Resolution is a Decision Procedure for Many Propositional Modal Logics

Renate A. Schmidt

MPI–I–97–2–002 January 1997

Author's Address

Max-Planck-Institut für Informatik, Im Stadtwald
66123 Saarbrücken, Germany
Email: schmidt@mpi-sb.mpg.de
URL: http://www.mpi-sb.mpg.de/~schmidt/

Publication Notes

This paper is the long version of 'Resolution is a Decision Procedure for Many Propositional Modal Logics: Extended Abstract', which is to appear in Kracht, M., de Rijke, M., Wansing, H. and Zakharyaschev, M. (eds), *Advances in Modal Logic '96*, CSLI Publications (1997).

Acknowledgements

I thank Hans Jürgen Ohlbach, Ullrich Hustadt, Andreas Nonnengart and Miroslava Tzakova who read early versions of this paper for their comments. Special thanks go to Ullrich for his permission to include Figures 6 and 7. This work is supported by the TRALOS-Project funded by the DFG.

Abstract

The paper shows satisfiability in many propositional modal systems can be decided by ordinary resolution procedures. This follows from a general result that resolution and condensing is a decision procedure for the satisfiability problem of formulae in so-called *path logics*. Path logics arise from propositional and normal uni- and multi-modal logics by the *optimised functional translation* method. The decision result provides an alternative decision proof for the relevant modal logics (including K, KD, KT and KB, their combinations as well as their multi-modal versions), and related systems in artificial intelligence. This alone is not interesting. A more far-reaching consequence of the result has practical value, namely, any standard first-order theorem prover that is based on resolution can serve as a reasonable and efficient inference tool for modal reasoning.

Keywords

Resolution decision procedures, propositional modal logics, translation methods, optimised functional translation.

Contents

1 Introduction

Theoremhood in the normal modal logic K and its extensions KD, KT, KTB, $S4$ and $S5$ (to mention just a few) is decidable. The standard filtration proof given in the popular textbooks of Chellas (1980) and Hughes and Cresswell (1984) on modal logic renders a decision procedure that is not practical. The proof has two parts. First we must make sure that our logic $K\Sigma$ is finitely axiomatisable, which is easy. And second, we must exhibit that $K\Sigma$ has the finite model property, which is usually more difficult. A logic has the finite model property if for every non-theorem a finite counter-model exists. The standard approach to exhibiting the finite model property uses filtrations for showing that $K\Sigma$ determined by a class of models is also determined by its subclass of finite models. Given an arbitrary modal formula φ, a systematic enumeration of theorems starting from the finite set of axioms using a finite number of rules (modus ponens and necessitation) will eventually produce φ, provided φ is a theorem in $K\Sigma$. If φ is not a theorem in $K\Sigma$, a complete enumeration of the class of finite models characterising the logic will eventually produce a model in which φ is false.

More practical are the semantic tableaux methods found in Kripke (1963), Hintikka (1957–1963) (for the references see Bull and Segerberg 1984), Fitting (1983) or Rautenberg (1983). These methods are very similar, but suffer from the non-determinism due to the implicit or explicit presence of accessibility relations and in many cases deduction can become quite unmanageable. Different from tableaux systems for first-order logic, modal tableaux systems are special purpose systems with the characteristic properties of accessibility built in. Fine (1975) uses a reduction to normal form from which finite models can be constructed. Other practical methods use resolution calculi. For example, Mints (1986, 1989, 1990) proposes a resolution calculus adapted and extended for modal propositional formulae.

We use a resolution calculus too. But in contrast to Mints, we use the classical resolution calculus for predicate logic without changing it. Our method is indirect in that we manipulate first-order translations of modal formulae. Translating modal formulae into predicate logic eliminates the modal connectives $\Box$ and $\Diamond$ and introduces new predicate and function symbols and variables. The standard *relational translation* maps modal formulae into a fragment of first-order logic with unary and binary predicates, function symbols resulting from Skolemisation and variables. Ordinary resolution (without any refinement strategies) is not a decision procedure for the relational translation. The problem is that very often infinitely many non-redundant resolvents are produced. We use a different translation, namely an *optimised* version of the *functional translation* for which infinite computation does not occur, as we will show.

The *functional translation* method was proposed independently by Ohlbach (1988a, 1988b), Fariñas Del Cerro and Herzig (1988), Herzig (1989) and Auffray and Enjalbert (1992) for modal predicate logics. Zamov (1989) claims to have a decision procedure for the logic *S4* using lock resolution. The *optimised functional translation* method applies only to propositional modal logics and is due to Ohlbach and Herzig (and it is also implicit in the paper by Zamov). Ohlbach and Schmidt (1995) show the optimisation can not only be applied to non-axioms, but also to axioms. An argument put forward by many authors in favour of the functional translation method is 'efficiency of automated deduction', without making precise what is meant by efficiency, providing neither a decision proof nor performance comparisons with other methods. The main contribution of this paper is a decision proof for many non-transitive modal logics. To emphasise the practical value the paper includes some benchmark results that compare the performance of resolution based theorem provers with that of systems based on other decision methods.

Important for decidability is that, the optimised functional translation eliminates in the clausal forms all Skolem functions other than Skolem constants. The optimised functional translation embeds modal formulae in logics called *path logics*. The *basic path logic* is a fragment of monadic first-order logic with function symbols. It includes only nullary functions and one designated binary function, namely juxtaposition. Terms are strings of variables and constants, and they encode accessibility as paths from the initial world. Terms, also called *paths*, satisfy the restriction that different occurrences of one variable in any clause have the same prefix. This property, called *prefix stability*, together with a *term depth bound assumption* are also essential for the decidability proof.

The paper proves a very general result. Namely, *resolution together with condensing provides a complete method for deciding whether or not a formula is a theorem in path logics for which*

(i) *a term depth bound exists for the input clauses as well as all resolvents*

(ii) *and unification is decidable.*

It follows that for any propositional and normal modal logics for which the term depth bound assumption holds ordinary resolution, or for modal logics with a non-empty theory, ordinary theory resolution is a decision procedure, provided theory unification is also decidable. The logics K, KD, KT and KB, their combinations as well as their multi-modal versions satisfy the assumption, so that for these logics resolution with condensing, or theory resolution with condensing, is a complete decision procedure for testing theoremhood or satisfiability.

The paper is organised as follows. We begin with two preliminary sections that provide the background for modal logic, the relational and functional translation methods and resolution. Section 2 defines the relational translation of modal logics, resolution decision procedures and condensing. It illustrates by way of modal examples what happens when standard resolution does not terminate for the relational translation because more and more non-redundant clauses are generated. Section 3 gives some background on the optimised functional translation method that give rise to path logics. The main parts of the paper are Sections 4 and 5. Section 4 defines basic path logic and its extensions associated with different normal propositional modal logics. Section 5 presents the decision proof. It studies the class of all variable indecomposable and condensed clauses of basic path logic built from a finite set of predicate and constant symbols. We prove the cardinality of this class has a finite upper bound by showing (in Theorem 5.20) that the cardinality of a corresponding class of terms has a finite upper bound. The section has several subsections. The first subsection gives an outline of the proof. In Section 6 we formally state the decidability results for modal logics that follow from the decidability result for basic path logic. Since modal logic has many applications, we mention and state the relevance of our results for some of them. This section also includes graphs of an empirical analysis of the performances of first-order theorem provers as compared to special purpose implementations for K. The conclusion raises topics for further research.

2 The relational translation and resolution

We are concerned with propositional modal logics. In this section we briefly recall the definition of the basic propositional modal logic and sketch its relational semantics on which the relational as well as the functional translation approaches are based.

The language of the basic modal logic is a propositional language over a finite set V of propositional variables $p, q, r, \ldots$ and the familiar connectives $\bot$ (false) and $\rightarrow$ (implication) plus one or more unary modal operators of the form $\Diamond$ or $\langle R_i \rangle$. A *modal formula* is defined inductively by:

(i) Every propositional variable p in V is a modal formula.

(ii) $\bot$ is a modal formula.

(iii) $\varphi \rightarrow \psi$ is a modal formula, when both φ and ψ are modal formulae.

(iv) $\Diamond\varphi$ or $\langle R_i \rangle\varphi$ is a modal formula, when φ is a modal formula.

The other propositional connectives $\neg$, $\vee$, $\wedge$, $\leftrightarrow$ and $\top$ (truth) are defined as usual. The modal box operators $\Box$ and $[R_i]$ are duals of the respective diamond operators, that is, by definition, $\Box\varphi = \neg\Diamond\neg\varphi$ and $[R_i]\varphi = \neg\langle R_i \rangle\neg\varphi$. As to keep the presentation simple, this paper treats uni-modal logics in detail and mentions the adaptations required for their multi-modal versions.

The basic uni-modal logic is the logic K. It is defined by the axioms and rules of propositional logic plus one additional axiom, namely the K-axiom

$$\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$$

and one additional rule, the rule of necessitation:

$$\text{if } \vdash p \text{ then } \vdash \Box p.$$

Modal logics that satisfy the K-axiom and necessitation are called *normal* modal logics. The logic K is the weakest normal uni-modal logic. Extensions of K are obtained by adding further axioms like

$$\begin{array}{ll} D & \Box p \rightarrow \Diamond p \\ T & \Box p \rightarrow p \\ B & p \rightarrow \Box\Diamond p \\ 4 & \Box p \rightarrow \Box\Box p \end{array}$$

Let Σ be a set of such modal axioms. By $K\Sigma$ we denote normal modal logics, in which the K-axiom and necessitation are true and each of the formulae in Σ are axioms. We also use the notation *KT4*, for example, for extensions of K with one modality for which both T and 4 are axioms.

The standard semantics of normal propositional modal logics is known as the *Kripke semantics* or *possible world semantics* which is given in terms of relational structures called *frames*. A (relational) frame of a uni-modal logic is a pair (W, R) of a non-empty set of worlds W and a binary relation R on W. R is the *accessibility relation* that determines the truth of modal formulae in possible worlds. The defining class of frames of a modal logic determines (and is determined by) a corresponding class of models. A (*relational*) *model* is a triple (W, R, ι) of a frame (W, R) and a valuation mapping ι. The model is said to be *based on the frame* (W, R). *Validity* in any model $\mathcal{M} = (W, R, \iota)$ and any world $x \in W$ is defined inductively by:

$$\begin{array}{ll}
\mathcal{M}, x \models p & \text{if } x \in \iota(p) \\
\mathcal{M}, x \not\models \bot & \\
\mathcal{M}, x \models \varphi \to \psi & \text{if } \mathcal{M}, x \models \varphi \text{ implies } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \Diamond\varphi & \text{if there is a } y \in W \text{ such that } (x, y) \in R \text{ and } \mathcal{M}, x \models \varphi
\end{array}$$

Multi-modal logics have a class of modalities $\{\langle R_i \rangle\}_i$, and accordingly, their frames are tuples of a set W of worlds and a set $\{R_i\}_i$ of binary relations on W and each accessibility relation R_i defines the corresponding modality $\langle R_i \rangle$.

The *relational translation* of modal logics imitates the relational semantics. In general, the relational translation is a mapping Π_r of modal formulae into second-order logic. For any modal logic $K\Sigma$, Π_r is a function defined by

$$\Pi_r(\Sigma) = \bigwedge_{\varphi \in \Sigma} \Pi_r(\varphi)$$

and

$$\Pi_r(\varphi) = \begin{cases} \forall P_1 \ldots P_n \; \forall x \; \pi_r(\varphi, x) & \text{if } \varphi \text{ is an axiom in } \Sigma, \text{ and} \\ \forall x \; \pi_r(\varphi, x) & \text{if } \varphi \text{ is any modal formula not in } \Sigma. \end{cases}$$

The P_i are unary predicate symbols uniquely associated with the propositional variables p_i occurring in φ as defined by the auxiliary function π_r from modal formulae and worlds to first-order formulae:

$$\begin{aligned}
\pi_r(p, x) &= P(x) \\
\pi_r(\bot, x) &= \bot \\
\pi_r(\varphi \to \psi, x) &= \pi_r(\varphi, x) \to \pi_r(\psi, x) \\
\pi_r(\Diamond\varphi, x) &= \exists y \; (R(x, y) \wedge \pi_r(\varphi, y))
\end{aligned}$$

(Many authors define a function denoted by ST, which is short for 'standard translation', instead of π_r. Usually ST is a function from modal formulae to first-order formulae with one free variable x.)

In general, φ is a theorem in $K\Sigma$ iff $\Pi_r(\Sigma) \to \Pi_r(\varphi)$ is a theorem in second-order logic. Because we do modal deduction by resolution based first-order theorem provers, for the relational translation, we restrict our attention to first-order definable logics. For first-order definable modal logics, $\Pi_r(\Sigma)$ can be taken to be a set of first-order correspondence properties and

(1) φ is a theorem in $K\Sigma$ iff $\Pi_r(\Sigma) \to \Pi_r(\varphi)$ is a first-order theorem
iff $\Pi_r(\Sigma) \wedge \neg\Pi_r(\varphi)$ is f.o. unsatisfiable.

