ x = x \circ 1 = x$ for all x.

1	commutative frames	$\dfrac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$
2	contractive frames	$\dfrac{\Gamma, A, A, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}$
3	expansive frames	$\dfrac{\Gamma, A, \Delta \vdash C}{\Gamma, A, A, \Delta \vdash C}$
4	monotonic frames	$\dfrac{\Gamma, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}$

Table 14.2: Correspondence between classes of frames and structural rules.

Definition 14.2.2 *We say that an information frame is:*

commutative	*if*	$x \circ y \sqsubseteq y \circ x$
contractive	*if*	$x \circ x \sqsubseteq x$
expansive	*if*	$x \sqsubseteq x \circ x$
monotonic	*if*	$x \sqsubseteq x \circ y$

In the sequel different substructural logics will be seen to correspond to different classes of information frames in the expected way, each condition on $\sqsubseteq$ being associated with a structural property of consequence relations. The correspondence is summarized in Table 14.2.

Truth, Falsity and valuations. To complete our analysis we have to make a few remarks on the notions of "truth" and "falsity". As said before we can think of an information token as a set of formulae (intuitively, the set of formulae which are verified by it). Under this canonical interpretation, that a formula A is verified by a token x simply means that $A \in x$. Hence, it is obvious that truth and falsity are preserved under intersections and unions of tokens.

In the context of abstract information frames it is convenient to decompose this property into the two properties that (a) truth and falsity are preserved under *finite* joins and meets and (b) that the functions "y is the truth-value of A at point x" (for each given formula A) are continuous. It follows that truth and falsity are preserved under arbitrary joins and meets.

In particular, this implies that if some information token verifies a sentence A, there exists *the least* information token that verifies A. Intuitively this represents the "information content" of A itself, and it is the set $\{B|A \vdash B\}$ under the canonical interpretation. We shall call such an information token *A-characteristic.*

These properties of the notions of truth and falsity can be expressed in terms of "valuations".

Let $\mathcal{L}$ be an information frame. A *valuation* over $\mathcal{L}$ is a function $v : \mathcal{F} \times P \mapsto \{T, F\}$ where $\mathcal{F}$ is the set of formulae of the language and and P is the set of information tokens in the frame $\mathcal{L}$, satisfying the following conditions:

1. For all formulae A, if $v(A, x) = T$ and $x \sqsubseteq y$, then $v(A, y) = T$.
2. For all formulae A, and all information tokens x, y: if $v(A, x) = T$ and $v(A, y) = T$, then $v(A, x \sqcap y) = T$.
3. For all formulae A, and all information tokens x, y: if $v(A, x) = F$ and $v(A, y) = F$, then $v(A, x \sqcup y) = F$.
4. For each given A, v is a continuous mapping, i.e. $\bigsqcup\{v(A, x)|x \in S\} = v(A, \bigsqcup S)$ and $\bigsqcup$

As usual, by $v(A, x) = T$ we mean "A is true at x" or "$x \Vdash A$", and by $v(A, x) = F$ we mean "A is false at x" or "$x \nVdash A$".

As we mentioned, it follows from our definitions that truth and falsity are "continuous", i.e. they are preserved under arbitrary joins and meets. In the next section we shall present specific valuation rules which analyse each logical operator. Note that valuations v are really assignments h in semantical terms. Compare with, for example, Definition 4.1.2.

Remark 14.2.3 *The reader is reminded that in this resource-sensitive context, the information token $x \sqcup y$, which is the lattice join of x and y contains all the information contained in either x or y, but might* not *contain all the information contained in* both x and y. *In other words, it is sufficient to to obtain whatever can be obtained from either x or y, but it is not sufficient, in general, to obtain what can be obtained from their composition $x \circ y$.*

As said before, we can always think of a valuation simply as *a relation between formulae and labels*. Under this more general interpretation, $v(A, x) = T$ means "A can be derived with label x" and $v(A, x) = F$ means "A cannot be derived with label x" (or, equivalently, it represents *goal*: "derive A with label x"). A closed tree for $FA : x$ will then be interpreted as a refutation of the assumption "A cannot be derived with label x" (or as a successful derivation of the goal).

As usual, a formula will be valid in a given logic when it is true in all valuations over every information frame belonging to a certain class (associated with the logic). Our labelled system of analytic deduction will prove theorems by showing that a *counterexample*, i.e. a falsifying valuation, is impossible.

In the next section we shall introduce **KE**-type *elimination* rules for labelled signed formulae. These rules will "analyse" the logical operators by specifying the *consequences* of labelled signed formulae, i.e. expressions of the form $TA : x$ and $FA : x$.

14.2.3 Labelled analytic deduction

Information frames, described in the previous section, play a key-role in our generalization of classical analytic deduction. In their "semantic" interpretation they provide a space to which the classical notions of truth and falsity are relativized (much as in Kripke's semantics for intuitionistic logic). In the more general, proof-theoretical, interpretation they provide a space of *labels* to which the notion of derivability is relativized. The use of labelled signed formulae in our system of analytic deduction brings out the correspondence between the two interpretations.

Different classes of information frames (as in Definition 14.2.2) will be identified by different *labelling algebras*. In the context of the present paper, a labelling algebra will be described by a finite set of expressions of the form $\alpha \sqsubseteq \beta$ where α and β contain variables which are intended as universally quantified. (Identities will of course be represented by pairs of such expressions.) For example, the labelling algebra corresponding to the class of commutative and contractive frames is described by the following expressions:

$$\begin{array}{ll} x \circ (y \circ z) \sqsubseteq (x \circ y) \circ z & \quad x \circ y \sqsubseteq y \circ x \\ (x \circ y) \circ z \sqsubseteq x \circ (y \circ z) & \quad 1 \circ x \sqsubseteq x \\ x \circ x \sqsubseteq x & \end{array}$$

Our next step consists in generalizing the rules of the classical refutation system **KE** to the new setting.

Generalized Bivalence and Non-contradiction The classical notions of truth and falsity are governed by two basic principles: the Principle of Bivalence (every proposition is either true or false) and the Principle of Non-Contradiction (no proposition is both true and false). In the **KE** system these principles are expressed directly as *rules*. The Principle of Bivalence is expressed by the only branching rule of the system[11], the rule PB, and the Principle of Non-Contradiction by the rule for closing a branch (as in the tableau method).

[11]This fundamental principle of classical semantics is not expressed by the rules of the tableau method if not in a very indirect way. On this point see [D'Agostino, 1992].

These principles still apply to our generalized framework except that we refer them to labelled propositions. Given an arbitrary proposition A and an arbitrary information token x, either it is true that x verifies A or it is false that x verifies A (i.e. $x \Vdash A$ or $x \nVdash A$). So the classical rule of bivalence can be generalized as follows:

$$\frac{}{TA : x \mid FA : x} \tag{14.15}$$

for every label x.

According to the proof-theoretical reading of signed formulae, the rule PB is a "lemma introduction" rule: if A can be derived with label x (i.e. if the right-hand subtree is closed), this fact can be used as a lemma (in the left-hand subtree).

A similar generalization applies to the Principle of Non-Contradiction: it is impossible that the same formula is verified and not verified by the same information token. Moreover, if a formula is verified by a token x it is also verified by every token y such that $x \sqsubseteq y$. Hence, the following generalized closure rule is justified:

$$\frac{\begin{array}{c} TA : x \\ FA : y \end{array}}{\times} \tag{14.16}$$

provided that $x \sqsubseteq y$.

Proof-theoretically speaking, if A can be derived with label x and the algebra of the labels is such that whatever can be derived with label x can also be derived with label y, then A can also be derived with label y and the goal expressed by $FA : y$ is achieved.

In fact, to cover all the logics in the family we are investigating, we need an even more general closure rule. Let us consider frames which are expansive but *not* monotonic. Suppose a formula A is verified by each of the tokens $x_1, \ldots, x_n$. By definition of valuation there is the *minimum* token at which A is true. Let a be such a minimum token. If the frame is expansive we have

$$a \sqsubseteq \overbrace{a \circ \cdots \circ a}^{n \text{ times}}$$

Therefore, since $a \sqsubseteq x_i$ for $i = 1, \ldots, n$, A is verified also by the token $x_1 \circ \cdots \circ x_n$. Hence, a branch containing all $TA : x_i$ and $FA : x_1 \circ \cdots x_n$ should be considered closed. This agrees with the fact that in any logic which allows expansion, the following rule can be derived:

$$\frac{\Gamma \vdash A \qquad \Delta \vdash A}{\Gamma, \Delta \vdash A}$$

This problem can be overcome by introducing a more general closure rule of which the previous one is a special case:

$$\frac{\begin{array}{c} TA : x_1 \\ \vdots \\ TA : x_n \\ FA : y \end{array}}{\times} \tag{14.17}$$

provided that $\bigsqcup$

Elimination rules for $\rightarrow$. It is natural to assume that if an information token x verifies a complex sentence of the form $A \rightarrow B$, and another information token y verifies A, then the composition of the two will verify B, i.e. :

$$x \Vdash A \rightarrow B \Longrightarrow \forall y, y \nVdash A \;\text{ or }\; x \circ y \Vdash B \tag{14.18}$$

This justifies immediately the following generalizations of the **KE** elimination rules for $TA \rightarrow B$:

$$\begin{array}{c} TA \rightarrow B : x \\ TA : y \\ \hline TB : x \circ y \end{array} \qquad \begin{array}{c} TA \rightarrow B : x \\ FB : x \circ y \\ \hline FA : y \end{array} \tag{14.19}$$

If we assume that the converse of (14.18) also holds, namely:

$$x \Vdash A \rightarrow B \Longleftarrow \forall y, y \nVdash A \;\text{ or }\; x \circ y \Vdash B \tag{14.20}$$

we can justify also the following generalization of the **KE** elimination rule for $FA \rightarrow B$:

$$\begin{array}{c} FA \rightarrow B : x \\ \hline TA : a \\ FB : x \circ a \end{array} \quad \boxed{\text{with } a \text{ a } \textit{new atomic} \text{ label}} \tag{14.21}$$

The new atomic label a stands for an information token which instantiates the y in the existentially quantified expression which is obtained by contraposing (14.20). In the sequel we shall see that a "liberalized" version of this rule can be conveniently adopted, to the effect that the atomic label introduced by the rule does not have to be new.

Again, the rules for implication given above are easily seen to be sound under the proof-theoretical interpretation. The reader can check that the implication rules (together with our PB rule) are equivalent to the if-and-only-if valuation clause resulting from the conjunction of (14.18) and (14.20), namely:

$$v(A \rightarrow B, x) = T \Longleftrightarrow \forall y, v(A, y) = F \;\text{ or }\; v(B, x \circ y) = T \tag{14.22}$$

This condition is formally identical to Urquhart's semantics of relevant implication [Urquhart, 1972]. However, here $\circ$ is not a semilattice join, but a monoid operation. Thus the above condition is sufficient to characterize the whole family of substructural implication systems and not just the implication fragment of **R**.

Labelled KE-trees for implicational logics. The operational rules described above are valid for all the logical systems in the family, no matter which structural rules they satisfy.

The extra-conditions on the $\sqsubseteq$ relation, which characterize a particular class of information frames, might be expressed as *structural* rules. Such rules are listed in Table 14.3. However we shall see that such structural rules are redundant and can be eliminated in favour of the general closure rule.

Definition 14.2.4

1. *A* derivation *is a tree of labelled signed formulae, or* LS-formulae *for short, constructed according to the rules of the system starting from a (posibly empty) sequence of LS-formulae called* initial *or* assumption *LS-formulae.*

2. *A branch ϕ is* closed for the labelling algebra $\mathcal{A}$ *if ϕ contains a set of LS-formulae $\{TA : x_1, \ldots, TA : x_n, FA : y\}$ such that*

$$\bigsqcup$$

Structural Rules	
Permutation	Contraction
$\dfrac{TA : z \circ x \circ y \circ v}{TA : z \circ y \circ x \circ v}$	$\dfrac{TA : z \circ x \circ x \circ y}{TA : z \circ x \circ y}$
Expansion	Weakening
$\dfrac{TA : z \circ x \circ y}{TA : z \circ x \circ x \circ y}$	$\dfrac{TA : z \circ x \circ v}{TA : z \circ x \circ y \circ v}$

Table 14.3: Structural rules (to be shown redundant).

5. *A formula A is* provable for the labelling algebra $\mathcal{A}$ *from the sequence of assumptions* Γ *if and only if there is a tree $\mathcal{T}$ such that (i) $\mathcal{T}$ is generated by applications of the rules starting from a sequence of initial LS-formulae*

$$TB_1 : b_1, \ldots, TB_n : b_n, FA : b_1 \circ \cdots \circ b_n$$

where $B_1, \ldots, B_n = \Gamma$; (ii) $b_i \neq b_j$ whenever $i \neq j$; (iii) $\mathcal{T}$ is closed for $\mathcal{A}$.

Redundancy of the structural rules. As mentioned above, the structural rules can be eliminated in favour of the branch closure rule, which allows us to formulate a *uniform* tree method for all the logics in the family, and reduce the difference between them to the conditions under which a branch is declared closed. In the sequel by "a tree for $\Gamma \vdash A$" we shall mean a tree starting with a sequence

$$TB_1 : b_1, \ldots, TB_n : b_n, FA : b_1 \circ \cdots \circ b_n$$

specified as above.

Proposition 14.2.5 *Every closed tree $\mathcal{T}$ for $\Gamma \vdash A$ can be transformed into an equivalent closed tree such that, in each branch, no application of an operational rule follows an application of a structural rule.*

Proof. We define the *rank* of a node n as follows: *rank*$(n) = 0$ if n is a leaf-node and *rank*$(n) = k+1$, where k is the maximal rank of the successors of n, for every non-leaf node.

Let us say that an LS-formula is *misplaced* if it is at the same time the conclusion of a structural rule and a premiss of an operational rule. Given a tree $\mathcal{T}$, let d the number of misplaced formulae of maximal rank, and r be equal to such maximal rank if $d > 0$ or to 0 if $d = 0$. We define $(r', d') < (r, d)$ if $r' < r$ or, $r' = r$ and $d' < d$. Let us denote by $\mu(\mathcal{T})$ the pair (r, d) associated with the tree $\mathcal{T}$. Clearly if $\mu(\mathcal{T}) = (0, 0)$ no LS-formula in the tree is at the same time the conclusion of a structural rule and the premiss of an operational rule. Although it may be the case that a formula which is the premiss of an operational rule *occurs* in a branch after a formula which is the conclusion of a structural rule, the former does not depend on the latter. In such a situation, the tree can always be re-arranged so that no application of an operational rule follows an application of a structural rule.

Now, let $\mathcal{T}$ be a tree such that $\mu(\mathcal{T}) > (0, 0)$. Let $TA : x$ be a misplaced LS-formula of maximal rank. Suppose $TA : w$, with $w \sqsubseteq x$, is the premiss of the application of the structural rule from which $TA : x$ results. For every branch ϕ passing through $TA : x$, let $TC_1 : z_1[x], \ldots, TC_n : z_n[x]$, where $z_i[x]$ means that the label x occurs in z_i, be all the LS-formulae in ϕ which result from the

Frames				Implication systems
commutative $x \circ y = y \circ x$	contractive $x \circ x \sqsubseteq x$	expansive $x \sqsubseteq x \circ x$	monotonic $x \sqsubseteq x \circ y$	
no	no	no	no	Lambek's
yes	no	no	no	Linear
yes	yes	no	no	Relevant
yes	yes	yes	no	Mingle
yes	yes	yes	yes	Intuitionistic

Table 14.4: Correspondence between implication systems and classes of information frames

application of a rule with $TA : x$ as one of the premises. Thus ϕ will have the following form:

$$\begin{array}{c} TA : w \\ TA : x \\ \vdots \\ TC_1 : z_1[x] \\ \vdots \\ TC_n : z_n[x] \\ \vdots \end{array}$$

Let $\mathcal{T}'$ obtained from $\mathcal{T}$ by: (i) deleting $TA : x$; (ii) replacing each $TC_i : z_i$ with the pair of LS-formulae $TC_i : z_i(x/w), TC_i : z_i$ (one after the other), where $z_i(x/w)$ denotes the result of substituting x with w in z_i. Then $\mathcal{T}'$ is a sound tree equivalent to $\mathcal{T}$. Moreover, $\mu(\mathcal{T}') < \mu(\mathcal{T})$. ■

The above proposition implies that the structural rules can be eliminated in favour of the general closure rule. To see this, consider that the structural rules always yield a formula signed with T as conclusion. Moreover, each structural rule re-labels a labelled signed formula $TA : x$ with a label y such that $x \sqsubseteq y$ in the class of frames associated with the structural rule. So, if a branch is closed by virtue of a set of LS-formulae $\{TA : x_1, \ldots, TA : x_n, FA : y$ with $\bigsqcup$

also be read off the same completed tree. In the following examples the labelled **KE**-rules used in the derivation are indicated on the right column. (For these rules we use the same names used for the classical rules; see Table 14.1 above.)

We shall adopt the following notational device. We shall write just $\vdash$ to express the fact that the consequence relation $\vdash$ under consideration is not assumed to be closed under any of the structural rules. We shall write $\vdash^{i_1 \cdots i_n}$ $(n \leq 4)$, to express the fact that the given consequence relation is closed under the structural rules corresponding to the rows $i_1, \ldots, i_n$ in Table 14.2. In turn, a substructural consequence relation $\vdash^{i_1 \cdots i_n}$ corresponds to the class of information frames such that their partial ordering $\sqsubseteq$ satisfies all the conditions specified in rows $i_1 \ldots i_n$. In this way each class of consequence relation is associated with a class of information frames[12].

Example 14.2.6 $\vdash (A \to B) \to ((C \to A) \to (C \to B))$

Proof.

(1)	$F(A \to B) \to ((C \to A) \to (C \to B)) : 1$	
(2)	$TA \to B : a$	E$F \to$(1)
(3)	$F(C \to A) \to (C \to B) : a$	E$F \to$(1)
(4)	$TC \to A : b$	E$F \to$(3)
(5)	$FC \to B : a \circ b$	E$F \to$(3)
(6)	$TC : c$	E$F \to$(5)
(7)	$FB : a \circ b \circ c$	E$F \to$(5)
(8)	$TA : b \circ c$	E$T \to$(4), (6)
(9)	$TB : a \circ b \circ c$	E$T \to$(2), (8)
$\times$		

■

Example 14.2.7 $\vdash^1 (A \to (B \to C)) \to (B \to (A \to C))$

Proof.

(1)	$F(A \to (B \to C)) \to (B \to (A \to C)) : 1$	
(2)	$TA \to (B \to C) : a$	E$F \to$(1)
(3)	$F(B \to (A \to C)) : a$	E$F \to$(1)
(4)	$TB : b$	E$F \to$(3)
(5)	$FA \to C : a \circ b$	E$F \to$(3)
(6)	$TA : c$	E$F \to$(5)
(7)	$FC : a \circ b \circ c$	E$F \to$(5)
(8)	$TB \to C : a \circ c$	E$T \to$(2), (6)
(9)	$TC : a \circ c \circ b$	E$T \to$(4), (8)
$\times$		

■

Notice how the previous derivation fails in every logic characterized by non-commutative frames since $a \circ c \circ b \sqsubseteq a \circ b \circ c$ does not hold true in general.

Example 14.2.8 $\vdash^{12} (A \to (A \to B)) \to (A \to B)$

(1)	$F((A \to (A \to B)) \to (A \to B)) : 1$	
(2)	$T(A \to (A \to B)) : a$	E$F \to$(1)
(3)	$F(A \to B) : a$	E$F \to$(1)
(4)	$TA : b$	E$F \to$(3)
(5)	$FB : a \circ b$	E$F \to$(3)
(6)	$T(A \to B) : a \circ b$	E$T \to$(2), (4)
(7)	$TB : a \circ b \circ b$	E$T \to$(6), (4)
$\times$		

[12]We have just considered four structural rules which arise from the Gentzen tradition. It is obvious that the list of structural rules could be extended *ad libitum* (for a more comprehensive list see [Dôsen, 1988]). However the general pattern which associates with each structural rule a condition on the partial ordering $\sqsubseteq$ is the same. So we shall restrict our discussion to the four structural rules listed above, though it should be clear how it extends to cover any collection of structural rules.

Notice how the previous derivation fails in every logic characterized by non-contractive frames, since $a \circ b \circ b \sqsubseteq a \circ b$ does not hold true in general.

Example 14.2.9 $\vdash^{123} A \to (A \to A)$

Proof.

(1)	$F(A \to (A \to A)) : 1$	
(2)	$T(A) : a$	E$F \to$(1)
(3)	$F(A \to A) : a$	E$F \to$(1)
(4)	$TA : b$	E$F \to$(3)
(5)	$FA : a \circ b$	E$F \to$(3)
$\times$		

■

Here the branch is closed because of the general closure rule (14.17). For, let $c =$ $\bigsqcup$

Notice that the examples given above can be used to show that our labelled deduction system is complete with respect to the implication logics in Table 14.4, since suitable subsets of the theorems proved provide axiom systems for such logics and the rule of Modus Ponens can be derived in the basic system with no additional conditions on $\sqsubseteq$. However this simulation of the axiomatic systems does not preserve the subformula property, because of the applications of the "cut rule" PB to simulate the Modus Ponens rule. A more informative completeness proof will be given in the sequel.

Elimination rules for $\neg$ Since we define $\neg A$ as $A \to \bot$, the elimination rules for $\neg$ can be immediately derived from the elimination rules for $\to$. So, the rule for eliminating $T\neg A : x$ will be:

$$\frac{\begin{array}{c} T\neg A : x \\ TA : y \end{array}}{T\bot : x \circ y}. \tag{14.23}$$

If we were dealing only with intuitionistic negation, completeness would be achieved by adding the following rule:

$$\frac{F\neg A : x}{\begin{array}{c} TA : a \\ F\bot : x \circ a \end{array}} \qquad \boxed{\text{with } a \text{ a } \textit{new atomic} \text{ label}} \tag{14.24}$$

However this would not yield a uniform method for dealing with the fragments involving *classical* negation, satisfying also the double negation law. It turns out that the LDS approach allows us to obtain such a uniform method by exploiting the power of the labelling algebra.

Let us say that an information token x is *consistent with* an information token y if $x \circ y \not\Vdash \bot$ or, in terms of valuations, $v(\bot, x \circ y) = F$. It follows from our definitions that, given an arbitrary information token x, there exists the *greatest* information token consistent with it. For, let S be the set of all information tokens consistent with x, i.e. $v(\bot, x \circ y) = F$ for all $y \in S$. It follows from the definition of valuation above (condition 3) that S is a directed set. Consider now $\bigsqcup S$. By the continuity of $\circ$, $x \circ \bigsqcup S = \bigsqcup \{x \circ y | y \in S\}$, so $v(\bot, x \circ \bigsqcup S) = v(\bot, \bigsqcup\{x \circ y | y \in S\})$. By the continuity of the notion of falsity, $v(\bot, \bigsqcup\{x \circ y | y \in S\}) = F$. So $\bigsqcup S$ is the greatest information token consistent with x. Notice that when the set S is empty, $\bigsqcup$ exists and is equal to the bottom element of the lattice.

Hence we can define:

$$x^* =_{\text{def}} \max\{y | v(\bot, x \circ y) = F\}.$$

Given our definition of the operation *, we can replace the rule (14.24) with the following:

$$\frac{F\neg A : x}{TA : x^*} \tag{14.25}$$

because if A is verified by a y such that $\bot$ is falsified by $x \circ y$, then A is verified by x^*, being x^* the greatest z such that $\bot$ is falsified by $x \circ z$.

This rule is equivalent to the original one provided that the following 0-premise rule is also available:

$$\frac{}{F\bot : x \circ x^*} \tag{14.26}$$

This rule is clearly sound and the pair of rules (14.25) and (14.26) are equivalent to the rule (14.24). Notice that the 0-premise rule for $\bot$ given above is equivalent to the following additional closure rule:

$$\frac{\begin{array}{c} T\bot : y_1 \\ \vdots \\ T\bot : y_n \end{array}}{\times} \tag{14.27}$$

provided that $\sqcup$

Notice that by definition:

$$y \circ (x/y) \sqsubseteq x \quad \text{and} \quad (x\backslash y) \circ y \sqsubseteq x$$

The operations / and \ are identical in every commutative frame. There is a strong analogy between these operations and algebraic division, as emerges from the following properties :

$$\begin{aligned} y \circ (x/y) &\sqsubseteq x && (14.30)\\ 1 &\sqsubseteq x/x && (14.31)\\ (x/y) \circ z &\sqsubseteq (x \circ z)/y && (14.32)\\ (x/y)/z &\sqsubseteq x/(y \circ z). && (14.33) \end{aligned}$$

similar properties hold for \.

Let x be an information token that verifies $A \otimes B$. Then we consider as part of the meaning of $\otimes$ that there are tokens y and z such that: (i) y verifies A; (ii) z verifies B and (iii) $y \circ z \sqsubseteq x$. Let a be such a token that verifies A, and b such a token that verifies B. Then, of course, $a \circ b$ verifies $A \otimes B$. Moreover, we have that $a \circ b \sqsubseteq x$. So x/a verifies B, where x/a is defined as above (i.e. as the *greatest* information token z such that $a \circ z \sqsubseteq x$).

This justifies the following rule:

$$\frac{TA \otimes B : x}{\begin{array}{l} TA : a \\ TB : x/a \end{array}} \quad \boxed{\text{with } a \text{ a } \textit{new atomic} \text{ label}} \qquad (14.34)$$

Moreover it is easy to see that following rules are also justified:

$$\frac{\begin{array}{l} FA \otimes B : x \\ TA : y \end{array}}{FB : x/y} \qquad \frac{\begin{array}{l} FA \otimes B : x \\ TB : y \end{array}}{FA : x\backslash y} \qquad (14.35)$$

The reader can check that our elimination rules (together with PB) are equivalent to the following valuation clause:

$$v(A \otimes B, x) = T \iff \exists y, v(A, y) = T \ \text{ and } \ v(B, x/y) = T \qquad (14.36)$$

When a consequence relation is commutative and contains a classical negation, namely it is closed under the Exchange rule and (14.5), then the following equivalence holds true:

$$A \otimes B \dashv\vdash \neg(A \to \neg B) \qquad (14.37)$$

It can be easily shown that, under such circumstances, the operators / and \ can be identified and assumed to satisfy the following identity:

$$x/y = (x^* \circ y)^* \qquad (14.38)$$

Elimination rules for $\uplus$. As mentioned in Section 14.2.1, the operator $\uplus$ arises naturally in logical systems with an involutive negation and closed under Exchange. It can be defined in terms of $\neg$ and $\to$ as follows:

$$A \uplus B =_{\text{def}} \neg A \to B.$$

So, its elimination rules can be derived from the rules for $\to$ and $\neg$ together with the assumption that possible models are restricted to the classical ones (i.e. those in which the operation * is an involution).

It is easy to see that the following rules are valid:

$$\frac{\begin{array}{l} TA \uplus B : x \\ FA : y^* \end{array}}{TB : x \circ y} \qquad \frac{\begin{array}{l} TA \uplus B : x \\ FB : x \circ y \end{array}}{TA : y^*} \qquad (14.39)$$

$$\begin{array}{l} TA\to B:x\\ TA:y\\ \hline TB:x\circ y\end{array} \qquad \begin{array}{l} TA\to B:x\\ FB:x\circ y\\ \hline FA:y\end{array} \qquad \begin{array}{l} FA\to B:x\\ \hline TA:a\\ FB:x\circ a\end{array}\ \text{(i)}$$

$$\begin{array}{l} TA\otimes B:x\\ \hline TA:a\\ TB:x/a\end{array}\ \text{(i)} \qquad \begin{array}{l} FA\otimes B:x\\ TA:y\\ \hline FB:x/y\end{array} \qquad \begin{array}{l} FA\otimes B:x\\ TB:y\\ \hline FA:x\backslash y\end{array}$$

$$\begin{array}{l} FA\Cup B:x\\ \hline FA:a^*\\ FB:x\circ a\end{array}\ \text{(i)} \qquad \begin{array}{l} TA\Cup B:x\\ FA:y^*\\ \hline TB:x\circ y\end{array} \qquad \begin{array}{l} TA\Cup B:x\\ FB:x\circ y\\ \hline TA:y^*\end{array}$$

$$\begin{array}{l} T\neg A:x\\ TA:y\\ \hline T\bot:x\circ y\end{array} \qquad \begin{array}{l} F\neg A:x\\ \hline TA:x^*\end{array} \qquad \begin{array}{l} \hline F\bot:x\circ x^*\end{array}$$

$$\begin{array}{l|l} \hline TA:x & FA:x\end{array} \qquad \begin{array}{c} TA:x_1\\ \vdots\\ TA:x_n\\ FA:y\\ \hline \times\end{array}\ \text{(ii)}$$

(i): for some new atomic label a

(ii): whenever $\bigsqcup$

Fact 14.2.13 *If $TA : a_1, \ldots, TA : a_n$ are labelled signed formulae occuring in the same branch, and all a_i's are* atomic labels*, then $a_1 = \cdots = a_n$.*

The above fact ensures that the domain of atomic labels introduced in a branch is not extended unnecessarily. We shall see that this property is crucial for the algorithmic formulation of some logics.

We can formulate the generative rules in a more convenient way so as to incorporate the content of Fact 14.2.13.

Definition 14.2.14 *We say that an* atomic label *a is A-*characteristic in a branch *ϕ if $TA : a$ occurs in ϕ.*

Exploiting this terminology, we can formulate "liberalized" versions of the generative rules. Consider again the rule (14.21). We can reformulate it as follows:

$$\frac{FA \to B : x}{\begin{array}{l} TA : a \\ FB : x \circ a \end{array}} \quad \boxed{\begin{array}{l}\text{for some } A\text{-}\textit{characteristic} \\ \text{atomic label } a\end{array}} \tag{14.42}$$

In this formulation the atomic label does not have to be *new*. For instance, this version of the rule would allow for the following expansion step:

$$\frac{FA \to B : x}{FB : x \circ a} \tag{14.43}$$

whenever a is an atomic label such that $TA : a$ occurs above in the branch. Only if no A-characteristic atomic label occurs in the branch, does the rule force us to introduce a *new* atomic label a.

Similar liberalized versions can be formulated for the other generative rules. The justification of this move is somewhat similar to the justification of the "liberalized" δ rule in first-order analytic tableaux (on this point see [Haehnle and Schmitt, 1992]).

Analytic restriction. All the elimination rules of the system $\mathbf{LKE}_S$ are *analytic*, i.e. they "analyse" complex formulae by specifying the *consequences* of their truth and falsity in terms of the truth and falsity of their subformulae. On the other hand the branching rule PB (which is clearly related to the cut rule) can introduce *arbitrary formulae* and *arbitrary labels.* So the problem of proof-search in our system depends crucially on *two* components: the choice of the formulae introduced via PB and the choice of the associated labels.

In the classical **KE** system — with no labels – the use of the cut rulle PB can be restricted to analytic applications, i.e. applications preserving the subformula property, without loss of completeness. Indeed, the choice of the "cut formulae", namely the formulae introduced by an application of PB, can be even further restricted so that the resulting refutations follow a regular pattern or *canonical form* and can be found by means of a simple systematic procedure, along the lines of the usual tableau-completion procedure. Moreover, such canonical refutations are often essentially shorter than the corresponding tableau refutations and *never* significantly longer (see [D'Agostino and Mondadori, 1994] for the related technical results in terms of polynomial simulations).

As we shall see, this property extends also to the labelled version of **KE**: we can restrict the applications of PB to subformulae without loss of completeness, and define a refutation procedure which is a labelled generalizzation of the classical one. However, the use of *labelled* formulae introduces a further degree of freedom in the application of the PB rule: the choice of the label. The semi-decision procedure described in the rest of this section restricts this freedom to some extent, but the space of the labels is still infinite. Of course, full mechanization can be achieved only when the complexity of the labels can be bound in one way or the other. This question will be addressed in the second part of the paper (forthcoming) where we discuss the decision problem. So the semi-decision procedure described below should be taken only as a convenient *first* step towards the mechanization of proof-search.