The first-order correspondence property for the logic KD is that R is a total (or serial) relation, for the logic KT it is that R is a reflexive relation, for the logic $S4$ (which coincides with $KT4$), it is that R is a quasi-order, and so on.

By way of two examples we will now explain modal inference by resolution combined with condensing and show that resolution without any refinement strategies is not a decision procedure for the relational translation of modal formulae. Resolution is first-order theorem proving method that establishes theoremhood by refutation. If φ is a theorem in a first-order definable $K\Sigma$, then any complete resolution procedure $\mathcal{R}$ will terminate having reduced $\Pi_r(\Sigma) \wedge \neg\Pi_r(\varphi)$ to the empty clause (the contradiction).

For example, the formula $\rho_1 = \Box p \to \Box\Diamond(\Diamond p \wedge \Diamond p)$ is a theorem in KT. The conjunction in ρ_1 is an intentional redundancy by which we will demonstrate how condensing eliminates unnecessary literals from clauses. We have:

$$\begin{aligned}
\Pi_r(T) &= \forall x\ R(x,x)\\
\neg\Pi_r(\rho_1) &= \exists x\ [\forall y\ (R(x,y) \to P(y)) \wedge \exists y\ (R(x,y) \wedge \forall z\ (R(y,z) \to\\
&\qquad \forall z'\ (R(z,z') \to \neg P(z')) \vee \forall z''\ (R(z,z'') \to \neg P(z''))))]
\end{aligned}$$

The Skolemised clausal form of $\Pi_r(T) \wedge \neg\Pi_r(\rho_1)$ is the set S_1 consisting of the following clauses (obtained by the standard transformation procedure that performs a conversion of first-order formulae to prenex conjunctive normal form and Skolemisation).

1. $R(x,x)$
2. $\neg R(\underline{x},y) \vee P(y)$
3. $R(\underline{x},\underline{y})$
4. $\neg R(\underline{y},z) \vee \neg R(z,z') \vee \neg P(z') \vee \neg R(z,z'') \vee \neg P(z'')$

$\underline{x}$ and $\underline{y}$ denote Skolem constants introduced for the existential quantifiers in $\neg\Pi_r(\rho_1)$. Evidently, the last clause is more complex than is necessary as it is logically equivalent to

$$4'.\ \neg R(\underline{y},z) \vee \neg R(z,z') \vee \neg P(z').$$

This is the kind of simplification achieved by condensing. Condensing is a redundancy elimination method due to Joyner Jr. (1976) and it is shown to be NP-hard by Gottlob and Fermüller (1993). Condensing replaces a clause by a simpler but logically equivalent clause. The *condensation* of a clause C, denoted by $\textsc{cond}(C)$, is the smallest subset of C which is also an instance of it. A clause is *condensed* if it does not subsume a proper subclause of itself, that is,

$$\text{if there is no substitution } \sigma \text{ such that } C\sigma \subsetneq C.$$

The condensation of a set S of clauses is the set

$$\textsc{cond}(S) = \{\textsc{cond}(C) \mid C \in S\}.$$

Up to variable renaming condensations are unique.

Clauses that are equal modulo variable renaming are called *variant* clauses. Two sets S and S' of clauses are *variants* of each other if every clause in S has a variant in S', and vice versa.

Applying the substitution $\{z'' \mapsto z'\}$ to clause 4 eliminates the redundant literals $\neg R(z,z'') \vee \neg P(z'')$. $4'$ is a condensation of 4. The other clauses are condensed, therefore, $\textsc{cond}(S_1) = \{1,2,3,4'\}$.

Depending on the definition we adopt, deduction based on the *resolution principle* does one or more steps producing the sum of instances of two clauses with complementary pairs of literals from the two clauses cancelling out. As usual, we use the symbols $C, C', D, D', \ldots$ for clauses. By definition, *dual literals* are complementary, that is, for A an atom, the duals of A and $\neg A$

are $A^{\neg} = \neg A$ and $(\neg A)^{\neg} = A$, respectively. The *dual of a clause* $C = L_1 \vee \ldots \vee L_n$ is the clause $C^{\neg} = L_1^{\neg} \vee \ldots \vee L_n^{\neg}$.

For propositional clauses or ground clauses deduction according to the resolution principle produces $D \vee D'$, from $D \vee C$ and $C^{\neg} \vee D'$. For non-ground clauses the generalisation is:

$$\text{Infer} \quad (D \vee D')\sigma, \quad \text{from} \quad D \vee C \quad \text{and} \quad C'^{\neg} \vee D',$$

provided $D \vee C$ and $C'^{\neg} \vee D'$ have no common variables and σ is the most general substitution that unifies C and C', that is, $C\sigma = C'\sigma$. If C and $C'^{\neg}$ are singleton sets, then they are referred to as the *literals resolved upon.* This deduction process is usually implemented as a combination of binary resolution and factoring.

A proof for $\Pi_r(T) \wedge \neg\Pi_r(\rho_1)$ is the following reduction of $\textsc{cond}(S_1)$ to the empty clause.

$$\begin{array}{ll}
5.\ [2.1,\ 3.1] & P(\underline{y}) \\
6.\ [5.1,\ 4'.3] & \neg R(\underline{y}, z) \vee \neg R(z, \underline{y}) \\
7.\ [1.1,\ 6.1, 6.2] & \emptyset
\end{array}$$

Inside the square brackets we indicate the literals resolved upon that produce the resolvents. For instance, clause 5 is the resolvent of clauses 2 and 3 with the most general unifier $\sigma = \{y \mapsto \underline{y}\}$.

Since resolution is a semi-decision procedure for first-order logic, the computation will not necessarily halt for non-theorems. The computation can halt without producing the empty clause when deduction produces no new clauses that are not variants of clauses already present. In that case, for any complete resolution procedure, the input set S is satisfiable. However, for many non-theorems resolution generates increasingly more non-variant clauses.

For example, neither

$$\rho_2 = \Box(p \rightarrow \Diamond p)$$

nor its negation $\neg\rho_2$ is a theorem of K. ρ_2 is satisfiable, which implies $\neg\rho_2$ is not a theorem and the formula

$$\neg\Pi_r(\neg\rho_2) = \exists x \forall y\ (\neg R(x, y) \vee \neg P(y) \vee \exists z\ (R(y, z) \wedge P(z)))$$

is first-order satisfiable. The Skolemised clausal form is the set S_2 consisting of the following two clauses.

$$1.\ \neg R(\underline{x}, y) \vee \neg P(y) \vee R(y, f(y))$$
$$2.\ \neg R(\underline{x}, y) \vee \neg P(y) \vee P(f(y)).$$

$\underline{x}$ is the Skolem constant introduced for the quantifier $\exists x$ and $f(y)$ is the Skolem term introduced for $\exists z$.

The clauses 1 and 2 have two resolvents.

$$3.\ [1.3,\ 2.1] \quad \neg R(\underline{x}, \underline{x}) \vee \neg P(\underline{x}) \vee \neg P(f(\underline{x})) \vee \neg P(f^2(\underline{x}))$$
$$4.\ [1.2,\ 2.3] \quad \neg R(\underline{x}, f(y)) \vee R(f(y), f^2(y)) \vee \neg R(\underline{x}, y) \vee \neg P(y).$$

Clause 4 resolves with clause 2 and yields:

$$5.\ [2.3,\ 4.4] \quad \neg R(\underline{x}, f^2(y)) \vee R(f^2(y), f^3(y)) \vee \neg R(\underline{x}, f(y)) \vee \neg R(\underline{x}, y) \vee \neg P(y).$$

This clause also resolves with 2, and again, the resulting resolvent resolves with 2, etcetera. Repeatedly resolving the new resolvents with clause 2 yields increasingly longer and more complex clauses. None of the clauses is redundant and can be deleted. In the limit, our sample S_2 has infinitely many non-variant resolvents. This shows standard resolution may not terminate for relational translations of modal formulae.

There are refinements of resolution that guarantee termination for the relational translation, for which refer to Fermüller, Leitsch, Tammet and Zamov (1993) and Hustadt (1997).

The idea of our decision proof for the optimised functional translation and basic path logic is that of Joyner Jr. (1976). We show that any resolution procedure combined with condensing produces for a finite input set S a finitely bounded set of non-variant resolvents, by showing that any class of path clauses built from finitely many predicate and constant symbols is finitely bounded. The second example demonstrates, one reason for an unbounded number of resolvents being produced is:

(i) The level of nesting of the terms increases during deduction (from $f(y)$ to $f^2(y)$ to $f^3(y)$, and so on).

Another reason is:

(ii) The number of literals in the resolvents increases during deduction.

(i) is not a problem in basic path logic as terms do not expand during deduction. We will show that (ii) does not happen, by showing that there is an upper bound on the number of variables that occur in any condensed clause that is built from finitely many predicate and constant symbols.

Formally, a *resolution procedure* is a function $\mathcal{R}$ from finite sets of clauses to finite sets of clauses. If S is a finite set of clauses then $\mathcal{R}(S)$ is a finite subset of the union of S and the (possibly infinite) set of instances of resolvents of pairs of clauses in S. Let

$$\begin{aligned}\mathcal{R}^0(S) &= S \quad \text{and}\\ \mathcal{R}^{n+1}(S) &= \mathcal{R}(\mathcal{R}^n(S)) \quad \text{if } n > 0.\end{aligned}$$

By $\mathcal{R}_{\text{COND}}$ we denote the procedure that combines any complete resolution procedure $\mathcal{R}$ and condensing. Define $\mathcal{R}_{\text{COND}}$ and $\mathcal{R}^n_{\text{COND}}$ by:

$$\begin{aligned}\mathcal{R}_{\text{COND}}(S) &= S \cup \text{COND}(\mathcal{R}(S) \setminus S),\\ \mathcal{R}^0_{\text{COND}}(S) &= \text{COND}(\mathcal{R}^0(S)) = \text{COND}(S), \text{ and}\\ \mathcal{R}^n_{\text{COND}}(S) &= \mathcal{R}_{\text{COND}}(\mathcal{R}^{n-1}_{\text{COND}}(S)) \quad \text{if } n > 0.\end{aligned}$$

In essence, $\mathcal{R}_{\text{COND}}$ is like the procedure $\mathcal{R}$ except that each resolvent is replaced by its condensation. Joyner Jr. (1976) shows condensing is compatible with any resolution procedure complete via semantic trees.

A resolution procedure $\mathcal{R}$ is (sound and) *complete* when, for some $n \geq 0$, the empty clause $\emptyset$ is in $\mathcal{R}^n(S)$ iff S is unsatisfiable. A complete resolution procedure may also halt without producing the empty clause. When at some level n, $\mathcal{R}^n(S)$ and $\mathcal{R}^{n+1}(S)$ coincide or are variants of each other, a complete resolution procedure halts and if the empty clause is not in $\mathcal{R}^n(S)$, then S is satisfiable.

Resolution decision procedures are defined for effectively specified classes of first-order formulae. Let $c(\varphi)$ denote the the clausal form of a first-order formula φ. A *resolution decision procedure* for a class $\mathcal{C}$ of formulas is a complete resolution procedure $\mathcal{R}$ for which the following is true.

> If $\varphi \in \mathcal{C}$ and φ is satisfiable, then for some $n \geq 0$, $\mathcal{R}^n(c(\varphi))$ and $\mathcal{R}^{n+1}(c(\varphi))$ are variants.

Such an $\mathcal{R}$ halts for any formula that belongs to the class $\mathcal{C}$. In this paper we focus on the class of path formulae that arise from the optimised functional translation.

3 The optimised functional translation

Like the relational translation approach, the functional translation approach and its optimisation are based on the relational semantics of modal logic.

The *functional semantics* of modal logics arises from the relational semantics by a reformulation of relations as sets of functions. Any binary relation R on a non-empty set W can be defined by a set AF of (partial or total) functions $\alpha : W \longrightarrow W$ as specified by

$$\forall xy\ (R(x,y) \leftrightarrow \exists\alpha\ (\alpha \in AF \wedge \alpha(x) = y)).$$

This formulation has the disadvantage that the variable α is a second-order variable. We simulate functional application with a designated non-associative binary operation $[\cdot,\cdot] : W \times AF \longrightarrow W$ by interpreting terms of the form

$$[x\alpha] \quad \text{as} \quad \alpha(x).$$

Complex terms of the form $\alpha_m(\ldots\alpha_2(\alpha_1(x)))$ are simulated by terms of the form $[[[[x\alpha_1]\alpha_2]\ldots]\alpha_m]$, or in our shorthand notation

$$[x\alpha_1\alpha_2\ldots\alpha_m].$$

Any binary relation R is defined then by a set AF of functions and the operation $[\cdot,\cdot]$, when

$$\forall xy\ (R(x,y) \leftrightarrow \exists\alpha\ (\alpha \in AF \wedge [x\alpha] = y)) \tag{2}$$

in which α is now a first-order variable. If R is a total relation then AF can be taken to be a set of total functions. For the general case that R is not total we introduce another designated predicate de, called the *dead end predicate*, and define any R by

$$\forall xy\ (R(x,y) \leftrightarrow (\neg de(x) \wedge \exists\alpha\ (\alpha \in AF \wedge [x\alpha] = y))), \tag{3}$$

with AF being a set of total functions. The right hand side of the equivalence reads, if x is not a dead end in R then there is a total function α that maps x to y.

Accordingly, a *functional model* is a tuple

$$(W, AF, [\cdot,\cdot], \iota) \quad \text{or} \quad (W, de, AF, [\cdot,\cdot], \iota),$$

depending on whether R is a total relation or not. AF is a set of *total* so-called *accessibility functions* defined by either (2) or (3). de is a subset of W. *Validity* in any functional model $\mathcal{M}$ and any world $x \in W$ is defined inductively by:

$$\begin{array}{l} \mathcal{M}, x \models p \quad \text{if } x \in \iota(p) \\ \mathcal{M}, x \not\models \bot \\ \mathcal{M}, x \models \varphi \to \psi \quad \text{if } \mathcal{M}, x \models \varphi \text{ implies } \mathcal{M}, x \models \psi \end{array}$$

for the propositional part, which is as in the relational semantics, and for serial and non-serial modal diamond operators, validity is defined by

$$\mathcal{M}, x \models \Diamond\varphi \quad \text{if there is an } \alpha \in AF \text{ such that } \mathcal{M}, [x\alpha] \models \varphi$$

and

$$\mathcal{M}, x \models \Diamond\varphi \quad \text{if } x \notin de \text{ and there is an } \alpha \in AF \text{ s.t. } \mathcal{M}, [x\alpha] \models \varphi,$$

respectively.

The truth of a formula φ in a world x depends only on the truth of subformulae of φ in R-successors of x. This means for the truth of φ in x, the predecessors of x or disconnected parts of the frame are irrelevant. This is made precise in the generated model property of normal modal logics. The property allows us to assume any frame is connected and has a starting world x, say. In the functional setting worlds are denoted by strings of the form $[x\alpha_1 \dots \alpha_n]$ and we note the notation displays also the *path* from the initial world x to that world. There is a shift of point of view from worlds to paths. In relational models for $\Diamond\varphi$ to be true at a world we require the existence of another accessible *world* at which φ is true. In functional models for $\Diamond\varphi$ to be true at a world we require the existence a *one step path* (or function) that leads to a world at which φ is true. Terms or strings of the form $[x\alpha_1 \dots \alpha_n]$ will also be referred to as *paths*, even though these terms define worlds.

The *functional translation* mapping is denoted by Π_f and mimics the functional semantics. Π_f maps modal formulae of $K\Sigma$ to second-order or first-order formulae as given by:

$$\Pi_f(\Sigma) = \bigwedge_{\varphi \in \Sigma} \Pi_f(\varphi)$$

and

$$(4) \qquad \Pi_f(\varphi) = \begin{cases} \forall P_1 \ldots P_n \; \forall x \; \pi_f(\varphi, x) & \text{if } \varphi \text{ is an axiom in } \Sigma, \text{ and} \\ \forall x \; \pi_f(\varphi, x) & \text{if } \varphi \text{ is not an axiom.} \end{cases}$$

π_f is the auxiliary function that maps modal formulae and worlds to first-order formulae. It defines the unary predicates P_i uniquely associated with the propositional variables p_i in φ.

$$\begin{aligned} \pi_f(p, s) &= Ps \\ \pi_f(\bot, s) &= \bot \\ \pi_f(\varphi \to \psi, s) &= \pi_f(\varphi, s) \to \pi_f(\psi, s) \\ \pi_f(\Diamond\varphi, s) &= \begin{cases} \exists\alpha \; \pi_f(\varphi, [s\alpha]) & \text{in } KD\Sigma \\ \neg de(s) \wedge \exists\alpha \; \pi_f(\varphi, [s\alpha]) & \text{in non-serial } K\Sigma. \end{cases} \end{aligned}$$

Like for the relational translation, in general, φ is a theorem in $K\Sigma$ iff $\Pi_f(\Sigma) \to \Pi_f(\varphi)$ is a theorem in second-order logic. For first-order definable modal logics

$$\begin{aligned} \varphi \text{ is a theorem in } K\Sigma \quad &\text{iff} \quad \Pi_f(\Sigma) \to \Pi_f(\varphi) \text{ is a first-order theorem} \\ &\text{iff} \quad \Pi_f(\Sigma) \wedge \neg\Pi_f(\varphi) \text{ is f.o. unsatisfiable.} \end{aligned}$$

There is no unique definition of a binary relation by a set of total functions which means the class of functional models that capture a complete modal logic is larger than the class of its relational models. The model theory for normal modal logics gives us choices both in the relational and the functional context. For example, by the generated model property a modal logic determined by standard relational models is also determined by the associated class of generated models. In Ohlbach and Schmidt (1995) we prove a modal logic determined by a class of relational or functional models is also determined by the associated class of so-called *maximal functional models.* A functional model is maximal iff AF is the set of *all* total functions that define R. In these models enough functions are available so that the following is true.

$$\exists\alpha\forall\beta \; \psi \leftrightarrow \forall\beta\exists\alpha \; \psi$$

This equivalence is exploited in the optimised functional translation.

The *optimised functional translation* of a modal formula φ is obtained by a sequence of two transformations. The first transformation maps a modal formula φ to the functional translation $\Pi_f(\varphi)$. The second transformation applies the so-called *quantifier exchange operator* Υ. It allows for the movement of existential functional quantifiers inward over universal functional quantifiers using the rule:

$$\exists\alpha\forall\beta\ \psi \quad \text{becomes} \quad \forall\beta\exists\alpha\ \psi.$$

Again, φ is a theorem in $K\Sigma$ iff $\Upsilon\Pi_f(\Sigma) \rightarrow \Upsilon\Pi_f(\varphi)$ is a theorem in second-order logic. One of the advantages of the optimised functional translation over the relational and functional translation is that more modal logics can be embedded in first-order logic. McKinsey's axiom ($\Box\Diamond p \rightarrow \Diamond\Box p$) which has no first-order relational correspondence property has a first-order optimised functional approximation. It is not an equivalent formulation of McKinsey's axiom, but the transformation preserves satisfiability, since

$$\begin{aligned} \varphi \text{ is a theorem in } K\Sigma \quad &\text{iff} \quad \Upsilon\Pi_f(\Sigma) \rightarrow \Upsilon\Pi_f(\varphi) \text{ is a first-order theorem} \\ &\text{iff} \quad \Upsilon\Pi_f(\Sigma) \wedge \neg\Upsilon\Pi_f(\varphi) \text{ is f.o. unsatisfiable.} \end{aligned}$$

This holds for $K\Sigma$ that are first-order definable in the relational setting and more, for example, for K extended with McKinsey's axiom. Further details can be found in Ohlbach and Schmidt (1995).

For the logics K and KD, $\Upsilon\Pi_f(\Sigma)$ is empty. For KDT, KDB and $KD4$, the defining characteristic functional properties in $\Upsilon\Pi_f(\Sigma)$ are first-order equations, namely

$$\begin{aligned} \textit{left identity}: \quad & [x\underline{e}] = x, \\ \textit{right inverse}: \quad & [x\alpha\alpha^{-1}] = x, \text{ and} \\ \textit{associativity}: \quad & [x\alpha\beta] = [x(\alpha \circ \beta)]. \end{aligned}$$

The equations are already in Skolemised clausal form. x, α and β denote first-order variables, $\underline{e}$ is a Skolem constant, $^{-1}$ a unary Skolem function and $\circ$ a binary Skolem function. The equations give a functional reformulation of the familiar correspondence properties. For T, left identity says $\underline{e}$ is the identity function in AF that maps any world x to x. For B, right inverse says every function α has an inverse α^{-1}, and for 4, associativity says functions may be composed.

The quantifier exchange operator Υ allows us to move existential quantifiers inside over universal quantifiers. We are not compelled to do all quantifier swaps that are possible. The completeness result of the translation $\Upsilon\Pi_f$ is true even if we choose to let Υ do nothing. However, moving all existential quantifiers inward as far as possible we can embed any modal non-axiom into a class of clauses that contains no complex Skolem terms other than constants. For a given non-axiom φ, S is defined to be the clausal form of the negation of the optimised version of the functional translation

$$S = c(\neg\Upsilon\Pi_f(\varphi)),$$

with the requirement that the only function symbols occurring in S are Skolem constants and the application operation $[\cdot,\cdot]$.

We consider two examples. The following formula is a theorem in *KD*.

$$\rho_3 = \underset{\alpha}{\Box}(\underset{\beta}{\Diamond}\underset{\gamma}{\Box}\neg p \vee \underset{\delta}{\Diamond}\underset{\epsilon}{\Diamond} p) \tag{5}$$

Its functional translation is

$$\Pi_f(\rho_3) = \forall x\ \forall\alpha\ (\exists\beta\forall\gamma\ \neg P[x\alpha\beta\gamma] \vee \exists\delta\exists\epsilon\ P[x\alpha\delta\epsilon]). \tag{6}$$

The boxes translate to universal functional quantifiers and the diamonds to existential functional quantifiers in such a way that each occurrence of a modal operator is uniquely associated with a functional variable, as indicated in the second line of (5). The propositional symbol p in ρ_3 becomes a monadic predicate P. Now apply the quantifier exchange operator in such a way that the Skolemised clausal form of the negation of $\Upsilon\Pi_f(\rho_3)$ does not contain any complex Skolem terms. This optimisation is always possible. The required optimisation of $\Pi_f(\rho_3)$ is:

$$\Upsilon\Pi_f(\rho_3) = \forall x\ \forall\alpha\ (\forall\gamma\exists\beta\ \neg P[x\alpha\beta\gamma] \vee \exists\delta\exists\epsilon\ P[x\alpha\delta\epsilon]).$$

Note the order of the quantifiers in the prefix of the first part of the disjunction is reversed. Negating, we then get the formula

$$\neg\Upsilon\Pi_f(\rho_3) = \exists x\ (\exists\alpha\exists\gamma\forall\beta\ P[x\alpha\beta\gamma] \wedge \exists\alpha\forall\delta\forall\epsilon\ \neg P[x\alpha\delta\epsilon]).$$

in which all existential quantifiers precede all universal quantifiers. Its clausal normal form is

1. $P[\underline{x}\underline{\alpha}\beta\underline{\gamma}]$
2. $\neg P[\underline{x}\underline{\alpha}\delta\epsilon]$.

Underlined symbols denote constants and non-underlined symbols denote variables. $\underline{x}$ is the initial world and because it occurs in every literal, we can safely omit $\underline{x}$. In what follows, we make $\underline{x}$ implicit in the notation by denoting the initial world in any $S = c(\neg\Upsilon\Pi_f(\varphi))$ by the empty string $[]$. One resolution step is possible. If we take the second clause and rename the variable δ by β and instantiate the variable ϵ with the constant $\underline{\gamma}$, we can resolve this clause with the first clause and get the empty clause. This proves the formula $\neg\Upsilon\Pi_f(\rho_3)$ is unsatisfiable, $\Upsilon\Pi_f(\rho_3)$ is a first-order theorem and because the optimised functional translation is sound and complete, we have shown that ρ_3 is a theorem of KD.

Consider again the formula $\rho_2 = \Box(p \to \Diamond p)$ from the previous section, where we saw unrefined resolution does not terminate for $\neg\Pi_r(\neg\rho_2)$. In this example, we use the non-serial definition of π_f.

$$\begin{aligned}
\neg\rho_2 &= \Diamond(p \wedge \Box(\neg p)) \\
\Pi_f(\neg\rho_2) &= \forall x\ (\neg de[x] \wedge \exists\alpha\ (P[x\alpha] \wedge (\neg de[x\alpha] \to \forall\beta\ \neg P[x\alpha\beta]))) \\
\Upsilon\Pi_f(\neg\rho_2) &= \forall x\forall\beta\exists\alpha\ (\neg de[x] \wedge P[x\alpha] \wedge (\neg de[x\alpha] \to \neg P[x\alpha\beta])) \\
\neg\Upsilon\Pi_f(\neg\rho_2) &= \exists x\exists\beta\forall\alpha\ (de[x] \vee \neg P[x\alpha] \vee (\neg de[x\alpha] \wedge P[x\alpha\beta])) \\
S &= \{\ 1.\ de[] \vee \neg P[\alpha] \vee \neg de[\alpha], \\
&\qquad 2.\ de[] \vee \neg P[\alpha] \vee P[\alpha\underline{\beta}]\ \}
\end{aligned}$$

For S no resolution step is possible, which proves S, and hence ρ_2, is satisfiable. (The literals $P[\alpha]$ and $P[\alpha'\underline{\beta}]$ are not unifiable, because they are the shorthand notation for $P[[]\alpha]$ and $P[[[]\alpha']\underline{\beta}]$ and the application operation is not associative.)