In order to describe the semi-decision procedure we need to introduce the notion of completed branch. In the classical case, when no labels are present, this notion is straightforward: a branch ϕ is completed whenever for each signed formula X occurring in it one of the following conditions is satisfied:

1. X has the form TP or FP with P atomic.
2. X has the form $FA \to B$ and both TA and FB occur in ϕ.
3. X has the form $TA \to B$ and either FA or TB occurs in ϕ.
4. X has the form $TA \wedge B$ and both TA and TB occur in ϕ.
5. X has the form $FA \wedge B$ and either FA or FB occurs in ϕ.
6. X has the form $FA \vee B$ and both FA and FB occur in ϕ.
7. X has the form $TA \vee B$ and either TA or TB occurs in ϕ.
8. X has the form $T\neg A$ and FA occurs in ϕ.
9. X has the form $F\neg A$ and TA occurs in ϕ.

When dealing with labelled signed formulae, this simple notion of a completed branch should be replaced by a more complex one.

Given a branch ϕ, we define *the domain of* ϕ as follows:

Definition 14.2.15 *The* domain of ϕ*, denoted by* D_ϕ *is the smallest set containing all the atomic labels* a *occurring in* ϕ *and closed under the operations* $\circ$, *, $/$ *and* $\backslash$.

Definition 14.2.16 *We say that a labelled signed formula* X *is* completely analysed *in a branch* ϕ *if one of the following conditions is satisfied:*

1. *X has the form $TA \to B : x$ and, for all $y \in D_\phi$, either $FA : y$ or $TB : x \circ y$ is in ϕ.*
2. *X has the form $FA \to B : x$ and for some atomic a, $TA : a$ occurs in ϕ and $FB : x \circ a$ occurs in ϕ.*
3. *X has the form $T\neg A : x$ and, for all $y \in D_\phi$, either $FA : y$ or $T\bot : x \circ y$ is in ϕ.*
4. *X has the form $F\neg A : x$ and $TA : x^*$ occurs in ϕ.*
5. *X has the form $TA \otimes B : x$ and for some atomic a, both $TA : a$ and $TB : x/a$ occur in ϕ.*
6. *X has the form $FA \otimes B : x$ and, for all $y \in D_\phi$, (i) either $FA : y$ or $FB : x/y$ is in ϕ and (ii) either $FB : y$ or $FA : x\backslash y$ is in ϕ.*

For the logics containing the operator $\uplus$, the obvious clause for this operator should be added.

We say that a branch ϕ *is* completed *if every LS-formula occurring in ϕ is completely analysed. A* tree *is said to be* completed *if all its branches are completed.*

From our definitions a completed branch must be *infinite*. In the second part of this paper we shall see when and how we can restrict our attention to a *finite* portion of a completed branch without loss of generality.

The following is a procedure which expands a branch *linearly*:

Procedure: linear completion ϕ;

Repeat until a rule is applicable or ϕ is closed

1. if ϕ contains two formulae of the form $TA \to B : x$ and $TA : y$, and does not contain $TB : x \circ y$, then append $TB : x \circ y$ to ϕ; endif

2. if ϕ contains two formulae of the form $TA \to B : x$ and $FB : x \circ y$, and does not contain $FA : y$, then append $FA : y$ to ϕ; endif

3. if ϕ contains a formula of the form $FA \to B : x$, then (i) if ϕ does not contain $TA : a$ for any atomic label a, then append $TA : a$ with a new atomic label a to ϕ; if ϕ contains $TA : a$ for some atomic label a and does not contain $FB : x \circ a$, then append $FB : x \circ a$ to ϕ; endif

4. if ϕ contains two formulae of the form $T\neg A : x$ and $TA : y$, and does not contain $T\bot : x \circ y$, then append $T\bot : x \circ y$ to ϕ; endif

5. if ϕ contains a formula of the form $F\neg A : x$, then if ϕ does not contain $TA : x^*$, then append $TA : x^*$ ϕ; endif

6. if ϕ contains a formula of the form $TA \otimes B : x$, then (i) if ϕ does not contain $TA : a$ for any atomic label a, append $TA : a$ for some new atomic label a to ϕ; (ii) if ϕ contains $TA : a$ with a atomic, and ϕ does not contain $TB : x/a$, then add $TB : x/a$ to ϕ; endif

7. if ϕ contains two formulae of the form $FA \otimes B : x$ and $TA : y$, and does not contain $FB : x/y$, then append $FB : x/y$ to ϕ; endif

8. if ϕ contains two formulae of the form $FA \otimes B : x$ and $TB : y$, and does not contain $FA : x\backslash y$, then append $FA : x\backslash y$ to ϕ; endif

Again, the procedure can be extended to cover the operator $\uplus$ in the obvious way.

Let us say that a branch ϕ is *linearly completed* if the result of applying the above procedure to ϕ is ϕ itself. A branch which is linearly completed, may not be completed in that there may be LS-formulae which are not completely analysed. Notice that such LS-formulae must be of the form $TA \to B : x$ or $T\neg A : x$ or $FA \otimes B : x$. In the first case this means that for some $y \in D_\phi$ neither $FA : y$ nor $TB : x \circ y$ are in ϕ. Then, we apply the branching rule PB as follows:

$$\frac{}{TA : y \mid FA : y}.$$

and expand the two new branches until they are both linearly completed. It is easy to see that each of these linearly completed branches will contain either $FA : y$ or $TB : x \circ y$. Similar considerations apply to the second case and third case. It is routine to define a systematic procedure which combines the linear completion procedure with applications of the splitting rule PB to the effect that either a closed or a completed (infinite) tree is always obtained.

Given a frame $\mathcal{L}$ and a valuation v over $\mathcal{L}$, let us say that v *satisfies an LS-formula* X if X has the form $TA : x$ and $v(A, x) = T$, or X has the form $FA : x$ and $v(A, x) = F$. We say that a set S of LS-formulae *has a model* if there are a frame $\mathcal{L}$ and a valuation v over $\mathcal{L}$ which satisfies all the LS-formulae in S. Then, we can show that:

Proposition 14.2.17 *Every completed open branch has a model.*

Proof.[sketch] Let ϕ be a completed open branch and v a valuation over a frame (including D_ϕ) satisfying:

1. $v(P, x) = T$ for all *atomic* P such that $TP : x$ occurs in ϕ;

2. $v(P, x) = F$ otherwise.

Then v is a valuation which satisfies — via the valuation clauses (14.22), (14.29), (14.36) and (14.41) — all the LS-formulae in ϕ. This can be easily shown by induction on the degree of the LS-formulae in ϕ. ■

Proposition 14.2.17 above implies that the semi-decision procedure does not cause any loss of deductive power with respect to the unrestricted system, i.e. the former is complete with respect to the latter. For, suppose there is a tree $\mathcal{T}$ for a set of initial LS-formulae S such that $\mathcal{T}$ has a completed open branch ϕ. Then there is a valuation v which satisfies all the LS-formulae in ϕ. Thus no $\mathbf{LKE}_S$-tree for the same set of initial LS-formulae (no matter whether the applications of PB are analytic or not) can be closed, otherwise such a valuation v would be impossible.

The procedure described above shows that every *closed* tree can be recognized in a finite number of steps, while the task of showing that a completed tree has an open branch is infinite. However, the search space for the PB-formulae (as opposed to that for the labels) is finite. We described this as a first step towards mechanization. In the next section we shall make a *second* step by introducing variables in the labels. Our final final step will consist in showing when and how the complexity of the labels can be bound so as to yield a full decision procedure. This problem will be discussed in the second part of this paper (forthcoming).

14.2.4 Soundness and completeness of $\mathbf{LKE}_S$

We have shown that in the system $\mathbf{LKE}_S$ we can restrict the PB rule (or cut) to *analytic* applications, i.e. applications involving only subformulae of the formulae occurring above in the branch, without any loss of completeness. We have also described a procedure which generates analytic deductions. We show now that (the unrestricted) $\mathbf{LKE}_S$ is sound and complete with respect to the family of substructural logics defined in Section 14.2.1. It follows, by the argument in the previous section, that the analytic cut procedure is also complete. For the sake of simplicity in this section we consider the standard rules given in Table 14.5 and not their liberalized version.

To prove the completeness of $\mathbf{LKE}_S$ we make a crucial use of substitutions of atomic labels in $\mathbf{LKE}_S$-trees. Let us say that an atomic label a is *critical for* a tree $\mathcal{T}$ if it is the new atomic label introduced by an application of a generative rule in $\mathcal{T}$. In other words, the only atomic labels which are non-critical are those associated with initial LS-formulae.

Lemma 14.2.18 (Substitution Lemma) *Let $\mathcal{T}$ be an $\mathbf{LKE}_S$-tree, and let $\mathcal{T}'$ be obtained from $\mathcal{T}$ by replacing each occurrence of a non-critical atomic label a with a label x such that x does not contain any atomic label which is critical for $\mathcal{T}$. Then $\mathcal{T}'$ is also an $\mathbf{LKE}_S$-tree. Moreover, if $\mathcal{T}$ is closed, $\mathcal{T}'$ is also closed.*

As before, we write $\Gamma \vdash^{i_1,\ldots,i_n} A$ to indicate that A is a consequence of Γ if the the consequence relation is closed under the structural rules identified by the string $i_1, \ldots, i_n$ (see Table 14.2) as well as under the general conditions on the logical operators. By $\Gamma \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_S} A$ we mean that A is $\mathbf{LKE}_S$-provable from Γ for the labelling algebra specified by the conditions on $\sqsubseteq$ identified by the string $i_1, \ldots, i_n$ (in the same table). Then, it is not difficult to show that:

Proposition 14.2.19 $\Gamma \vdash^{i_1,\ldots,i_n} A \Longrightarrow \Gamma \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_S} A$

Proof. The proof is easily obtained by showing that the relation $\vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_S}$ is closed under all the relevant conditions, namely (i) the basic conditions of Identity and Surgical cut, (ii) the general conditions on the logical operators $\mathrm{C}_\otimes$, $\mathrm{C}_\rightarrow$, $\mathrm{C}_\neg$ and (iii) the structural rules identified by the string $i_1, \ldots, i_n$. We show only how to prove closure under surgical cut, contraction and expansion and leave the rest to the reader. Suppose $\Gamma \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_S} A$ and $\Delta, A, \Lambda \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_S} B$. We want to show that $\Delta, \Gamma, \Lambda \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_S} B$. Let us write $T\Gamma : \gamma$ to denote a sequence of LS-formulae obtained by the sequence of formulae Γ by (i) signing them with T and (ii) attaching to each of them a distinct atomic label. We shall also use γ° to denote the $\circ$-concatenation of these atomic labels. Similarly, we shall use the abbreviations $T\Delta : \delta$, δ°, $T\Lambda : \lambda$ and λ°. By hypothesis there is a closed $\mathbf{LKE}_S$-tree $\mathcal{T}_1$ with initial LS-formulae $T\Gamma : \gamma, FA : \gamma^\circ$, and there is also a closed $\mathbf{LKE}_S$-tree $\mathcal{T}_2$ with initial LS-formulae $T\Delta : \delta, TA : a, T\Lambda : \lambda, FB : \delta^\circ \circ a \circ \lambda^\circ$. We can assume, without loss of generality, that the sets of the atomic labels occuring in $\mathcal{T}_1$ and $\mathcal{T}_2$ are disjoint.

Then the following tree is closed:

$$\begin{array}{c}
\begin{array}{l}
T\Delta : \delta \\
T\Gamma : \gamma \\
T\Lambda : \lambda \\
FB : \delta^\circ \circ \gamma^\circ \circ \lambda^\circ
\end{array} \\
\diagup \qquad \diagdown \\
\begin{array}{cc}
TA : \gamma^\circ & FA : \gamma^\circ \\
\mathcal{T}_2[a/\gamma^\circ] & \mathcal{T}_1
\end{array}
\end{array}$$

Where $\mathcal{T}_2[a/\gamma^\circ]$ denotes the result of substituting every occurrence of the atomic label a in $\mathcal{T}_2$ with the label γ°. By the substitution lemma above $\mathcal{T}_2[a/\gamma^\circ]$ is closed if $\mathcal{T}_2$ is closed. Hence, the above closed tree is the required $\mathbf{LKE}_S$-proof of $\Delta, \Gamma, \Lambda \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_\mathcal{S}} B$.

To show closure under contraction, assume $\Gamma, A, A, \Delta \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_\mathcal{S}} B$. We want to show $\Gamma, A, \Delta \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_\mathcal{S}} B$ provided $2 \in (i_1, \ldots, i_n)$, i.e. the extra-condition $a \circ a \sqsubseteq a$ on the frame is satisfied.

By hypothesis there is a closed $\mathbf{LKE}_S$-tree $\mathcal{T}_1$ with initial LS-formulae

$$T\Gamma : \gamma, TA : a, TA : b, T\Delta : \delta, FB : \gamma^\circ \circ a \circ b \circ \delta^\circ$$

By the substitution lemma stated above we can replace every occurrence of b in this tree with an occurrence of a and the result will be a closed $\mathbf{LKE}_S$-tree with initial LS-formulae:

$$T\Gamma : \gamma, TA : a, T\Delta : \delta, FB : \gamma^\circ \circ a \circ a \circ \delta^\circ$$

So, if the frame is contractive, i.e. $a \circ a \sqsubseteq a$, the following tree will also be closed:

$$\begin{array}{c}
\begin{array}{l}
T\Gamma : \gamma \\
TA : a \\
T\Delta : \delta \\
FB : \gamma^\circ \circ a \circ \delta^\circ
\end{array} \\
\diagup \qquad \diagdown \\
\begin{array}{cc}
TB : \gamma^\circ \circ a \circ a \circ \delta^\circ & FB : \gamma^\circ \circ a \circ a \circ \delta^\circ \\
\times & \mathcal{T}_1[b/a]
\end{array}
\end{array}$$

To show closure under expansion, assume $\Gamma, A, \Delta \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_\mathcal{S}} B$. We want to show that $\Gamma, A, A, \Delta \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_\mathcal{S}} B$, provided that $3 \in (i_1, \ldots, i_n)$, i.e. the extra condition $a \sqsubseteq a \circ a$ on the frame is satisfied.

By hypothesis there is a closed $\mathbf{LKE}_S$-tree $\mathcal{T}_1$ with initial LS-formulae $T\Gamma : \gamma, TA : a, T\Delta : \delta, FB : \gamma^\circ \circ a \circ \delta^\circ$. Moreover, by the substitution lemma, the tree $\mathcal{T}_1[a/b]$ obtained from $\mathcal{T}_1$ by replacing every occurrence of a with an atomic label b which does not occur in $\mathcal{T}_1$ is also closed. Then the following tree will also be closed:

$$\begin{array}{c}
\begin{array}{l}
T\Gamma : \gamma \\
TA : a \\
TA : b \\
T\Delta : \delta \\
FB : \gamma^\circ \circ a \circ b \circ \delta^\circ
\end{array} \\
\diagup \qquad \diagdown \\
\begin{array}{cc}
TB : \gamma^\circ \circ a \circ \delta^\circ & FB : \gamma^\circ \circ a \circ \delta^\circ \\
\diagup \qquad \diagdown & \mathcal{T}_1 \\
\begin{array}{cc}
TB : \gamma^\circ \circ b \circ \delta^\circ & FB : \gamma^\circ \circ b \circ \delta^\circ \\
\times & \mathcal{T}_1[a/b]
\end{array} &
\end{array}
\end{array}$$

The leftmost branch of the tree is closed by virtue of the general closure rule applied to $TB : \gamma^\circ \circ a \circ \delta^\circ$, $TB : \gamma^\circ \circ b \circ \delta^\circ$ and $FB : \gamma^\circ \circ a \circ b \circ \delta^\circ$. For,

$\sqcup$

Remark 14.2.20 *Notice that in the above proof of the closure of $\vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_{\mathcal{S}}}$ under expansion the simple special case of the closure rule, as in (14.16), would be sufficient if the frame was assumed to be a monotonic one, i.e. such that $a \sqsubseteq a \circ b$. In this case the branch would be closed by virtue of the pair $TB : \gamma^\circ \circ a \circ \delta^\circ, FB : \gamma^\circ \circ a \circ b \circ \delta^\circ$. Moreover, a simple analysis of the proof of the above proposition shows that the special case is always sufficient for completeness except for the class of mingle frames, i.e. those which are expansive but non-monotonic. The algorithmic problems raised by our use of the labels in the closure rule (both for the general and the special case) will be discussed in the second part of this paper.*

It follows from our previous discussion that the semi-decision procedure is also complete and, therefore, the applications of PB can be restricted as indicated in this procedure.

The soundness (or "faithfulness") of the system $\mathbf{LKE}_{\mathcal{S}}$ with respect to the substructural consequence relations stems from our discussion in the previous sections and from the consideration of what we have called "the canonical interpretation" of an information frame. According to this interpretation:

- the information tokens in P are sets of formulae closed under $\vdash$;
- the composition operation $\circ$ is defined as follows:
$$x \circ y =_{\text{def}} \{A \otimes B | A \in x, B \in y\};$$
- the identity 1 is defined as the set of theorems $\{A | \vdash A\}$;
- the partial ordering $\sqsubseteq$ is set-inclusion.

Then, P is a complete lattice under $\sqsubseteq$ and all the other conditions in the definition of an information frame (Definition 14.2.1) are satisfied. The *canonical valuation* v is then defined as follows:

$$v(A, x) = T \ \text{ iff } \ A \in x$$

It is routine to show that all the $\mathbf{LKE}_{\mathcal{S}}$-rules are sound under the canonical interpretation when the LS-formulae $TA : x$ and $FA : x$ are interpreted as $A \in x$ and $A \notin x$ respectively. So, if $A_1, \ldots, A_n \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_{\mathcal{S}}} B$, i.e. there is a closed tree for $TA_1 : a_1, \ldots, TA_n : a_n, FB : a_1 \circ \cdots \circ a_n$, this means that $B \in a_1 \circ \cdots a_n$ where the a_i's are the sets $\{C | A_i \vdash C\}$. Thus, $A_1 \otimes \cdots \otimes A_n \vdash B$ and, by $C_\otimes$, $A_1, \ldots, A_n \vdash B$. This is sufficient to show:

Proposition 14.2.21 $\Gamma \vdash^{i_1,\ldots,i_n}_{\mathbf{LKE}_{\mathcal{S}}} A \Longrightarrow \Gamma \vdash^{i_1,\ldots,i_n} A$

14.2.5 Free variables in the labels

Each label occurring in a tree is built up from atomic labels and the relevant operations of the labelling algebra. New atomic labels are introduced by applications of the rules for analysing formulae of the form $FA \to B : x$ and $TA \otimes B : x$, and the propagation of the labels is uniquely determined by the tree rules. By contrast, the rule

$$\frac{}{TA : x \mid FA : x}$$

is sound for *every* choice of x and we only know that, for every valid sequent $\Gamma \vdash A$, *there exists* a set of choices for the labels generated by the application of PB which leads to a closed tree. It is therefore convenient in practice to apply the rule PB with a *variable* label x and postpone the evaluation of this variable until enough information is available. For this purpose we need some new notions.

Definition 14.2.22 *We enrich our labelling language with a denumerable set of* variables *denoted by ν_1, ν_2, ν_3 etc. A* label-scheme *α is a label containing variables. A* potential closure set *is a set of LS-formulae of the form $\{TA : \alpha_1, \ldots, TA : \alpha_n, FA : \beta\}$. A* potentially closed *branch is a branch containing a potential closure set. A* tree *$\mathcal{T}$ is* potentially closed *if all its branches are potentially closed.*

Notice that a potentially closed branch may contain more than one potential closure set. So, every potentially closed branch ϕ determines a finite set I_ϕ of inequalities of the form

$$\bigsqcup$$

Example 14.2.24 *Consider the formula* $(\neg A \to \neg B) \to (B \to A)$. *This is a theorem of linear and relevance logic (as well as, of course, classical logic) but not a theorem of intuitionistic logic. We show that there are no countermodels for the first two logics while showing at the same time that there is one for the third.*

$$\begin{tikzcd}
 & F(\neg A \to \neg B) \to (B \to A) : 1 \arrow[d, no head] & \\
 & T\neg A \to \neg B : a \arrow[d, no head] & \\
 & FB \to A : 1 \circ a \ (= a) \arrow[d, no head] & \\
 & TB : b \arrow[d, no head] & \\
 & FA : a \circ b \arrow[dl, no head] \arrow[dr, no head] & \\
T\neg A : x \arrow[d, no head] & & F\neg A : x \arrow[d, no head] \\
T\neg B : a \circ x \arrow[d, no head] & & TA : x^* \\
T\bot : a \circ x \circ b & &
\end{tikzcd}$$

In this tree, the lefthand branch is closed for all commutative labelling algebras under the substitution $x = (a \circ b)^*$. *Under this substitution the righthand branch is closed only for* classical *models, since* $(a \circ b)^{**} = a \circ b$.

Example 14.2.25 *The formula $\neg\neg A \rightarrow A$ is a theorem of linear and relevance logic, but not of intuitionistic logic.*

$$\begin{tikzcd}
 & F\neg\neg A \rightarrow A : 1 \arrow[d, dash] & \\
 & T\neg\neg A : a \arrow[d, dash] & \\
 & FA : 1 \circ a \ (= a) \arrow[dl, dash] \arrow[dr, dash] & \\
T\neg A : x \arrow[d, dash] & & F\neg A : x \arrow[d, dash] \\
T\bot : a \circ x & & TA : x^*
\end{tikzcd}$$

The lefthand branch is closed under the substitution $x = a^*$. The righthand branch is closed under the same substitution only for classical models, where $a^{**} = a$.

Example 14.2.26 *The formula $(A \rightarrow \neg A) \rightarrow \neg A$ is a theorem of relevance and intuitionistic logic but not of linear logic.*

$$\begin{tikzcd}
F(A \rightarrow \neg A) \rightarrow \neg A : 1 \arrow[d, dash] \\
TA \rightarrow \neg A : a \arrow[d, dash] \\
F\neg A : a \arrow[d, dash] \\
TA : a^* \arrow[d, dash] \\
T\neg A : a \circ a^* \arrow[d, dash] \\
T\bot : a \circ a^* \circ a^*
\end{tikzcd}$$

This one-branch tree is closed for all contractive models, since in all such models $a \circ a^* \circ a^* \sqsubseteq a \circ a^*$.

Example 14.2.27 *The formula $(A \otimes B \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$ is valid in all frames.*

$$\begin{tikzcd}
 & F(A \otimes B \to C) \to (A \to (B \to C)) : 1 \arrow[d, dash] & \\
 & TA \otimes B \to C : a \arrow[d, dash] & \\
 & FA \to (B \to C) : a \arrow[d, dash] & \\
 & TA : b \arrow[d, dash] & \\
 & FB \to C : a \circ b \arrow[d, dash] & \\
 & TB : c \arrow[d, dash] & \\
 & FC : a \circ b \circ c \arrow[dl, dash] \arrow[dr, dash] & \\
TA \otimes B : x \arrow[d, dash] & & FA \otimes B : x \arrow[d, dash] \\
TC : a \circ x & & FB : x/b
\end{tikzcd}$$

The left-hand branch of the above tree is closed under the substitution $x = b \circ c$. Since $c = 1 \circ c \sqsubseteq b/b \circ c \sqsubseteq (b \circ c)/b$ (see above, p. 398), the righthand branch is closed under the same substitution.

Chapter 15

Combining Labelled Deductive Systems

15.1 Introduction and motivation

Among the many advantages of working within the *LDS* framework, there is one of great usefulness. It is the capability of combining any two *LDS* logics in a very easy and simple way. This allows us to combine any twologics, provided we have an *LDS* formulation for them. Since we have general methods for finding an *LDS* formulation for almost arbitrary logics, we should, in principle, be able to develop general methods for combining arbitrary logics.

In many simple cases, such as modal and intuitionistic logics, the methods of fibring and combining them, whcih arise out of *LDS* considerations can be directly applied without detour through *LDS*. Other logics may actually require an *LDS* presentation.

Let us begin by presenting the master problem of combining logics.

Master Problem: General Weaving of Logics Problem

Let $\mathbf{L}$ be a logic with some connectives. Let $\mathbf{L}_i, i \in I$ be a family of logics which are conservative extensions of $\mathbf{L}$. Assume $\mathbf{L}_i \cap \mathbf{L}_j = \mathbf{L}$, for $i \neq j$. Assume each $\mathbf{L}_i$ has, among others, the set $\mathcal{C}_i = \{\sharp^i_1, \ldots, \sharp^i_{k(i)}\}$ of additional connectives to those of $\mathbf{L}$. Then the general weaving of logics problem is to characterise the set of all logics $\{\mathbf{L}^\alpha_I \mid \alpha$ is a name for a way of combining $\mathbf{L}_i, i \in I\}$ which are built on the connectives of $\mathbf{L}$ and $\bigcup_{i=1}^n \mathcal{C}_i$ and which are conservative extensions of each $\mathbf{L}_i, i \in I$.

In particular we seek to characterise the minimal such logic $\mathbf{L}^{\min}_I$, as well as some distinguished other special logics $\mathbf{L}^\delta_I$, which possibly represent some special ways of combining $\mathbf{L}_i, i \in I$. We use the notation $\mathbf{L}^\delta_I = \otimes^\delta_{i\in I}\mathbf{L}_i$.

The above general problem requires the formulation and solution of several secondary problems.

Problem 1:

Characterise the notion of a logical system (i.e. define the 'entities' (logics) involved in the weaving of logics 'problem').

Problem 2:

Present methodologies for combining any two logics, independent of how they are formulated (e.g. one may be formulated as a Hilbert system, the other through a semantical interpretation, the third may be an algebraic logic, etc.).

Problem 3:

Investigate transfer properties. If all $\mathbf{L}_i, i \in I$ share some property (decidability, interpolation, etc.) do they transfer to $\mathbf{L}_I$?

Problem 4:

Compare the combined logics thus obtained with existing known combinations of logics, which are abundant in the literature.

Problem 5:

Study possible natural interactions between the logics $\mathbf{L}_i, i \in I$ during the combination process, possibly leading to non-conservative combinations. Try to identify special ways of non-conservatively combining logics, which are meaningful (e.g. combinations leading to temporal logics, n-dimensional cross products, combining a logic with its 'metalevel', etc.).

Some attempts at a methodical solution to some of these problems for the case of normal modal logics can be found in [Kracht and Wolter, 1991, Fine and Schurz,] and for intuitionistic modal logics in [Amati and Pirri, 199]. In addition there are many papers presenting particular combinations of particular logics to suit particular applications. These range from introducing modality into relevance logic to the fuzzification of an arbitrary logic. It is time to seek a general methodology for combining logics (which we refer to as 'weaving' of logics) and attempt to classify existing combinations within such a methodology. This is the task of this series of papers.

Upon reflection, it seems that the easiest way to combine logics is when they are presnted semantically. The semantics characterises each connective individually and so any choice of connectives from a variety of systems can in principle be combined. The only problem might arise is when the semantics are completely different in flavour. Can we do something in this case? Our best bet is to use algebraic methods, as we shall see in a later section.

Our main tool in this chapter is the notion of fibred semantics and fibred labelling. . We combine logics by fibring their semantics or labels.

Example 15.1.1 (Motivating example for fibred semantics) *This example introduces the idea of fibred semantics. Imagine we want to form a combined logic with the intuitionistic $\Rightarrow$ and the* **K** *modality $\Box$. We know the Kripke semantics for $\Rightarrow$ and the Kripke semantics for $\Box$. We also know the syntax of the combined logic, namely all wffs built up from the atoms using any of $\{\Rightarrow, \Box\}$ as connectives. What we do not have is any axioms or semantics for the combined system.*

Let us proceed nevertheless and take an arbitrary formula of the combined language, say $B = (p \Rightarrow \Box(q \Rightarrow r))$. We do not have models for the combined language, but we do notice that the main connective of B is $\Rightarrow$ and that $B = (p \Rightarrow A)$ with $A = \Box(q \Rightarrow r)$. We do have intuitionistic models for $\Rightarrow$ so let us take such a model of the form $\mathbf{m} = (S, \leq, a, h)$. S is the set of possible worlds, $\leq$ is a reflexive and transitive relation on S, $a \in S$ is the actual world and h is the assignment to the atoms, satisfying $t \in h(q)$ and $t \leq s$ imply $s \in h(q)$, where $h(q) \subseteq S$.

We can evaluate, or try to evaluate, B in $\mathbf{m}$, since its main connective is $\Rightarrow$. For B to hold in $\mathbf{m}$ we must have that it holds in the actual world a. Thus we continue:
$a \vDash_{\mathbf{m}} B$ iff $a \vDash_{\mathbf{m}} (q \Rightarrow A)$ iff for all t such that $a \leq t$ and $t \vDash_{\mathbf{m}} p$ we have $t \vDash_{\mathbf{m}} A$.

We know how to evaluate $t \vDash_{\mathbf{m}} p$, because p is atomic:
$t \vDash_{\mathbf{m}} p$ iff (def) $t \in h(p)$.

The problem is to evaluate $t \vDash_{\mathbf{m}} A$, which is $t \vDash_{\mathbf{m}} \Box(q \Rightarrow r)$. $\Box$ is not in the intuitionistic language and so we do not have a semantic recursive evaluation clause for it. Nor is A atomic so we cannot use the assignment h. So what shall we do?

We notice that in order to complete the evaluation of $a \vDash_{\mathbf{m}} B$ all we need is to have an answer, for each $t \in S$ such that $a \leq t$, to the question of whether $t \vDash_{\mathbf{m}} A$. Any answer will do! It is at this stage that we can introduce the basic idea of fibring! We notice that $A = \Box(q \Rightarrow r)$ is a formula beginning with the modality $\Box$. So if we take any Kripke model for $\Box$ (and we do have a sematnics for $\Box$) then we can evaluate A there and get a value, and this value we give to $t \vDash_{\mathbf{m}} A$. So let us associate with each $t \in S$, a modal Kripke model $\mathbf{n}_t = (T^t, R^t, b^t, h^t)$ and agree that $t \vDash_{\mathbf{m}} A$ be the value of $\mathbf{n}_t \vDash A$. Note that we also have dependence on $\mathbf{m}$, i.e. we should write $\mathbf{n}_{\mathbf{m},t}$ etc, but since $\mathbf{m}$ is fixed, we omit that.

Let us now expand on $\mathbf{n}_t \vDash A$. By definition we need to check
$\mathbf{n}_t \vDash A$ iff $b^t \vDash_{\mathbf{n}_t} \Box(q \Rightarrow r)$ iff for all $s \in T^t$ such that $b^t R^t s$ we have $s \vDash_{\mathbf{n}_t} (q \Rightarrow r)$.

We now have a similar problem as before. We want to evaluate in a modal model an intuitionistic formula. The model cannot give us a value. Again we notice that all we need is to get a value somehow. Let us use the same idea and associate with each $s \in T^t$ an intutionisitc Kripke model $\mathbf{m}_{t,s} = (S^{t,s}, \leq^{t,s}, a^{t,s}, h^{t,s})$ and evaluate $q \Rightarrow r$ in this model. We continue:
$s \vDash_{\mathbf{n}_t} (q \Rightarrow r)$ is the same by agreement as $\mathbf{m}_{t,s} \vDash (q \Rightarrow r)$ which by definition of intuitionistic satisfaction, is the same as $a^{t,s} \vDash (q \Rightarrow r)$, which holds iff for all $w, a^{t,s} \leq^{t,s} w$ and $w \vDash_{\mathbf{m}_{t,s}} q$ imply $w \vDash_{\mathbf{m}_{t,s}} r$.

The latter can now be evaluated because q and r are atomic.

*To summarize, we needed to have associated (*or fibred*) with each intuitionistic world a modal model and with each modal world an intuitionistic model. When we were forced to evaluate a formula D in a world belonging to the other semantics, we continued the evaluation in the associated model.*

The basic fibring can be done by a two place function $\mathbf{F}$. If $\mathbf{m}$ is a model and t is a possible world in $\mathbf{m}$ then $\mathbf{F}(\mathbf{m}, t)$ is a Kripke model of the other semantics. We have $t \vDash_{\mathbf{m}} A$ iff $\mathbf{F}(\mathbf{m}, t) \vDash A$, iff $a^{\mathbf{F}(\mathbf{m},t)} \vDash A$ where A is a formula with main connective not in the logic of the model $\mathbf{m}$ and $a^{\mathbf{F}(\mathbf{m},t)}$ is the actual world of the model $\mathbf{F}(\mathbf{m}, t)$.