The structure of modal formulae determines an invariance on variables and constants in paths (Ohlbach (1988a, 1988b)), which is essential for the decision proof. The variables in the translation $\Pi_f(\varphi)$ of any modal formula φ are ordered in a certain way. Compare, for instance, ρ_3 and $\Pi_f(\rho_3)$ in (5) and (6). The modal operators and the functional variables form unique pairs (α is associated with the first occurrence of a box, etcetera) and the structure of ρ_3 determines an ordering on the variable occurrences in the terms $[x\alpha\beta\gamma]$ and $[x\alpha\delta\epsilon]$ of $\neg\Pi_f(\rho_3)$. The characteristic ordering of the variables in the terms is formalised as the *prefix stability* property (also called *unique path property* by Auffray and Enjalbert (1992)). Any set S of clauses is prefix stable for variables and constants iff

(i) all occurrences of a variable in any term of any clause in S have the *same prefix*, and

(ii) all occurrences of a constant in any term occurring in S have the *same prefix* (modulo variable renaming).

Path logics satisfy only the first restriction for variables, since condition (ii) is not preserved by forming resolvents.

4 Path logics

The optimised functional translation maps propositional modal formulae into logics called *path logics* (not to be confused with the path logic of Auffray and Enjalbert or Fariñas Del Cerro and Herzig (1995)). Path logics are monadic fragments of first-order logic with one special function symbol, namely $[\cdot, \cdot]$, for defining accessibility by juxtaposition. A formula of path logic is further restricted in that its clausal form may only contain variables and constant symbols, and it is required to satisfy prefix stability for variables.

More formally, the vocabulary of the *basic path logic* includes unary predicates, variables, constants and one binary function. $P, Q, \ldots$ denote unary predicates. Functional variables are denoted by Greek letters $\alpha, \beta, \ldots$ and functional constants by underlined Greek letters $\underline{\alpha}, \underline{\beta}, \ldots$. The letters $x, y, \ldots$ denote world variables. There is a special world constant, denoted by the empty string $[]$, for the initial world. $[\cdot, \cdot]$ is a binary left-associative operation. Terms, also called *paths*, are of the form

$$[[[[[]u_1]u_2]\ldots]u_m] \quad \text{or} \quad [[[[xu_1]u_2]\ldots]u_m],$$

or in shorthand notation

$$[u_1 u_2 \ldots u_m] \quad \text{or} \quad [x u_1 u_2 \ldots u_m].$$

The u_i denote functional variables or constants.

The following notation and definitions will be used in the next section. The symbols $u, u_1, u_2, \ldots$ and also $v, v_1, v_2, \ldots$ are reserved for either functional variables or constants. The symbols $s, t, \ldots$ are reserved for any terms of the form $[]$, u or $[u_1, u_2, \ldots]$. Usually the term $[s\,t]$ is malformed since juxtaposition is assumed to be left-associative. When we write $[s\,t]$ we mean the term $[s\,u_1 \ldots u_i]$, where $t = [u_1 \ldots u_i]$. Given a path s, let $s|_0 = []$ and define all $s|_i$ for $i > 0$ by $s = [s|_1 \ldots s|_m]$. By definition, the *length* of a term $[u_1 \ldots u_m]$ is m.

Our definition of prefixes and suffixes extends the conventional definition. Consider the term

$$t = [suu_{i+1} \dots u_m].$$

The subterm s of t is a *prefix in the term* t. s may be the empty string. By definition, the *prefix of a variable or constant* u in the term t is the term s. The prefix of u in $[uu_{i+1} \dots u_m]$ is the empty string $[]$. Similarly, we define suffixes. $[u_{i+1} \dots u_m]$ is a *suffix in the term* t. The *suffix of a variable or constant* u in the term t is $[u_{i+1} \dots u_m]$. u_m has no suffix in t. The *suffix of a subterm* s in t is $[uu_{i+1} \dots u_m]$.

A set T of terms is said to be *prefix stable (for variables)* if any variable α occurring in T has exactly one prefix. A clause is said to be prefix stable (for variables) if the set of terms occurring in the clause has this property. The *prefix of a variable or constant* u *in a prefix stable clause* (or *set of terms*) is the unique prefix of u in any term of the clause (or set).

The logical connectives and quantifiers of basic path logic are those of first-order logic. By definition, a formula φ is a *path formula* iff it is a formula φ over the language just defined and its clause form $c(\varphi)$ satisfies prefix stability for variables. This defines the syntax of the basic path logic associated with K and KD.

Path logics associated with modal logics with non-empty theory Σ are extensions of the basic path logic with equational theories determined by $\Upsilon\Pi_f(\Sigma)$. The modal logics *KDT*, *KDB* and *KD4* are embedded in extensions in which the theories are defined by left identity, right inverse and associativity, respectively.

The semantics of path logics is defined as usual for first-order logic.

Any set S of clauses resulting from the optimised functional translation of a modal non-axiom is well-formed in basic path logic and all its extensions. Ohlbach (1988a, 1991) proves that prefix stability for variables is preserved by resolution, factoring and subsumption. It is not difficult to see prefix stability is also preserved by condensing. Hence, any $\mathcal{R}^n(S)$ and $\mathcal{R}^n_{\text{COND}}(S)$ (with $\mathcal{R}$ being unrefined resolution) are well-formed in path logics.

Inference for basic path logics is facilitated by resolution with syntactic unification, whereas inference for non-basic path logics is facilitated by theory resolution. Theory resolution combines resolution with a more sophisticated unification algorithm for solving equations modulo a given theory. For T, B and 4 unification is modulo the left identity law, the inverse law and the associativity law, respectively.

Syntactic unification does not expand terms, as bindings either rename variables or instantiate variables with constants. Unification for T and B does not expand terms. For example, the clauses

$$P[\alpha]$$
$$\neg P[\alpha'\beta] \vee Q[\alpha'\beta]$$

(which do not resolve modulo the empty theory) have a theory resolvent modulo left identity. A most general unifier modulo left identity of the terms $[\alpha]$ and $[\alpha'\beta]$ is $\{\alpha' \mapsto \alpha, \beta \mapsto \underline{e}\}$. In essence, the unification algorithm of Ohlbach (1988a) immediately deletes variables that are instantiated with $\underline{e}$, so that the theory resolvent is $Q[\alpha]$. Theory resolution for right inverse is similar, combining unification and deleting terms as in the following example. A resolvent of

$$P[\alpha]$$
$$\neg P[\alpha'\underline{\beta}\gamma] \vee Q[\alpha'\underline{\beta}\gamma]$$

is $Q[\alpha]$, the most general unifier being $\{\alpha' \mapsto \alpha, \gamma \mapsto \underline{\beta}^{-1}\}$.

For 4 new variables are introduced during resolution and terms expand. For example, a resolvent of

$$P[\alpha\gamma] \vee Q[\alpha\gamma]$$
$$\neg P[\alpha'\underline{\beta}] \vee R[\alpha'\underline{\beta}]$$

is $Q[\alpha(\gamma' \circ \underline{\beta})] \vee R[(\alpha \circ \gamma')\underline{\beta}]$. The most general unifier is $\{\gamma \mapsto \gamma' \circ \underline{\beta},\ \ \alpha' \mapsto \alpha \circ \gamma'\}$ with γ' being a new variable.

5 The decision proof

This section proves that any complete procedure $\mathcal{R}_{\text{COND}}$ is a decision procedure for any finite formulae of the basic path logic. We will be concerned with the procedure $\mathcal{R}_{\text{COND}}$ in which $\mathcal{R}$ is ordinary resolution without any refinement strategy. To prove that $\mathcal{R}_{\text{COND}}$ produces a finitely bounded number of resolvents, we show that every set of non-variant clauses built from given finite sets of predicate and constant symbols and terms with bounded depth is finite.

What happens is this. Given a finite set S of input clauses of basic path logic, we form its condensation $\text{COND}(S)$ and then we repeatedly apply

$\mathcal{R}_{\text{COND}}$. All clauses formed at any level contain literals that are made up of only the finite set of predicate symbols and Skolem constants occurring in S, and variables. We saw in the example of Section 2, one of two things, or both, can occur bearing an unbounded number of non-variant clauses.

(i) Either literals of increasingly greater term depth appear in the clauses, or

(ii) clauses having increasingly more literals are produced.

From the outset we consider only basic path logic in which terms do not grow. Formulated more generally, we consider only path logics for which the maximal term depth in any literal is bounded. We prove (ii) does not happen, that is, we prove *there is a bound on the cardinality (size) of any condensed clause of basic path logic* (Theorem 5.20). Then it follows that there are finitely many non-variant clauses which means $\mathcal{R}_{\text{COND}}$ does not bear an unbounded number of non-variant clauses and terminates always for the path logics under consideration.

The existence of a size bound of any condensed clause follows from the existence of a bound on the number of its variables. For the proof of the existence of a variable bound in any clause we use Joyner's (1976) method of variable partitioning, or splitting (without requiring splitting as an inference rule). The *variable partition* of any clause C is the finest partition of C into disjoint subclauses that do not share common variables. We say the subclauses in the variable partition are *variable indecomposable* or *split*. For example, the variable partition of the clause

$$P(\underline{a}) \vee P(\underline{b}) \vee \neg Q(z') \vee \neg R(x,y) \vee \neg R(y,z) \vee R(x,z)$$

is the following.

$P(\underline{a})$	$P(\underline{b})$	$\neg Q(z')$	$\neg R(x,y) \vee \neg R(y,z) \vee R(x,z)$

$P(\underline{a})$ and $P(\underline{b})$ are in separate blocks because no two ground literals share variables (here we deviate from Joyner). The transitivity subclause is variable indecomposable because its variables form a chain across the literals. And, $\neg Q(z')$ is in a different block from the other subclauses as the variable z' occurs in none of the other literals.

Let C be a variable indecomposable and condensed clause in the class of clauses of basic path logic built from a finite number of predicate symbols and function symbols with bounded term depth. *When there is also a bound on the number of variables that occur in C*, then it follows C has finite size. Then there are finitely many non-variant variable indecomposable clauses in the class and this implies that any condensed clause in the class has finite size.

5.1 Overview

Our aim is to prove the existence of a bound on the number of variables that occur in any variable indecomposable and condensed clause of basic path logic built from finitely many predicate symbols and constant symbols. We do not give an explicit variable bound. Our strategy is, to show there is an embedding of the variables of a condensed and variable indecomposable clause into a finite Boolean algebra. It is natural to look for an embedding that encodes the variables of a given clause in terms of the finite entities of that clause, that is, in terms of the constants and the predicate symbols of the clause. The problem is how to deal with condensing. Ideally we want the encoding to have a natural property that captures the condensedness property of the clause. The encoding we use is weaker. It has a natural property that is necessary but not sufficient for condensedness. We encode variables by sets of tuples that represent the set of suffixes of the given variable and the necessary property for the clause to be condensed is the *antichain property*. It requires that the encodings of variables with the same prefix form an antichain. An *antichain* is a subset of pairwise incomparable elements in a poset $(P, \subseteq)$. For example, the set of singleton subsets of a set with more than two elements is an antichain.

The encoding of variables will be by a family of embeddings $\{f_s\}$. Each f_s will map any variable with prefix s to a tuple-encoding of the set of suffixes of the variable. The encoding exploits the property of prefix stability and induces finer partitions on clauses than the variable partition. The finer partitioning is by prefixes. A block in a *prefix partition* is a subclause containing all literals sharing a common prefix. We show there is a one-one correspondence between variable encodings and blocks in prefix partitions of the clause. The most difficult part of the proof is showing that, if there is a subset relationship between the encodings of the pairs of variables (that is, the antichain property is not true) then one of the corresponding blocks is redundant (Theorem 5.12).

For reasons of uniformity, improved readability and ease of formulation we develop our argument for sets of terms and not for clauses. We encode clauses as *sets of terms all with the same length.* More precisely, assuming that m denotes the maximal length of any term in the input set S, we encode any clause in S or $\mathcal{R}^n_{\text{COND}}(S)$ by a set of terms of length $m+1$. For example, if $m = 3$ the clause

$$(7) \qquad P[\alpha\underline{\beta}] \vee \neg P[\alpha\gamma\underline{\delta}] \vee Q[\underline{\epsilon}\gamma'] \quad \text{is reperesented by} \quad \begin{array}{l}[\alpha\,\underline{\beta}\,\underline{b}\,\underline{p}]\\ [\alpha\,\gamma\,\underline{\delta}\,\underline{p}']\\ [\underline{\epsilon}\,\gamma'\,\underline{b}\,\underline{q}]\end{array} .$$

$\underline{b}$ is a special blank constant, and $\underline{p}$, $\underline{p}'$ and $\underline{q}$ are new and different constant symbols associated with P, $\neg P$ and Q, respectively.

In general, the terms of a clause have different lengths. By introducing new blank symbols which we append to terms shorter than m we obtain terms with uniform length. We convert literals to terms by introducing additional new constant symbols for positive and negative occurrences of predicate symbols. We append the new 'predicate constant' that represents the 'head' of the literal to the term in the argument. (By the *head* of a literal we mean the string obtained by deleting the arguments of the literal.)

In the described way we convert any clause C to a set T of terms all with equal length. T can be viewed as a matrix when we write the terms below each other as in (7). This *matrix term representation* of a clause helps visualising prefix partitions.

Reformulated for the term representation our goal is to show that *the number of variables occurring in any condensed and variable indecomposable set T of terms that is prefix stable is finitely bounded.* By definition, a set T of terms is *condensed* if there is no substitution σ such that $T\sigma \subsetneqq T$.

We define two operations for dividing and reassembling T. Let $T(s)$ be the *set of suffixes* of s occurring in T. Formally, let s be a term and define

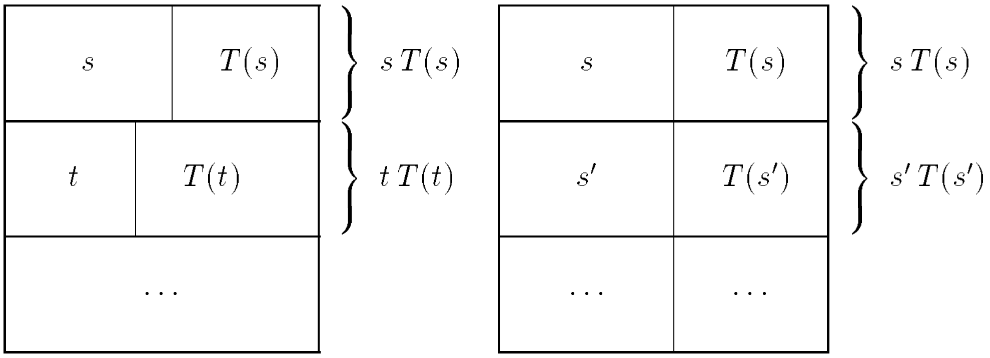
$$T(s) = \{[u_1 \dots u_i] \mid [su_1 \dots u_i] \in T\}.$$

$T(s)$ may be viewed as a sub-matrix of T. For example, if T is the matrix of (7) then

$$T([\alpha]) = \begin{array}{l}[\underline{\beta}\,\underline{b}\,\underline{p}]\\ [\gamma\,\underline{\delta}\,\underline{p}']\end{array} .$$

Forming sub-matrices satisfies a kind of associativity: For any terms s and t occurring in T,

$$(T(s))(t) = T([st]).$$



(a) The variable partition T/θ_v (b) The prefix partition T/θ_i

Figure 1: Partitioning T

We can construct the subset of terms in T that have s as a common prefix from the suffixes in $T(s)$ by prepending s with the following operation:

$$s\,T' = \{[su_1 \dots u_i] \mid [u_1 \dots u_i] \in T'\}.$$

Then, $s\,T(s)$, more precisely $s\,(T(s))$, denotes the subset of all terms in T that have prefix s. For example, for T of (7)

$$[\alpha]\,T([\alpha]) = \alpha\,T(\alpha) = \begin{matrix}[\alpha\underline{\beta}\,\underline{b}\,\underline{p}] \\ [\alpha\gamma\,\underline{\delta}\underline{p}']\end{matrix}\ .$$

Prepending terms is monotone, that is,

$$T' \subseteq T'' \quad \text{implies} \quad s\,T' \subseteq s\,T''.$$

Because of prefix stability, any variable indecomposable set of terms has the form $s\,T(s)$, and when the set is not ground, it has the form $[s\alpha]\,T([s\alpha])$. The variable partition of a set T of terms (given by an equivalence relation θ_v) can be viewed as depicted in Figure 1(a). Each row in the sub-matrix marked with s is the term s, and the sub-matrix marked $T(s)$ is the matrix of all suffixes of s in T.

The opposite picture 1(b) depicts a a prefix partition of T by an equivalence relation θ_i. The blocks of the partition are $s\,T(s)$, $s'\,T(s')$, etcetera, and are such that $s, s', \dots$ are all terms of length i.

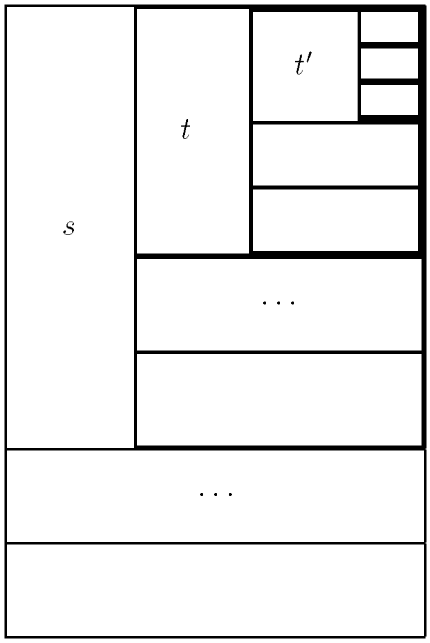


Figure 2: Nested partitioning of T

For example, the θ_1-partition of the matrix in (7) consists of two blocks:

$$\begin{matrix}[\alpha\,\underline{\beta}\,\underline{b}\;\underline{p}] \\ [\alpha\gamma\,\underline{\delta}\,\underline{p}']\end{matrix} \quad \text{and} \quad [\underline{\epsilon}\,\gamma'\,\underline{b}\,\underline{q}].$$

The common prefix of length one in the first block is $[\alpha]$ and the common prefix in the second block is $[\underline{\epsilon}]$. The θ_2- and θ_3-partitions are identical and consist of three blocks:

$$[\alpha\underline{\epsilon}\,\underline{b}\,\underline{p}], \quad [\alpha\delta\,\underline{\gamma}\,\underline{p}'] \quad \text{and} \quad [\underline{\epsilon}\,\gamma'\,\underline{b}\,\underline{q}].$$

The prefix partitions defined by θ_2 and θ_3 are finer than the prefix partition defined by θ_1. This nesting of increasingly finer prefix partitions induces a tree-like devision on any prefix stable set of terms depicted in Figure 2.

The nested partitioning of T is exploited in the inductive definition of the family $\{f_s\}$ of variable embeddings into a finite Boolean algebra. Each function f_s maps any variable α to a set of tuples that encodes the set of suffixes of α in T. s in the index of f_s is the unique prefix of α in T. That is, any f_s maps any variable α with prefix s to a tuple encoding of the submatrix $T([s\alpha])$. Prefix stability ensures that every variable of T is in the domain of exactly one function f_s.

For illustration, we consider again T of (7). For the variables γ and α, the f_s are defined in Figure 3. The encodings are defined inductively starting

$$
\begin{array}{lll}
[\alpha\ \underline{\beta}\ \underline{b}\ \underline{p}] & & f_{[]}(\alpha) = \{(\underline{\beta}, \underline{b}, \underline{p}), \\
[\alpha\ \gamma\ \underline{\delta}\underline{p}'] & f_{[\alpha]}(\gamma) = \{(\underline{\delta}, \underline{p}')\} & \qquad (\{(\underline{\delta}, \underline{p}')\}, \underline{\delta}, \underline{p}')\} \\
[\underline{\epsilon}\ \gamma'\ \underline{b}\ \underline{q}] & f_{[\underline{\epsilon}]}(\gamma') = \{(\underline{b}, \underline{q})\} &
\end{array}
$$

Figure 3: The variable encodings of (7)

with variables closest to the end of any term. We start with the variables γ and γ'. The suffix of γ is $[\underline{\delta}\underline{p}']$ and its encoding by $f_{[\alpha]}$ (because $[\alpha]$ is the prefix of γ) is the singleton set containing the tuple $(\underline{\delta}, \underline{p}')$. Similarly, the encoding of the suffix $[\underline{b}\,\underline{q}]$ of γ' by $f_{[\underline{\epsilon}]}$ is the tuple $(\underline{b}, \underline{q})$. The variable α occurs twice and the encoding of its two suffixes by $f_{[]}$ is $(\underline{\beta}, \underline{b}, \underline{p})$ and $(f_{[\alpha]}(\gamma), \underline{p}'), \underline{\delta}, \underline{p}') = (\{(\underline{\delta}, \underline{p}')\}, \underline{\delta}, \underline{p}')$.

Note the exact correspondence between $f_{[\alpha]}(\gamma) = \{(\underline{\delta}, \underline{p}')\}$ and the term $[\gamma\underline{\delta}\underline{p}']$ and also between $f_{[]}(\alpha) = \{(\underline{\beta}, \underline{b}, \underline{p}),\ (\{(\underline{\delta}, \underline{p}')\}, \underline{\delta}, \underline{p}')\}$ and the set $\alpha T(\alpha) = \{[\alpha\underline{\beta}\underline{b}\underline{p}], [\alpha\gamma\underline{\delta}\underline{p}']\}$. This correspondence is a general phenomenon. *In any condensed set T of terms there is a bijection between the encoding $f_s(\alpha)$ of any variable α with prefix s and the subset $[s\alpha]\,T([s\alpha])$ of terms of T that have prefix $[s\alpha]$* (Theorem 5.16). If each f_s is an embedding of variables into a finite structure then each block $[s\alpha]\,T([s\alpha])$ is finite and then it follows that T must also be finite.

The proof that for condensed T each f_s is an embedding (an injection), that is, if $f_s(\alpha) = f_s(\beta)$ then $\alpha = \beta$, is by contradiction. We prove (in Theorem 5.12): *If $f_s(\alpha) \subseteq f_s(\beta)$ for different variables α and β then the block $s\,T(s)$ and consequently T is not condensed*, or formulated contrapositively, *every condensed set T satisfies the antichain property: for no two different variables α and β does $f_s(\alpha) \subseteq f_s(\beta)$ hold.* The proof is by induction on the length of common prefixes starting with longer prefixes in the base case and increasingly shorter prefixes in the inductive steps.

We sketch the idea of the proof with the sample sets of Figure 4. All terms in every matrix have prefix s. Recall, we assume every term in any

$$
\begin{array}{llll}
\text{(i)} & [s\alpha_1\underline{p}] & f_s(\alpha_1) = \{\underline{p}\} & \\
 & [s\alpha_2\underline{q}] & f_s(\alpha_2) = \{\underline{q}\} & \\
 & [s\alpha_3\underline{r}] & f_s(\alpha_3) = \{\underline{r}\} & \\
 & & & \\
\text{(ii)} & [s\alpha_1\underline{p}] & f_s(\alpha_1) = \{\underline{p}, \underline{q}\} & \\
 & [s\alpha_1\underline{q}] & & \\
 & [s\alpha_2\underline{p}] & f_s(\alpha_2) = \{\underline{p}, \underline{r}\} & \\
 & [s\alpha_2\underline{r}] & & \\
 & [s\alpha_3\underline{q}] & f_s(\alpha_3) = \{\underline{q}, \underline{r}\} & \\
 & [s\alpha_3\underline{r}] & & \\
 & & & \\
\text{(iii)} & [s\alpha_1\underline{p}] & f_s(\alpha_1) = \{\underline{p}\} & \\
 & [s\alpha_2\underline{q}] & f_s(\alpha_2) = \{\underline{q}\} & \\
 & [s\alpha_3\underline{r}] & f_s(\alpha_3) = \{\underline{r}\} & \\
 & [s\alpha_4\underline{p}] & f_s(\alpha_4) = \{\underline{p}, \underline{q}\} & \\
 & [s\alpha_4\underline{q}] & & \\
 & & & \\
\text{(iv)} & [s\alpha_1\underline{\beta p}] & & f_s(\alpha_1) = \{(\underline{\beta}, \underline{p}),\ (\{\underline{q}\}, \underline{q})\} \\
 & [s\alpha_1\gamma_1\underline{q}] & f_{[s\alpha_1]}(\gamma_1) = \{\underline{q}\} & \\
 & & & \\
\text{(v)} & [s\alpha_1\underline{\beta p}] & & f_s(\alpha_1) = \{(\underline{\beta}, \underline{p}),\ (\{\underline{q}\}, \underline{q})\} \\
 & [s\alpha_1\gamma_1\underline{q}] & f_{[s\alpha_1]}(\gamma_1) = \{\underline{q}\} & \\
 & [s\alpha_2\gamma_2\underline{q}] & f_{[s\alpha_2]}(\gamma_2) = \{\underline{q}\} & f_s(\alpha_2) = \{(\{\underline{q}\}, \underline{q})\}
\end{array}
$$

Figure 4: Sample sets of terms and their variable encodings

set T has length $m+1$. The base case of the proof is when s has length $m-1$ as in (i), (ii) and (iii). The sets in (i) and (ii) are condensed and we note in both cases $f_s(\alpha_1)$, $f_s(\alpha_2)$ and $f_s(\alpha_3)$ form an antichain. For (iii) the antichain property fails, since

$$f_s(\alpha_1) \subseteq f_s(\alpha_4) \qquad \text{and} \qquad f_s(\alpha_2) \subseteq f_s(\alpha_4). \tag{8}$$

The set in (iii) is not condensed. Its condensation is that of (i) and it is obtained by applying the bindings

$$\alpha_1 \mapsto \alpha_4 \qquad \text{and} \qquad \alpha_2 \mapsto \alpha_4.$$

Note, the redundancy eliminating bindings can be read off from the subset relationships of the variable encodings in (8).