The model $\mathbf{F}(\mathbf{m}, t)$ can be presented in many forms. If the semantics of the two logics involved are very similar, the model $\mathbf{F}(\mathbf{m}, t)$ may be a slight modification of the model $\mathbf{m}$ itself.

It may even be the case that $\mathbf{F}(\mathbf{m}, t)$ is $\mathbf{m}$ itself with a change in evaluation procedures. For example fibring a classical model to an intuitionistic model can be achieved by a change of the evaluation procedure.

The intuitive idea of the fibred semantics can also be understood in terms of classical model theory. Suppose we are dealing with classical models of a binary relation R and unary relation Q. These models have the form (S, R, Q) where S is a set and $R \subseteq S \times S$ and $Q \subseteq S$. Where do R and Q come from? For the purpose of the model theory of a binary and unary relations, we do not care where R and Q come from. All we need is a subset of $S \times S$ and a subset of S. However, we could get R and Q in a more elaborate way. We could, for example, map S onto a group G using $f : S \to G$ and let xRy hold in S iff $f(x) \cdot f(y) = f(y) \cdot f(x)$ in G and let $x \in Q$ hold if for example $f(x)^2 = 1$. Our way of looking at this is to say that we are fibring group structure onto S.

In computer science terms this can be looked upon as opening a window or what is called 'delegation' in object oriented programming. To compute whether xRy holds, we open a window and place $f(x), f(y)$ in the window (i.e. go to the associated group) and then compute something in the widnow (e.g. commutativity) and then come back. The group G itslef may be further fibred, for example, into a field F. We need a function $g : G \to F$ with some translation of the group operations into field operations. For example $x \cdot y = z$ holds in G iff $g(x) \cdot g(y) = g(z)$ holds in F. (This means that G is viewed as a multiplicative group of a field.)

The above is not exactly fibring, it is more like representation. Fibring occurs when we double back into S. For example we can map G into S using a mapping $h : G \to S$ and require that for some formula $\Psi(u_1, u_2, u_3)$ of S the following holds: $x \cdot y = z$ in G iff $\Psi(h(x), h(y), h(z))$ holds in S.

Let us now do this operation more systematically. Suppose we are given models of a binary relation, say $(S_1, R_1), \ldots, (S_n, R_n), \ldots$ and we want to create models of a binary relation and a unary relation. The simplest way of achieving this is to take the models (S_n, R_n) and to add a subset $Q_n \subseteq S_n$ to form (S_n, R_n, Q_n). Now let us ask, where does Q_n come from? It can be arbitrary, or we can fibre (S_{n+1}, R_{n+1}) into (S_n, R_n). Let $\Psi_n(R, Q, x)$ be a formula with R, Q and one free variable x, let $f_n : S_n \to S_{n+1}$ embed S_n in S_{n+1} and let $(S_n, R_n) \vDash x \in Q$ iff $(S_{n+1}, R_{n+1}) \vDash \Psi_n(R, Q, f_n(x))$. Notice that of course Ψ_n depends on x (through $f_n(x)$).

Of course this evaluation may not terminate if each Ψ_n contains Q but it will terminate if Ψ_n does not contain Q for some n.

We do have fibring here because the value of $x \in Q$ at (S_n, R_n) is determined by going to (S_{n+1}, R_{n+1}).

In programming terms, we have a program π_1, relying on a subset Q which can be computed by a program π_2. π_2 also relies on Q but sends the computation to π_3 etc. The program will terminate if some π_n is able to compute Q itself.

Example 15.1.2 (a) *Consider any sophisticated word processor. It will allow you, in the middle of a letter, to open a window and access a spreadsheet. This is in order to enable you to do a calculation which arises in the text. In fact the more sophisticated the program is, the more interaction there is between the word processor and the spreadsheet and the more you can import into your letter.*
(b) *A prolog program may use the predicate prime(x). It may send the predicate to a Pascal program to check whether x is prime. The pascal program may use assembly language.*

The next series of definitions will formally present the concept of fibring of this chapter. Compare with section ???.

Definition 15.1.3 (Logics and Semantics) *1. A propositional logical language* $\mathbf{L}$ *is comprised of an infinite set of atoms and a set of connectives* $\{\sharp_1, \ldots, \sharp_k\}$*. The connective* $\sharp_j$ *has* n_j *places.*

2. *The wffs of* $\mathbf{L}$ *are defined by induction*
   - *An atom* q *is a wff*
   - *If* $A_1, \ldots, A_{n_j}$ *are wffs then* $\sharp_j(A_1, \ldots, A_{n_j})$ *is a wff with main connective* $\sharp_j$*.*
3. *A consequence relation* $\mid\!\sim$ *in the language* $\mathbf{L}$ *is a binary relation on wffs of* $\mathbf{L}$ *of the form* $A \mid\!\sim B$ *satisfying:*
   - *Identity*
   $$A \mid\!\sim A$$
   - *Transitivity*
   $$A \mid\!\sim B \text{ and } B \mid\!\sim C \Rightarrow A \mid\!\sim C.$$
4. *A classical monadic logic language* $\mathbf{CL}$ *for a possible world semantics for a logic* $\mathbf{L}$ *with connectives* $\{\sharp_1, \ldots, \sharp_k\}$ *has the non logical predicates* $\varphi_1, \ldots, \varphi_k, R_1, \ldots, R_k$ *where* R_i *is* n_i+1 *place predicate (corresponding to the* n_i *place connective* $\sharp_i$*) and* φ_i *is* $m_i (m_i \leq n_i + 1)$ *place predicate supporting (or connected to) the connective* $\sharp_i$*.*
5. *An interpretation for the connectives* $\sharp_i$ *is a classical first-order formula* $\Psi_i(t, Q_1, \ldots, Q_{n_i})$ *of the langauge* $\mathbf{CL}$ *where* t *is the only free element variable in* Ψ_i *and* Q_j *are monadic variables (subset variables).*
6. *A possible world structure for the logic* $\mathbf{L}$ *in the langauge* $\mathbf{CL}$ *with interpretation* $\{\Psi_i\}$ *is any classical structure* $\mathbf{m} = (S^{\mathbf{m}}, \varphi_j^{\mathbf{m}}, R_j^{\mathbf{m}}, \mathbf{e}^{\mathbf{m}}, h^{\mathbf{m}})$*, where* $\mathbf{e} \in S, \varphi_j$ *and* $R_j, \ldots$ *are relations as in (4) above, and* h *is an assignment giving a subset* $h(q) \subseteq S$*, to each atom* q *of* $\mathbf{L}$*.*
7. *Given a structure* $\mathbf{m}$*, the function* h *can be extended to an arbitrary wff* A *of* $\mathbf{L}$ *by structural induction as follows*
   $$h(\sharp_j(A_1, \ldots, A_{n_j})) = \{t \mid \mathbf{m} \vDash \Psi_j(t, h(A_1), \ldots, h(A_{n_j}))\}$$
   We also write $t \vDash_h A$ *when* $t \in h(A)$*.*

   We are abusing notation here. '$\vDash$*' is classical satisfiability.* $\Psi_j(t, Q_1, \ldots, Q_{n_j})$ *is a formula with free variable* t *and free subset variables* $Q_1, \ldots, Q_{n_j}$*. By substituting* $h(A_i)$ *for* Q_i *we get a wff which we can evaluate at* $\mathbf{m}$ *as a classical model.*
8. $\mathbf{m}$ *is said to satisfy* $A, \mathbf{m} \vDash A$*, if* $\mathbf{e} \in h(A)$*.*
9. *We say that a class of model* $\mathcal{K}$ *characterises a consequence relation* $\mid\!\sim$ *(or* $\mid\!\sim$ *is complete for* $\mathcal{K}$*) if we have for all* A, B*:*
   $$A \mid\!\sim B \text{ iff for all } \mathbf{m} \in \mathcal{K} \text{ (if } \mathbf{m} \vDash A \text{ then } \mathbf{m} \vDash B\text{)}.$$

Definition 15.1.4 (Fibred semantics) *Let* $\mathbf{L}_i$ *be logic languages with connectives* $\{\sharp_1^i, \ldots, \sharp_{k_i}^i\}$*, each of* n_j^i *places and* $\mathbf{CL}_i$ *be the corresponding monadic classical logic languages with interpretations* Ψ_j^i *for the respective connectives. Let* $\mathbf{L}$ *be a langauge containing (and based on) the disjoint union of the connectives of* $\mathbf{L}_i$*. Let* $\mathbf{CL}$ *be a monadic classical language based on the disjoint union of the predicates of* $\mathbf{CL}_i$*. We define a special class of structures for* $\mathbf{CL}$*, obtained by 'fibring' the structures of* $\mathbf{CL}_i$*.*

Let $\mathcal{K}_i$ be a class of structures for $\mathbf{L}_i$ of the form $\mathbf{m} = (S^{\mathbf{m}}, \varphi_j^{\mathbf{m}}, R_j^{\mathbf{m}}, \mathbf{e}^{\mathbf{m}}, h^{\mathbf{m}})$. We can assume that all sets of possible worlds $S^{\mathbf{m}}$ of all structures in $\cup_i \mathcal{K}_i$ are all pairwise disjoint. We now construct the fibred structures for $\mathbf{L}$.

Let $W = \bigcup_{\mathbf{m} \in \cup_i \mathcal{K}_i} S^{\mathbf{m}}$. Since all $S^{\mathbf{m}}$ are pairwise disjoint, for each $w \in W$ there exists a unique $\mathbf{m}, i$ such that $w \in S^{\mathbf{m}}$ and $\mathbf{m} \in \mathcal{K}_i$. Let $\tau(w)$ be the function giving this unique i. Let $\mathbf{a}(w) = \mathbf{e}^{\mathbf{m}}$, i.e. $\mathbf{a}$ gives the actual world of the model $\mathbf{m}$ to which w belongs. Let $\mathbf{F}(i, w)$ be an arbitrary function giving for each i and each w a value $w' = \mathbf{e}^{\mathbf{m}}, \mathbf{m} \in \mathcal{K}_i$. The function $\mathbf{F}$ is a fibring function, associating with each possible world w and each semantic class $\mathcal{K}_i$ a structure $\mathbf{m} \in \mathcal{K}_i$. Let h be an assignment into W defined by

$$w \in h(q) \text{ iff } w \in h^{\mathbf{m}}(q), \text{ for the unique } \mathbf{m} \text{ such that } w \in S^{\mathbf{m}}.$$

We call any $\mathbf{n} = (W, \tau, \mathbf{a}, \mathbf{F}, w_0, h)$ a fibred structure.

Let $\mathcal{K}$ be the class of all fibred structures. Let A be a wff of $\mathbf{L}$. We define the notion of satisfaction as follows:

- *$w \vDash_h q$ iff $w \in h(q)$, for q atomic.*
- *$w \vDash_h \sharp_j^i(A_1, \ldots, A_{n_j^i})$ iff*
    1. *$\tau(w) = i$ and for the unique $\mathbf{m} \in \mathcal{K}_i$, such that $w \in S^{\mathbf{m}}$ we have $\mathbf{m} \vDash \Psi_j^i(h(A_1) \cap S^{\mathbf{m}}, \ldots, h(A_{n_j^i}) \cap S^{\mathbf{m}})$, or*
    2. *$\tau(w) \neq i$ and $w' = \mathbf{F}(i, j)$ and $w' \vDash \sharp_j^i(A_1, \ldots, A_{n_j^i})$.*
- *we say $\mathbf{n} \vDash A$ iff $w_0 \vDash A$*
- *Let $\mathcal{K}$ be a class of fibred structures. We say $\mathcal{K} \vDash A$ iff for every $\mathbf{n}$ in $\mathcal{K}, \mathbf{n} \vDash A$.*

Fibring the semantics of a modal logic to itself can be considered as a special case of two dimensional modal logics.

Definition 15.1.5 (Fibred Kripke Semantics) *Let S be a nonempty set of possible worlds. Consider $T = S^2$, the set of all 2 dimensional vectors (t_1, t_2) with $t_i \in S$. Consider an accessibility relation $R \subseteq T^2$, of the form $(t_1, t_2)R(s_1, s_2)$ and an assignment h of the form $h(q) \subseteq T$. Then $(T = S^2, R, h)$ is a Fibred Kripke model.*

The above definition seems to be a simple generalisation of the usual Kripke semantics. Instead of taking structures of the form (S, R, h), where S is a set of possible worlds and R is the accessibility relation on S, we are generalsiing and taking $T = S^2$ and letting R be an accessibility relation on 2-vectors from S. This is technically correct, however, the way we are going to manipulate the semantics is going to rely heavily on the internal structure of S^2. The relation R is between pairs of elements, $(w_1, t)R(w', s)$. If we understand w as naming a Kripke structure with a set of possible worlds $S(w)$ and understand t as a possible world $t \in S(w)$, then we can read the relation $(w, t)R(w', s)$ as meaning that the Kripke structure named w', with the set of possible world $S(w')$ containing point $s \in S(w')$, is associtated with $t \in S(w)$.

The point $s \in S(w')$ can be interpreted as the actual world of $S(w')$.

Thus we have:

- w is a model iff $\exists t, y, z(w, x)R(y, z)$.
- $t \in S(w)$ iff $\exists y, z(w, t)R(y, z)$
- y is a model related to (w, t) iff $\exists z(w, t)R(y, z)$
- $z \in S(y)$ iff $(w, t)R(y, z)$.

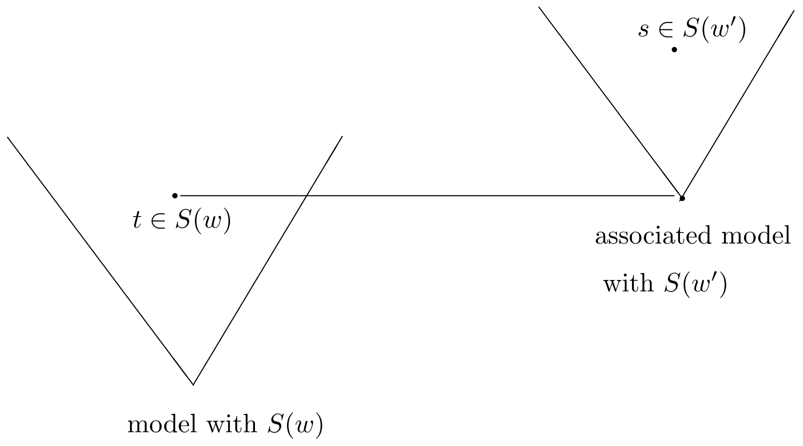


Figure 15.1:

Figure 15.1 illustrates this notion.

Let us now examine how the previous example of sequence of models (S_n, R_n) fits into Definition 15.1.4. We can assume without loss of generality that all the S_n are subsets of a single set.

Denote this set by W and think of it as a set of labels. We can also identify the indices 1, 2, 3, ..., n, ... as elements of W. We can thus assume that for each $w \in W$, $S_w \subseteq W$ is a set of possible worlds and $R_w \subseteq S_w^2$ is a binary relation on S_w. Let $\mathbf{s}(n) = n+1$ be the successor function. Since we identified the indices $\{n\}$ with a subset of W, the successor function becomes some function * in $W \mapsto W$.

The function $f_w : S_w \to S_{w*}$ can be realised by a binary function $\mathbf{F}(w,x) : W \times W \to W$ satisfying for $s \in S_w$ that $\mathbf{F}(n,x) \in S_{w*}$.

The binary relation R of Definition 15.1.5 can be realised as $(w,x)R(w*, f_w(x))$ or in our new notation $(w,x)R(\mathbf{a}(w), \mathbf{F}(w,x))$. where $\mathbf{a}(w) = w*$.

Example 15.1.6 *We show by way of example how the Kripke modal semantics can be viewed as fibred in the sense of previous definitions. Let* $\mathbf{m} = (T, R, a, h)$ *be a Kripke model. For each* $t \in T$ *associate the Kripke model* $\mathbf{m}_t = (T^t, R^t, t, h^t)$*, where*

$$\begin{aligned} &T^t = \{s \in T \mid t = s \vee tRs\} \\ &R^t = R \restriction T^t \times T^t \\ &h^t = h \restriction T^t \end{aligned}$$

clearly for any A

$$t \vDash_{\mathbf{m}} \Box A \text{ iff } t \vDash_{\mathbf{m}_t} \Box A.$$

This paper discusses relationships between the consequence relations of $\mathbf{L}_i$ and the consequence relation of $\mathbf{L}$, for various conditions on $\mathbf{F}$ and h. Let $\delta(\mathbf{F}, h)$ be a condition on the combination (i.e. on $\mathbf{F}$ and h). We denote by $\mathbf{CL}^\delta$ the class of fibred structures obtained and by $\mathbf{L}^\delta$ the resulting logic (consequence relation). For some δ's, $\mathbf{L}^\delta$ is a conservative extension of each $\mathbf{L}_i$. For other δ's, we get a specific combined logic. The general problem to study is given a δ and $\mathbf{L}_i$, can we characterise $\mathbf{L}^\delta$? If we have syntactical axiomatiations of $\mathbf{L}_i$, can we put them together to obtain an axiomatization of $\mathbf{L}^\delta$? Can we identify known combined logics from the literature and present them as the result of some special conditions δ on the fibring?

Our method is to show how we can fibre logics $\mathbf{L}_i$ into a logic $\mathbf{L}^F$ through the most general fibring construction. Once we get semantics and axiomatisation for this general logic $\mathbf{L}^F$, any combined logic existing in the literature can, in principle, be given a semantics which is an extension of the $\mathbf{L}^F$ semantics. The situation is similar to that of modal logic $\mathbf{K}$. Any modal logic stronger than

K can, in principle, be given semantics by restricting that of **K**. So, by studying the most general combination $\mathbf{L}^F$ of logics $\mathbf{L}_i$, any existing axiomatic combined logic can, in principle, be given semantics. Converely, we can try and axiomatise any restriction on the $\mathbf{L}^F$ semantics.

For example, if we restrict our choice of fibred structures and require that for any atomic q and any w an i

$$w \vDash q \text{ iff } \mathbf{F}(i,w) \vDash q$$

we get a construction which we call dove-tailing. Does this condition correspond to an axiom on **L**? We shall see that in modal logic this requirement is very natural.

Example 15.1.7 (Sample Problem) *Devise a logic with the classical connective '$\wedge$', the intutionistic implication '$\Rightarrow$', and the* **S4** *modality '$\Box$'. In other words, we want a logic* **L** *with connectives $\{\wedge, \Rightarrow, \Box\}$ where these connectives behave in* **L** *respectively as they behave in their original logics. (Actually '$\wedge$' is the same in all three logics.)*

To solve the above problem, we adopt an inutitive way of thinking. Given two logics, $\mathbf{L}_1$ *and* $\mathbf{L}_2$*, how do we combine them? Never mind the question of whether we get a (minimal) conservative combination; we just want to put them together in some natural way.*

Well, whether we can do anything depends on how the the logics are presented to us and what we know about them. Here are some possibilities:

1. *If* $\mathbf{L}_1$ *and* $\mathbf{L}_2$ *are presented in the same methodology, say proof theoretically as Gentzen systems, or say as Hilbert systems, then we can let* **L** *be the union of the languages and axioms and rules and allow substitution in* **L** *in both sets of axioms and rules. Thus if* $\mathbf{L}_1$ *is a modal* **K4** *for $\Box_1$ and* $\mathbf{L}_2$ *is modal* **K4** *for $\Box_2$ and the common language is classical logic then* **L** *will have both modalities with* **K4** *axioms and rules for each. There is no interaction between the two modalities. In fact, if we add the interactive axioms*

$$A \to \Box_2 \Diamond_1 A$$
$$A \to \Box_1 \Diamond_2 A$$

we get the temporal logic $\mathbf{K}_t$ *(with $\Box_1$ future and $\Box_2$ past).*

2. *It may be that* $\mathbf{L}_1$ *is presented as a Hilbert system and* $\mathbf{L}_2$ *as a Gentzen system, how do we combine them?*

3. *Worse still, suppose* $\mathbf{L}_1$ *is presented proof theoretically and* $\mathbf{L}_2$ *semantically, how do we combine them?*

4. *If both* $\mathbf{L}_1$ *and* $\mathbf{L}_2$ *are faithfully translatable into a third logic* **M** *then we can combine the two logics in* **M** *and translate back. Let τ_1 and τ_2 translate (respectively)* $\mathbf{L}_1$ *and* $\mathbf{L}_2$ *into* **M***. Assume that the translation is done by translating each connective $\sharp$ of* $\mathbf{L}_i$ *of the form $\sharp(x_1, \ldots, x_n)$ into a formula $\varphi_\sharp(x_1, \ldots, x_n) = \tau_i(\sharp)(x_1, \ldots, x_n)$ of* **M***.*

   Thus any formula of $\mathbf{L}_{1,2}$ *can be translated into* **M** *by applying recursively either τ_1 or τ_2 depending on the connective. Let τ be this 'combined' translation. Thus*

   - $\tau(\sharp(A_1, \ldots, A_n)) = \tau_i(\sharp)(\tau(A_1), \ldots, \tau(A_n)), \sharp \in \mathbf{L}_i$
   - $\tau(x) = x$*, for x atomic.*

   Define $\mathbf{L}_{1,2}$ *by*

   $$\mathbf{L}_{1,2} \vdash A \textit{ iff (def) } \mathbf{M} \vdash \tau(A).$$

   For example both $\Box_1$ and $\Box_2$ are translatable into classical logic via transitive possible world relations R_1 and R_2. Combining them in **M** *gives a translation into (essentially Kripke semantics)* **M** *using both relations. In semantical terms we get Kripke semantics of the form (S, R_1, R_2). In this semantics $t \vDash \Box_i A$ iff for all s, tR_is implies $s \vDash A$. Do we get the same system as when we combine their Hilbert Style formulation? In this case the answer is yes.*

Going back to the example with $\{\wedge, \Box\}$ of modal logic **S4** *and $\{\Rightarrow\}$ of intuitionistic logic. We note that there are several ways of combining them. Are they identical?*

- *Combine through the translation into classical logic via their respective Kripke semantics.*
- *Translate modal logic via the Kripke semantics into classical logic* **M**. *Consider the translation as a syntactical translation into* **M** *as a Hilbert system. Weaken* **M** *into* **I** *(intutionistic predicate logic). Let* **L** *of $\{\wedge, \Box, \Rightarrow\}$ be defined via the translation into* **I**.
- *Combine $\wedge, \Box$ and $\Rightarrow$ via the translation of intutionistic logic into modal* **S4**.
- *Combine $\{\wedge, \Box\}$ and $\{\Rightarrow\}$ as Hilbert systems. Take the union of languages and all instances of axioms and rules involving the connectives. Alternatively, take a natural deduction or Gentzen formulation of the two systems and combine the two proof theories.*
- *Combine the two systems by fibring their semantics as proposed in the present paper.*

In this chapter we develop several case studies of combining logics:

- Combining two modalities;
- intutitionistic modal logic;
- modal linear logic;
- modal many valued logic;
- intuitionistic many valued logic.
- combining any two *LDS*'s

The first case study, combining two modalities, is the simplest. This should be straightforward, as we are combining two logics of the same kind which, in general, have no special requirement on the assignments to the atoms.

The second case combines two slightly different logics; namely modal logic and intuitionistic logic. the novelty in this case is that intuitionistic logic has a restriction on the assignment to atoms which has to be satisfied. This slight complication has to be addressed.

The third case study combines two much more different logics, having differnet natural presentations. Modal logic has good semantics while linear logic is primarily proof theoretical. The question to ask is does the combination depend on which semantics we choose for the logic?

The fourth and fifth case studies address the problem where the two logics are really far apart in style. Łukasiewitz many valued logic has an algebraic, truth table presentation, while modal and intuitionistic logics have a possible world semantics. How do we combine them? Do we have several options? Do we need a general definition of fibring algebraic logics?

We will introduce several methods of combining logics. Each method will have its own name. So for example we shall define what it means to 'fibre' two logics or to 'dovetail' two logics or to 'fuzzle' (make fuzzy) a logic by another or to 'braid' two logics, etc. Our purpose now is to give an intuitive idea of what these methodologies are. We explain the concepts by taking a simple example. Suppose we want to combine two modal logics $\mathbf{L}_1$ and $\mathbf{L}_2$. Let $\mathcal{K}_1, \mathcal{K}_2$ be the respective Kripke semantics of the logic. Let $\mathbf{m}$ be a model in $\mathcal{K}_1$ and let t be a possible world of $\mathbf{m}$. The semantic construction which combines the logics associates a model $\mathbf{n}$ with t. The different methodologies of combination differ on the kind of model $\mathbf{n}$ that we use.

1. For *fibring* (type 0 combination) logics, we require that $\mathbf{n}$ be any model in $\mathcal{K}_2$.
2. For *dovetailing* (type 1 combination), we require that $\mathbf{n}$ be a model of $\mathcal{K}_2$ such that for any *atomic q*
$$t \vDash q \text{ iff } \mathbf{n} \vDash q$$
(i.e. the fibred model must agree with the values t gives to atom).

3. For *braiding* (type 2 combination), we do not require that $\mathbf{n}$ is a model of $\mathcal{K}_2$ but that $\mathbf{n}$ can be obtained from a model of $\mathcal{K}_2$ in the *same way* as the model at t is derived from $\mathbf{m}$. So for example if $a^{\mathbf{m}}$ is the actual world of $\mathbf{m}$ and k is the smallest integer such that $a^m(R^{\mathbf{m}})^k t$. Then we require that for some model $\mathbf{n}_2 \in \mathcal{K}_2$, with actual world $a^{\mathbf{n}_2}$, there is an $s \in S^{\mathbf{n}_2}$ such that $a^{\mathbf{n}_2}(R^{\mathbf{n}_2})^k s$ and $\mathbf{n}$ is the model obtained from $\mathbf{n}_2$ by changing the actual world to s. $\mathbf{n}$ may not be in the semantics $\mathcal{K}_2$.

4. For *joining* (type 3 combination) we form a multi-dimensional system out of the participating logics.

5. For *fuzzling* (type 4 combination), we simply assign at t values which are elements of the algebraic logic $\mathbf{L}_2$ instead of just $\{0, 1\}$ values (e.g. boolean valued models are an example of 'making fuzzy' by weakening the 'crisp' $\{0,1\}$ values into boolean values).

Section 2 Fibres modal Logics. Section 3 Fibres modal and intuitionistic logic. Section 4 asks the following general question: given two arbitrary logics, for which we may not have a semantic, how do we combine them? The answer is that since one can always generate algebraic semantics, it is useful to show how to combine or fibre algebraic semantics. By way of application, this section shows how to combine modal logic with many valued logic, and present a general method of making logics fuzzy. Section 5 deals with multi-dimensional logics.

15.2 Case study 1: Combining modal logics

This section deals with combinations of modal logics. The problem in its general form is how to construct a multimodal logic containing several unary modalities, each coming with its own system of specific axioms. For example, how to combine a necessity $\Box A$ with belief BA and knowledge KA systems, or how to combine necessity $\Box A$ with temporal connectives GA and HA and the like.

There are many such combined systems in the literature but no methodology for generating them. Each combined system was presented and motivated by a different author for different reasons and different applications.

The method presented in this section for modal logics is as follows. Most known modal systems already have good semantics. Thus each logic $\mathbf{L}_i$ can be identified via a class of Kripke structures $\mathcal{K}_i$, which characterises it. Thus we will give methodology for fibring (combining) classes of Kripke structures and study its properties. We shall then investigate what other modes of syntactical combinations are available. If each $\mathbf{L}_i$ is axiomatizable and $\mathbf{L}$ is the logic of the semantic combination of the semantics of $\mathcal{K}_i$ of $\mathbf{L}_i$, can we axiomatize $\mathbf{L}$ by some combination of the individual axiomatizations of $\mathbf{L}_i$? We continue to study other features such as decidability, finite-model property, etc.

We begin by combining arbitrary extensions of modal $\mathbf{K}$. It makes no difference to our methodology whether thee extensions are normal or not. We find out how to axiomatise the combination in terms of the theorems (axiomatizations) of the combined components. We see that in certain cases, the combined axiomatization is very simple.

The fibring construction is very simple intuitively but unfortunately very awkward to write on paper.

The rest of the section is structured as follows.

We begin with a short discussion and present a canonical fibring construction. Definition 15.2.1 gives the general definition of fibred Kripke semantics for modal logics. Its idea is essentially to replace the notion of a possible world as a 'unit' by another Kripke structure. Armed with this definition, we can fibre arbitrary modal logics through their semantics. We need a completeness theorem for the combined logic, and this is done in Theorem 15.2.3. the proof is involved notationally, though simple conceptually. It does require an intermediate construction for an inductive argument to go through. This is called 'grafting' and is defined in Definition 15.2.2. Examples 15.2.4 and 15.2.5 show how two modal logics, which are quite typical and arbitrary, can be combined and axiomatised, using the previous completeness theorem. Theorem 15.2.6 gives more properties of the construction.

The second part of this section is concerned with a more specific construction, called dovetailing. Here the combination of the logics is done in a tighter manner, though the combined logic is still a conservative extension of each of its components. Again, Definitions 15.2.7 and 15.2.8 describe the notion and Theorem 15.2.9 gives the completeness theorem. Properties of dovetailing and some examples are given following the completeness theorem. We especially look for conditions on the logics which make dovetailing the same as the previous notion of fibring.

The next series of definitions and theorems have to do with fibring fragments of modal logics. It is important as a preliminary to introducing modality into other systems such as intuitionistic logic or relevance or linear logics. These other logics do not have all the classical connectives and so they 'take in' only the modality. For this reason we must develop techniques of combining pure modalities. We may have both modalities $\Box$ and $\Diamond$ in modal intuitionistic logic or modal relevance logic and because of lack of negation, one is not reducible to the other, and in fact, one may not have all the dual properties of the other. Thus, for all practical purposes we have here a fragment with combined pure modalities. Example 15.2.18 introduces the problems involved and Theorem 15.2.20 is the completeness theorem for this case.

We conclude the section with Definition 15.2.23 and Theorem 15.2.24 dealing with the question of when is the disjunction property preserved in a combination.

Fibring Modalities

Let us now start the business of this section. Imagine we want to combine two modalities $\Box_1$ and $\Box_2$. We start with the simple case where both $\Box_i$ are **K**-modalities, i.e. they each satisfy the axioms below.

1. All truth functional tautologies
2. Modus ponens
$$\frac{\vdash A; \vdash A \to B}{\vdash B}$$
3. Necessitation
$$\frac{\vdash A}{\vdash \Box A}$$
4. Axiom schema
   (a) $\Box(A \to B) \to (\Box A \to \Box B)$

The above is a Hilbert system presentation. Such presentations are standard in modal logics. One may be tempted to take the combined system with both modalities and the **K** axioms and rules for each of them and let this be, together with substitution, the desired fibred system $\mathbf{K}_{1,2}$. This procedure, however, depends on the proof formulation of the components. **K** may be formulated, for example, without necessitation, as a system with an infinite number of axioms, namely all **K**-theorems. If $\Box_1$ and $\Box_2$ are fibred under this new Hilbert formulation we get a combined system with less theorems because necessitation is not available. We cannot derive $\Box_1\Box_2(A \to A)$ for example unless we put in some additional rules. The lesson we learn from the above example is that we should not just formally combine proof systems but that we need some semantical interpretation, an intended meaning, to guide us in doing the fibring. This intuition can also help us with the choice of axioms for the resulting combined system.

Turning now to combining modal logics, we approach the problem semantically. The logic **K** is complete for the Kripke semantics with possible world structures of the form (S, R, a, h) where $R \subseteq S \times S, a \in S$ is the actual world and h is the assignment to the atoms ($h(q) \subseteq S, q$ atomic). To distinguish between the semantics for $\Box_1$ and $\Box_2$, we write (S^1, R^1, a^1, h^1) and (S^2, R^2, a^2, h^2) respectively. We now study in detail how to fibre several modalities. We begin with the following:

Canonical Fibring Construction:
Let $\mathcal{K}_i$ be the class of all Kripke structures of the modal logics $\mathbf{L}_i$ with modality $\Box_i$, for $i \in I$.