In the matrices of (iv) and (v) we assume s is shorter and has length $m-2$. (iv) is a condensation of (v). The redundancy eliminating substitution is

$$\sigma = \{\alpha_2 \mapsto \alpha_1, \gamma_2 \mapsto \gamma_1\}.$$

We have that $f_s(\alpha_2)$ is a subset of $f_s(\alpha_1)$, which means the antichain property fails to hold for (v). But, γ_1 and γ_2 do not have the same prefix, so that we cannot compare $f_{[s\alpha_1]}(\gamma_1)$ and $f_{[s\alpha_2]}(\gamma_2)$ (because the functions must have the same index). Applying $\{\alpha_2 \mapsto \alpha_1\}$ to $f_{[s\alpha_2]}(\gamma_2)$ first, then γ_1 and γ_2 have a common prefix and we see that $f_{[s\alpha_1]}(\gamma_1) = f_{[s\alpha_1]}(\gamma_2)$. In general, if $f_s(\alpha) \subseteq f_s(\beta)$ then the redundancy eliminating substitution σ for $[s\beta]\, T([s\beta])$ can be constructed by induction.

This concludes the informal preview of the results to come. We proceed as follows. First we formally define the encoding of a clause as a set or matrix of terms. Then we study variable partitions and prefix partitions and their interrelationships. And finally we define the family $\{f_s\}$ of variable embeddings and do the boundedness proofs.

5.2 The matrix term representation of clauses

Given any prefix stable clause C, we take the set T' of its terms and transform T' to a set T'' by appending blanks. Then we append to the terms in T'' special predicate constants that are uniquely associated with the heads of the literals in C to obtain a set T.

Let K be the set of constant symbols in the term representation of the input set S. That is, K is a finite non-empty set of Skolem constants, predicate constants and the blank symbol. Let $X_1, \dots, X_m$ be a sequence of pairwise disjoint non-empty sets of variables. Define $\mathcal{T}(X_1, \dots, X_m, K)$ to be the set of all terms that are built from the variables in X_i and constants in K with the restriction that

$$s \in \mathcal{T}(X_1, \dots, X_m, K) \quad \text{iff} \quad s = [u_1 \dots u_m u_{m+1}],$$

where $u_{m+1} \in K$ and for each $1 \leq i \leq m$,

$$u_i \in \begin{cases} X_i, & \text{when } u_i \text{ is a variable, and} \\ K, & \text{when } u_i \text{ is a constant.} \end{cases}$$

Observe that, each X_i contains the variables that occur in position i of any term in $\mathcal{T}(X_1, \dots, X_m, K)$, and all terms have equal length, namely $m + 1$. Let T be a subset of $\mathcal{T}(X_1, \dots, X_m, K)$. Each variable in T occurs at one fixed position in any term, but T is not necessarily prefix stable. We view T as a matrix with $m + 1$ columns and as many rows as there are terms in T. The order of the rows is arbitrary.

In the following two theorems we view T and C as multi-sets. In general, by a 'set' of terms we mean a 'multi-set' of terms. If T is condensed then it is a set.

Theorem 5.1 Let T be any set of terms with maximal length m in the basic path logic. There is a one-one correspondence between T and a set T' of terms all of length m such that the following is true.

(i) T is prefix stable iff so is T', and

(ii) T and T' have the same number of variables.

Proof. Encode each term in T of length shorter than m by appending a suitable number of blank symbols. The resulting set is

$$T' = \{[st] \mid s \in T, t = [\underline{b} \dots \underline{b}] \text{ or } t = [], \text{ and } \tau(s) + \tau(t) = m\},$$

where $\tau(t)$ is the length of a term t. Define a mapping $g : T' \longrightarrow T$ by

$$g([st]) = s \quad \text{where } t = [\underline{b} \dots \underline{b}] \text{ or } t = [] \text{ and no blanks } \underline{b} \text{ occur in } s.$$

g is a function and it is a bijection. Consequently, (i) and (ii) follow. $\square$

For a given prefix stable clause C let K_0 be a set of new constant symbols given by

$$\{\underline{p} \mid \exists s\ Ps \in C\} \cup \{\underline{p}' \mid \exists s\ \neg Ps \in C\}.$$

$\underline{p}$ and $\underline{p}'$ are the *predicate constants.* We assume K_0 is a finite set contained in K.

Theorem 5.2 Let C be any clause in the basic path logic with maximal term length m. Let $K_0 \subseteq K$ be a set of new constant symbols defined as above. Then there is a one-one correspondence between C and a set T of terms in $\mathcal{T}(X_1, \ldots, X_m, K)$ such that

(i) C is prefix stable iff so is T, and

(ii) C and T have the same number of variables.

Proof. Let T' be the set of terms occurring in C. By the previous result T' can be transformed to a set T'' of terms all with length m and there is a one-one correspondence between the two sets. Let T be the union of the following sets of terms:

$$\{[s\underline{p}] \mid Pt \in C \text{ for } g(s) = t \text{ in } T' \text{ and } s \in T''\}$$
$$\{[s\underline{p}'] \mid \neg Pt \in C \text{ for } g(s) = t \text{ in } T' \text{ and } s \in T''\}$$

s is the term with a suffix of blank symbols that corresponds to t in C. The bijection between C and T is $g' : T \longrightarrow C$ given by

$$g'([st\underline{p}]) = Ps$$
$$g'([st\underline{p}']) = \neg Ps$$

where s does not contains blanks, t is a string of blanks only or t is empty and $\underline{p}$ and $\underline{p}'$ are the predicate constants in K_0 that are associated with P and $\neg P$, respectively. T is a subset of $\mathcal{T}(X_1, \ldots, X_m, K)$ and satisfies prefix stability iff T'' does. Also, the number of variables remains unchanged. $\square$

The described transformation from clauses to sets of terms preserves prefix stability because we append the new predicate constant. Prepending the new constants does not preserve prefix stability.

5.3 Variable partitioning

The *variable partition* of a set T of terms is the finest partition into subsets of terms that do not have any variables in common. Different from Joyner Jr. (1976) ground terms will belong to separate blocks that are singleton sets. A set of terms is *variable indecomposable* (or *split*) iff its variable partition is a singleton set.

Let θ_v be a binary relation on a set T of terms defined by:

$$s\theta_v t \quad \text{iff} \quad s \text{ and } t \text{ have a variable in common or } s = t.$$

In general, θ_v is not an equivalence relation, because transitivity may fail. For example,

$$[\alpha\beta]\theta_v[\beta\gamma] \quad \text{and} \quad [\beta\gamma]\theta_v[\delta\gamma],$$

but we do not have that $[\alpha\beta]\theta_v[\delta\gamma]$. However, θ_v defined over prefix stable terms is transitive, and it is an equivalence relation.

We use the standard notation. If θ is an equivalence relation over a set T then T/θ denotes the set of *blocks* s/θ for s in T, where $s/\theta = \{t \in T \mid s\theta t\}$. T/θ is a set of pairwise disjoint sets (the blocks s/θ) that exhaust T. T/θ is said to be a *partition* of T and θ *partitions* T.

θ_v defined over terms in the path logic has the following properties:

Lemma 5.3 Let θ_v be defined on T for T a set of terms in the path logic. Let s and t be terms in T with s not ground. Then

(i) $s\theta_v t$ iff for some i, $s|_i = t|_i$ is a variable.

(ii) $s\theta_v t$ iff s and t have a common prefix $[u_1 \dots u_i]$ with u_i a variable.

(iii) θ_v is an equivalence relation.

(iv) For every s in T, s has a prefix ending with a variable that is common to all terms in s/θ_v.

If s is ground then

(v) $s\theta_v t$ iff $s = t$.

(vi) s/θ_v is a singleton set.

Proof. Suppose $s = [u_1 \ldots u_i]$ and $t = [v_1 \ldots v_j]$ are terms in T with $i \leq j$. Suppose also s is not ground.

(i) By prefix stability the common variables of s and t occur at the same position. Thus, assuming $s\theta_v t$, there is a position $k \leq i$ such $s|_k$ is a variable and $s|_k = t|_k$. The converse direction is evident.

(ii) Again, the right to left direction is evident. Suppose $s\theta_v t$. By prefix stability it follows that s and t have a common prefix and by (i) there is a common prefix that ends with a variable.

(iii) Evidently, θ_v is reflexive and symmetric. Suppose $s\theta_v t$ and $t\theta_v t'$ with $t' = [w_1 \ldots w_k]$. Then there exist two positions l_1 and l_2 such that $s|_{l_1}$ and $t|_{l_2}$ are variables and $s|_{l_1} = t|_{l_1}$ and $t|_{l_2} = t'|_{l_2}$. Suppose $l_1 \leq l_2$. By (ii) $t = [u_0 \ldots u_{l_1} v_{l_1+1} \ldots v_j]$ and $t' = [u_0 \ldots u_{l_1} v_{l_1+1} \ldots v_{l_2} w_{l_2+1} \ldots w_k]$. Evidently, $s|_{l_1} = t'|_{l_1}$ and hence $s\theta_v t'$. Therefore θ_v is also transitive and an equivalence relation.

(iv) is by (ii).

(v) and (vi) for s ground, are evident. $\square$

Theorem 5.4 Let T be a set of terms in the path logic. Then

(i) T/θ_v coincides with the variable partition of T.

(ii) Each block in T/θ_v that is not ground is uniquely associated with a term $[u_1 \ldots u_i]$ with u_i a variable that is the prefix of every term in the block.

(iii) Any ground block in T/θ_v is a singleton set.

Proof. (i) Since θ_v is an equivalence relation, T/θ_v is a partition of T. Evidently, no two different blocks in T/θ_v share a variable. Because the blocks cannot be partitioned further, $T/\theta_v = \{s/\theta_v \mid s \in T\}$ is the variable partition of T.

(ii) and (iii) are immediate by (ii) and (vi) of the above lemma. $\square$

Property (ii) is important. It says that, any block in the partition T/θ_v that is not ground, is of the form $s\,T(s)$. Property (iii) says, ground blocks also have this form, because ground blocks are singleton sets.

Corollary 5.5 Let T be a set of terms in the path logic.

(i) Each block in the variable partition has the form $s\,T(s)$. Every non-ground block is of the form $[s\alpha]\,T([s\alpha])$, for some variable α.

(ii) If T is variable indecomposable then all terms in T have a common prefix and there is a prefix s in T such that $T = s\,T(s)$.

Proof. By the previous theorem, all terms in the variable partition T/θ_v have one common prefix. □

This substantiates the claim that the variable partition of a set T may be viewed as a matrix arranged and divided as in Figure 1(a).

5.4 Prefix partitioning

For any set T of terms from $\mathcal{T}(X_1, \ldots, X_m, K)$ we define a sequence of m equivalence relations θ_i, one for each of the first m positions (columns in the matrix representation of T). In contrast to θ_v, which is determined by common variables, the relations θ_i are determined by a common prefix of length i. Compare the variable partition T/θ_v in Figure 1(a) with the prefix partition T/θ_i in Figure 1(b).

For any $1 \leq i \leq m$, we define θ_i to be the relation on T given by

(9) $$\begin{aligned} s\theta_i t \quad &\text{iff} \quad s \text{ and } t \text{ have a common prefix of length } i \\ &\text{iff} \quad s|_j = t|_j \quad \text{for every } j \leq i. \end{aligned}$$

We say a set T of terms is *prefix indecomposable* if $s/\theta_1 = T$ for every $s \in T$.

Lemma 5.6 Let T be a subset of $\mathcal{T}(X_1, \ldots, X_m, K)$. Let $\{\theta_i\}_i$ be the family of relations on T defined by (9). Then:

(i) For every i, θ_i is an equivalence relation.

(ii) For every i, T/θ_i is a partition of T.

(iii) The sequence $\theta_1, \ldots, \theta_m$ is a descending chain, that is, $\theta_m \subseteq \theta_{m-1} \subseteq \ldots \subseteq \theta_1$.

(iv) If T is prefix stable then $\theta_v \subseteq \theta_1$.

(v) If T is prefix stable and variable indecomposable then T is prefix indecomposable.

Proof. (i) and (ii) are straightforward.

(iii): By definition, $s\theta_j t$ iff for every $k \leq j$, $s|_k = t|_k$. For any $i \leq j$ we have $s|_k = t|_k$ for $k \leq i$, that is, $s\theta_i t$. Thus, for any i and j if $i \leq j$ then $\theta_j \subseteq \theta_i$.

(iv) For every non-ground terms $s, t \in T$, $s\theta_v t$ means s and t share a variable. Prefix stability for variables implies $s\theta_1 t$, that is, s and t have a common prefix of at least length one. Otherwise, for ground terms $s\theta_v t$ iff $s = t$ iff $s\theta_1 t$ by (i) and Lemma 5.3 (v).

(v) is by (iv) since $\theta_v \subseteq \theta_1$ iff $s/\theta_v \subseteq s/\theta_1$. □

Theorem 5.7 Let T be a set of terms in the path logic and let s and t be any terms occurring in T. Then

(i) The blocks s/θ_i in T/θ_i are of the form $t\,T(t)$ where t is the prefix of length i of s.

(ii) The blocks in $T(s)/\theta_i$ are of the form $t\,T([st])$ for some t.

Therefore, like in the blocks in the variable partition, the blocks in a prefix partition have the form $s\,T(s)$. A prefix partition of T by θ_i can be viewed as in Figure 1(b), where s, s', etcetera all have length i.

Lemma 5.6 proves $\theta_1, \ldots, \theta_m$ define a sequence of increasingly finer prefix partitions (recall also the example from the overview). This sequence or nesting of increasingly finer prefix partitions induces a tree-like structure on any prefix stable set of terms as depicted in Figure 2. Take a variable indecomposable set of terms $s\,T(s)$ and view it as a matrix. $T(s)$ is the sub-matrix that represents the set of suffixes of the term s in $s\,T(s)$. The next two theorems prove prefix stability and condensedness are preserved by both forming sub-matrices and prepending prefixes, and hence it follows that, the sub-matrix $T(s)$ has the same properties as $s\,T(s)$.

Theorem 5.8 Let T be a set of terms in $\mathcal{T}(X_1, \ldots, X_m, K)$. The following statements are equivalent:

(i) T is prefix stable.

(ii) For every i, every block s/θ_i in T/θ_i is prefix stable.

(iii) For every s, $s\,T(s)$ is prefix stable.

(iv) For every s, $T(s)$ is prefix stable.

(v) For every s and t, $t\,T([st])$ is prefix stable.

Proof. (i) iff (ii): Prefix stability is preserved by taking subsets and forming the union of sets of terms that do not share variables (implicit by Ohlbach 1988a), which imply the ($\Rightarrow$) and ($\Leftarrow$) directions, respectively.

(ii) iff (iii) follows from the previous result.

(iii) iff (iv): The function $f : T(s) \longrightarrow s\,T(s)$ defined by $f(t) = [st]$ is a bijection. Thus the ($\Rightarrow$) direction is straightforward and the converse direction follows since s and $T(s)$ do not share variables or constants.

(iv) iff (v) follows from (i) iff (ii). $\square$

Lemma 5.9 Let T be any set of terms. If every subset T' of T is condensed then T is condensed.

Proof. Since $T' = T$ is a special case. $\square$

The converse is not true in general. For example, both the singleton sets $\{[\alpha\beta]\}$ and $\{[\alpha\gamma]\}$ are condensed, but their union is not.

Theorem 5.10 Let T be a set of terms in the path logic. The following statements are equivalent:

(i) T is condensed.

(ii) For every s, $s\,T(s)$ is condensed.

(iii) For every s, $T(s)$ is condensed.

(iv) For every s and t, $t\,T([st])$ is condensed.

Proof. Since $t\,T([st]) = t\,(T(s))(t)$, (iii) iff (iv) follows from (i) iff (ii). (iii) implies (i), since $T = T([])$.

We prove that if $T(s)$ is not condensed then $s\,T(s)$ is not condensed, which implies T is not condensed, that is, (i) implies (ii) implies (iii). Suppose $T(s)$ is not condensed. Then there is a substitution σ such that $T(s)\sigma$ is a proper subset of $T(s)$. Since prepending a term s is order preserving we conclude $(s\,T(s))\sigma = s(T(s)\sigma)$ is a proper subset of $s\,T(s)$ and this means $s\,T(s)$ is not condensed. Since $s\,T(s)$ and $T \setminus s\,T(s)$ have no common variables (by prefix stability) we derive that

$$T\sigma = (T \setminus s\,T(s))\sigma \cup s\,T(s)\sigma = (T \setminus s\,T(s)) \cup s\,T(s)\sigma.$$

This is a proper subset of $(T \setminus s\,T(s)) \cup s\,T(s) = T$. Hence T is not condensed. $\square$

Consequently, any sub-matrix $T(s)$ is prefix stable and condensed if $s\,T(s)$ is. So, $T(s)$ can be partitioned further either by variable partitioning or prefix partitioning and each block has the form $t\,T'(t)$ where $T' = T(s)$. Again, every sub-matrix $T'(t)$ of $T(s)$ is prefix stable and condensed and may be divided further. This process of repeatedly partitioning sub-matrices has a finitely bounded number of levels, because the original matrix T has finitely bounded width. It remains to show each partition consists of a bounded number of blocks with bounded size.

5.5 The embedding of variables

The embedding of the variables occurring in any condensed and variable indecomposable set T of terms of the basic path logic is given by a family $\{f_s\}$ of functions. For technical reasons we define the f_s also for constants.

Let $T \subseteq \mathcal{T}(X_1, \ldots, X_m, K)$ be prefix stable. Define f_s recursively as follows: For every prefix $[su]$ in (any term of) T

(i) if s has length $m-1$ and $u \in X_m \cup K$ then

$$f_s(u) = \begin{cases} \{\underline{p} \mid [su\underline{p}] \in T\} & \text{if } u \text{ is a variable} \\ u & \text{otherwise, and} \end{cases}$$

(ii) if s has length $0 \leq i < m-1$ and $u \in X_{i+1} \cup K$ then:

$$f_s(u) = \{(f_{[su]}(u_{i+2}), \ldots, f_{[suu_{i+2}\ldots u_{m-1}]}(u_m), \underline{p}) \mid [suu_{i+2}\ldots u_m\underline{p}] \in T\}$$

if u is a variable, and $f_s(u) = u$ otherwise.

The restrictions of each f_s to constants are identity mappings. Restricted to variables each f_s maps $u = \alpha$ to a set of tuples that encodes the set of suffixes of α in T. s in the index of f_s is the unique prefix of α in T. For sample encodings refer back to Figure 4.

Prefix stability ensures that every variable of T is in the domain of exactly one function f_s. Denote the domain of each f_s restricted to variables by X_s. Assuming that s is a prefix of length i occurring in T then X_s is a subset of X_{i+1} with

$$\alpha \in X_s \quad \text{iff} \quad [s\alpha] \text{ is a prefix in } T.$$

In words, X_s is the set of variables of T that all have prefix s. In the base case (i) the restriction of f_s to variables is a mapping from X_s to a subset of the set K_0 of predicate constants. For example

$$f_s(\beta) = \{\underline{p}, \underline{q}\} \quad \text{for} \quad \begin{array}{l} [s\beta\underline{p}] \\ [s\beta\underline{q}] \end{array} .$$

If s has length $m-2$ as in

$$\begin{array}{l} [s\alpha\beta\underline{p}] \\ [s\alpha\beta\underline{q}] \\ [s\alpha\underline{\gamma}\underline{q}] \end{array}$$

the restriction of f_s to variables is a mapping from X_s to a subset of the product $(K \cup \mathbf{2}^K) \times K$. In this example, $f_s(\alpha) = \{(\{\underline{p}, \underline{q}\}, \underline{p}),\ (\{\underline{p}, \underline{q}\}, \underline{q}),\ (\underline{\gamma}, \underline{q})\}$. That is, the restriction of f_s to variables is a mapping

$$\begin{array}{ll} X_s \longrightarrow \mathbf{2}^K & \text{if } s \text{ has length } m-1, \\ X_s \longrightarrow \mathbf{2}^{(K \cup \mathbf{2}^K) \times K} & \text{if } s \text{ has length } m-2, \\ X_s \longrightarrow \mathbf{2}^{(K \cup \mathbf{2}^{(K \cup \mathbf{2}^K) \times K}) \times (K \cup \mathbf{2}^K) \times K} & \text{if } s \text{ has length } m-3, \end{array}$$

etcetera. In general, the restriction of f_s to variables is a mapping $X_s \longrightarrow \mathbf{2}^{B_{i+1}}$ where s is a prefix in T of length $0 \leq i < m$ and B_{i+1} is defined recursively by

$$\begin{aligned} B_m &= K \\ B_i &= (K \cup \mathbf{2}^{B_{i+1}}) \times \ldots \times (K \cup \mathbf{2}^{B_m}) \times K \quad \text{for any } 1 \leq i < m. \end{aligned}$$

Since K is finite, each $\mathbf{2}^{B_{i+1}}$ is a finite Boolean algebra. The size of each $f_s(\alpha)$ is bounded by $2^{|B_{i+1}|}$ for some i. Provided the restriction of f_s to variables is an embedding of X_s into $\mathbf{2}^{B_{i+1}}$, it follows that X_s is finite. If this is true for all s then each X_i is finite and the number of variables in T is finite and we have completed the decision proof.

Clearly, f_s is not an embedding for every T, especially not if T is not condensed, because both variables α_1 and α_2 in

$$\begin{array}{l} [s\alpha_1\underline{p}] \\ [s\alpha_2\underline{p}] \end{array}$$

for example, are mapped to $\{\underline{p}\}$. However, for condensed T each f_s restricted to variables is an embedding. Proving this is the technical part of the paper. When we say a function is an embedding we mean it is an injection.

We need the following mapping. Let α be an arbitrary variable in $X_s \subseteq X_{i+1}$ with prefix s. For any T (not necessarily condensed) define a mapping

$$h_\alpha : \alpha\, T([s\alpha]) \longrightarrow f_s(\alpha)$$

from the block of suffixes of a variable α (including the variable) to the encoding of that variable specified by

$$h_\alpha([\alpha u_{i+2} \dots u_m \underline{p}]) = (f_{[s\alpha]}(u_{i+2}), \dots, f_{[s\alpha u_{i+2} \dots u_{m-1}]}(u_m), \underline{p}).$$

That is, h_α maps any suffix $[\alpha u_{i+2} \dots u_m \underline{p}]$ beginning with α to the tuple of encodings of the u_{i+j} following α.

Lemma 5.11 Let $T \subseteq \mathcal{T}(X_1, \dots, X_m, K)$ be prefix stable. Let α be a variable in X_{i+1} with prefix s. Then h_α is a surjective function.

Proof. We readily see that h is well-defined, total and single-valued. That h is onto, that is, for any tuple $\overline{a}$ in $f_s(\alpha)$ a term t exists in $\alpha\, T([s\alpha])$ such that $h_\alpha(t) = \overline{a}$, follows from the definition of f_s. □

We will show that h_α is a bijection, provided T (or $\alpha\, T([s\alpha])$) is condensed (Theorem 5.16).

The next theorem is important as it relates condensedness to the antichain property of the variable encodings.

Theorem 5.12 Let $T \subseteq \mathcal{T}(X_1, \dots, X_m, K)$ be prefix stable. Let α and β be two different variables in T with a common prefix s. If $f_s(\alpha) \subseteq f_s(\beta)$ then there is a non-trivial substitution σ such that

$$(\alpha\, T([s\alpha]))\sigma \subseteq \beta\, T([s\beta]).$$

Proof. We construct σ by induction on the length of s. Example (iii) of Figure 4 illustrates the base case and example (v) the inductive step.

In the base case we assume s has length $m-1$ and $\alpha, \beta \in X_m$. We have

$$f_s(\alpha) = \{\underline{p} \mid [s\alpha\underline{p}] \in T\}$$
$$\alpha\, T([s\alpha]) = \{[\alpha\underline{p}] \mid \underline{p} \in f_s(\alpha)\}.$$

Let σ be the binding $\{\alpha \mapsto \beta\}$. (i) When $f_s(\alpha) \subseteq f_s(\beta)$, then $(\alpha\, T([s\alpha]))\sigma = \{[\beta \underline{p}] \,|\, \underline{p} \in f_s(\alpha)\}$ is a subset of $\beta\, T([s\beta]) = \{[\beta \underline{p}] \,|\, \underline{p} \in f_s(\beta)\}$.

The inductive hypothesis is: For any $\gamma, \delta \in X_{i+j+1}$ with a common prefix t in $\alpha\, T([s\alpha])$ and $\beta\, T([s\beta])$, respectively (with α replaced by β), if $\gamma \neq \delta$ and $f_t(\gamma) = f_t(\delta)$ then there is a substitution θ such that

$$(\gamma\, T'([t\gamma]))\theta = \delta\, T([t\delta]).$$

T' is $T\{\alpha \mapsto \beta\}$. The equality symbols instead of the inclusion symbols is not a mistake. It turns out we not need the more general condition for the inductive step.

In the inductive step we assume α, β are in X_{i+1} and the length of their common prefix s is i. Suppose $\alpha \neq \beta$ and $f_s(\alpha) \subseteq f_s(\beta)$. We aim at showing $(\alpha\, T([s\alpha]))\sigma$ for a suitable σ is a subset of $\beta\, T([s\beta])$. σ will include the binding

$$\sigma' = \{\alpha \mapsto \beta\}$$

plus other substitutions θ from the inductive hypothesis.

Given that $f_s(\alpha) \subseteq f_s(\beta)$, the sets $\alpha\, T([s\alpha])$ and $\beta\, T([s\beta])$ can be viewed as depicted in Figure 5. $\alpha\, T([s\alpha])$ is the union of the set

$$(\alpha) \qquad \{[\alpha \ldots \underline{p}] \,|\, [s\alpha \ldots \underline{p}] \in T \text{ with the `\ldots' being constants only}\}$$

and the sets

$$(\alpha\alpha) \qquad [\alpha t \gamma]\, T([s\alpha t\gamma])$$

for γ any variable following α so that at the positions between α and γ only constants occur. The term t is a string of constants only. Analogously, $\beta\, T([s\beta])$ is the union of a (β)-part of terms in which β is followed by constants only, and a $(\beta\beta)$-part of terms in which β is followed by some variable.