We want to create a multi modal logic $\mathbf{L}_I^F$,which is a conservative extension of each $\mathbf{L}_i$. We can assume that the sets of possible worlds of each model of each semantics class $\mathcal{K}_i$ are all disjoint and that therefore each model can be uniquely identified by its actual world. We note that in case of normal modal logics such as **K, S4** etc., for any model (S, R, a, h) with actual world a, and any $t \in S$, the structure (S, R, t, h) is also a model of the same logic. Even in the non-normal case, this model will be a model of the basic logic **K**. We can therefore assume that for each $i \in I$ and each $(S, R, a, h) \in \mathcal{K}_i$ and each $t \in S$, there exists a $j \in I$ such that $(S, R, t, h) \in \mathcal{K}_j$.

Let $\mathcal{K} = \cup_{i \in I} \mathcal{K}_i$.

Let W be the set of all worlds in any model in $\mathcal{K}$. We can assume that each $w \in W$ identifies a unique Kripke model (S^w, R^w, w, h^w) in $\mathcal{K}$ (i.e. in one of the semantics of $\mathcal{K}_i$). With S^w its set of possible worlds, R^w its accessibility relation and h^w its assignment. Let $\tau(w) \in I$ be the function identifying to which semantics w belongs.

We now need to associate with each w and each $t \in S^w$ and each $j \in I$ such that $j \neq \tau(w)$, a model in the semantics $\mathcal{K}_j$. Let this model be $w' \in W$. In case $j = \tau(w)$ let $\mathbf{F}(j, w, t) = t$, i.e. let $w' = t$. We thus have a fibring funciton $\mathbf{F}(j, w, t) = w'$. We can also write $w' = \mathbf{F}_j(w, t)$ and perceive $\mathbf{F}_j$ as the fibring function for the logic $\mathbf{L}_j$. We require $\tau(w') = j$. We also require that $\tau(w) = \tau(y)$ for all $y \in S^w$. The above construction shall be referred to as the *canonical fibring construction.*

Definition 15.2.1 (Fibring of propositional modal logics) *1. Let $\mathbf{L}_i, i \in I$ be a family of modal logics with $\Box_i, i \in I$, respectively. Let $\mathcal{K}_i$ be a class of models for which $\mathbf{L}_i$ is complete.*

A fibred model for the logic $\mathbf{L}_I^F$ has the form $(W, S, R, \mathbf{a}, h, \mathbf{F}, \tau, w_0)$ where W is a set of labels. S is a function giving for each w a set of possible worlds (labels) $S^w \subseteq W$. R is a function giving for each w, a relation $R^w \subseteq S^w \times S^w$. $\mathbf{a}$ is a function giving the actual world $\mathbf{a}^w$ of the model labelled by w. (In our preparatory discussion above, $\mathbf{a}^w$ was equal to w).

h is the assignment function $h^w(q) \subseteq S^w$, for each atomic q. τ is the semantical identifying function $\tau : W \to I$. $\tau(w) = i$ means that the model $(S^w, R^w, \mathbf{a}^w, h^w)$ is a model in $\mathcal{K}_i$. $\mathbf{F}$ is the fibring function, for each w and $t \in S^w$ and $i \in I$, such that $\tau(w) \neq i$, $w' = \mathbf{F}(i, w, t) \in W$ and $\tau(w') = i$. $w_0 \in W$ is the 'actual' model.

In case $\tau(w) = i$ we do not need to fibre. However, we can still define the function $\mathbf{F}$. We can let $\mathbf{F}(i, w, t)$ give us the part of the model w truncated at t. In other words, if w is the model $(S, R, w, h) \in \mathcal{K}_i$ and $t \in S$ then $\mathbf{F}(i, w, t)$ is essentially $(s, R, t, h \restriction S_t)$, where $S_t = \{s \in S \mid \exists n \geq 0 t R^n s\}$. Formally we define $\mathbf{F}(i, w, t) = w'$ and $\mathbf{a}^{w'} = t$ and $S^{w'} = \{s \in S^w \mid ((\exists n \geq 0) t (R^w)^n s)\}$ and $R^{w'} = R^w \cap (S^{w'} \times S^{w'})$ and $h^{w'} = h^w \restriction S^{w'}$. The model $\mathbf{F}(i, w, t)$, for the case $\tau(w) = i$ may not be in the semantics $\mathcal{K}_i$, for modal logics which are not normal, violating the requirement on the fibring function $\mathbf{F}$. Thus to have a strictly correct definition, we must assume that $\mathcal{K}_i$ is such that all truncated models obtained from models in $\mathcal{K}_i$ are also in $\mathcal{K}_i$.

2. *Satisfaction is defined as follows:*
   - *$(w, t) \vDash q$ iff $t \in h^w(q)$, for q atomic and $t \in S^w$.*
   - *$(w, t) \vDash \Box_i A$ iff $\tau(w) = i$ and for all $s(tR^w s \to s \vDash A)$ or $\tau(w) = j \neq i$ and $(w', t') \vDash \Box_i A$ where $w' = \mathbf{F}(i, w, t)$ and $t' = \mathbf{a}^{w'}$.*
   - *Note that because of our requirement on $\mathbf{F}(i, w, t)$ for the case of $\tau(w) = i$, the second clause above is identical to the first clause for this case.*

3. *We say A is valid if for all models of $\mathbf{L}_I^F$, with actual world $w_0 \in W$ we have $(w_0, \mathbf{a}^{w_0}) \vDash A$.*

Definition 15.2.2 (Fibring and Grafting Models) *We would like to define certain constructions with fibred structures.*

1. *Let $\mathbf{L}_i, i \in I$ be logics. Let $\mathbf{m}_0 = (S_0, R_0, a_0, h_0)$ be a model of one of the logics, say $\mathbf{L}_{i_0}$. For each $t \in S_0$ and each $i \neq i_0$ let $\mathbf{m}_{i,t} = (W^{i,t}, S^{i,t}, R^{i,t}, \mathbf{a}^{i,t}, h^{i,t}, \mathbf{F}^{i,t}, \tau^{i,t}, w_0^{i,t})$ be fibred structures with $\tau^{i,t}(w_0^{i,t}) = i$.*

We want to fibre $\mathbf{m}_{i,t}, i \neq i_0, t \in S_0$ onto $\mathbf{m}_0$ as follows:

First assume S_0 and $W^{i,t}$ are all pairwise disjoint sets. Also assume that $\mathbf{m}_{i,t}$ is named by its actual world $w_0^{i,t}$. We define the fibred model $\mathbf{m} = (W, S, R, \mathbf{a}, h, \mathbf{F}, \tau, a_0)$ as follows

- *Let $W = S_0 \cup \bigcup_{i \neq i_o, t \in S_0} W^{i,t}$*
- *Let S^w, for $w \in W$ be*

$$\begin{array}{rcl} S^w & = & (S^{i,t})^w \text{ for } w \in W^{i,t} \\ & = & S_{o,w} \text{ for } w \in S_0 \\ & & \text{where } S_{o,w} = \{t \in S_0 \mid (\exists n \geq 0) w R_0^n t\}. \end{array}$$

- *Let R^w, for $w \in W$ be*

$$\begin{array}{rcl} R^w & = & (R^{i,t})^w \text{ for } w \in W^{i,t} \\ R^w & = & R_0 \upharpoonright S_{0,w} \text{ for } w \in S_0 \end{array}$$

- *Let h^w, for $w \in W$ be*

$$\begin{array}{rcl} h^w & = & (h^{i,t})^w, \text{ for } w \in W^{i,t} \\ h^w & = & h_0 \upharpoonright S_{0,w} \text{ for } w \in S_0 \end{array}$$

- *Let $\mathbf{a}$ be defined as follows*

$$\begin{array}{rcl} \mathbf{a}^w & = & (\mathbf{a}^{i,t})^w, \text{ for } w \in W^{i,t} \\ \mathbf{a}^w & = & w \text{ for } w \in S_0 \end{array}$$

- *Let τ be defined as follows*

$$\begin{array}{rcl} \tau(w) & = & \tau^{i,t}(w), \text{ for } w \in W^{i,t} \\ \tau(w) & = & w \text{ for } w \in S_0 \end{array}$$

- *Let $\mathbf{F}(j, w, s)$, for $s \in S^w$ be:*
  $\mathbf{F}(j, w, s) = \mathbf{F}^{i,t}(j, w, s)$ for $w \in W^{i,t}$.
  $\mathbf{F}(j, w, s) = w_0^{j,s}$ for $w \in S_0$ and $j \neq i_0$
  $\mathbf{F}(i_o, w, s) = s$ for $w \in S_0$

The following holds at the new fibred model.

- *For any formula A of $\mathbf{L}_{i_0}$ and any $w \in S_0$ we have*
  $w \vDash A$ in $\mathbf{m}_0$ iff $w \vDash A$ in $\mathbf{m}$.
- *For any formula $\Box_i B$ of $\mathbf{L}_I^F, i \neq i_0$ and any $w \in S_0$ we have*
  $w \vDash \Box_i B$ in $\mathbf{m}$ iff $\mathbf{m}_{i,w} \vDash \Box_i B$.

2. *Let $\mathbf{m}$ be a fibred model with actual world w_0, with $\tau(w_0) = i_0$. Let $\mathbf{m}_i$ for $i \neq i_0$ be fibred models. We want to* graft *$\mathbf{m}_i$ onto $\mathbf{m}$ at w_0 by replacing $\mathbf{F}^{\mathbf{m}}(i, \mathbf{m}, w_0)$ by the model $\mathbf{m}_i$.*

$$\begin{array}{rcl} \text{Let } \mathbf{m} & = & (W^0, S^0, R^0, \mathbf{a}^0, h^0, \mathbf{F}^0, \tau^0, w_0) \\ \mathbf{m}_i & = & (W^i, S^i, R^i, a^i, h^i, \mathbf{F}^i, \tau^i, w_i) \end{array}$$

We can assume all w^i, W^0 are pairwise disjoint and that they are named by their actual worlds, i.e. $\mathbf{m} = w_0$ and $\mathbf{m}_i = w_i$.

Let $\mathbf{m}_{i,t}$, for $t \in S^{o,w_0}, t \neq w_0$ be $\mathbf{F}^0(i, w_0, t)$. Let $\mathbf{m}_{i,w_0} = \mathbf{m}_i = w_i$.

Then the grafting of $\mathbf{m}_i$ onto $\mathbf{m}$ is defined to be the same as the fibring of $\mathbf{m}_{i,t}$ onto $(S^{0,w_0}, R^{0,w_0}, w_0, h^{0,w_0})$

Theorem 15.2.3 (Completeness theorem for the fibred logic $\mathbf{L}_I^F$) *Let $\mathbf{L}_i, i \in I$ be modal logics with semantical classes of structures $\mathcal{K}_i$ and set of theorems $\mathbf{T}_i$. (i.e. $\mathbf{T}_i = \{A \text{ of } \mathbf{L}_i \mid A \text{ is valid in all } \mathcal{K}_i \text{ models }\}$). Let $\mathbf{T}_I^F$ be the following set of wffs of $\mathbf{L}_I^F$.*

1. $\mathbf{T}_i \subseteq \mathbf{T}_I^F$

2. **Modal Fibring Rule:**
If $\Box_i$ is the modality of $\mathbf{L}_i$ and $\Box_j$ of $\mathbf{L}_j, i, j$ arbitrary $i \neq j$ and $C = \bigwedge_{k=1}^n \Box_i A_k \to \bigvee_{k=1}^m \Box_i B_k \in \mathbf{T}_I^F$ then for all d, $\Box_j^d C \in \mathbf{T}_I^F$.

3. *$\mathbf{T}_I^F$ is the smallest set closed under (1), (2), modus ponens and substitution.*

Then $\mathbf{T}_I^F$ is the set of all wffs of $\mathbf{L}_I^F$ valid in all the fibred structures of $\mathbf{L}_I^F$.

Proof.

1. The case of soundness is easy to verify.
2. To show completeness we use induction on the number of nested different language modalities in the formula.

Case $n = 1$
B is a formula with only one nested different modalities. This means B is a wff of $\mathbf{L}_i$ for some $i \in I$. If $\mathbf{L}_I^F \not\mid\!\sim B$, then $\mathbf{L}_i \not\mid\!\sim B$. Hence there is a model in $\mathcal{K}_i$ which validates B. This model can easily be built up into a fibred model (using the canonical fibring construction) in which B holds.

Case $n > 1$
We can assume B is obtained from formulas of the form

- $B_i(\overline{x}^1, \overline{x}^2, \ldots \overline{x}^k, \overline{y})$,
- $\Box_{\tau(1)} A_1^1(\overline{y}), \ldots, \Box_{\tau(1)} A_{n(1)}^1(\overline{y})$

  $\vdots$
- $\Box_{\tau(k)} A_1^k(\overline{y}), \ldots, \Box_{\tau(k)} A_{n(k)}^k(\overline{y})$

Where $\Box_{\tau(j)}$ is the modality of $\mathbf{L}_{\tau(j)}$ and $\overline{x}^j$ is the sequence of variables $x_1^j, \ldots, x_{n(j)}^j$ and B_i is a formula of the logic $\mathbf{L}_i$ and B is obtained from B_i by the substitution Θ where $\Theta(x_j^r) = \Box_{\tau(r)} A_j^r$ and $\Theta(y_i) = y_i$.

Thus

$$B = B_i(x_j^r / \Box_{\tau(r)} A_j^r, \overline{y}).$$

Subcase a:
In B_i all of x_j^r are within the scope of the modality $\Box_i$.

In this subcase, by the induction hypothesis, each $\Box_{\tau(r)} A_j^r$ has lower number of nested modalities than B. We assume B is $\mathbf{L}_I^F$ consistent. Hence B_i is $\mathbf{L}_i$ consistent. We want to show that B has a fibred model. Our strategy is to consider all $\mathbf{L}_i$ models of B_i and try to augment at least one of them into a fibred model of B.

Let $\mathbf{m} = (S^{\mathbf{m}}, R^{\mathbf{m}}, a^{\mathbf{m}}, h^{\mathbf{m}})$ be any model of B_i. Since B is $B_i\Theta$, we might be able to augment $\mathbf{m}$ into a fibred model of B if we can realise the assignment $h^{\mathbf{m}}$ by fibred models of $\Theta(x_r^j)$. To be more specific, let $t \in S^{\mathbf{m}}$ be any possible world in the model $\mathbf{m}$. Consider the assignment $h^{\mathbf{m}}(t, x_r^j)$, for any atomic proposition x_r^j of B_i, of the sequence $\overline{x}^j$. $\Theta(x_r^j)$ is a formula $\Box_{\tau(j)} A_r^j$ of the logic $\mathbf{L}_{\tau(j)}$. In the fibred semantics, any such $\Box_{\tau(j)} A_r^j$ is evaluated at a fibred model. The question is whether we can realise the truth values $h^{\mathbf{m}}(t, x_r^j)$ in a suitable model $\mathbf{n}_j$, for all the atoms of $\overline{x}^j$. If we can do that for all j, then we can build a fibred model for B. Of course we cannot realise the values $h^{\mathbf{m}}(t, x_r^j)$, i.e. if we fibre an arbitary model $\mathbf{n}'_t$ at the point t, we might change the value of $B = B_1\Theta$ at $a^{\mathbf{m}}$.

We must consider two possibilities

- t is a point such that the values of the atoms x_r^j at t do not affect the truth value of $B_1(x_r^j, \overline{y})$ at $a^{\mathbf{m}}$. In this case we don't mind what model $\mathbf{n}'_t$ we fibre at t, because even though $h^{\mathbf{m}}(t, x_r^j)$ is not the same as the value of $\Theta(x_r^j)$ at $\mathbf{n}'_t$, the values of $B = B_1\Theta$ at $a^{\mathbf{m}}$ will not change.

- t is a point such that the values of x^j_r at t do affect the value of $B_1(x^j_r, \overline{y})$ at $a^{\mathbf{m}}$. In this case the model $\mathbf{n}_t$ must realise these values.

Our first observation is that the points t we need worry about are the points at distance less than $d+1$ from $a^{\mathbf{m}}$ (i.e. $a^{\mathbf{m}}(R^{\mathbf{m}})^k t$ holds for $k \leq d+1$), where d is the maximal depth of nested modalities in B_1. This can be proved by induction. for $d = 0$ there are no modalities and so $a^{\mathbf{m}}$ is all that matters. For $\Box E$ evaluated at $a^{\mathbf{m}}$, the values that matter are values of E at points s such that $a^{\mathbf{m}} R^{\mathbf{m}} s$. By the induction hypothesis the value of E at s is not affected by values of atoms at points at distance more than d from s.

We thus conclude that we need realise all values $h^{\mathbf{m}}(t, x^j_r)$, for all points t at a distance less than $d+1$ from $a^{\mathbf{m}}$.

How can we fail to do that? Let us check! Let

$$D^j_t = \bigwedge_{s=1}^{n(j)} [x^j_s]^{\varepsilon(j,s)}$$

$$C^j_t = \bigwedge_{s=1}^{n(j)} [\Box_{\tau(j)} A^j_s]^{\varepsilon(j,s)} = D^j_t \Theta.$$

where $\varepsilon(j,s) = h^{\mathbf{m}}(t, x^j_s)$. We use the notation: $\varphi^\varepsilon = \varphi$ if $\varepsilon = 1$ and $\sim \varphi$ if $\varepsilon = 0$. If the above C^j_t is consistent, then by the induction hypothesis a fibred model would exist. If C^j_t is inconsistent, then no fibred model exists. Assume now that for every model $\mathbf{m} \in \mathcal{K}_i$, there exists a j and a $t \in S^{\mathbf{m}}$ at distance $d(t)$ (with $d(t) \leq d+1$) from the actual world $a^{\mathbf{m}}$ (i.e. $a^{\mathbf{m}}(R^{\mathbf{m}})^{d(t)} t$) such that C^j_t is inconsistent. Since the number of possible different C^j_t is finite, this means that for some finite number of wffs $D_1, \ldots, D_m$ built up as conjunctions of atoms and their negations from among $\overline{x}^j$, we have that $\{D_s \Theta\}$ covers all the C^j_t and we have $\vdash \sim D_s \Theta, s = 1, \ldots m$. Therefore we have that

$$\vdash_{\mathbf{L}_i} (\bigwedge_{s=1}^{m} \Box_i^{d(s)} \sim D_s) \to \sim B_i$$

since in each model $\mathbf{m}$ of B_i one of the D_i is satisfied.

Hence

$$\vdash_{\mathbf{L}_I^F} (\bigwedge_{s=1}^{m} \Box_i^{d(s)} \sim D_s \Theta) \to \sim B_i \Theta$$

but $B_i \Theta$ is B and $\vdash_{L_I} \sim D_s \Theta$. Hence (by axiom (2) of $\mathbf{L}_I$)

$$\vdash_{\mathbf{L}_I} \Box_i^{d(s)} \sim D_r \Theta$$

and hence

$$\vdash_{\mathbf{L}_I} \sim B$$

which contradicts the assumptions.

We therefore conclude that there exists a model $\mathbf{m}$ of B_i for which all C^j_t, for all j and all $t \in S^m$ such that $d(t) \leq d+1$ are consistent. By the induction hypothesis there exist fibred models $\mathbf{W}^j_t$ for each C^j_t. In this fibred model we have for each x^j_r and $t \in S^m$

$$h^{\mathbf{m}}(t, x^j_r) = \text{ value of } \Box_{\tau(j)} A^j_r \text{ in } \mathbf{W}^j_t.$$

For points t such that $d(t) > d+1$, let $\mathbf{W}^j_t$ be an arbitrary fibred models. These models wil not necessarily realise the values of $h^{\mathbf{m}}(t, x^j_r)$ but we know that these values do not affect the value of B_1 at $a^{\mathbf{m}}$. Now that we have $\mathbf{W}^j_t$ for all $t \in S^{\mathbf{m}}$, we can assume that $\mathbf{W}^j_t$ are all pairwise disjoint. We can fibre $\mathbf{m}$ with $\mathbf{W}^j_t$ and get a new fibred model for $B = B_i \Theta$, as we do in Definition 15.2.2.

Subcase b:

We now examine the subcase where the conditions of *Subcase a* are not necessarily satisfied. Let B be an arbitrary formula. Certainly there exists a classical boolean combination $\Psi(\overline{x}_1, \ldots, \overline{x}_n, \overline{y})$, where $\overline{x}_i = (x^i_1, \ldots, x^i_{k(i)})$ and $\overline{y} = (y_1, \ldots, y_k)$ and a substitution Θ such that $B = \Psi\Theta$, where $\Theta x^j_r = \Box_{\tau(j)} A^j_r$ and $\Theta y_i = y_i$. Since we are trying to find a model for Ψ, we can assume that Ψ is in a disjunctive normal form $\Psi = \bigvee_s \alpha_s$.

Let α be such a disjunct.

$$\alpha_s = \bigwedge_{r,j} [x^j_r]^{\varepsilon(j,r,s)} \wedge \bigwedge_r y^{\varepsilon(r,s)}$$

At least for one such $\alpha_s, \alpha_s\Theta$ must be $\mathbf{L}_I$ consistent otherwise we will have $\vdash_{\mathbf{L}_I} \sim \Psi\Theta$.

Let this disjunct be α. (We omit reference to s.) Consider an arbitrary i and consider the formula

$$E_{\tau(i)} = \bigwedge_{r=1}^{k(i)} [\Box_{\tau(i)} A^i_r]^{\varepsilon(i,r)}$$

this is a consistent formula which has the form

$$E_{\tau(i)} = B_{\tau(i)}(\overline{z}_1, \ldots, \overline{z}_m, \overline{u})\Theta_1$$

with substitution Θ_1 as in *Subcase a* above. It therefore has a fibred model $\mathbf{W}_i$.

The models $\mathbf{W}_i$ can now be fibred into a model of $\alpha\Theta$ and hence of $\Psi\Theta$, as we have done in Definition 15.2.2.

This completes the induction and the proof of the theorem. ■

Example 15.2.4 *Consider two logics $\mathbf{K}_1$ and $\mathbf{K}_2$ with two $\mathbf{K}$ modalities $\Box_1$ and $\Box_2$. Consider the fibred cobination $\mathbf{K}^F_{1,2}$. By the previous theorem it can be axiomatised by taking all theorems of $\mathbf{K}_1$ and $\mathbf{K}_2$ together with the following modal fibring rule:*

$$\frac{\vdash \bigwedge_k \Box_i A_k \to \bigvee_k \Box_i B_k}{\vdash \Box_j [\bigwedge_k \Box_i A_k \to \bigvee_k \Box_i B_k]}$$

where $i, j \in \{1, 2\}$, $i \neq j$.

Note that necessitation is not available. This example investigates under what conditions necessitation is admissible.

The logic $\mathbf{K}$ has the following special properties:

1. $\vdash \Box\top$
2. $\vdash \Box(A \wedge B) \leftrightarrow \Box A \wedge \Box B$
3. *The disjunction property*

$$\frac{\vdash \Box A \to \Box B \vee \Box C}{\vdash (\Box A \to \Box B) \text{ or } \vdash (\Box A \to \Box C)}$$

The above three properties can be used to prove that the modal fibring rule is equivalent to necessitation.

First the rule can be reduced to

$$\frac{\vdash \Box_i A \to \Box_i B}{\vdash \Box_j(\Box_i A \to \Box_i B)}$$

using (2) and (3) above.

Second, since $(\top \to A) \leftrightarrow A$ we get from (1) that $(\Box\top \to \Box A) \leftrightarrow \Box A$.

This can be used to derive

$$\frac{\vdash \Box_i A}{\vdash \Box_j \Box_i A}$$

Since necessitation is admissible in each component we get that

$$\frac{\vdash A}{\vdash \Box_i A}$$

is admissible in the combination $\mathbf{K}_{1,2}$.

The key property which we have used is the disjunction property. This holds for $\mathbf{K}, \mathbf{K}4$, *and some other system and for all of these systems the above reduction of the modal fibring rule to necessitation stands.*

Example 15.2.5 *This example deals with non normal logics. The example we give is typical. Consider the following axiomatisation of a modal logic* $\mathbf{L}$.

0. *all substitution instances of truth functional tautologies.*
1. $\Box A$ *if* A *is a truth functional tautology.*
2. $\Box(A \wedge B) \leftrightarrow (\Box A \wedge \Box B)$
3. $\Box(\Box(A \wedge B) \leftrightarrow (\Box A \wedge \Box B))$
4. $\Box A \to \Box\Box A$
5. $\Box(\Box A \to \Box\Box A)$
6. $\Box A \to A$

We allow the rule of modus ponens but not necessitation.

This system can derive all theorems of $\mathbf{K4}$ *together with* $\Box A \to A$ *and all their modus ponens consequences. We cannot derive* $\Box(\Box A \to A)$ *because we do not have necessitation. This logic is complete for all frames of the form* (S, R, a) *where* R *is transitive and* aRa *holds. We have:*

$$\vdash_{\mathbf{L}} A \textit{ iff } a \vDash A \textit{ in all } (S, R, a, h).$$

It is interesting to see what happens if we fibre $\mathbf{L}$ *with itself. We get two* $\mathbf{L}$ *modalities* $\Box_1$ *and* $\Box_2$ *together with all* $\mathbf{L}$ *theorems for each* $\Box_1$ *and* $\Box_2$ *and the modal fibring rule. Unlike the previous example this rule cannot be simplified.*

Theorem 15.2.6

1. *Assume* $\mathbf{L}_i, i \in I$ *has the finite model property then so does* $\mathbf{L}_I^F$.
2. *Assume* $\mathbf{L}_i, i \in I$ *are finitely axiomatizable then so is* $\mathbf{L}_I^F$.
3. *Assume* $\mathbf{L}_i, i \in I$ *have the disjunction property then so does* $\mathbf{L}_I^F$.
4. *Assume* $\mathbf{L}_i, i \in I$ *are decidable, then so is* $\mathbf{L}_I^F$.

Proof. Follows from the completeness proof and (for decidability) the way the theorems are generated. ■

Dove-tailing Modal Logics

Definition 15.2.7 (Dove-tailing of propositional modal logics)

1. *Let* $\mathbf{L}_i, i \in I$ *be modal logics as in the previous definition, with* $\mathcal{K}_i$ *the class of models for* $\mathbf{L}_i$. *Let* $\mathbf{L}_I^D$ *(the dovetailing combination of* $\mathbf{L}_i, i \in I$*) be defined semantically through the class of all (dove-tailed) models of the form* $(W, \mathbf{R}, a, h)$, *where* W *is a set of worlds,* $a \in W$, h *is an assignment into* W *and for each* $i \in I, \mathbf{R}(i) \subseteq W \times W$. *We require that for each* i $(W, \mathbf{R}(i), a, h)$ *is a model in* $\mathcal{K}_i$. *We further require the following:*
   Let $t \in W$ *be such that there exists* $n_1, \ldots, n_k$ *and* $i_1, \ldots, i_k$ *such that* $a\mathbf{R}^{n_1}(i_1) \cdot \mathbf{R}^{n_2}(i_2) \ldots \cdot \mathbf{R}^{n_k}(i_k)t$ *holds.*
   Let $i \neq i_k$ *and let* $W^t = \{y \in W \mid \textit{for some } n \geq 0, t\mathbf{R}^n(i)y\}$
   Then the model $(W^t, \mathbf{R}(i) \upharpoonright W^t \times W^t, t, h \upharpoonright W^t)$ *is in* $\mathcal{K}_i$.

2. *We define the notion of* $w \vDash A$ *by induction.*
   - $w \vDash q$ *if* $w \in h(q)$ *for* q *atomic.*
   - $w \vDash \Box_i A$ *if for all* $y \in W$*, such that* $w\mathbf{R}(i)y$ *we have* $y \vDash A$.
   - $\vDash A$ *iff for all models and actual worlds* $a \vDash A$.

Definition 15.2.8 (Dove-tailing Models) *Let* $\mathbf{L}_i, i \in I$ *be logics. Let* $\mathbf{m}_0 = (S_0, R_0, a_0, h_0)$ *be a model in the logic* $\mathbf{L}_{i_0}$. *For each* $i \neq i_0$ *and* $t \in S_0$, *let*

$$\mathbf{m}_{i,t} = (S^{i,t}, \mathbf{R}^{i,t}, a^{i,t}, h^{i,t})$$

be a dove-tailed model. We want to dove-tail the models $\mathbf{m}_{i,t}$ *onto* $\mathbf{m}_0$. *We can assume that* S_0 *and* $S^{i,t}$ *are all pairwise disjoint. We define a model* $\mathbf{m} = (S, \mathbf{R}, a_0, h)$ *as follows:*

- *Let* $S = S_0 \cup \bigcup_{\substack{i \neq i_0 \\ t \in S^0}} S^{i,t}$
- *Let* $\mathbf{R}$ *be defined as follows:*
  $x\mathbf{R}(j)y$ *iff* $x \in S^{i,t}$ *and* $x\mathbf{R}^{i,t}(j)y$.
  or $x \in S_0$ *and* $j \neq i_0$ *and* $a^{j,x}\mathbf{R}^{j,x}(j)y$
  or $x \in S_0$ *and* $j = i_0$ *and* xR_0y.
- *Let* $h = h_0 \bigcup_{\substack{i \neq i_o \\ t \in S_0}} h^{i,t}$.

What we have basically grafted is the situation in the following Figure 15.2.

The following holds at the new dovetailed model

- *For any formula* A *of* $\mathbf{L}_{i_0}$ *and any* $w \in S_0$ *we have* $w \vDash A$ *in* $\mathbf{m}_0$ *iff* $w \vDash A$ *in* $\mathbf{m}$.
- *For any formula* $\Box_i B$ *of* $\mathbf{L}^D_I, i \neq i_0$ *and any* $w \in S_0$ *we have* $w \vDash \Box_i B$ *in* $\mathbf{m}$ *iff* $w \vDash \Box_i B$ *in* $\mathbf{m}_{i,w}$.

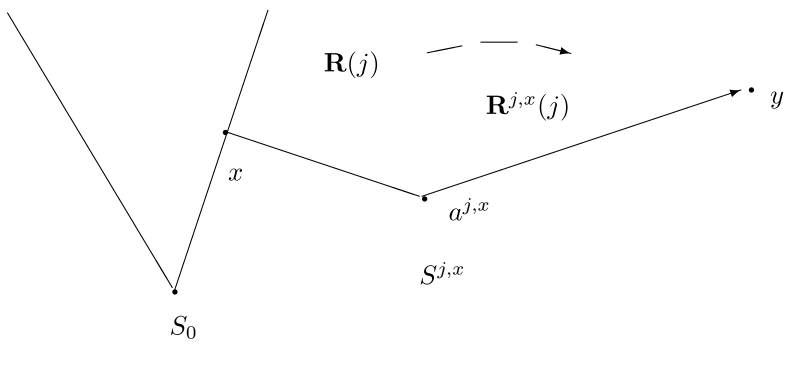


Figure 15.2:

Theorem 15.2.9 (Completeness theorem for the Dove-tailed logic $\mathbf{L}^D_I$) *Let* $\mathbf{L}_i, i \in I$ *be modal logics with semantical classes of structures* $\mathcal{K}_i$ *and set of theorems* $\mathbf{T}_i$. *Let* $\mathbf{T}^D_I$ *be teh following set of wffs of* $\mathbf{L}^D_I$.

1. $\mathbf{T}_i \subseteq \mathbf{T}^D_I$

2. **Modal Dove-tailing Rule***:*
If $\Box_i$ is the modality of $\mathbf{L}_i$ and $\Box_j$ of $\mathbf{L}_j, i, j$ arbitrary $i \neq j$ and

$$C = \bigwedge_{k=1}^{n} \Box_i A_k \wedge \bigwedge_{k=1}^{m} \Diamond_i \sim B_k \to \bigvee_{k=1}^{r} q_r \in \mathbf{T}_I^D$$

then for all d $\Box_j^d C \in \mathbf{T}_I^D$. Where q_r are atoms or their negations, and $q_1, \ldots, q_k$ list all the atoms or their negations appearing in any A_k or $B_k, k = 1, 2, \ldots$.

3. *$\mathbf{T}_I^D$ is the smallest set closed under (1), (2) modus ponens and substitution.*

Then $\mathbf{T}_I^D$ is the set of all wffs of $\mathbf{L}_I^D$ valid in all the dove-tailed structures of $\mathbf{L}_I^D$.

Proof.

1. The case of soundness is easy to verify.

2. To prove completeness we proceed along the lines of the proof of the previous theorem, the completeness theorem for $\mathbf{L}_I^F$.

The case to examine is *subcase a* of case $n > 1$. We have a model $\mathbf{m} = (S^{\mathbf{m}}, R^{\mathbf{m}}, a^{\mathbf{m}}, h^{\mathbf{m}})$ of $\mathbf{L}_i$ of the formula B_i and we want to augment it into a dove-tailed model of B.

For this purpose we have to realise the assignment of $h^{\mathbf{m}}(t, x_r^j)$, for points t such that $d(t) \leq d+1$ for all x_r^j by a fibred model for the substitution $\Theta(x_r^j) = \Box_{\tau(j)} A_r^j$ and in addition the model must realise all the values $h^{\mathbf{m}}(t, y_i)$, for the atoms y_i because the model will be dove-tailed at t. Thus the formulas C_t^j and D_t^j to be considered in our case are the following:

$$D_t^j = \bigwedge_{s=1}^{n(j)} [x_s^i]^{\varepsilon(j,s)} \wedge \bigwedge_r [y_r]^{\varepsilon(r)}$$

$$C_t^j = \bigwedge_{s=1}^{n(j)} [\Box_{\tau(j)} A_s^j]^{\varepsilon(j,s)} \wedge \bigwedge_r [y_r]^{\varepsilon(r)}$$

where

$$\varepsilon(j, s) = h^{\mathbf{m}}(t, x_s^j)$$
$$\varepsilon(r) = h^{\mathbf{m}}(t, y_r).$$

and

$$C_t^j = D_t^j \Theta.$$

Just like the previous proof, if for every $\mathbf{m}$ there is a $t \in S^{\mathbf{m}}$ and a j for which C_t^j is inconsistent, then for some finite $D_1, \ldots, D_m$ we have

$$\vdash_{\mathbf{L}_i} (\bigwedge_{s=1}^{m} \Box_i^{d(s)} \sim D_s) \to \sim B_i$$

Hence

$$\vdash_{\mathbf{L}_I^D} (\bigwedge_{s=1}^{m} \Box_i^{d(s)} \sim D_s \Theta) \to \sim B$$

But from the inconsistency of C_t^j for all j, t we get

$$\vdash \sim D_s \Theta.$$

The modal dove-tailing rule allows us to conclude

$$\vdash_{\mathbf{L}_I^D} \Box_i^{d(s)} \sim D_s \Theta$$

and so we get

$$\vdash_{\mathbf{L}_I^D} \sim B$$

a contradiction.

We can thus assume that for some model $\mathbf{m}$ of B_i, all the formulas C_t^j are realisable by a model $\mathbf{m}_t^j = (S^{j,t}, \mathbf{R}^{j,t}, a^{j,t}, h^{j,t})$.

We can now dove-tail these models with $\mathbf{m}$ into a model of B, as done in Definition 15.2.8. Note that for each t and atom $y, h^{\mathbf{m}}(t, y) = h^{j,t}(a^{j,t}, y)$.

Subcase b of $n > 1$ is proved also similarly to the fibred case. We assume there exists a classical boolean combination $\Psi(\overline{x}_1, \ldots, \overline{x}_n, \overline{y})$ and a Θ such that $B = \Psi\Theta$, where $\Theta(x_r^j) = \Box_{\tau(j)} A_r^j$, and $\Theta(y_i) = y_i$. Write Ψ in a disjuctive normal form and let α be any disjunct. At least one such α must be $\mathbf{L}_I^D$ consistent. Assume α is such a disjunct.

Consider the subformula of α.

$$E_{\tau(i)} = \bigwedge_{r=1}^{k(i)} [\Box_{\tau(i)} A_r^i]^{\varepsilon(i,r)} \wedge \bigwedge_r y_r^{\varepsilon(r)}.$$

This formula is consistent and has the form

$$E_{\tau(i)} = B_{\tau(i)}(\overline{z}_1, \ldots, \overline{z}_m, \overline{u})\Theta_1$$

with substitution Θ_1 falling under *subcase a*. It therefore has a dove-tailed model $\mathbf{m}_i$.

The models all agree on the value to $\overline{y}$ and can all be dove-tailed into a single model of α and hence of $B = \Psi\Theta$ as in Definition 15.2.8. ■

Theorem 15.2.10 *Assume $\mathbf{L}_i, i \in I$ all admit necessitation, then $\mathbf{L}_I^D$ (the dovetailing of $\mathbf{L}_i$) can be axiomatised by taking the union of the axioms and rules of $\mathbf{L}_i$*

Proof. The modal dove-tailing rule can be reduced to necessitation in this case. ■

Theorem 15.2.11 *If $\mathbf{L}_i, i \in I$ admit necessitation and satisfy the disjunction property, then $\mathbf{L}_I^F = \mathbf{L}_I^D$.*

Proof. Follows from the considerations of Example 15.2.4 and the previous theorem. ■

Theorem 15.2.12 *If $\mathbf{L}_i, i \in I$ all have the finite model property (are finitely axiomatizable, have the disjunction property) so is $\mathbf{L}_I^D$ and $\mathbf{L}_I^F$.*

Proof. From the completeness theorem.

1. If all participating models are finite then the fibred semantics will contain models which may not be finite but are made (fibred) from finite models. Any particular formula A of the fibred logic when evaluated at a fibred model, requires only a finite depth of fibring relative to the actual world and the model can be changed to a finite fibred model.

2. The transfer of finite axiomatizability and disjunction property are immediate from the completeness theorem.

3. The case of decidability is more tricky. If each $\mathbf{L}_i$ has its own totally different decision procedure, we may not have a decision procedure for $\mathbf{L}_I^D$ or $\mathbf{L}_I^F$. However, if they are all logics of the same type with the same decision procedure we may be able to transfer decidability. We can transfer dedidability if the procedure is tableaux like. The argument goes as follows. Take any formula A of the mixed language. A is of the form $B_i(D_1, \ldots, D_k)$ where $B_i(x_1, \ldots, x_k)$ is in the logic $\mathbf{L}_i$ and $D_1, \ldots, D_k$ are in some other of the fibring logics. Apply the tableaux method characteristic for the logic $\mathbf{L}_i$, regarding $D_1, \ldots, D_k$ as atomic. The tableaux will require assignment of values to the 'atoms' $D_1, \ldots, D_k$. When such values are needed, apply the tableaux methods of the logic $\mathbf{L}_j$ of the external connectives of each D_j. This can be done because D_j itslef is of the form $D_j^*(C_j^1, \ldots, C_j^{m_j})$ where $D_j^*(x_1, \ldots, x_{m_j})$ is in the logic $\mathbf{L}_j$ and C_j^r are in some other logic. The process will terminate because the complexity goes down.

The above is not a proper proof. We leave decidability and interpolation to a later part of this series. ■

Problem
Find sufficient conditions on $\mathbf{L}_i$ such that the fibred logic $\mathbf{L}_I^F$ is the same as the dove tailed logic $\mathbf{L}_I^D$.

To study this problem we examine two typical examples. First we want to know under what conditions fibring is identical with dovetailing. The condition on the semantics is that it be *forward looking.*

Definition 15.2.13 *1. A model (S, R, a) is* forward looking *iff for any assignment h, any t and any h' such that $h \upharpoonright S_t = h' \upharpoonright S_t$ and any wff $A(\Box B_1, \ldots, \Box B_k)$ we have $t \vDash_h A$ iff $t \vDash_{h'} A$, where $S_t = \{s \mid \exists n, tR^n s, n \geq 1\}$.*

2. $\mathcal{K}_i$ is said to be forward looking *if all of its structures are forward looking.*

3. A logic is forward looking *if it is complete for a forward looking semantics.*

Example 15.2.14 *1. If R is reflexive then it is not forward looking.*

*2. modal **K** and **K4** are forward looking.*

Theorem 15.2.15 *Let $\mathbf{L}_i, i \in I$ be logics with semantics $\mathcal{K}_i$. Let A be a wff of $\mathbf{L}_I^F$ which has a fibred model. Then A has a dove-tailed like model. The dove-tailed like model may not satisfy the condition that $(S, \mathbf{R}((i), a, h) \in \mathcal{K}_i)$. This condition can be guaranteed when each $\mathcal{K}_i$ is forward looking, in which case we get a proper dove-tailed model.*

Proof. We show that any fibred model be $(W, S, R, \mathbf{a}, h, \mathbf{F}, \tau)$ can be turned into an equivalent dovetailed model $(S, \mathbf{R}(i), a, h^D)$. Note that the dove-tailed model relies on the particular h^D, and that $(S, \mathbf{R}(i), a)$, for any fixed i may *not* be in the class $\mathcal{K}_i$.

Turning to the definition of $(S, \mathbf{R}_i, a, h^D)$, let S be W. We may assume that for any $x \in W$ there exists a unique w such that $x \in S^w$. Let $x, y \in W$. We let $x\mathbf{R}(i)y$ hold iff either

1. for some w, we have that $x, y \in S^w$ and $\tau(w) = i$, and $xR^w y$.
2. $x \in S^w$ and $y \in S^{w'}$ and $w' = \mathbf{F}(i, w, x)$ and $\mathbf{a}^{w'} R^{w'} y$.
   Let $h^D(q) = \cup_{w \in W} h^w(q)$

Notice that in clause (2) we took the model $(S^{w'}, R^{w'}, \mathbf{a}^{w'})$ and effectively replaced it by the model $(S^{w'}, R^{w'}, x)$, as we dove-tailed the possible worlds of $S^{w'}$ (except $\mathbf{a}^{w'}$) onto x. When the semantics of the logic of $\tau(w')$ is forward looking, this new model can also be assumed to be in the semantics and hence the dove-tailed model is propert, according to Definition 15.2.7.

Lemma 15.2.16 *For any (w, y) and A*

$$(w, y) \vDash_F A \text{ iff } y \vDash_D A$$

where $\vDash_F$ is satisfaction in the fibred model and $\vDash_D$ is satisfaction in the dovetailed model.

Proof. By induction.

Note that since $\forall y \exists! w (y \in S^w)$ holds, we can write $y \vDash_F A$ in place of $(w, y) \vDash_F A$.

The lemma holds for atomic q by definition.

$y \vDash_F \Box_i A$ iff $\mathbf{a}^{w'} \vDash_F \Box_i A$, where $w' = \mathbf{F}(i, w, y)$ iff $\forall x[\mathbf{a}^{w'} R^{w'} x \to x \vDash_F A]$ iff $\forall x[yR(i)x \to x \vDash_F A]$ iff $y \vDash_D \Box_i A$, by the induction hypothesis. ■

The lemma just proved that $(S, \mathbf{R}(i), a, h^D)$ is an equivalent model to $(W, S, R, \mathbf{a}, h, \mathbf{F}, \tau)$.

This equivalence holds for any semantics. We have not used in the proof of the lemma the assumption that the semantics we have fibred is forward looking. We need this assumption to show that under any assignment h^* (other than h^D) the structure $(S, \mathbf{R}(i), a, h^*)$ satisfies only $\mathbf{L}_I^F$ theorems. ■

Example 15.2.17 *We fibre a* $\mathbf{T}$*-modality* $\Box_1$ *with a* $\mathbf{K}$*-modality* $\Box_2$*, to show the difference between dovetailing and fibring.*

Let q *be atomic. Consider*

$$\Box_2\Box_1 q \wedge \Diamond_2 \sim q.$$

In the fibred semantics the above formula has a model. We can have a pair (w, y)*, such that* $wR^w y, (w, y) \vDash \sim q$ *and in the fibred model* $w' = \mathbf{F}(w, w, y)$ $\mathbf{a}^{w'} \vDash \Box_1 q$*. It is true that in* w' *by reflexivity* $\mathbf{a}^{w'} \vDash q$ *but this leads to no contradiction. In the dovetailed model, however, we will need to have at some* y *such that* wR_2y $y \vDash \sim q \wedge \Box_1 q$*, which is impossible.*

15.2.1 Fibring and Dove-tailing modal fragments

Example 15.2.18 (Fibring modal fragments) *The previous theorems dealt with fibring and dove-tailing full modal logics, with modalities* $\Box_i$ *and all the boolean connectives. We found theorems telling us how to fibre the semantics* $\mathcal{K}_i$ *of* $\mathbf{L}_i$ *and at the same time how to generate the theorems of* $\mathbf{L}_i, i \in I$*. The proof also shows that any semantics* $\mathcal{K}_i$ *for* $\mathbf{L}_i$ *will do for the purpose of fibring. If* $\mathbf{L}_i$ *is complete for* $\mathcal{K}_i$ *and for another semantics* $\mathcal{K}'_i$*, then fibring* $\mathcal{K}_i, i \in I$ *or* $\mathcal{K}'_i, i \in I$ *yields the same resulting logic and the same set of theorems for* $\mathbf{L}^F_I$ *or* $\mathbf{L}^D_I$ *respectively. We refer to this property as sematnical independence of the fibring process.*

This independence does not hold if we are dealing with modal fragments . Consider for example two modal fragments $\mathbf{L}_1$ *and* $\mathbf{L}_2$*, where* $\mathbf{L}_1$ *is the fragment of modal* $\mathbf{K}$ *with* $\{\Diamond_1\}$ *only while* $\mathbf{L}_2$ *is the fragment of* $\mathbf{K}$ *with* $\{\Box_2, \wedge\}$*. Each fragment is complete for the class* $\mathcal{K}$ *of all* $\mathbf{K}$ *Kripke models with arbitrary binary relations* R*. What do we mean by complete? Here we must use the consequence relation. Let* $A_1, \ldots, A_n \vDash_{\mathcal{K}} B$ *mean that in every Kripke model* $(S, R, , a, h) \in \mathcal{K}$*. If* $a \vDash A_i$*, for all* i *then* $a \vDash B$*. Completeness for the fragment with* $\Diamond_1$ *means that for all* A_i, B *in the fragment*

$$\{A_i\} \vdash_{\mathbf{K}} B \text{ iff } \{A_i\} \vDash_{\mathcal{K}} B$$

where $\vdash_{\mathbf{K}}$ *is provability in some syntactical formualtion of the full* $\mathbf{K}$*. If we fibre* $\mathbf{L}_1$ *and* $\mathbf{L}_2$ *according to (i.e. using) the* $\mathbf{K}$ *Kripke semantics, we get what is expected i.e. we get the* $\{\Diamond_1, \{\Box_2, \wedge\}\}$ *fragment of the full fibring of two full versions of* $\mathbf{K}$ *namely* $\{\{\Diamond_1, \wedge, \sim\}, \{\Box_2, \wedge, \sim\}\}$*. However because* $\mathbf{L}_1$ *with* $\Diamond_1$ *is a very weak language, all we can express in it is sequents of the form*

$$\Diamond_1^{n_i} q_i \vdash_{\mathbf{L}_1} \Diamond_1^n q$$

where q_i, q *are atomic.*

The above will hold in $\mathbf{K}$ *iff for some* $i, q_i = q$ *and* $n_i = n$*.*

Another semantics for $\mathbf{L}_1$ *is the class of all Kripke models of the form* $(\mathcal{N}, <_1, 0, h)$ *where* $\mathcal{N}$ *is the set of natural numbers and* $x <_1 y$ *iff* $y = x + 1$*. This semantics gives* $\Diamond_1 q$ *the reading 'q holds the next day'.*

If we fibre $\mathbf{L}_1$ *and* $\mathbf{L}_2$ *using the natural numbers Kripke sematnics for* $\mathbf{L}_1$ *we get the new theorem*

(*) $\Diamond_1\Box_2 q_1 \wedge \Diamond_1\Box_2 q_2 \vdash_{\mathbf{L}_{1,2}} \Diamond_1\Box_2(q_1 \wedge q_2)$

This theorem does not hold if we fibre $\mathbf{L}_1$ *and* $\mathbf{L}_2$ *using the* $\mathbf{K}$ *semantics for* $\mathbf{L}_1$*. The reason that we can get (*) is that conjunction becomes available through the fibring, and* $\Diamond_1$ *with* $\wedge$ *is sensitive to the different semantics chosen for* $\mathbf{L}_1$*, while* $\mathbf{L}_1$ *on its own is not senstive to the differences.*

It is instructive to see why the inductive argument of the fibring completeness theorem works for the full language, in the case of the natural numbers semantics.

Following our strategy we try to show that $\Diamond_1\Box_2 q_1, \Diamond_1\Box_2 q_2 \not\vdash \Diamond_1\Box_2(q_1 \wedge q_2)$ *in the fibred semantics.*

This form is a substitution instance of the form

$$\Diamond_1 x, \Diamond_1 y \not\vdash \Diamond_1 z$$

using the substitution Θ*, with* $\Theta(x) = \Box_2 q_1, \Theta(y) = \Box_2 q_2$ *and* $\Theta(z) = \Box_2(q_1 \wedge q_2)$*.*

A natural numbers counter model $(\mathcal{N}, <_1, 0, h)$ *can be obtained with* $h(1, x) = h(1, y) = 1$ *and* $h(1, z) = 0$*.*

The realisation of h for the substitution Θ will give us a fibred countermodel for the original consequence form. We need to fibre a model yielding

$$h(1, \Box_2 q_1) = h(1, \Box_2 q_q) = 1$$
$$\textit{and } h(1, \Box_2(q_1 \wedge q_2)) = 0$$

This is not possible because in $\mathbf{L}_2$

(**) $\Box_2 q_1 \wedge \Box_2 q_2 \vdash_{\mathbf{L}_2} \Box_2(q_1 \wedge q_2)$.

We now want to use this fact, (**), to show that (*) must hold.

When the full boolean connectives are available to us in $\mathbf{L}_1$ *we can write the form:*

$$\Box_1(x \wedge y \to z), \Diamond_1 x, \Diamond_1 y \vdash \Diamond_1 z$$

the addition of $\Delta_1 = \Box_1(x \wedge y \to z)$ *excludes the above counter model. If we find such a wff* D_r *for each unrealisable model, we get*

(***) $\{D_r\}, \Diamond_1 x, \Diamond_1 y \vdash \Diamond_1, z$

and after substitution we get

$$\{D_r\Theta\}, \Diamond_1\Box_2 q_1, \Diamond_1\Box_2 q_2 \vdash \Diamond_1\Box_2(q_1 \wedge q_2)$$

Since $\vdash_{\mathbf{L}_{1,2}} D_r\Theta$, *we get* $\Diamond_1\Box_2 q_1, \Diamond_1\Box_2 q_2 \vdash_{\mathbf{L}_{1,2}} \Diamond_1\Box_2(q_1 \wedge q_2)$. *Thus showing (*) is provable.*

Without boolean operations in $\mathbf{L}_1$ *we cannot write* D_r. *We may try to wrie the Gentzen Rule:*

$$\frac{x, y \vdash_{\mathbf{L}_1} z}{\Diamond_1 x, \Diamond_1 y \vdash_{\mathbf{L}_1} \Diamond_1 z}$$

but the meaning of the above as a rule involves validity in all models and is not what we want.

Remark 15.2.19 *Given two logics* $\mathbf{L}, \mathbf{M}$ *with* $\mathbf{L}$ *complete for several different semantics* $\mathcal{L}_1, \mathcal{L}_2, \ldots$ *and* $\mathbf{M}$ *is complete for several semantics* $\mathcal{M}_1, \mathcal{M}_2, \ldots$. *Then the fibring of the semantics* $\mathcal{L}_i$ *and* $\mathcal{M}_j$ *might yield different fibred logical systems. If we want to define a reasonable notion of semantically independent fibring of* $\mathbf{L}$ *and* $\mathbf{M}$ *we might wish to take the 'most general' semantics for each logic and fibre these. The investigations into this topic is postponed to the section on algebraic fibring. Our strategy would be to consider the algebraic logics (Lindebaum algebra?) of* $\mathbf{L}$ *and* $\mathbf{M}$ *and fibre these.*

The above discussion shows that fibring modal fragments may depend on the semantics chosen for the fragments. The reason being that we may have a fragment $\mathbf{L}$ which is syntatically too weak to distinguish between two differnt semantics $\mathcal{K}_1$ and $\mathcal{K}_2$ but when fibred with another logic $\mathbf{M}$, the fibred result can make the distinction. Thus we will have no way of predicting the set of theorems of the fibred result. simply by looking at the axiomatic formulation of $\mathbf{L}$ and $\mathbf{M}$. The question still remains of what happens in cases where all the logics involved, i.e. $\mathbf{L}$ and $\mathbf{M}$, are so weak that fibring does *not* help distinguish between the choice of semantics? Can we get a completeness theorem and thus establishing the semantical independence of the fibring process? The answer is yes under certain conditions.

The weakest (least expressive) modal logic is the *pure modality* logic with just $\Box$ and $\Diamond$ and atoms and no other connectives. The wffs of such a logic have the form $\overline{M}x$, where $\overline{M}$ is a string of modalities and x atomic. Any syntactical fibring of logics with pure modalities, $\mathbf{L}_i, i \in I$, will yield similar mixed wffs $\overline{M}x$, where x is atomic and $\overline{M}$ a string of modalities from some $\mathbf{L}_i, i \in J, J \subseteq I, J$ finite. Such fragments are of independent interest because when fibring modalities into other logics such as modal intuitionistic logic, one tends to fibre only the pure modal fragment. A completeness theorem for fibring pure modalities will therefore be very useful.

Theorem 15.2.20 (Completeness theorem for fibring and dovetailing pure modalities) *Let $\mathbf{L}_i, i \in I$ be modal logics in a language with connectives $\Box_i$ and/or $\Diamond_i$ together with atomic propositional variables. We assume for convienence that $\top, \bot$ are available.*

Let $\mathcal{K}_i$ be semantics (class of Kripke models) for which $\mathbf{L}_i$ is complete. Let $\mathbf{L}_I^F$ be the fibring (resp. Let $\mathbf{L}_I^D$ be the dovetailing) of $\mathbf{L}_i, i \in I$. Let $\mid\!\sim_i$ be the consequence relation of $\mathbf{L}_i$.
Let $\mid\!\sim$ be the consequence relation of $\mathbf{L}_I^F$ (resp. $\mathbf{L}_I^D$) defined by the following rules.

1. *Any substitution instance of $\Delta_i \mid\!\sim_i \Gamma_i$, where $\Delta_i \mid\!\sim \Gamma_i$ is in the language of $\mathbf{L}_i$.*

2. Pure Modalities fibring (resp. dovetailing) rules

$$\frac{A_1, \ldots, A_n \mid\!\sim C_1, \ldots, C_m}{\Box A_1, \ldots, \Box A_n, \Diamond B \mid\!\sim \Diamond C_1, \ldots, \Diamond C_m}$$

In case of fibring, A_i and C not atomic.

$$\frac{A_1, \ldots, A_n \mid\!\sim C, B_1, \ldots, B_m}{\Box A_1, \ldots, \Box A_n, \mid\!\sim \Box C, \Diamond B_1, \ldots, \Diamond B_m}$$

In case of fibring, A_i, B_j and C not atomic.

3. *$\mid\!\sim$ is the smallest consequence relation (monotonic, reflexive and transitve) closed under the above rules.*

   The following holds.
   Let Δ, Γ be two finite sets of wffs of $\mathbf{L}_I^F$ (resp. $\mathbf{L}_I^D$). If $\Delta \not\mid\!\sim \Gamma$, then there exists a fibred model $\mathbf{m}$ of $\mathbf{L}_I^F$ (resp $\mathbf{L}_I^D$) such that

   - *$A \in \Delta$ implies $\mathbf{m} \vDash A$*
   - *$A \in \Gamma$ implies $\mathbf{m} \nvDash A$*

Proof. The proof is by induction on the modal complexity of wffs in $\Delta \cup \Gamma$. Each formula of the language has the form $\overline{M}^i y_i$, with $\overline{M}^i$ a string of modalities and y_i atomic. Let $l(i)$ be the length of the string. The induction is on the number $n = \max_i l(i)$.

The inductive assumption on finite sets of wffs Δ, Γ of max $l(i) \leq n$ is:
Completeness
$\Delta \not\mid\!\sim \Gamma$ implies for some fibred model $\mathbf{m}, \mathbf{m} \vDash$ all of Δ and $\mathbf{m} \nvDash$ any $B \in \Gamma$.
Case $n = 0$
In this case all wffs of $\Delta \cup \Gamma$ are atomic. We therefore have that $\Delta \mid\!\sim \Gamma$ iff $\Delta \cap \Gamma \neq \varnothing$ clearly both inductive assumptions hold.
Case $n \geq 1$
In this case the elements of $\Delta \cup \Gamma$ have the form $\alpha_i = N_i \overline{M}^i y_i$, where N_i is the top front modality, which may not exist if α_i is atomic. $\overline{M}^i$ is the rest of the string, which may be empty. For a given $\Delta \not\mid\!\sim \Gamma$, we need to show a fibred (resp. dovetailed) model $\mathbf{m}$ as desired. We first observe that if $q_1, \ldots, q_k$ are all the atoms involved in $\Delta \cup \Gamma$, then we can assume that they are distributed among $\Delta \cup \Gamma$. In fact, if Ω is any set of formulas, we can assume that $\Omega \subseteq \Delta \cup \Gamma$ and $\Delta \not\mid\!\sim \Gamma$. This follows from the cut rule, because if for all partitions of Ω into Ω_1 and Ω_2 we have $\Delta \cup \Omega_1 \mid\!\sim \Gamma \cup \Omega_2$, then $\Delta \mid\!\sim \Gamma$ by cut.

We second observe that we can assume that all modalities N_i for all i belong to the same logic, say $\mathbf{L}_{\Box,\Diamond}$. The reason we can do that is that we can decompose Δ into $\Delta_1 \cup \Delta_2 \cup \ldots$ and Γ into $\Gamma_1 \cup \Gamma_2 \cup \ldots$ such that $\Delta_1 \not\mid\!\sim \Gamma_1, \Delta_2 \not\mid\!\sim \Gamma_2$ etc, and Δ_j, Γ_j are maximal in containing all the atoms and all the formulas of the same top modalities N_i of the same logic. If we find countermodels $\mathbf{m}_j$ for $\Delta_j \not\mid\!\sim \Gamma_j$, then we can put them together as done in Definition 15.2.2. Since Δ_i, Γ_i contain all atoms, the case of dovetailing is also covered, i.e. the models can be dovetailed.

We therefore assume that all N_i are of the same logic, $i = 1, 2, \ldots$.

We therefore need to find a model $\mathbf{m}$ of the logic $\mathbf{L}$ such that

- $\mathbf{m} \vDash A$ for $A \in \Delta$

- $\mathbf{m} \not\vDash A$ for $A \in \Gamma$

The third step in our analysis is to ask whether we can find a countermodel $\mathbf{m}$ in which $\Diamond\top$ holds. If such an $\mathbf{m}$ exists then we can assume $\Diamond\top \in \Delta$. If such a model does not exist then all fibred Kripke models which satisfy Δ and falsity Γ have only one possible world, the actual world, which is not accessible to anything.

In this latter case $\Delta \mathrel{|\!\!\sim} \Box\bot$ and Γ contains only wffs $\Diamond A$. We thus found a very special countermodel for $\Delta \mathrel{|\!\!\not\sim} \Gamma$.

We have now finished our initial analysis of Δ, Γ.

Let us assume that

$$\Delta = \{\Box D_1, \ldots, \Box D_{r_1}, \Diamond E_1, \ldots, \Diamond E_{r_2}\} \cup \Omega_1$$

and

$$\Gamma = \{\Box D'_1, \ldots, \Box D'_{r'_1}, \Diamond E'_1, \ldots, \Diamond E'_{r'_2}\} \cup \Omega_2$$

and that $E_1 = \top$. $\Omega_1 \cup \Omega_2$ cover all basic atoms appearing in wffs of $\Delta \cup \Gamma$.

We can assume that D_j, E_j, D'_j, E'_j are wffs of the form $\overline{M}^i_1 y_i$, where $\overline{M}^i_1$ are the tails of the strings of modalities in $\Delta \cup \Gamma$.

We need a fibred (resp. dovetailed) Kripke model $\mathbf{m} = (W, S, R, \mathbf{a}, h, \mathbf{F}, \tau, w_0)$ such that $\tau(w_0) = \mathbf{L}$ and $(S^{w_0}, R^{w_0}, w_0, h^{w_0}) \in \mathcal{K}$ such that for all $s \in S^{w_0}, w_0 R^{w_0} s \to (s \vDash D_j$ and $s \not\vDash E'_j)$ and for some t_j and $s_j, w_0 R^{w_0} t_j$ and $w_0 R^{w_0} s_j$ and $t \vDash E_j$ and $s_j \not\vDash D'_j$.

If for all j we have that both (*1) and (*2) below hold, then by the induction hypothesis we can find appropriate countermodels and take their disjunctive amalgamation, and complete the induction step.

(*1) $\{D_1, \ldots, D_{r_1}, E_j\} \mathrel{|\!\!\not\sim} \{E'_1, \ldots, E'_{r'_2}\}$

(*2) $\{D_1, \ldots, D_{r_1}\} \mathrel{|\!\!\not\sim} \{E'_1, \ldots, E'_{r'_2}, D'_j\}$

Otherwise either

$$(1**) \qquad \Box D_1, \ldots, \Box D_{r_1}, \Diamond E_j \mathrel{|\!\!\sim} \Diamond E'_1, \ldots, \Diamond D'_{r'_2}$$

or

$$(2**) \qquad \Box D_1, \ldots, \Box D_{r_1} \mathrel{|\!\!\sim} \Diamond E'_1, \ldots, \Diamond E'_{r'_2}, \Box D'_j$$

In either case this contradicts $\Delta \mathrel{|\!\!\not\sim} \Gamma$.

This completes the inductive step and the proof of the theorem. ■

15.2.2 The disjunction property

The logics $\mathbf{L}_i$ may be complete for several possible semantics. We may have that $\mathbf{L}_i$ is complete for a class of models $\mathcal{K}_i$ which is *disjunctive*, (defined below). In such a case the fibred semantics will also be disjunctive.

Definition 15.2.21 (Disjunctive Semantics) *Let $\mathcal{K}$ be modal or intuitionistic semantics. $\mathcal{K}$ is said to be* disjunctive *iff for any finite number of models $(S_i, R_i, a_i, h_i) \in \mathcal{K}$ there exists a model $(S, R, a, h) \in \mathcal{K}$ such that the following holds:*

$$S_i - \{a_i\} \subseteq S - \{a\}$$

- $R_i \restriction (S_i - \{a_i\})^2 = R \restriction (S_i - \{a_i\})^2$
- $h_i \restriction (S_i - \{a_i\}) = h \restriction (S - \{a\})$
- *$x \in S_i$ and $a_i R_i x$ imply aRx*
- *$x \in S$ and aRx imply for some $i, a_i R_i x$.*

The model (S, R, a, h) is called a disjunctive amalgamation *of (S_i, R_i, a_i, h_i).*

Example 15.2.22 *There are many logics with disjunctive semantics, e.g.* **K, K4, T**, *etc.*

Definition 15.2.23 (Disjunctive amalgamation of fibred models) *Let $\mathbf{L}_i, i \in I$ be logics with semantics $\mathcal{K}_i$. Let $\mathbf{L}_I^F$ be their fibred logic through their semantics. Assume that $\mathcal{K}$ is disjunctive. let*

$$\mathbf{m}_i = (W^i, S^i, R^i, \mathbf{a}^i, h^i, \mathbf{F}^i, \tau^i, w_0^i)$$

$i = 1, \ldots, k$ be k fibred models such that W^i are all pairwise disjoint. Assume we identify the name of a fibred model with its actual world. We also assume that $\tau(w_0^i) = \mathbf{L}_0$, i.e. $\mathbf{n}_i^0 = (S^{i,w_0^i}, R^{i,w_0^i}, \mathbf{a}^{i,w_0^i}, h^{i,w_0^i})$ are all in the class of models of $\mathcal{K}_0$ (here we identify each fibred model by its actual world). We want to define a disjunctive amalgamation $\mathbf{m}$ of the fibred models $\mathbf{m}_i$.

Since $\mathcal{K}_0$ is disjunctive, there exists a model $\mathbf{n}^0 \in \mathcal{K}_0$ which amalgamates $\mathbf{n}_i^0$. This means that

$$\begin{array}{l} \mathbf{n}^0 = (S_0, R_0, a_0, h_0) \\ S^{i,w_0^i} \subseteq S_0, \\ a_0 R w_0^i \\ R^{i,w_0^i} = R_0 \upharpoonright S^{i,w_0^i} \\ h^{i,w_0^i} = h_0 \upharpoonright S^{i,w_0^i} \end{array}$$

The above for $i = 1, \ldots, k$.

We can now define $\mathbf{m}$.

$$\mathbf{m} = (W, S, R, \mathbf{a}, h, \mathbf{F}, \tau, a_0)$$

The idea being that in the model $\mathbf{n}_0$, some of the points are from S^{i,w_0^i} and some are new. For the old points we fibre the models $\mathbf{m}_i$. For the new points, we fibre arbitrarily, say $\mathbf{m}_1$. The definition is as follows:

$$\begin{array}{l} W = S_0 \cup \bigcup_i W^i \\ X^w = X^{i,w} \text{ for } w \in W^i, w \neq w_0^i \\ X^{w^{i_0}} = X_0 \\ X^w = X_0 \text{ otherwise (for } w \in -S_0 - \bigcup_i W^i). \end{array}$$

The above for X being either $S, R, \mathbf{a}$ or h.

$$\begin{array}{l} \tau(w) = \tau^i(w) \text{ for } w \in W^i \\ \tau(w) = \mathbf{L}_0 \text{ otherwise} \\ \mathbf{F}(j, w, t) = \mathbf{F}^i(j, w, t) \text{ for } w \in W^i, t \in S^w \\ F(j, w, t) = \mathbf{F}^1(j, w_0^1, t), \text{ for } w = a_0 \text{ and } t \in S_0 - \bigcup_i W^i \end{array}$$

Theorem 15.2.24 (Completeness theorem for disjunctive semantics) *Let $\mathbf{L}_i, \mathcal{K}_i$ be as in Theorem 15.2.20 and assume further that each $\mathcal{K}_i$ is disjunctive. Then the fibred (resp. dovetailed) logic $\mathbf{L}_I^F$ (resp. $\mathbf{L}_I^D$) is also disjunctive. It can be axiomatised respectively by the following clauses (1D), (2D), and (3) of 15.2.20. The induction hypothesis also assumes the disjunction property, namely if $\Delta \mathrel{|\!\sim} \{\overline{M}^i y_i\}$ then for some $i_0, \Delta \mathrel{|\!\sim} \overline{M}^{i_0} y_{i_0}$.*

$$(1D) \qquad \frac{A_1, \ldots, A_n \mathrel{|\!\sim} C}{\Box A_1, \ldots, \Box A_n, \Diamond B \mathrel{|\!\sim} \Diamond C}$$

$$(2D) \qquad \frac{A_1, \ldots, A_n \mathrel{|\!\sim} C}{\Box A_1, \ldots, \Box A_n \mathrel{|\!\sim} \Box C}$$

where in the case of fibring, A_i and C are assumed not atomic.

Proof. The proof follows the same lines as that of 15.2.20.

We reach the stage in the proof where we want to show that (*1) and (*2) hold.

We proceed to use (1D) and (2D).

If (*1) does not hold then by the disjunction property, for some j_0

$$\{D_1, \ldots, D_{r_1}, E_j\} \mathrel{|\!\sim}_{\mathbf{L}} E'_{j_0}$$

hence $\{\Box D_i, \Diamond E_j\} \vdash_{\mathbf{L}} \Diamond E'_{j_0}$ hence $\Delta \mathrel{|\!\sim} \Gamma$.

If (*2) does not hold then either

$$\{D_1, \ldots, D_{r_1}\} \mathrel{|\!\sim}_{\mathbf{L}} D'_j$$

and hence

$$\{\Box D_i\} \mathrel{|\!\sim}_{\mathbf{L}} \Box D'_j$$

and hence

$$\Delta \mathrel{|\!\sim} \Gamma$$

or

$$\{D_1, \ldots, D_{r_1}\} \mathrel{|\!\sim}_{\mathbf{L}} E_j$$

and hence (since $E_1 = \top$)

$$\{\Box D_i, \Diamond E_1\} \mathrel{|\!\sim}_{\mathbf{L}} \Diamond E'_j$$

and hence $\Delta \mathrel{|\!\sim} \Gamma$.

This completes the inductive step and the proof of the theorem. ■

15.3 Case study 2: Intuitionistic modal logics

This section studies the combination of pure modality $\Box$ and or $\Diamond$ into intuitionistic logic. It begins with a general discussion of what is involved leading to definition 15.3.1 — fibred semantics for intuitionistic modal logic. This is the most general combination. The completeness theorem is proved in Theorem 15.3.3. The proof is rather involved, mainly because classical negation is not available, and we need to deal with negative information in a roundabout way. However, the effort is worthwhile, as Example 15.3.7 shows.

Let us now begin. Consider a language with atomic propositions and the connectives $\wedge, \Box, \Rightarrow$. We want $\wedge$ to be intuitionistic conjunction, $\Box$ to be **S4** modality and $\Rightarrow$ to be intuitionistic implication. Thus $\mathbf{L}_1$ is intuitionistic logic with $\{\wedge, \Rightarrow\}$ and $\mathbf{L}_2$ is **S4** modal logic with $\{\Box\}$, both based on the same atoms. We are looking for $\mathbf{L}_{1,2}$. Let us try and find semantics for this logic. Consider a wff in the language. Say $q \Rightarrow \Box A$.

Since the main connective is intutionistic, let us evaluate it at an intuitionistic Kripke model, say $\mathbf{m} = (S, \leq, 0, h)$. S is the set of possible worlds. $\leq$ is the reflexive and transitive accessibility relation. $0 \in S$ is the actual world and h is the assignment to the atoms. h satisfies the assignment restriction

$$(*) \qquad t \in h(q) \wedge t \leq s \to s \in h(q)$$

for all atoms q.

Let us now evaluate our formula in the model.

$\mathbf{m} \vDash (q \Rightarrow \Box A)$ iff by definition $0 \vDash (q \Rightarrow \Box A)$, where 0 is the actual world.

$0 \vDash (q \Rightarrow \Box A)$ iff for all t, if $0 \leq t$ and $t \vDash q$ then $t \vDash \Box A$.

We can check whether $t \vDash q$ because this holds iff $t \in h(q)$. However, we do not know how to evaluate $t \vDash \Box A$. The connective $\Box$ is not recognised in $\mathbf{m}$. However to answer whether $\mathbf{m} \vDash q \Rightarrow \Box A$ all we need is a value for $t \vDash \Box A$ for every t. How we get it is our problem.

Let us then attach to t a modal logic model

$$\mathbf{n}_t = \mathbf{F}(\mathbf{m}, t) = (T^t, R^t, a^t, h^t)$$

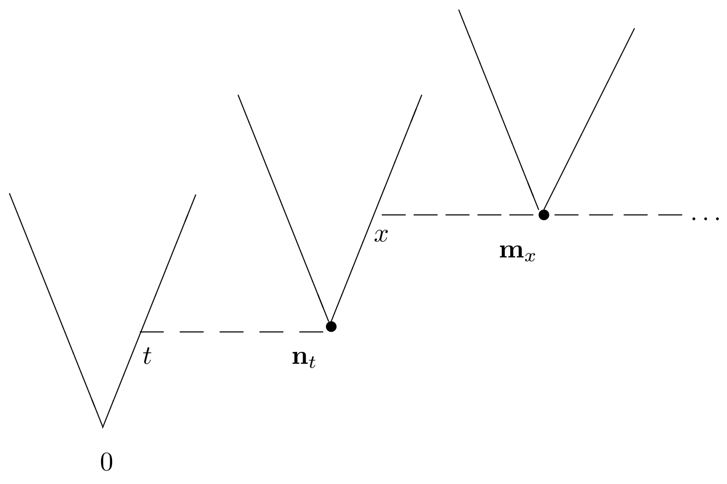


Figure 15.3:

where $\mathbf{n}_t$ depends on $\mathbf{m}$ and on $t \in S$ and where T^t is the set of possible worlds of $\mathbf{n}_t, R^t$ is the $\mathbf{S4}$ accessibility relation. a^t is the actual world and h^t is the assignment.

We stipulate:

$$t \vDash_{\mathbf{m}} \Box A \text{ iff (definition) } \mathbf{n}_t \vDash \Box A$$

The latter is the same as $a^t \vDash_{\mathbf{n}_t} \Box A$.

Since $\mathbf{n}_t$ is a modal model, a value can be found.

We continue the evaluation: $a^t \vDash \Box A$ iff for all $x \in T^t, a^t R^t x$ implies $x \vDash A$.

If A is in the modal language we can continue to evaluate. If A is atomic we get our value from h^t. What if A contains the intuitionistic $\Rightarrow$? ie $A = (p \Rightarrow r)$? Then we need to evaluate $x \vDash_{\mathbf{n}_t} (p \Rightarrow r)$.

Since $\mathbf{n}_t$ is a modal model, we do not know how to evaluate the above. However, all we need is a value. If we associate with x an intuitionistic model

$$\mathbf{m}_x = \mathbf{F}(\mathbf{n}_t, x) = (S^x, \leq^x, 0^x, h^x)$$

then we can continue our evaluation. $\mathbf{m}_x$ is functionally dependent on x and $\mathbf{n}_t$ through the function $\mathbf{F}$. We can represent the dependence as $\mathbf{F}(t, x)$.

We stipulate:

$$x \vDash_{\mathbf{n}_t} p \Rightarrow r \text{ iff (definition) } \mathbf{m}_x \vDash (p \Rightarrow r)$$

which is evaluated at the actual world of $\mathbf{m}_x$.

Thus we continue:

$$\mathbf{m}_x \vDash p \Rightarrow r \text{ iff } 0^x \vDash (p \Rightarrow r) \text{ iff for all } y \in S^x; 0^x \leq y \text{ and } y \vDash p \text{ implies } y \vDash r.$$

This process can be continued as many times as we need. To carry out the evaluation of arbitrary mixed formulas of $\mathbf{L}$, we need Kripke models of each logic and functions associating models of the other logic to each possible world, and infinitum, as illustrated in Figure 15.3.

These fibred models are not properly defined at the moment but are intuitively defined by the needs of combining logics. Note that we cannot put arbitrary modal models $\mathbf{n}_t$, for points $t \in S$, because they must satisfy the intuitionistic compatibility conditions

$$(*) \qquad t \leq s \text{ and } \mathbf{n}_t \vDash \Box A \text{ imply } \mathbf{n}_s \vDash \Box A$$

for any $\Box A$ of the modal language.

We need to secure the above condition (for arbitrary A):

$$\text{Let}\quad \begin{aligned}\mathbf{n}_t &= (T^t, R^t, a^t, h^t)\\ \mathbf{n}_s &= (T^s, R^s, a^s, h^s)\end{aligned}$$

We can assume without loss of generality that our Kripke models $\mathbf{n} = (T, R, a, h)$ satisfy the condition:

$$x \in T \text{ iff } (\exists m \geq 0) aR^m x$$

where xR^0y iff $x = y$ and $xR^{m+1}y$ iff $\exists z(xRz \wedge zR^m y)$.

We will argue that we can conservatively ensure that condition (*) holds for arbirary A by asking that we have:

(**) $t \leq s$ implies $T^t \supseteq T^s$ and $R^t \supseteq R^s$, and $h^s \supseteq h^t \restriction T^s$.

First note that the modal language $\mathbf{L}_2$ contains only atoms and modality $\Box$. So all wffs of $\mathbf{L}_2$ have the form $\Box^n q, q$ atomic. Assume $a^t \vDash \Box A$. This is equivalent to $\forall y(a^tR^ty \to y \vDash A)$. Similarly $a^s \vDash \Box A$ is equivalent to $\forall y(a^sR^sy \to y \vDash A)$. We thus have that condition (*) reduces to

$$(*1) \quad \forall y(a^tR^ty \to y \vDash A) \to \forall y(a^sR^sy \to y \vDash A)$$

Since A is arbitrary, it can be taken to be $\Box^m q$ for arbitrary m and atomic q. Thus we get:

$$(*2) \quad \forall q \forall m[\forall y(a^t(R^t)^m y \to y \in h^t(q)) \to \forall y \forall m(a^s(R^s)^m y \to y \in h^s(q))]$$

In particular we cannot have for some atomic q, t and s and y that $t \leq s$ and a^tR^ty and a^sR^sy and $y \in h^t(q)$ and $y \notin h^s(q)$. Thus the assignments h^t are not arbitrary but are increasing in assigning truth.

To ensure that his happens for every q we have two options:

1. Restrict h^t and h^s by requiring (*2) to hold, in which case we do not have arbitrary truth increasing assignment.
2. Allow for arbritary truth increasing assignments. In which case (*2) would imply that

   $$(*3) \quad \forall y \forall m[a^s(R^s)^m y \to a^t(R^t)^m y] \text{ and } h^s(q) \supseteq h^t(q) \restriction T^s$$

   which means that

   $$(**) \quad t \leq s \text{ implies } T^s \subseteq T^t \text{ and } R^s \subseteq R^t, \text{ and } h^s \supseteq h^t \restriction T^s.$$

Condition (**) is consistent with any Kripke model satisfying (*1). In other words, any Kripke model satisfying (*1) can be transformed into a Kripke model satisfying (**) which validates the same formulas. To prove this let

$$T^{*t} = T^t \cup \bigcup_{t \leq s} T^s$$

and let

$$a^tR^{*t}y \text{ iff def } a^tR^ty \vee \bigvee_{t \leq s} a^sR^sy.$$

Let

$$h^{*t}(q) = h^t(q) \cup \bigcup_{t \leq s} h^s(q).$$

Claim: For any $\Box A$

$$a^t \vDash \Box A \text{ iff } a^t \vDash^* \Box A$$

Proof. Assume $a^t \vDash \Box A$, then for all y such that a^tR^ty $y \vDash A$. Hence by (*1) we have for all y such that for some $s, t \leq s$ and a^sR^sy we have $y \vDash A$, hence for all y such that $a^tR^{*t}y$ we have $y \vDash A$ and hence $a^t \vDash *\Box A$. The other direction is easy. ■

We can thus assume that (**) holds in our semantics.

Consider the fibred intuitionistic modal logic with $\Rightarrow$ and $\Diamond$ (instead of $\Rightarrow$ and $\Box$). We need to secure a condition [*] on the fibred semantics, very much like the condition (*) of the previous example, namely

[*] $t \leq s$ and $\mathbf{n}_t \vDash \Diamond A$ imply $\mathbf{n}_s \vDash \Diamond A$.

Very much like the previous example one can show that to secure this condition one can assume condition [**] below.

[**] $t \leq s$ implies $T^t \subseteq T^s$ and $R^t \subseteq R^s$ and $h^t = h^s \restriction T^t$.

$$\begin{array}{rcl} \text{where } \mathbf{n}_t & = & (T^t, R^t, a^t, h^t) \\ \mathbf{n}_s & = & (T^s, R^s, a^s, h^s). \end{array}$$

In case the fibred modal logic contains both $\Box$ and $\Diamond$ we get equality in [**].

Definition 15.3.1 *(1)* **Fibred Semantics for Intuitionistic Modal Logic**
Let $\mathbf{L}_i, i \in I$ be a family of logics. Each $\mathbf{L}_i$ is either a fragment of modal logic with either a modality $\Box_i$ or a modality $\Diamond_i$ or both (we may not have negation in the fragment) or a fragment of some extension of intutionistic logic (i.e. an intermediate logic). A fibred model for the logic $\mathbf{L}_I^F$ has the form $(W, S, R, \mathbf{a}, h, \mathbf{F}, \tau, w_0)$ as in Definition 2.1. We add the following additional fibring conditions.

If $\tau(w) = i$ and $\mathbf{L}_i$ is an intuitionistic fragment and $\mathbf{L}_j$ is a modal or another intuitionistic fragment and $t, s \in S^w$ and $tR^w s$ and $t' = \mathbf{F}(j, w, t)$ and $s' = \mathbf{F}(j, w, s)$ (i.e. t' and s' are the fibred models at t and s respectively) then (**) and [**] hold whenever $\Box$ and/or $\Diamond$ are the modalities respectively present at $\mathbf{L}_j$, or (**) holds if $\mathbf{L}_j$ is intuitionistic

(**) $S^{t'} \supseteq S^{s'}, R^{t'} \supseteq R^{s'}, h^{s'} = h^{t'} \restriction S'$

*[**]* $S^{t'} \subseteq S^{s'}, R^{t'} \subseteq R^{s'}, h^{t'} = h^{s'} \restriction T'$.

*(2)***Dove-tailed Semantics for Intutionistic Modal Logic**
Let $\mathbf{L}_i$ be as before. A dove-tailed model for the logic $\mathbf{L}_I^D$ has the form $(W, \mathbf{R}, a, h)$, where W is a set of worlds, $a \in W$, h is an assignment into W and for each $i \in I, \mathbf{R}(i) \subseteq W \times W$ is the accessibility relation. For each $i \in I$, and $t \in W$, $(W, \mathbf{R}(i), a, h)$ is a model of the logic $\mathbf{L}_i$ (modal or intutionistic).

We also require for any intutionistic $\mathbf{L}_i$ and any $\mathbf{L}_j$ that

1. *If $\mathbf{L}_j$ is intuitionistic or $\Box$ only modal that $t\mathbf{R}(i)s \wedge s\mathbf{R}(j)t'$ imply $t\mathbf{R}(j)t'$*
2. *If $\mathbf{L}_j$ is $\Diamond$ modal then $t\mathbf{R}(i)s \wedge t\mathbf{R}(j)t'$ imply $sR(j)t'$*
3. *If $\mathbf{L}_j$ is $\Box, \Diamond$ modal then both (1), (2) hold.*

We also require the following:
Let $t \in W$ be such that there exist $n_1, \ldots, n_k$ and $i_1 \ldots, i_k$ such that $a\mathbf{R}^{n_1}(i_1) \cdot \mathbf{R}^{n_2}(i_2) \cdot \ldots \cdot \mathbf{R}^{n_k}(i_k)t$ holds. Assume $i \neq i_k$ and let

$$W^t = \{y \in W \mid t\mathbf{R}^m(i)y \text{ holds}\}.$$

Then the model $(W^t, \mathbf{R}(i) \restriction W^t, t, h \restriction W^t)$ is in $\mathcal{K}_i$.

Lemma 15.3.2 *1. Let $\mathcal{K}_I$ be the fibred semantics of the previous definition and assume that each $\mathcal{K}_i$ is disjunctive. Let Φ and Ψ be two finite sets of formulas of the form $A \Rightarrow_i B$, where $\Rightarrow_i$ is the implication of $\mathbf{L}_i$. Assume $\mathbf{L}_i$ does not contain disjunction. Assume further that there is a fibred countermodel $\mathbf{m}_\alpha$ for $\Phi \vdash \alpha$ for every $\alpha \in \Psi$, then there is a fibred counter model $\mathbf{m}$ for $\Phi \vdash \Psi$.*

2. *The same holds if Φ, Ψ are assumed to contain wffs of the form $\Box_j A, \Diamond_j A$ all in some $\mathbf{L}_j$.*

Proof. A similar construction to that which proves the disjunction property in ordinary modal or intutionistic logic. ■

We are now ready to state a completeness theorem for the fibring and dovetailing of modalities with intuitionistic connectives. We want a general fibring of an intutionistic or intermediate logic connective $\Rightarrow$ with one or several modalities $\Box_1$ and/or $\Diamond_2$. Thus for example we would like to fibre the $\Rightarrow$ of Dummett's **LC** with the **K** modalities $\Box$ and $\Diamond$. The result of the fibring, as well as the axiomatisation, is very much dependent on the semantics we take for the modalities and on the additional connectives which happen to be present, such as $\wedge, \vee$ and $\neg$. Thus no general theorem can be easily given. We will therefore choose one type of modality, **K** modality and only $\Rightarrow$ and do the fibring and dovetailing for that. The methodology will be clear from the proofs and we shall later indicate what happens if the modalities come from a stronger (possibly non-normal) extension of **K**.

The reader will notice throughout that the proofs use properties of intuitonistic logic in an essential way and one needs to think what one can do in a general case. This we postpone to a later section.

We now fibre several intermediate logics $\Rightarrow_i$ with several **K** modalities $\Box_j, \Diamond_j$. Since we might not have $\neg, \wedge$ and $\vee$, we need to formulate the logics to be fibred using the notion of consequence relation.

Theorem 15.3.3 (Completeness theorem for fibred (resp. dovetailed) intuitionistic modal logics)

1. *Let $\mathbf{L}_i$ be a logic with either an intermediate intutionistic implication $\Rightarrow_i$ (and possibly $\wedge, \vee$ and $\bot$) or with one or both of the $\mathbf{L}_i$ modalities $\Box_i, \Diamond_i$. Let $\mathcal{K}_i$ be the semantics for $\mathbf{L}_i$. We make the following assumptions.*

   (*) For intuitionistic $\mathbf{L}_i$, we assume that either disjunction is available or the semantics $\mathcal{K}_i$ is disjunctive.

   Let $\mid\!\sim_i$ be the consequence relation of $\mathbf{L}_i$. Then $\{A_j\} \mid\!\sim_i \{B_k\}$ means that in every Kripke model of $\mathbf{L}_i$ (modal or intuitionistic) of the form (S, R, a, h) whenever for all $j, a \vDash A_j$ then for some $k, a \vDash B_k$. We regard the consequence relation as a relation between finite sets of formulas. The completeness theorem characterises the consequence relation of $\mathbf{L}_I^F$ (denoted by $\mid\!\sim$) in terms of the consequence relations of $\mathbf{L}_i$ (denoted by $\mid\!\sim_i$).

2. *Let $\mid\!\sim$ be the smallest consequence relation containing $\mid\!\sim_i$ for $i \in I$, closed under substitution (and of course under reflexivity, monotonicity and cut) and closed under the following rules, where $\Rightarrow$ is a connective of an arbitrary $\mathbf{L}_{i_1}$ and $\Box, \Diamond$ of an arbitrary $\mathbf{L}_{i_2}$.*

   (a)

$$\frac{A_1, \ldots, A_n \mid\!\sim C_1, \ldots, C_m}{\Box A_1, \ldots, \Box A_n, \Diamond B \mid\!\sim \Diamond C_1, \ldots, \Diamond C_m}$$

   for the case of fibring A_i and C must begin with $\Box, \Diamond$ or $\Rightarrow$.

$$\frac{A_1, \ldots, A_n \mid\!\sim C, B_1, \ldots, B_m}{\Box A_1, \ldots, \Box A_n \mid\!\sim \Box C, \Diamond B_1, \ldots, \Diamond B_m}$$

   for the case of fibring A_i, B_j and C must begin with $\Box, \Diamond$ or $\Rightarrow$.

   (b) Deduction theorem for each $\Rightarrow_i$.
   Then $\mid\!\sim$ is the consequence relation of all the fibred models of $\mathbf{L}_I^F$ (resp. dovetailed models of $\mathbf{L}_I^D$).

Proof.

1. Soundness is easy to verify.

2. Completeness can be proved along similar lines to the proof of Theorem 15.2.3.

   In the case of modal logics, completeness was proved by induction on the number of nested modalities of different logics. Thus one would expect that we use induction in the modal intuitionistic case on the number of nested different connectives of the different logics. Thus for a formula of the form $\Box_2(A \Rightarrow_1 \Box_2\Diamond_3 B) \Rightarrow_4 B$ one may expect the inductive nesting number to be 4. This does not work, however, because the modal logics with $\Box$ and $\Diamond$ only do not have enough expressive power to carry the inductive case step. It is more convenient to count only nestings of intuitionistic $\Rightarrow$, because we can use the expressive power of the $\Rightarrow$ to carry the induction through. Thus the above formula falls under the inductive case for $n = 2$. The nestings are $A \Rightarrow_1 \Box_2\Diamond_3 B$ and the full $\Box_2(A \Rightarrow_1 \Box_2\Diamond_3 B) \Rightarrow_4 B$. Modalities do not count and they join the outer $\Rightarrow$ governing them. The above will be analysed as:

   Case $n = 1$
   Substitution into
   $$B_1(x, y) = [x \Rightarrow_1 \Box_2\Diamond_3 y]$$
   by
   $$\Theta_1(x) = A, \Theta_2(y) = B$$

   Case $n = 2$
   Substitution into
   $$B_2(x, y) = [\Box_2 x \Rightarrow_4 y]$$
   by
   $$\begin{array}{rcl}\Theta_1(x) & = & A \Rightarrow_1 \Box_2\Diamond_3 B \\ \Theta_2(y) & = & B.\end{array}$$

   Case n is described inductively as composed of wffs $B_n(\overline{x}, \overline{y})$, whose outer connective is some intuitionistic $\Rightarrow_i$ of the logic $\mathbf{L}_i$ and within it some modalities and atoms from $\mathbf{L}_j, j \in J$ (abbreviated $\mathbf{L}_J$). A substitution $\Theta_n(x_i)$ gives wffs of case $n - 1$.

Let $\Delta \not\vdash \Gamma, \Delta, \Gamma$ sets of wffs of $\mathbf{L}_I^F$ (resp. $\mathbf{L}_I^D$). We show by induction on the maximal number n of nested intuitionistic connectives (of different logics) in $\Delta \cup \Gamma$ that there exists a fibred (resp. dovetailed) model in which all members of Δ hold and all members of Γ do not hold.
Case n = 0
This case means that we have only connectives of the form $\Box$ and $\Diamond$ of various modal logics $\mathbf{L}_j, j \in J$. Let $\mathbf{L}_J$ be the logic with modalities of $\mathbf{L}_j, j \in J$. We need a lemma:

Lemma 15.3.4 *Let $A_1 \dots A_n \not\mid\!\sim B_1, \dots, B_k$ in $\mathbf{L}_J$. Then for some fibred (resp. dovetailed) model* $\mathbf{m}$ *of* $\mathbf{L}_J^F, \mathbf{m} \vDash A_i$ *and* $\mathbf{m} \not\vDash B_j$.

Proof. From the theorem on fibring (resp. dovetailing) pure modalities in the previous section, there exists a fibred modal model. The intuitionistic fibred part can be added arbitrarily.
Case $n = 1$
We prove that every Δ, Γ s.t. $\Delta \not\mid\!\sim \Gamma$ has a fibred (resp. dovetailed) countermodel. Our strategy is to regard all modally prefixed atoms in $\Delta \cup \Gamma$ as new atoms in a purely intuitionistic form $\Delta' \not\mid\!\sim \Gamma'$. We can find an intuitionistic countermodel for that form and realise the assignments to the new atoms by fibring (resp. dovetailing) modal models. This would give us a fibred countermodel for $\Delta \not\mid\!\sim \Gamma$. We now go through the details.

Let (S, R, a, h) be any intuitionistic countermodel for $\Delta' \not\mid\!\sim \Gamma'$ and let $\{A_1, \dots, A_m, B_{m+1}, \dots, B_k\}$ be the new modal atoms. Thus
$$\Delta' = \{\alpha_i(\overline{x}, \overline{y})\}, \Gamma' = \beta_j\{(\overline{x}, \overline{y})\}$$
and
$$\Theta(x_1) = A_1, \dots, \Theta(x_m) = A_m, \Theta(x_{m+1}) = B_{m+1}, \dots, \Theta(x_k) = B_k.$$

m is an arbitrary division to be chosen later. Let $t \in S$ and assume $h(t, x_i) = 1, i \leq m(t)$ and $h(t, x_i) = 0, m(t) < i \leq k$. If we can realise, for arbitrary t (m depends on t) $\{A_i\} \not\mid\!\sim \{B_j\}$ then we get a fibred (resp. dovetailed) countermodel for case $n = 1$. Otherwise $\{A_i\} \mid\!\sim \{B_j\}$. By the property (*) assumed in Theorem 15.3.3, $\{A_i\} \mid\!\sim B$ where B is either $\bigvee B_j$ or B_{j_0} for some j_0. Let $\delta_t = x_1 \Rightarrow_i (x_2 \Rightarrow_i \ldots, (x_{m(t)} \Rightarrow_i x_{j_0}) \ldots)$. There is only a finite number of possibilities for δ_t. Hence

$$\{\delta_t\} \cup \Delta' \vdash_{\mathbf{L}_i} \Gamma'$$

so

$$(\delta_t \Theta), \Delta' \Theta \mid\!\sim \Gamma' \Theta.$$

Since $\mid\!\sim \delta_t \Theta$ and $\Delta = \Delta' \Theta, \Gamma = \Gamma' \Theta$ we get $\Delta \mid\!\sim \Gamma$, a contradiction.

Thus for some intuitionistic countermodel (S, R, a, h) of $\Delta' \not\mid\!\sim \Gamma'$, all assignments at t are realisable by fibred models, $\mathbf{m}_t$. These models can be grafted onto (S, R, a, h) as in Definition 15.2.2. This grafted model can be turned to satisfy the requirements [*] and [**] of Definition 15.3.1 by following the construction at the beginning of the Section.

This completes the proof for case $n = 1$.

Case $n > 1$

We assume Δ, Γ are obtained from a Δ_i, Γ_i of a pure intuitionistic modal component logic $\mathbf{L}_J$ as in case $n = 1$ via a substitution Θ. In other words

$$\Delta_i = \{B_i^r \mid r = 1, \ldots, k\}$$
$$\Gamma_i = \{B_i^r \mid r = k + 1 \ldots m\}$$

and each B_i^r has the form

$$B_i^r(\overline{x}^r, \overline{y}^r)$$

Let $\Theta_0(x_r) = \overline{M}^r z_r$, and let $\Theta_1(z_r) = A_r \Rightarrow_j B_r$. Let $\Theta = \Theta_0 \Theta_1$ (Θ_0 first then Θ_1). Thus $\Theta(x_r) = \overline{M}^r(A_r \Rightarrow_j B_r)$, where $\overline{M}^r$ is a string of modalities from $\mathbf{L}_J$ and $A_r \Rightarrow_j B_r$ are from a logic $\mathbf{L}_j$.

We assume *subcase a* that each x variable is under the scope of the $\Rightarrow_i$ connective. Since $\Delta_i \Theta \not\vdash \Gamma_i \Theta$ we can assume $\Delta_i \Theta_0 \not\mid\!\sim_{\mathbf{L}_J} \Gamma_i \Theta_0$ and hence there are $\mathbf{L}_J$ models $\mathbf{m} = (S^{\mathbf{m}}, R^{\mathbf{m}}, a^{\mathbf{m}}, h^{\mathbf{m}})$ in which $\Delta_i \Theta_0$ holds and $\Gamma_i \Theta_0$ does not hold as shown in case $n = 1$. Reasoning very much like the case of Theorem 15.2.3 we want to show that there exists at least one $\mathbf{m}$ in which we can realise the assignment $h^{\mathbf{m}}(t, x_s^{r,j})$ for all t by a fibred model for $\Theta(x_s^{r,j})$, of the logic $\mathbf{L}_j$. Assume this is not possible. Then by the induction hypothesis for every $\mathbf{m}$, there exists a j and t such that $\Phi_{j,t} \Theta \vdash \Psi_{j,t} \Theta$, where

$$\Phi_{j,t} = \{x_s^{r,j} \mid h^{\mathbf{m}}(t, x_s^{r,j}) = 1, r, s \text{ varying}\} \vdash \Psi_{j,t}$$
$$= \{x_s^{r,j} \mid h^{\mathbf{m}}(t, x_s^{r,j}) = 0, r, s \text{ varying}\}.$$

Our aim is to reach a contradiction by using the fact that for every $\mathbf{m}$ there exists a j and $t \in S^{\mathbf{m}}$ such that $\Phi_{j,t} \mid\!\sim \Psi_{j,t}$. The contradiction is to show that $\Delta \mid\!\sim \Gamma$ contrary to assumptions. To achieve this end we need to extract some formulas $\delta_1, \delta_2, \ldots$ so that $\Delta_i \Theta_0, \delta_1 \Theta_0, \delta_2 \Theta_0, \ldots \mid\!\sim \Gamma_i \Theta_0$ and $\mid\!\sim \delta_1 \Theta, \mid\!\sim \delta_2 \Theta, \ldots$ and hence by cut, $\Delta = \Delta_i \Theta \mid\!\sim \Gamma = \Gamma_i \Theta$. We need a small lemma to the effect that we can assume that $\Psi_{j,t}$ contains (or is equivalent in $\mathbf{L}_j$ to) only one element $\beta_{j,t}$. Using this fact we can write

$$\delta_{j,t} = \varphi_1 \Rightarrow_j (\varphi_2 \Rightarrow_j \ldots \Rightarrow (\varphi_{k(j,t)} \Rightarrow_j \beta_{j,t} \ldots))$$

where

$$\Phi_{j,t} = \{\varphi_1, \ldots, \varphi_{k(j,t)}\}.$$

Since there is only a finite number of different possibilities for $\delta_{j,t}$, for different models $\mathbf{m}$, we get by case $n = 1$ that for some $\delta_1, \delta_2 \ldots, \Delta_i \Theta_0, \delta_1 \Theta_0, \delta_2 \Theta_0, \ldots \mid\!\sim \Gamma_i \Theta_0$.

Hence

$$\Delta_i \Theta, \delta_1 \Theta, \ldots, \vdash \Gamma_i \Theta$$

and since $\mid\!\sim \delta_r \Theta$ we get $\Delta \mid\!\sim \Gamma$ a contradiction.

We now prove the lemma.

Lemma 15.3.5 *We can assume that $\Psi_{j,t}$ contains only one element.*

Proof. Note that all the elements of $\Phi_{j,t}\Theta$ and $\Psi_{j,t}\Theta$ are of the form $(A \Rightarrow B)$. If $\mathbf{L}_j$ contains no disjunction in the language we get from 4.4 and the induction hypothesis that for some $x \in \Psi_{j,t}$ we have $\Phi_{j,t}\Theta \vdash \Theta(x)$. If $\mathbf{L}_j$ does contain disjunction then clearly $\Phi_{j,t}\Theta \vdash \bigvee \Psi_{j,t}\Theta$.

In either case we get a wff $\beta_{j,t}$ such that $\Phi_{j,t}\Theta \vdash \beta_{j,t}\Theta$. We can thus assume that $\Psi_{j,t}$ contains only one element. ■

We have thus concluded the proof of case $n > 1$ *subcase a.*
Subcase b
We now examine the subcase where the conditions of *subcase a* are not necessarily satisfied. We thus have $\Delta \not\vdash \Gamma$ and Δ, Γ are obtained by substitution into $\Delta_i \not\vdash_{\mathbf{L}_J} \Gamma_i$ and the formulas in $\Delta_i \cup \Gamma_i$ are of the form $B_i^r(\overline{M}, \overline{x}, \overline{y})$ as in *subcase a.*

We do not have in this case that each variable x is under the scope (in $B_i^r(\ldots x \ldots)$) of intutionistic connecitve $\Rightarrow_j$. This means that $\Delta_i \cup \Gamma_i$ is comprised of several formulas whose outer connectives are different intuitionistic ones $\Rightarrow_{j_1}, \Rightarrow_{j_2}$. Since these formulas are evaluated in different fibred models, we can apply *subcase a* to them separately. The following lemma explains

Lemma 15.3.6 *Let $j \neq j'$ and assume*

$$A \Rightarrow_j B, A' \Rightarrow_{j'} B' \vdash C \Rightarrow_j D, C' \Rightarrow_{j'} D'$$

then either

$$A \Rightarrow_j B \vdash C \Rightarrow_j D$$

or

$$A' \Rightarrow_{j'} B' \vdash C' \Rightarrow_{j'} D'.$$

■

Example 15.3.7 *The above theorem allows us to fibre or dovetail the following:*

1. *The intermediate logic* **KC** *obtained by adding the schema $\neg A \vee \neg\neg A$ to intuitionistic logic with* **S4** *modality $\Box$, and $\Diamond$.*
2. *Dummetts* **LC** *(obtained by adding the schema $(A \to B) \vee (B \to A)$ with the* **K** *modality $\Diamond$.*

The axioms of the combined system are obtained from the completeness theorem. For example, dovetailing the systems in (1) yields:

1. **KC** *axioms with Modus Ponens.*
2. $\Box A \wedge \Box(A \to B) \to \Box B$
   $\Box(A \to B) \wedge \Diamond A \to \Diamond B$
   $\Diamond\Diamond A \to \Diamond A$
   $\Box A \to \Box\Box A$
   $\Box A \to \Diamond A \vee \Box B$
   $\Diamond\neg A \vee \Box\neg\neg A$
   $\Diamond\neg\neg A \vee \Box\neg A$
   $\Box A \to A$
   $A \to \Diamond A$
3. $\dfrac{\vdash A}{\vdash \Box A}$

In the above examples we assumed the fibring is done by a function $\mathbf{F}(w, t) = z$. Thus the value of a new connective (e.g. $\Box$) is evaluated at a single model labelled by z. It is more general to associate with (w, t), not a single model z but a family of such models in which case $\mathbf{F}$ becomes a set fucntion or a relation $\mathbf{R}$. We thus get in $\mathbf{m}$:

$$t \vDash \sharp(A_i) \text{ iff for all } \mathbf{n} \text{ such that } (\mathbf{m}, t)\mathbf{R}\mathbf{n}, \text{ we have } \mathbf{n} \vDash \sharp(A_i)$$

The advantage of such an approach is that it is more general. Imagine some system where classical disjunction is a new connective. Then we do not necessarily want to have

$$\frac{A \mid\!\sim C, B \mid\!\sim C}{A \vee B \mid\!\sim C}.$$

15.4 Algebraic fibring of logics

Previous sections discussed fibring of modal and intutionistic logic through their possible world semantics. Many logics do not have clear possible world semantics and are best presented through their algebraic semantics. A typical example is many valued logics, where the algebraic presentation is the most natural. Thus for example to fibre modal logic with many valued logic we need to know how to construct a fibred many valued model. We need to define the notion of fibring of two algebraic semantics as well as the notion of fibring of an algebraic semantics with a possible world semantics. Furthermore, since many logics have both a possible world and an algebraic semantics, as for example is the case with modal logics and intuitionistic logic, we need to give careful definitions which would essentially yield the same fibred result no matter how we fibre the logics — algebraically, semantically or mixed. Thus for example the fibring of modal and intuitionistic logic should be indifferent to whether we fibre them through their Kripke semantics or through their algebras or in a mixed way. Another bonus to the algebraic formulation of fibring may come in connection with the problem of the semantical independence of the fibring process. We have seen in the case of fragments of modal logics that some fragments were complete for several different semantics and depending on the choice of semantics, the fibring of the semantics yielded different fibred logics with different sets of theorems. It is hoped that by taking the Lindenbaum algebraic semantics for these fragments and fibring them we would get a unique fibred system which we can regard as *the* fibring of the logics.

We begin by describing what we mean by algebraic semantics for a logic. We want a very general definition that would equally apply to many systems, whether they are monotonic or non-monotonic. There are many systems which have none of the classical connectives and which satisfy only some very restricted proof theory. There is no agreed standard definition of what is algebraic logic or semantics. We shall compare our approach to those of others as we go along. Our primary concern in this chapter is the fibring of logics, so any good definition of algebraic semantics will do. As it turns out our definitions are of independent interest. The new notion of algebraic logics for general consequence relations was introduced in Chapter 8, section 5.

This applies algebraic and shows how to fibre, dovetail and fuzzle various logics.

Definition 15.4.1 (Fibring and dovetailing algebraic logics) *A fibred algebra has the form* $(W, A, f_i, \leq T, h, \mathbf{F})$*, where* W *is the set of labels. For each* $w \in W, A^w \subseteq W$ *is the set of elements of the algebra labelled by* w. f_i^w *is the i-th function of the algebra of* w *and* $\leq^w, T^w$ *are its order and truth set.* $\mathbf{F}$ *is a fibring function, giving for each* w *and* $t \in A^w$ *and each logic* j *another algebra indexed by* $w' = \mathbf{F}(j, w, t)$*. The following must be satisfied*

$t \leq^w t'$ *implies* $\mathbf{F}(j, w, t)$ *is homomorphic onto* $\mathbf{F}(j, w, t')$*.*

We need an assignment function $h^w(q)$ *giving each atomic* q *a filter* $h^w(q)$ *in* A^w*.* h^w *can be extended to arbitrary wffs. We use the fibring function when we want to evaluate* $h^w(\sharp(q))$ *for a connective* $\sharp$ *which is in another logic* $\mathbf{L}_j$*. In this case we look at all* $w'_t = \mathbf{F}(j, w, t)$ *for all* $t \in A^w$*. We check at which point* t *the formula* $\sharp(q)$ *holds at the algebra* w'_t *i.e. we check* $\{t \mid h^{w'_t}(\sharp(q)) \in T^{w'_t}\}$*. The set of these points is a filter (because* $t \leq s$ *implies* w'_t *is homomorphic onto* w'_s *and hence if* $\sharp$ *holds at* w'_t *it will hold at* w'_s*). We let* $h^w(\sharp(q))$ *be this filter. The case of dovetailing of algebras can be distinguished from that of fibring by the following extra requirement:*

For all w *and all* $t \in A^w$ *and* $w'_t = \mathbf{F}(j, w, t)$ *we have for all atomic* q *that* $t \in h^w(q)$ *iff* $h^{w'_t}(q) \in T^{w'_t}$*.*

Example 15.4.2 *Let* $\mathbf{m}$ *be a fibred modal logic. Assume the fibring is done through its Kripke semantics. How would the fibring be done as an algebra?*

A Kripke model (S, R) *gives rise to an algebra where the elements are subsets* $X \subseteq S$. $X \leq Y$ *means* $X \supseteq Y$. *We need to fibre an algebra to each subset* $X \subseteq S$. *We do have Kripke models* $\mathbf{m}_x$ *fibred for each points* $x \in X$. *These give rise to algebras* $\mathbf{a}_x$. *Let* $\mathbf{a}_X = \Pi_{x \in X} \mathbf{a}_x$. *Clearly* $X \leq Y$ *means* $X \supseteq Y$ *and so* $\mathbf{a}_Y$ *is a subalgebra of* $\mathbf{a}_X$.

Thus we associate with each fibred modal model a fibred algebraic modal model through the above indicated translation. In the case of dovetailing, there is the commitment that the fibred model agrees on atomic values. This means algebraically that $X \vDash q$ *iff* $\mathbf{a}_X \vDash q$ *for all atoms.*

Example 15.4.3 (Many valued modal logic) *This is an example of fibring semantical models (modal logic) with algebraic models (Lukasiewitz many valued logic). We consider the many valued language with* $\{\wedge, \vee, \rightarrow, \neg\}$ *with truth values at the real interval* $[0\ 1]$. *The algebraic models are linearly ordered abelian groups which are embeddable in* $[0\ 1]$. *So it is sufficient to consider assignments* μ *of values and truth table for values in* $[0\ 1]$. *The following are the algebraic functions:*

- *The domain is* $[0\ 1]$.
- $\leq$ *is numerical* $\leq$.
- $T = \{0\}$ *(0 is truth).*
- $f_\wedge(x, y) = max(x, y)$.
- $f_\vee(x, y) = min(x, y)$.
- $f_\neg(x) = 1 - x$
- $f_\rightarrow(x, y) = max\ (0, y - x)$.

Strictly speaking, the assignments μ *into an algebra* $\mathbf{a} = (A, f_\wedge, f_\vee, f_\rightarrow, f_\neg, \{0\}, \mu)$ *should be to filters, i.e. sets of points closed under* $\leq$. *These filters can be characterised by a single point* $x \in [0\ 1]$ *either as* $\mu(q) = [x\ 1]$ *or* $\mu(q) = (x\ 1]$ *(left open or closed intervals).*

We will write $\mu(q) = x$ *for* $[x\ 1]$ *and* $x-$ *for* $(x\ 1]$.

We now turn to fibring and to dovetailing.

Let $\mathbf{m} = (S, R, a, h)$ *be a Kripke model for* $\Box$. *The fibring function* $\mathbf{F}$ *associates with each* $t \in S$ *an algebraic model* $\mathbf{a}_t = (A_t, \leq, f_\wedge, f_\vee, f_\rightarrow, f_\neg, \{0\}, \mu_t)$. *Since* $A_t = [0\ 1]$, *fibring algebras* $\mathbf{a}_t$ *to* t *is nothing more than associating with each* t *an arbitrary many valued assignment* μ_t *to the atoms* q *of the modal language. In case the semantics is dovetailed, we further require that for atoms* q *we have* $\mu_t(q) = 0$ *iff* $t \vDash q$.

We can further fibre to each element $x \in A_t$ *of the algebra a Kripke model* $\mathbf{n}_{x,t} = (S_{x,t}, R_{x,t}, a_{x,t}, h_{x,t})$. *The requirement is that for* $x \leq y, \mathbf{n}_{x,t}$ *is homomorphic onto* $\mathbf{n}_{y,t}$. *This means that the following holds for* $x \leq y$:

(*) For all $\Box A, \mathbf{n}_{x,t} \vDash \Box A$ *implies* $\mathbf{n}_{y,t} \vDash \Box A$.

This can be achieved as in the beginning of Section 3 by requiring that:
$x \leq y$ *implies* $S_{x,t} \supseteq S_{y,t}$ *and* $R_{x,t} \restriction S^2_{y,t} \supseteq R_{y,t}$ *and* $h_{y,t} = h_{x,t} \restriction S_{y,t}$. *The reason the assignment cannot be truth increasing but must remain constant is that we have negation in the language. We can write both* $\Box q$ *and* $\Box \sim q$ *and neither can change value.*

Let us now evaluate $\Box(q \rightarrow \Diamond q), q$ *atomic, at the model* $\mathbf{m}$.

- $a \vDash \Box(q \rightarrow \Diamond q)$ *iff for all* $t \in S$ *such that* $aRt, t \vDash q \rightarrow \Diamond q$.
- *Since the main connective of* $q \rightarrow \Diamond q$ *is many valued, we have* $t \vDash q \rightarrow \Diamond q$ *iff* $\mathbf{a}_t \vDash q \rightarrow \Diamond q$ *iff* $\mu_t(q \rightarrow \Diamond q) = 0$ *iff* $max(0, \mu_t(\Diamond q) - \mu_t(q)) = 0$ *iff* $\mu_t(q) \leq \mu_t(\Diamond q)$.
- $\mu_t(q)$ *is directly given since* q *is atomic. However* $\mu_t(\Diamond q)$ *is not directly given.* $\Diamond$ *is a modal operator and we need to go t the fibred modal models* $\mathbf{n}_{x,t}, x \in A_t$ *(recall that* $A_t = [0\ 1]$*).*

We let

$$Q_t = \{x \mid \mathbf{n}_{x,t} \vDash \Diamond q\}$$

Q_t is a filter because of (*) above. It has the form either $[\alpha_t\ 1]$ or $(\alpha_t\ 1]$.

Thus $\mu_t(\Diamond q) = \alpha_t$ (or α_t-). Thus we need to check whether $\mu_t(q) \leq \alpha_t$.

Example 15.4.4 *We would like to highlight a point in the previous example, which will be of importance later. Consider the top level fibring. We start with $\mathbf{m} = (S, R, a, h)$. Then with each $t \in S$, we fibre an algebra $\mathbf{a}_t$. If all the algebras have the same domain, the fibring reduces to μ_t, the assignment. Let us stop at this stage and consider the entity (S, R, a, h, μ) and let us try to evaluate $t \vDash \Box q$. Since $\Box q$ contains no mnay valued connectives, we get $t \vDash \Box q$ holds iff $\forall s(tRs$ implies $s \vDash q)$. Consider the wff $Iq = def\ (q \to q) \to q$. Really Iq is q but it is formally a many valued wff. So we have to evaluate it at the algebra $\mathbf{a}_t$. We have $\mathbf{a}_t \vDash I(q)$ iff $\mu_t(q) = 0$. $t \vDash \Box I(q)$ iff for all s (tRs implies $\mu_t(q) = 0$).*

We now have the opportunity to make $t \vDash \Box q$ fuzzy (i.e. 'fuzzle' the satifaction $\vDash$ or 'fuzzle' the modal logic) by extending μ_t to $\Box q$:

($\sharp$) $\mu_t(\Box q) = \mathrm{Sup}_{\{s \mid tRs\}} \mu_s(q)$.

Using ($\sharp$) we can fuzzle any wff of the modal logic and extend μ_t to all wffs. We take the many valued table for $\wedge, \vee, \neg$ and $\to$. We have thus by understable intuitive definition, through ($\sharp$), turned (S, R, a, μ) into a sort of modal many valued logic by changing the crisp $\{0\ 1\}$ assignment h into a fuzzy μ. Note that what we are getting is not fibring or dovetailing. It is something new.

Example 15.4.5 *Let us continue the previous example and try now to fuzzle many valued logic with modal logic. Consider the many valued model $\mathbf{a}_t = ([0\ 1], \ldots, \mu_t)$. To each $x \in [0\ 1]$, a modal model $\mathbf{n}_{x,t}$ was fibred. For $x \leq y$ we had $S_{x,t} \supseteq S_{x,y}$ and $R_{x,t} \upharpoonright S^2_{y,t} \supseteq R_{y,t}$ and $h_{y,t} = h_{x,t} \upharpoonright S_{y,t}$. Clearly $\mathbf{n}_{0,t}$ is the largest model. Let q be atomic. The value $x \vDash q$ in the algebra $\mathbf{a}_t$ is obtained by looking whether $\mu_t(q) \leq x$. Formally we can write*

$$val_t(x, q) = 1 \text{ iff } \mu_t(q) \leq x$$

$val_t(x, q)$ is a 'crisp' $\{0,1\}$ function. We now 'fuzzle' it by giving values in the modal algebra. Let val^*_t be the new fuzzy function.

$$val^*_t(x, q) = \{s \in S_{x,t} \mid s \in h_{x,t}(q)\}.$$

Since the modal models $\mathbf{n}_{x,t}$ satisfy (**) of the beginning of Section 2, we can assume that they all have the form $(S_t, R_{x,t}, a_t, h_t)$ and that $x \leq y$ implies $R_{x,t} \supseteq R_{y,t}$. There is no need to assume that S gets smaller with increasing $\leq$ because what is not connected by R in the model is practically non existent. We can also assume that $h_{x,t}(q) = h_t(q) \cap S_{x,t}$.

for $\alpha \in S_t$ let $\mu^\alpha_t(q)$ be $Inf_x \alpha \in h_{x,t}(q)$. Clearly in our particular case $\mu^\alpha_t(q)$ is either 0 or 1.

$$val^*_t(x, q) = \{s \in S_{x,t} \mid \mu^s_t(q) \leq x\}.$$

We can also turn $R_{x,t}$ into R^*_t a many valued relation by defining

$$R^*_t(\alpha, \beta) = Inf_x[R_{x,t}(\alpha, \beta) \text{ holds}]$$

for $\alpha, \beta \in S$, and of course

$$R_{x,t}(\alpha, \beta) \text{ holds iff } R^*_t(\alpha, \beta) \leq x.$$

We can thus assume that we are dealing with structures of the form $(S_t, R^*_t, a_t, h_t, \mu^\alpha_t, \mu_t)$ where $R^*_t(\alpha, \beta) \in [0\ 1]$.

$$\mu^\alpha_t(q) \in [0\ 1]$$

and we define

- $R_{x,t}(\alpha, t) = 1$ *iff* $R^*_t(\alpha, \beta) \leq x$

- *$\alpha \in h_{x,t}(q)$ iff $\mu_t^\alpha(q) \leq x$.*
- *$\mu_t(q)$ is the many valued assignment for q, which for the case of dovetailing, equals $\mu_t^{a_t}(q)$. (Note that since $\mu_t^{a_t}(q)$ is $\{0,1\}$ valued,dovetailing would collapse μ_t to a classical assignment.)*

We want to extend $\mu_t^\alpha(A)$ to arbitrary A . We do that by induction. We assume that for A and $B, \mu_t^\alpha(A) = x$ and $\mu_t^\alpha(B) = y$ are defined and we have

$$val_t^*(x, A) = \{s \in S_t \mid \mu_t^s(A) \leq x\}$$

Similarly for $val_t^*(x, B)$.

The obvious way to extend μ_t^α is the following

$$\begin{aligned} \mu_t^\alpha(\Box A) &= Inf_x \forall_\beta [R_{x,t}(\alpha, \beta) = 1 \text{ implies } \beta \in \ val_t^*(x, A)] \\ & Inf_x \forall_\beta [R_{x,t}(\alpha, \beta) \leq x \text{ implies } \mu_t^\beta(A) \leq x] \end{aligned}$$

Clearly also

$$\begin{aligned} \mu_t^\alpha(A \wedge B) &= \ max(\mu_t^\alpha(A), \mu_t^\alpha(B)) \\ \mu_t^\alpha(A \vee B) &= \ min(\mu_t^\alpha(A), \mu_t^\alpha(B)) \\ \mu_t^\alpha(A \to B) &= \ max(0, \mu_t^\alpha(B) - \mu_t^\alpha(A)) \\ \mu_t^\alpha(\neg A) &= 1 - \mu_t^\alpha(A) \end{aligned}$$

In other words, we are reading the connectives pointwise as should be.

Example 15.4.6 (A 'fuzzled' modal logic) *The previous two examples show that modal and many valued logic can be put together in two different ways. If we start with a modal model (S, R, a, h) then we can fuzzle (make fuzzy) h by changing it into a many valued assignment μ and extend to the entire modal langauge. If we start with a many valued model μ then we can fuzzle μ by changing it into a function into elements of a modal algebra. This turned out to be equivalent to looking at modal models where the possible world relation is fuzzy but the assignment is crisp. i.e. models of the form (S, R^*, a, μ) where $R^*(\alpha, \beta) \in [0\ 1]$, while μ is a {0,1} assignmnt. μ can be extended to all wffs, in which case it becomes a $[0\ 1]$ valued function.*

The obvious combination of the two approaches is to make both R^* and μ^* fuzzy. This leads us to the following definition.

Definition 15.4.7 *An algebraic fuzzled many valued modal model has the form (S, R^*, a, μ^*), where $R^* : S^2 \to [0\ 1]$ is a fuzzy possible world relation and for each $s \in S$ and atomic $q, \mu_s^*(q) \in [0\ 1]$.*

μ_s^* can be extended to arbitrary formulas as follows:

$$\mu_s^*(A \sharp B) = f_\sharp(\mu_s^*(A), \mu_s^*(B))$$

where $\sharp \in \{\wedge, \vee, \to, \neg\}$ and $f_\sharp$ is the many valued truth table for $\sharp$

$$\mu_s^*(\Box A) = \ Inf_x[\text{ for all } \alpha, R^*(s, \alpha) \leq x \text{ implies } \mu_s^*(A) \leq x]$$

The above definition should be compared with the case of dovetailing.

The above examples indicate how we can define the notion of one logic **L** being *fuzzled* (made fuzzy) by another logic **Z**. In the modal-many valued logic examples above, modal logic was made fuzzy by Łukasiewicz many valued logic. The possible world relation of the modal semantics was replaced by a function to truth filters of the many valued algebra and the assignments h of **L** was replaced by an algebraic many valued assignment.

15.5 Label dependent connectives

We have alredy introduced the possiblity that certain constants be label dependent. For example when we skolemise the data item

$$\alpha : \exists x \Diamond A(x)$$

we introduce a constant

$$c^\alpha \text{ and get}$$

$$\alpha : \Diamond A(c^\alpha)$$

We can thus create a $\beta, \alpha < \beta$ and get:

$$\beta : A(c^\alpha)$$

The above notation ensures that the data item

$$\alpha \Diamond \exists x A(x)$$

is distinguished from the previous one. We can introduce a $\beta, \alpha < \beta$ and get

$$\beta : \exists x A(x)$$

then skolemise and end up with

$$\beta : A(c^\beta)$$

This section studies label dependent connectives. If the labels are representing time, then we are dealing with time dependent connectives. It is easier to give examples of the latter.

Example 15.5.1 *Consider the temporal connective $U(A, B)$, reading "B until A". Its classical truth table is:*

$U(A, B)$ holds at t iff for some $s > t, A$ holds at s, and for all U between t and s we have that BR holds at U.

We can write the equivalence

$t : U(A, B)$ is equivalent to $t : F(A \wedge H^t B)$

HB reads "B was always true in the past".

$H^t B$ reads "B was always true in the past between now and t".

H^t is a time dependent connective.

Example 15.5.2 *The following are examples for making $\Box^t$ label dependent.*

1. *The basic meaning for $\Box A$ is "always A". It is not label dependent. The corresponding* LDS *rule is:*

$$\frac{s : \Box A, s < r}{r : A}$$

2. *The basic meaning for $\Box^t A$ is "always A up to t".*
   The corresponding rule is:

$$\frac{s : \Box^t A, s < r < t}{r : A}$$

3. *The basic meaning of $\Box^t A$ is "always A except at t"*
   The corresponding rule is:

$$\frac{s : \Box^t A, t < s, s \neq t}{s : A}$$

4. *The basic meaning of $s : \Diamond^t A$ is "A is true at t and $s < t$". The corresponding rules are:*

$$\frac{s : \Diamond^t A}{s \leqq t}$$

and

$$\frac{s : \Diamond^t A}{t : A}$$

another example is that of *joining* two logics to form a two-dimensional product.

Example 15.5.3 (Joining modal logics) *Let* $\mathbf{L}$ *be the extension of the modal logic* $\mathbf{K}$. *This logic is complete for the class of all Kripke structures* (S, R, a, h).

Let $\mathbf{L}_1, \mathbf{L}_2$ *be two copies of* $\mathbf{L}$ *with modalities* $\Box_1$ *and* $\Box_2$. *Let* $\mathbf{L}_{1,2}$ *be the langauge with both modalities and consider the following* $\mathcal{K}_{1,2}$ *semantics for* $\mathbf{L}_{1,2}$.

The structures have the form $(S = S_1 \times S_2, R_1, R_2, (a_1, a_2), h)$ *where* (S_i, R_i, a_i) *are* $\mathbf{L}_i$ *Kripke structures and* h *is an assignment into* S, *i.e.* $h(q) \subseteq S$, *for each atomic* q.

Satisfaction under h *is dfined as follows:*

$(t, s) \vDash_h q$ *if* $(t, s) \in h(q)$.

$(t, s) \vDash_h \Box_1 A$ *iff for all* $t', tR_1t' \Rightarrow (t', s) \vDash_h A$

$(t, s) \vDash_h \Box_2 A$ *iff for all* $s', sR_ws' \Rightarrow (t, s') \vDash_h A$.

We write $\vDash A$ *if* $(a_1, a_2) \vDash_h A$ *for all Kripke structures.*

To be continued.

Chapter 16

Abduction in Labelled Deductive Systems

16.1 Intuitive theory of labelled abduction

This section will introduce our intuitive theory of abduction within the framework of *Labelled Deductive Systems*, and give some simple examples.

The new ideas we shall put forward are:

- abduction depends on proof procedures
- abductive principles can be part of the data. In other words, a declarative item of data can be either a formula or a principle of abduction

The more precise machinery for these concepts will be developed in later sections. This section will discuss the intuitive ideas.

The basic situation we are dealing with can be presented as

$$\begin{array}{ll} \Delta & \vdash ?!Q \\ \text{data} & ?\text{query or } !\text{ input} \end{array}$$

It is a relationship between a database and a formula. The relationship is either declarative (ie $?Q$, Q a query) or imperative ($!Q$, Q is an input or a demand to perform abduction or a demand for explanation etc). In the imperative case there is an interaction between Δ and Q and a new database Δ' emerges.

We have argued in previous chapters that the most general and useful database is the one where the data is structured and the proof procedures use the structure.

In this set up, the abduction rules are extra moves that help answer the query or help change the database as a result of the query or input.

Thus to do abduction we need more precise proof procedures or update procedures for structured databases and then on top of that we can define the extra abductive rules.

The exact proof procedures can be conviently formalised in the framework of *LDS*. Meanwhile let us illustrate our ideas through a series of examples. The reader should compare with section 1.7, where a practical system, which includes abduction, is presented.

Example 16.1.1 The database below is a Horn clause database. It is labelled in the sense that each clause is named. The query is D. The query does not follow from the database as it is. We are going to use it to illustrate principles of abduction.

	Data	**Query**
a_1	$I \wedge T \to D$	$?\ D$
a_2	$L \to I$	
a_3	$L \wedge S \to T$	
a_4	$O \wedge P \to T$	
a_5	L	

The database literals have no meaning. Let us give them a meaning. In the Stanford University English Department, there are two main ways of getting a PhD title. One can either put forward a thesis, stay in the department for 4-5 years acquiring and displaying an immense breadth of knowledge and pass an interview, or one can write a very good publication and get a job offer from another university in the top ten in the country. The database then becomes:

	Data
a_1	Interview $\wedge$ Thesis $\to$ Degree
a_2	Lecture $\to$ Interview
a_3	Lecture $\wedge$ Scholarly Survey $\to$ Thesis
a_4	(Job) Offer $\wedge$ Publications $\to$ Thesis
a_5	Lecture

Query

? Degree

Another interpretation for the same database is a component interpretation. To do the laundry (D) one needs a washing machine (T) and washing powder (I). For washing powder one can use dishwashing soap (L). For a washing machine one may use a dishwasher (S) and dishwashing soap or one may handwash (P) but then one at least needs a spinner (O).

We thus get in this case

Data

a_1 Washing Powder $\wedge$ Washing Machine $\to$ Laundry

a_2 Dishwashing Soap $\to$ Washing Powder

a_3 Diswashing Soap $\wedge$ Dishwasher $\to$ Washing Machine

a_4 Spinner $\wedge$ Handwash $\to$ Washing Machine

a_5 Diswashing Soap.

Query

? Laundry

We now list several possible abductive principles for the query $?D$. The principles depend on the computation, so let us suppose that we compute the query prolog like, where the pointer always starts at the top clause (assume $a_1 > a_2 > a_3 > a_4 > a_5$.)

We note that in logic programming [Eshghi and Kowalski, 1989] abduction for Horn clause programs is done via a system of the form (Δ, I, A), where Δ is the program, I is a set of integrity constraints and A is a set of literals which are *abducible.* Whenever an abducible literal is encountered in the computation (eg $?D$) it is immediately added to the database provided it does not violate the integrity constraints.

Let us now examine our options:

Example 16.1.2 [Possible principles of Abduction]

1. The first option is to abduce on anything as soon as needed. This corresponds in our case, to no integrity constraints and every literal is abducible.
   In this case we add D to the database, ie the Abduction principle yields D. In the component example such abduction makes no sense. I want to know which parts are missing so that we can get them and wash our clothes.

2. The second option is to abduce on literals which are not heads of clauses. In this case, we add S. This is because S is the first literal encountered in the top down order of execution. Note that we do not use here a set of abducibles. The structure of the database determines what we add.

3. If our underlying logic is not classical logic but some other resource logic, we will not succeed by adding S to the database because that would require the "use" of L twice. Once to make I succeed in clause a_2 and once to make T succeed in clause a_3. In the component example we need more dishwashing soap if we use a dishwasher, and we have only one lot of it (ie a_5).

   Note that the database is structured and thus we can add
   $a_6 \quad L$
   and $\{a_1, \dots, a_5\}$ is *not* the same database as $\{a_1, \dots, a_6\}$.

   Anyway, if the underlying logic is a resource logic, the result of our abduction will be $O \wedge P$, unless we are prepared to add another copy of L.

4. If we require the weakest logical assumption (in classical logic) which makes the goal succeed then we must add $S \vee (O \wedge P)$. This abduction principle is independent of the computation.

5. In co-operative answering, the abduction principle takes the top level clause. In this case the answer is T. To the query "$?D$" we answer "yes if T". Think of the thesis example. If an ordinary student wants to know what is missing to get a PhD, the obvious answer is "a thesis" and not "a paper and a job offer from Harvard".

6. The power of our labelling mechanism can be easily illustrated by a more refined use of the labels. If atoms are labelled, for example, by cost (laundry example) the abduction principle can aim for minimal cost. One can also "cost" the computation itself and aim to abduce on formulas giving maximal provability with a least number of modus ponens instances.

Example 16.1.3 To show that the abduction depends on the computation let us change the computation to forward chaining or Gentzen like rules. From

$$\text{Data} \vdash ?D$$

we get

$$I \wedge T \to D, I, S \to T, O \wedge P \to T \qquad \vdash ?D$$

which reduces to

$$T \to D, S \to T, O \wedge P \to T, \qquad \vdash ?D$$

which reduces to the following by chaining:

$$S \to D, O \wedge P \to D, \qquad \vdash ?D$$

As we see, not many abduction possibilities are left!

So far we have discussed the possibilities of abduction principles being added to proof rules. We now come to our second new idea, namely:

- Abduction principles are data!

Example 16.1.4 Consider the following database and query:

a_1 A

a_2 $A \to (B \to S)$

a_3 B

....

a_4 X, abduce on structrure to find X.

a_5 $B \to D$
The goal is $?S \otimes D$.

By writing $S \otimes D$ for the goal we are saying we want to partition the database, which is a list of assumptions, into two parts, the first part must prove S and the second part must prove D. This is done in resource logics, where one pays attention to the question of which part of the database proves what.

Such considerations arise in many areas for example in what is known as *parsing as logic.*

Consider the text:
Mary hit John. He cried.

The way this can be analysed is by assuming that each word is assigned a wff of some resource logic (actually concatenation logic, see [Gabbay and Kempson, 1991]) with a label. This assignement is done at the lexical level. Thus a noun n is assigned $n' : NP$. An intranstive verb v_1 is assigned $v_1' : NP \to S$. A transitive verb v_2 is assigned $v_2' : NP \to (NP \to S)$. The pronoun 'he' is assigned an abduction principle. Our problem becomes:

Data

1. Mary$'$: NP
2. hit$'$: $NP \to (NP \to S)$
3. John$'$: NP
4. he: Abduce on structure. Take the first literal up the list.
5. cried$'$: $NP \to S$.

Query
Prove $?S$ or $S \otimes S$ or $?S \otimes S \otimes S \ldots$ etc, in order to show we have a text of sentences.

We are thus saying that Anaphora resolution makes use of structural abduction.

The reader should note that anaphora resolution is a complex area and we are not making any linguistic claims here beyond the intuitive example that abduction principles can be treated as data. We do admit however that logical principles underlying database management do seem to be operative in natural language understanding and we are working full steam ahead in making our case.

Coming back to our view of abduction as data, we are really saying:

- A database can either display data items or give us pointers to where to get them.

Thus a labelled database can appear as below:

$$\boxed{\begin{array}{ll} n_1\text{:} & \text{data } m \\ \vdots & \vdots \\ n_k\text{:} & \text{get datum from} \ldots \\ \vdots & \vdots \end{array}}$$

Abductive Labelled Database

I would like to give next a combined example of planning and parsing, based on ideas of [Gabbay and Kempson, 1991].

Example 16.1.5 (Planning) *Consider the situation described by the diagram below.*

There are three languages involved

1. *The database language containing the predicates* $On(x, y)$ *and* $Free(x)$
2. *The imperative (Input) command language with the predicates* $Move(x, y)$.

3. *The mixed metalanguage with the connectives "$\wedge$" for "and" and "$\Rightarrow$" for "precondition and action imply postcondition".*

a			t_1: $On(a, b)$	
b	c		t_1: $On(b, tab)$	$\leftarrow Move(a, c)$
			t_1: $On(c, tab)$	
	a		t_2: $On(a, c)$	$\leftarrow Move(a, tab)$
b	c			
b	c	a	t_3: $On\ (a, tab)$.	

$$On(x, y) \wedge\ Free(x) \wedge\ Free(z) \wedge Move(x, z) \Rightarrow\ On(x, z) \wedge Free(y) \wedge\ Free(x)$$

The diagram describes the initial layout of the blocks ready to respond to command. t_1 labels all data true at the initial situation and t_2 and t_3 the additional data after each of the actions. We have

$$t_1 < t_2 < t_3.$$

If we query the system with

$$?\ On(a, x)$$

we get three answers, with different labels, indicating where the answer was obtained in the database, namely:

$$\begin{aligned} &\vdash t_1 : \ On(a, b) \\ &\vdash t_2 : \ On(a, c) \\ &\vdash t_3 : \ On(a, tab) \end{aligned}$$

The reply to the user is determined by the system as the answer proved with the stronger label, namely:

$$On(a, tab)$$

Call the deductive system governing the planning consideration LDS_1[1]*. This system involves proving where the blocks are after which action. This system accepts commands in logical form $Move(x, y)$. It does not accept commands in English. If the command comes in English, which we can represent as* move x onto y, *it needs to be parsed into the* LDS_1 *language. This is done in a parsing logic* LDS_0*. The following diagram explains the scheme, see Figure 16.1:*

The following diagram describes the database-query problem of LDS_0*:*

move′:	$NP \rightarrow (NP \rightarrow S)$	$?S + S$
a′:	NP	
c′:	NP	
move′:	$NP \rightarrow (NP \rightarrow S)$	
it:	*use abduction. First use structural abduction to get the first NP higher in the list, then use inferential abduction to try and get maximal inferential effects in* LDS_1.	
tab′:	NP	

Notice that the abduction principle in LDS_0 *also uses inferential effect in* LDS_1*. Intuitively we are trying to abduce on who "it" refers to. If we choose "it" to be a block which is already on the table, it makes no sense to move it onto the table. Thus the command when applied to the database will produce no change. The abduction principle gives preference to abduced formulas which give some effect.*

[1]We remark in passing that this approach offers a *conceptual* (not computational) solution to the frame problem. Conceptually, given an initial labelled database and a sequence of actions to be performed, we model the sequence by another labelled database; the database obtained by adding the results of the actions to the initial database. We label the additions appropriately. This idea will be pursued elsewhere. There are several such "non-monotonic" solutions in the literature. This is probably the most general

$$\begin{tikzcd}[row sep=small]
\textbf{English Input:} \\
\textit{move a onto c. move it onto table} \arrow[d] \\
\boxed{\textit{Parsing Logic } \mathrm{LDS}_0} \arrow[d] \\
\textit{move } (a,c);\ \textit{move } (a, tab) \\
\boxed{\textit{Planning Logic } \mathrm{LDS}_1}
\end{tikzcd}$$

Figure 16.1:

From the logical point of view we are using the following principle, (see also Example 16.1.9):

- Abduction principles can serve as items of data in one database Δ_0, being a pointer to another database Δ_1 where some computation is carried out jointly involving both databases and the result is the abduced data item in Δ_0.

Example 16.1.6 (Logic Programming Abduction) *The abductive system in logic programming can be schematically put into our form by making use of the way the Prolog pointer scans the database. An abductive system has the form (Δ, I, A), where $\Delta = (C_1, \ldots, C_n)$ is a sequence of clauses and literals, where I are the integrity constraints and where A is the set of abducible atoms. This system can be translated into the following Horn clause database:*

(0) C_0 = *Abduce on the goal by checking whether the goal is in A and whether when added it satisfies the integrity constraints.*

(1) C_1

$\vdots$ $\vdots$

(n) C_n

We are now ready for our next conceptual step. If an abductive principle is to be considered as a declarative item of data, say $\mathcal{Q}_{Abduce}$, then what would be the meaning of

$$\Delta \ ?! \ \mathcal{Q}_{Abduce}$$

For the imperative interaction $!\mathcal{Q}_{Abduce}$, the meaning is obvious, we simply apply the abductive principle to the database and get a new database. However the query meaning of the abductive principle is a bit more difficult to explain in general, and we may need to provide an explanation for each specific case. For example, if the abduction prinicple abduces a formula B, then $\Delta ? \mathcal{Q}_{Abduce}$ would mean $\Delta ? B$. This seems all right at first sight, however, the problem here can be that the abduction process by its nature tries to find B's which are not available in the database and so the answer to the query $\Delta ? \mathcal{Q}_{Abduce}$ will always be no. This is clearly unacceptable. We must seek another meaning for the query.

Let us for the time being, accept only the imperative reading of $\Delta ! \mathcal{Q}_{Abduce}$. We can immediately allow ourselves to write databases with clauses containing $\mathcal{Q}_{Abduce}$ in them. Let us see through a few examples what this means.

Example 16.1.7 Let $\Delta = \{\mathcal{Q}_{Abduce} \wedge B \to D\}$. Think of the above as a database, and assume the computation procedure to be Prolog like. We consider the following query

$$\Delta ? D$$

which reduces to

$$\Delta?(\mathcal{Q}_{Abduce}, B)$$

which reduces to

$$\Delta, B?B$$

which succeeds.

Here we assumed that $\mathcal{Q}_{Abduce}$ yields B. Our database is similar, in this case, to the Prolog database:

$$\mathit{Assert}\ (B) \wedge B \rightarrow D$$

Indeed, asserting is a form of unconditional abduction.

Example 16.1.8 [Abduction and Negation by Failure] From our point of view, negation by failure is abduction. This point has also been made in [Eshghi and Kowalski, 1989]. However, we want to make our position crystal clear to avoid confusion. We believe as stated in Chapter 1, that abduction is a principle of reasoning of equal standing to deduction and that every logical system is comprised of both proof rules and various mechanisms including abductive rules. This view has developed through our interaction with the logics of common sense reasoning and our work in natural language understanding [Gabbay and Kempson, 1991]. Negation by failure is not central to our scheme, though it is an interesting example from our point of view.

We begin with a precisely specified proof system. The query $\Delta?Q$ can be algorithmically checked. If the algorithm succeeds, the answer is yes. The algorithm may loop or it may fail. We may be able to prove that for the particular $\Delta?Q$, the algorithm must fail (eg in a case where none of the rules can even be applied). In this case we can say $\Delta?Q$ *finitely fails* (relative to the algorithm). Thus the notion of *finite failure* can be defined for any proof theoretic system.

Given a system, we can consider the following abduction principles which we call $\mathit{Fail}(Q, B)$:

If $\Delta?Q$ finitely fails then abduce (or assert) B.

To make our example specific, let us choose a language and computation procedures. By an atomic literal let us understand either an atom $\{p, q, r, \ldots\}$ or an abduction principle $\mathit{Fail}(a, b)$, where a, b are atoms. By clauses let us understand Horn clauses of literals. Goals are conjunctions of literals. Thus we can write the following clauses:

1. $q \wedge\ \mathit{Fail}(a, b) \wedge c \rightarrow p$
2. $\mathit{Fail}(q, r)$
3. $a \rightarrow\ \mathit{Fail}(b, b)$.

To explain the computational meaning, we will translate into Prolog. Ordinary Prolog is not expressive enough for our purpose, so we use N-Prolog [Olivetti and Terracini, 1991, Gabbay and Reyle, 1984, Gabbay, 1985] with negation by failure, mainly because it allows hypothetical reasoning, ie embedded implications.

We translate:

$$\mathit{Fail}(a, b) \mapsto (\neg a \rightarrow b).$$

After translation, the database becomes:

1. $q \wedge (\neg a \rightarrow b) \wedge c \rightarrow p$
2. $\neg q \rightarrow r$
3. $a \wedge \neg b \rightarrow b$

which is meaningful computationally in goal directed N-Prolog, see also chapter 4, section 3.

A Horn clause with negation by failure of the form:

$$a \wedge \neg b \rightarrow c$$

can be translated back into our abductive language as

$$a \to \mathit{Fail}(b, c)$$

A Prolog goal of the form $\neg a$ an be translated as $\mathit{Fail}(a, \varnothing)$, $\varnothing$ is truth.

N-Prolog is not as expressive as our abductive language. In our abductive language we also have the imperative meaning of

$$\Delta ?\ \mathit{Fail}(a, b)$$

which means *apply* the abduction to Δ.

This would correspond to

$$\mathit{Assert}\ (\neg a \to b)$$

in N-Prolog. N-Prolog does not allow for that. The syntax is defined in such a way that we do not get goals of the form $\Delta?(\neg a \to b)$. The N-Prolog computation rule would require in this case to add $\neg a$ to Δ, which is not meaningful.

The connection between abduction and negation by failure was observed in [Eshghi and Kowalski, 1989]. Since their abductive systems have the restricted form (Δ, I, A) as described in Example 16.1.6, they need to rewrite the Horn clause program into a more convenient form, translating the Prolog $\neg a$ as a^* and adding the integrity constraint:

$$a \wedge a^* \to .$$

Example 16.1.9 (A Conversation between two Intelligent Databases) *We have the logical means to allow for two* LDS *databases to negotiate and reach an understanding. Imagine two databases S and H exchanging formulas continuously. At time n, the databases have evolved through the sequences*

$$\Delta_1^S, \ldots, \Delta_n^S$$

and

$$\Delta_1^H, \ldots, \Delta_n^H.$$

At time n, database S sends a logical input I_n^S to database H and conversely, database H sends an input I_n^H to database S. The two databases use abduction to update themselves. Thus

$$\Delta_{n+1}^H = \text{Abduce}(\Delta_n^H, I_n^S)$$

and

$$\Delta_{n+1}^S = \text{Abduce}(\Delta_n^S, I_n^H).$$

To continue and communicate we need action principles for sending the next input. This is also part of our abduction scheme as hinted at in Example 16.1.5

16.2 Labelled abductive mechanisms

The basic *deductive* apparatus of *LDS* is a precise system of rules allowing one to show (or fail to show) whether $\Delta \vdash \Gamma$, for Δ a data structure and Γ a goal structure. Most useful among the goal structures is the unit structure of the form $t : G$. Thus for the purpose of explaining what abduction is going to be in our framework, we assume that the notion of

$$\Delta \vdash t : G$$

is precisely algorithmically defined, thus yielding a particular *LDS* system **L**. We now schematically explain how abduction principles fit into this framework. Consider a database Δ, containing $\alpha : X$ inside it, which we write as $\Delta[\alpha : X]$ schematically with $t : X$. Somewhere in the structure Δ, α is a label variable and X is a propositional variable standing for a wff. For any particular choice of X, and of α say the choice of $\alpha = \alpha_0$ and $X = A$, $\Delta[\alpha : A]$ is a proper database.

Suppose we want to prove a goal $t : G$. Then, for some (maybe none) wffs A_i and labels α_i we may have

$$\Delta[\alpha_i : A_i] \vdash t : G$$

A principle of abduction

$$Abduce(\alpha : X)$$

is a computation (algorithm) that can choose one or more of the α_i and A_i above.

Of course for different G we get different α_i, A_i.

The importance of the above point of view is:

1. Databases can take abductive principles as part of their data, slotted at the right places.
2. The abductive principle is relative to the computation procedure and the rest of the database. Thus when new data is put in, the abductive principle changes. We get a strong learning component in the database.

An inductive principle is a special case of abductive principle which learns a rule $A \to B$ as opposed to a fact. (atom q). Mathematically there is no difference.

Bibliography

[Alchourrón *et al.*, 1985] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50:510–530, 1985.

[Allwein and Dunn, 1993] G. Allwein and J.M. Dunn. Kripke models for linear logic. *the Journal of Symbolic Logic*, 58:514–545, 1993.

[Amati and Pirri, 199] G. Amati and F. Pirri. Uniform tableaux methods for intuitionistic modal logic I. *Studia Logica*, 199? To appear.

[Anderson and Belnap Jr, 1975] A. R. Anderson and N.D. Belnap Jr. *Entailment: the Logic of Relevance and Necessity*, volume 1. Princeton University Press, Princeton, 1975.

[Anderson and Belnap, 1975] A. R. Anderson and N. D. Belnap. *Entailment.* Princeton University Press, 1975.

[Avron, 1988a] A. Avron. The semantics and proof theory of linear logic. *Theoretical Computer Science*, 57:161–184, 1988.

[Avron, 1988b] A. Avron. The semantics nd proof theory of linear logic. *Theoretical Computer Science*, 57:161–184, 1988.

[Avron, 1991] A. Avron. Simple consequence relations. *Journal of Information and Computation*, 92:105–139, 1991.

[Babb, 1990] E. Babb. An incremental pure logic language with constraints and classical negation. In T. Dodd, R. Owens, and S. Torrance, editors, *Logic Programming: Expanding the Horizon.* Blackwell, 1990.

[Barkley-Rosser, 1960] J. Barkley-Rosser. Axiomatisation of infinite valued logic. *Logique et Analyse*, pages 137–153, 1960.

[Blok and Pigozzi, 1989] W. J. Blok and D. Pigozzi. *Algebraizable Logics.* Memoires of AMS No 396, American Mathematical Society, 1989.

[Brachman *et al.*, 1989] R. J. Brachman, H. J. Levesque, and R. Reiter (eds.). *KR '89: Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann, San Mateo, California, 1989.

[Clark and McCabe, 1984] K. Clark and F. McCabe. *Micro-Prolog.* Prentice Hall, 1984.

[Console *et al.*, 1991] L. Console, D. T. Dupre, and P. Torasso. On the relationship between deduction and abduction. *Journal of Logic and Computation*, 1:661–690, 1991.

[Curry and Feys, 1958] H. Curry and R. Feys. *Combinatory Logic, Volume 1.* North Holland, 1958.

[Curry, 1934] H. Curry. Functionality in combinatory logic. In *Proceedings of the National Academy of Sciences of the USA, Vol 20*, pages 154–180, 1934.

[Curry, 1963] H. Curry. *Foundations of Mathematical Logic.* McGraw Hill, 1963.

[D'Agostino and Gabbay, 1994] Marcello D'Agostino and Dov Gabbay. Labelled Tableaux, *Journal of Automated Reasoning, to appear.*

[D'Agostino and Mondadori, 1994] Marcello D'Agostino and Marco Mondadori. The taming of the cut. To appear in the Journal of Logic and Computation, vol. 4, n.3., 1994.

[D'Agostino, 1992] Marcello D'Agostino. Are tableaux an improvement on truth-tables? Journal of Logic, Language and Information*, 1:235–252, 1992.*

[D'Agostino, 1994] Marcello D'Agostino. On the efficiency of proof search in classical analytic deduction. Forthcoming. (Preliminary version presented at the Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, Marseille, 28–30 April 1993), 1994.*

[Dôsen, 1988] Kosta Dôsen. Sequent systems and groupoid models I. Studia Logica*, 47:353–385, 1988.*

[Dôsen, 1989] Kosta Dôsen. Sequent systems and groupoid models II. Studia Logica*, 48:41–65, 1989.*

[Dunn, 1976] J. M. Dunn. Intuitive semantics for first-degree entailment and coupled trees. Philosophical Studies*, 29:149–168, 1976.*

[Eshghi and Kowalski, 1989] K. Eshghi and R. A. Kowalski. Abduction compared with negation by failure. In Proc. 6th ICLP 89, MIT Press, 1989.

[Fine and Schurz,] K. Fine and G . Schurz. Transfer theorems for stratified multimodal logics, ? To appear.

[Fitting, 1983] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* Reidel, Dordrecht, 1983.

[Sambin, 1993] G. Sambin. The semantics of pretopologies. In K. Dôsen and Peter Schroeder-Heister, editors, *Substructural Logics*, pages 293–307. Oxford University Press, 1993.

[Gabbay and Kempson, 1991] D. M. Gabbay and R. Kempson. Natural language content: a proof theoretic perspective. In *Proceedings of the Eighth Amsterdam Colloquium*, pages 173–196, 1991.

[Gabbay and Kempson, 1991] D. M. Gabbay and R. Kempson. Labelled Abduction adn Relevance, unpublished draft, 1991.

[Gabbay and Ohlbach, 1993a] Dov M. Gabbay and Hans J. Ohlbach. An algebraic fine structure for logical systems. In Dov Gabbay and Franz Guenthner, editors, *What is a Logical System?* Oxford University Press, 1993. To appear.

[Gabbay, 1985] D. M. Gabbay. N-Prolog, Part II, *Journal of Logic Programming*, 5: 251–283, 1985.

[Gabbay and Ohlbach, 1993b] Dov M. Gabbay and Hans J. Ohlbach. From hilbert calclus to possible-world semantics. In Krysia Broda, editor, *Proceedings of ALPUK Logic Programming Conference 1992*, pages 218–252. Springer, 1993. Lecture Notes in Computer Science.

[Gabbay and Reyle, 1984] D. M. Gabbay and U. Reyle. N-Prolog, An extension of Prolog with hypothetical implication. *Journal of Logic Programming*, 4: 319–355, 1984.

[Gabbay and Reyle, to appear] D. M. Gabbay and U. Reyle. Direct deductive computation on discourse represenation structures. *Linguistics and Philosophy*, to appear.

[Gabbay *et al.*, 1994] D. M. Gabbay, L. Giordano, A. Martelli, and N. Olivetti. Conditional logic programming. In *ICLP 94*, 1994. to appear.

[Gabbay, 1969] D. Gabbay. The Craig interpolation theorem for intuitionistic logic I and II. In *Logic Colloquium '69*, 1969.

[Gabbay, 1976] D. Gabbay. *Investigations in Modal and Temporal Logics*, D. Reidel, 1976.

[Gabbay, 1985] D. Gabbay. Theoretical foundations for nonmonotonic reasoning in expert systems. In K. Apt, editor, *Expert Systems, Logics and Models of Concurrent Systems*, pages 439–459. Springer-Verlag, Berlin, 1985.

[Gabbay, 1981] D. M. Gabbay. *Semantical Investigations in Heyting's Intuitionistic Logic.* D. Reidel, 1981.

[Gabbay, 1992a] D. Gabbay. Theory of algorithmic proof. In *Handbook of Logic in Theoretical Computer Science, Volume 1*, pages 307–408. Oxford University Press, 1992.

[Gabbay, 1992b] D. M. Gabbay. Fibred semantics and the combination of logics, August 1992. Lectures given at Logic Colloquium 1992, Veszprém, Hungary.

[Gabbay, 1993a] D. M. Gabbay. Fibred semantics and the weaving of logics, part 1, 1993. manuscript, Imperial College.

[Gabbay, 1993b] Dov Gabbay. Fibred semantics and the weaving of logics, I. Technical Report, University of Stutgart, 1993.

[Gabbay, 1993c] Dov M. Gabbay. Classical versus non-classical logics. In Dov Gabbay, Chris Hogger, and J.H. Robinson, editors, *Handbook of Logic in AI and Logic Programming.* Oxford University Press, 1993. To appear.

[Gabbay, 1994] D. M. Gabbay. LDS and the traditional fallacies, 1994. In preparation.

[Gärdenfors, 1988] P. Gärdenfors. *Knowlege in Flux: Modeling the Dynamics of Epistemic States.* MIT Press, Cambridge, MA, 1988.

[Girard, 1976] J.-Y. Girard. Three valued logic and cut elimination: the actual meaning of Takeuti's conjectures. *Dissertationes mat*, CXXXVI, 1976.

[Girard, 1987] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Haehnle and Schmitt, 1992] Reiner Haehnle and Peter Schmitt. The liberalized δ-rule in free variable semantic tableaux. Manuscript, 1992.

[Hähnle, 1992] Reiner Hähnle. *Tableau-Based Theorem-Proving in Multiple-Valued Logics.* PhD thesis, Department of Computer Science, University of Karlsruhe, 1992. To appear with Oxford University Press.

[Hindley and Seldin, 1986] R. Hindley and J. Seldin. *Intoduction to combinators and λ-calculus.* Cambridge University Press, 1986.

[Kamp, 1968] H. Kamp. *Tense Logic and the theory of linear order.* PhD thesis, University of California, Los Angeles, 1968.

[Kong and Williams, 1991] Q. Kong and M. H. Williams. Incomplete information in a logic database. In T. Dodd, R. Owens, and S. Torrance, editors, *Logic Programming, Expanding the Horizons*, pages 124–146. Ablex, 1991.

[König, 1994] E. König. A hypothetical reasoning algorithm for linguistic analysis. *Journal of Logic and Computation*, 4:1–22, 1994.

[Kowalski, 1979] R. A. Kowalski. *Logic for Problem Solving.* North Holland, Amsterdam, 1979.

[Kracht and Wolter, 1991] M. Kracht and F. Wolter. Properties of independently axiomatizable bimodal logics. *Journal of Symbolic Logic*, 56:1469–1485, 1991.

[Kraus *et al.*, 1990] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990. A preliminary version, with authors Kraus and Lehmann only, was presented under the title "Nonmonotonic logics: models and proofs" to *JELIA: European Workshop on Logical Methods in Artificial Intelligence*, Roscoff France, June 1988. Another preliminary version, with all three authors, appeared under the title "Preferential models and cumulative logics", Technical Report TR 88-15, Department of Computer Science, Hebrew University of Jerusalem, November 1988.

[Lambek, 1989] J. Lambek. Multicategories revisited. *Contempory Mathematics*, 92:217–239, 1989.

[Lauritzen and Spiegelhalter, 8] S. L. Lauritzen and D. J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems (with discussion). *The Journal of the Royal Statistical Society B*, 50:157–224, 8?

[Lehmann and Magidor, 1992] D. Lehmann and M. Magidor. What does a conditional knowledge base entail? To appear in *Artificial Intelligence*, 1992. This paper gathers together and extends the material of a paper by Lehmann alone, with the same title, in [Brachman *et al.*, 1989], 212-222, and Technical Report 88-16 by Lehmann and Magidor, Department of Computer Science, Hebrew University of Jerusalem, November 1988.

[Makinson, 1989] D. Makinson. General theory of cumulative inference. In M. Reinfrank et al., editors, *Non-Monotonic Reasoning*, volume 346 of *Lecture Notes on Artificial Intelligence*, pages 1–18. Springer-Verlag, Berlin, 1989.

[Makinson, 1993] D. Makinson. General patterns in nonmonotonic reasoning. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in AI and Logic Programming, Vol 2.* Oxford University Press, 1993.

[Maslov, 1969] S. Yu. Maslov. Invertible sequent variant of constructive predicate calculus. *Seminar in Mathematics. V.A. Steklov Mathematical Institute, Leningrad*, 4:36–42, 1969.

[McRobbie and Belnap, 1979] Michael A. McRobbie and Nuel D. Belnap. Relevant analytic tableaux. *Studia Logica*, XXXVIII:187–200, 1979.

[Ng and Subrahmanian, 1994] R. Ng and V. Subrahmanian. Dempster-Shafer logic programs and stable semantics. In J. N. Crossley and J. B. Remmel, editors, *Logical Methods*, pages 654–704. Birkhaüser, 1994.

[Nute, 1986] D. Nute. LDR – A Logic for Default Reasoning. Research Report 01-0013, ACMC, 1986.

[Olivetti,] N. Olivetti. Circumscription.

[Olivetti and Gabbay, 1993] N. Olivetti and D. M. Gabbay. To restart or not to restart. Technical report, Imperial College, 1993.

[Olivetti and Terracini, 1991] N. Olivetti and L. Terracini. N-Prolog, Part II, *Journal of Logic Programming*, 5:251–283, 1985.

[Ono, 1993] H. Ono. Semantics for substructural logics. In K. Dôsen and Peter Schroeder-Heister, editors, *Substructural Logics*, pages 259–291. Oxford University Press, 1993.

[Perlis, 1985] D. Perlis. Languages with self reference I: foundations. *Artificial Intelligence*, 25:301–22, 1985.

[Perlis, 1988] D. Perlis. Languages with self reference II: knowledge, belief and modality. *Artificial Intelligence*, 34:179–212, 1988.

[R. and Routley, 1972] R. and V. Routley. The semantics of first degree entailment. *Noûs*, VI:335–359, 1972.

[Rose, 1956a] A. Rose. Formalisation du calcul propositional implicatif a m valuers de lukasiewica. *Comptes rendus hebdomadaires des seances de'la'Academie des sScineces*, 243:1263–1264, 1956.

[Rose, 1956b] A. Rose. Formalisation du calcul propositional implicatif a x_0 valuers de lukasiewicz. *Comptes rendus hebdomadaires des seances de'la'Academie des Scineces*, 243:1182–1185, 1956.

[Routley and Meyer, 1973] R. Routley and R.K. Meyer. The semantics of entailment, I. In H. Leblanc, editor, *Truth, Syntax and Semantics.* North Holland, 1973.

[Scott, 1974] D. Scott. Completeness and axiomatizability in many valued logics. In *Proceedings of the Tarski Symposium*, pages 411–436, Providence, Rhode Island, 1974. American Mathematical Society.

[Tarski, 1956] A. Tarski. On the concept of logical consequence, polish 1936. In Translation in:*Logic, Semantics, Metamathematics.* Oxford University Press, 1956.

[Thistlewaite *et al.*, 1988] P. B. Thistlewaite, M. A. McRobbie, and B. K. Meyer. *Automated Theorem Proving in Non Classical Logics.* Pitman, 1988.

[Torsun and Manning, 1990] I. Torsun and K. Manning. Execution and application of temporal modal logic. Technical report, University of Bradford, 1990.

[Urquhart, 1972] Alasdair Urquhart. Semantics for relevant logic. *The Journal of Symbolic Logic*, 37:159–170, 1972.

[Valerius, 1989] R. Valerius. *The logic of frame- and stop-rules in Lorenzen Games.* PhD thesis, University of Stuttgart, 1989.

[van Benthem, 1992] J. van Benthem. The logic of programming. *Fundamenta Informatica*, 1992.

[Vermeir and Laenens, 1990] D. Vermeir and E. Laenens. An overview of ordered logic. In *Third Logical Biennial*, Varga, Bulgaria, 1990.

[Wallen, 1990] L.A. Wallen. *Automated Deduction in Non-Classical Logics.* The MIT Press, Cambridge, Mass., 1990.

[Wansing, 1990] H. Wansing. Formulas as types for a hierarchy of sublogics of intuitionistic propositional logic. Technical Report 9/90, Free University of Berlin, 1990.

[Wójcicki, 1989] R. Wójcicki. *An Axiomatic treatment of nonmonotonic arguments.* 1989.

[Wójcicki, to appear] R. Wójcicki. An axiomatic treatment of non monotonic arguments. *Studia Logica*, to appear.

MPI-I-94-216	P. Barth	Linear 0-1 Inequalities and Extended Clauses
MPI-I-94-209	D. A. Basin, T. Walsh	Termination Orderings for Rippling
MPI-I-94-208	M. Jäger	A probabilistic extension of terminological logics
MPI-I-94-207	A. Bockmayr	Cutting planes in constraint logic programming
MPI-I-94-201	M. Hanus	The Integration of Functions into Logic Programming: A Survey
MPI-I-93-267	L. Bachmair, H. Ganzinger	Associative-Commutative Superposition
MPI-I-93-265	W. Charatonik, L. Pacholski	Negativ set constraints: an easy proof of decidability
MPI-I-93-264	Y. Dimopoulos, A. Torres	Graph theoretical structures in logic programs and default theories
MPI-I-93-260	D. Cvetković	The logic of preference and decision supporting systems
MPI-I-93-257	J. Stuber	Computing Stable Models by Program Transformation
MPI-I-93-256	P. Johann, R. Socher	Solving simplifications ordering constraints
MPI-I-93-250	L. Bachmair, H. Ganzinger	Ordered Chaining for Total Orderings
MPI-I-93-249	L. Bachmair, H. Ganzinger	Rewrite Techniques for Transitive Relations
MPI-I-93-243	S. Antoy, R. Echahed, M. Hanus	A needed narrowing strategy
MPI-I-93-237	R. Socher-Ambrosius	A Refined Version of General E-Unification
MPI-I-93-236	L. Bachmair, H. Ganzinger, C. Lynch, W. Snyder	Basic Paramodulation
MPI-I-93-235	D. Basin, S. Matthews	A Conservative Extension of First-order Logic and its Application to Theorem Proving
MPI-I-93-234	A. Bockmayr, F. J. Radermacher	Künstliche Intelligenz und Operations Research
MPI-I-93-233	A. Bockmayr, S. Krischer, A. Werner	Narrowing Strategies for Arbitrary Canonical Rewrite Systems
MPI-I-93-231	D. Basin, A. Bundy, I. Kraan, S. Matthews	A Framework for Program Development Based on Schematic Proof