Because $f_s(\alpha) \subseteq f_s(\beta)$, the (α)-part of $\alpha\, T([s\alpha])$, modulo a substitution, is a subset of the (β)-part of $\beta\, T([s\beta])$. For any term $t_1 = [\alpha \ldots \underline{p}]$ in (α) there is a term $t_2 = [\beta \ldots \underline{p}]$ in $\beta\, T([s\beta])$. The dots in t_1 and t_2 denote the same string of constants. Thus, $t_1\sigma' = t_2$. If $\sigma' \subseteq \sigma$, then σ unifies the (α)-part of $\alpha\, T([s\alpha])$ with a subset of the analogous (β)-part of $\beta\, T([s\beta])$.

We now consider the sets of the form $(\alpha\alpha)$ or $(\beta\beta)$ and apply the inductive hypothesis to $\gamma\, T([s\alpha t\gamma])$. $f_{[s\alpha t]}(\gamma)$ is the set

$$\{(f_{[s\alpha t\gamma]}(u_{i+j+2}), \ldots, f_{[s\alpha t\gamma u_{i+j+2}\ldots u_{m-1}]}(u_m), \underline{p}) \,|\, [\gamma u_{i+j+2} \ldots u_m \underline{p}] \in \gamma\, T([s\alpha t\gamma])\}$$

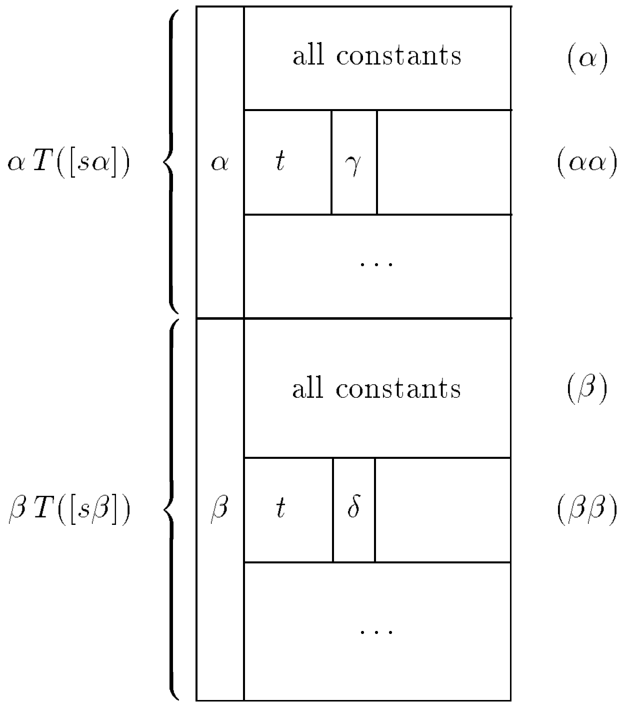


Figure 5: $\alpha\,T([s\alpha]) \cup \beta\,T([s\beta])$

and for any $(a_{i+j+2},\dots,a_m,\underline{p}) \in f_{[s\alpha t]}(\gamma)$, any

$$\overline{b} = (\alpha,\dots,f_{[s\alpha t]}(\gamma),a_{i+j+2},\dots,a_m,\underline{p})$$

that is in $f_s(\alpha)$ and is also in $f_s(\beta)$. Let F be the set of such tuples $\overline{b}$, that is,

$$F = \{(\dots,f_{[s\alpha t]}(\gamma),a_{i+j+2},\dots,a_m,\underline{p}) \in f_s(\alpha) \mid (a_{i+j+2},\dots,a_m,\underline{p}) \in f_{[s\alpha t]}(\gamma)\}.$$

Note the arguments before $f_{[s\alpha t]}(\gamma)$ are all constants, since t is a string of constants. We have

$$F \subseteq f_s(\alpha) \subseteq f_s(\beta).$$

Since h_β is onto (Lemma 5.11), for any tuple $\overline{b} \in F$, there is a term t' in $\beta\,T([s\beta])$ such that $h_\beta(t') = \overline{b}$. This means, there is a variable δ in $\beta\,T([s\beta]$ with prefix $[s\beta t]$ (as depicted in $(\beta\beta)$) such that

$$f_{[s\alpha t]}(\gamma) = f_{[s\beta t]}(\delta)$$

and $F = h_\beta([\beta t\delta]\,T([s\beta t\delta]))$. Instantiate α in T with β to get $T\sigma'$, in which then $f_{[s\beta t]}(\gamma) = f_{[s\alpha t]}(\gamma)$ and hence

$$f_{[s\beta t]}(\gamma) = f_{[s\beta t]}(\delta).$$

By the inductive hypothesis a substitution θ exists such that

$$(**) \qquad (\gamma\,(T\sigma')([s\beta t\gamma]))\theta = \delta\,(T\sigma')([s\beta t\delta]).$$

θ contains the binding $\gamma \mapsto \delta$.

Therefore, a substitution σ that is such that $(\alpha\,T([s\alpha]))\sigma \subseteq \beta\,T([s\beta])$ as required by the theorem is

$$\sigma' \cup \{\theta \mid \theta \text{ satisfies } (**) \text{ with } \gamma \text{ a variable in some } (\alpha\alpha)\text{-part of } \alpha\,T([s\alpha])\}.$$

This completes the proof. □

If there is substitution σ such that $(\alpha\,T([s\alpha]))\sigma \subseteq \beta\,T([s\beta])$ and α and β are different then the block $[s\alpha]\,T([s\alpha])$ of T is redundant, and both $s\,T(s)$ and T are not condensed. For condensed T and $s\,T(s)$ a necessary requirement is then that for every pair of different variables

$$f_s(\alpha) \not\subseteq f_s(\beta).$$

In other words:

Corollary 5.13 Let $T \subseteq \mathcal{T}(X_1, \ldots, X_m, K)$ be prefix stable. If T is condensed then the set of f_s images of X_s form an antichain (the *antichain property* of f_s).

Furthermore:

Corollary 5.14 Let $T \subseteq \mathcal{T}(X_1, \ldots, X_m, K)$ be prefix stable and condensed. Let α and β be variables in X_{i+1} with a common prefix s. Then

$$f_s(\alpha) \subseteq f_s(\beta) \quad \text{implies} \quad \alpha = \beta.$$

Proof. If $\alpha \neq \beta$ then by the previous result there is a substitution σ such that $(\alpha\, T([s\alpha]))\sigma \subseteq \beta\, T([s\beta])$ and $([s\alpha]\, T([s\alpha]))\sigma \subseteq [s\beta]\, T([s\beta])$, which implies $s\, T(s)$ is not condensed. Then by Theorem 5.10, T is also not condensed. □

The antichain property of the f_s is a necessary but not a sufficient condition for condensedness. The problem are constants. The set

$$\begin{array}{c} [s\alpha\underline{\gamma}\,\underline{p}] \\ [s\beta\delta\,\underline{p}] \end{array}$$

is not condensed but it does satisfy the antichain property, because $f_s(\alpha) = \{(\underline{\gamma}, \underline{p})\}$ and $f_s(\beta) = \{(\{\underline{p}\}, \underline{p})\}$ are not comparable. The redundancy eliminating substitution is $\{\delta \mapsto \underline{\gamma}\}$ and involves binding a constant which is not conveyed by the encoding.

A special case of Corollary 5.14 is

$$f_s(\alpha) = f_s(\beta) \quad \text{implies} \quad \alpha = \beta,$$

which says that the restriction of f_s to variables is one-one.

Corollary 5.15 Let $T \subseteq \mathcal{T}(X_1, \ldots, X_m, K)$ be prefix stable and condensed. For any prefix s occurring in T, the restriction of f_s to X_s is an embedding into $\mathbf{2}^{B_{i+1}}$.

Proof. We consider $f_s : X_s \longrightarrow \mathbf{2}^{B_{i+1}}$ with X_s non-empty. f_s is total and it is single-valued (that is, if $\alpha = \beta$ then $f(\alpha) = f(\beta)$), hence, it is a function. By the previous corollary f_s is an injection. □

Now it follows that for condensed T, h_α is an injection, too.

Theorem 5.16 Let $T \subseteq \mathcal{T}(X_1, \ldots, X_m, K)$ be prefix stable. Let α be any variable in X_{i+1} with prefix s. If T is condensed then h_α is a bijection.

Proof. In Lemma 5.11 we showed that h_α with α a variable in T (not necessarily condensed) is a surjection. Let

$$t_1 = [\alpha u_{i+2} \ldots u_m \underline{p}] \quad \text{and} \quad t_2 = [\alpha v_{i+2} \ldots v_m \underline{q}]$$

be arbitrary terms in $\alpha\, T([s\alpha])$ and assume $h_\alpha(t_1) = h_\alpha(t_2)$. This means $f_{[s\alpha]}(u_{i+2}) = f_{[s\alpha]}(v_{i+2})$, $f_{[s\alpha u_{i+2}]}(u_{i+3}) = f_{[s\alpha v_{i+2}]}(v_{i+3})$, etcetera. For every constant $\underline{\beta}$ occurring in $\alpha\, T([s\alpha])$, $f_s(\underline{\beta}) = \underline{\beta}$. This means t_1 and t_2 are equal at all positions that are filled with constants, in particular, $\underline{p} = \underline{q}$. Thus, if t_1 contains no variable, neither does t_2 and $t_1 = t_2$.

If t_1 does contain a variable, let γ be the first variable following α, that is, let

$$t_1 = [\alpha t' \gamma u_{i+j+1} \ldots u_m \underline{p}].$$

t' is a string of constants. Then

$$t_2 = [\alpha t' \delta v_{i+j+1} \ldots v_m \underline{p}]$$

for some suitable $v_{i+j+1}, \ldots, v_m$. Since $h_\alpha(t_1) = h_\alpha(t_2)$, $f_{[\alpha t']}(\gamma) = f_{[\alpha t']}(\delta)$ and this implies using Corollary 5.14 that, $\gamma = \delta$. Repeat the argument for the variables following γ.

We conclude that if $h_\alpha(t_1) = h_\alpha(t_2)$ then $t_1 = t_2$, that is, h_α is injective. $\square$

This proves there is a one-one correspondence between $\alpha\, T([s\alpha])$ and $f_s(\alpha)$, provided T is condensed. Or, if T is not condensed then there is a one-one correspondence between the condensation of $\alpha\, T([s\alpha])$ and $f_s(\alpha)$.

Corollary 5.17 Let $T \subseteq \mathcal{T}(X_1, \ldots, X_m, K)$ be prefix stable and condensed. Let α be a variable in X_{i+1} with prefix s. Then $\alpha\, T([s\alpha])$ and $f_s(\alpha)$ have the same cardinality.

Proof. h_α is a one-one function from $\alpha\, T([s\alpha])$ onto $f_s(\alpha)$. $\square$

Lemma 5.18 Let $T \subseteq \mathcal{T}(X_1, \ldots, X_m, K)$ be prefix stable and condensed. Let K be a finite set of constants. Then:

(i) For every variable α occurring in T, the size of $f_s(\alpha)$ is finitely bounded.

(ii) For every variable α occurring in T, the size of $\alpha\, T([s\alpha])$ is finitely bounded.

(iii) For every constant $\underline{\beta}$ occurring in T, the size of $\underline{\beta}\, T([s\underline{\beta}])$ is finitely bounded.

(iv) For every prefix s occurring in T, the size of $s\, T(s)$ is finitely bounded.

(v) The size of every block $u\, T(u)$ in the θ_1-partition of T is finitely bounded.

(vi) The size of every block s/θ_v in the variable partition of T is finitely bounded.

Proof. (i) K is finite and the size of $f_s(\alpha)$ is limited by $2^{|B_{i+1}|}$, for some i.

(ii) is then immediate by the previous corollary.

(iii) $\underline{\beta}\, T([s\underline{\beta}])$ can be viewed like $\alpha T([s\alpha])$ in the proof of Theorem 5.12 (see Figure 5) as the union of a $(\underline{\beta})$-part of the terms in which $\underline{\beta}$ has no variable in the suffixes, and a $(\underline{\beta}\underline{\beta})$-part of terms in which $\underline{\beta}$ does have variables in the suffixes. The size of the constant part $(\underline{\beta})$ is bounded, since K is and because T is condensed. By (ii), the size of the non-constant part $(\underline{\beta}\underline{\beta})$ is also bounded. Thus a size bound exists for $\underline{\beta}\, T([s\underline{\beta}])$.

By (ii) and (iii) a size bound exists for any $u\, T([su])$, where u a variable or a constant. $u\, T([su])$ and $[su]\, T([su])$ have the same cardinality. Thus (iv) follows.

(v) is a special case of (iv).

And, (vi) is by Lemma 5.6 (iv) and (v). □

(v) and (vi) allow us to conclude that, given that T is built from finitely many constant symbols and predicate symbols,

(i) if T is prefix indecomposable then the size of T is finitely bounded, and consequently,

(ii) if T is variable indecomposable then the size of T is finitely bounded, too.

Theorem 5.19 Let T be any variable indecomposable and condensed set of terms in the path logic in which finitely many constant symbols occur and in which the maximal term depth is finitely bounded. Then the size of T is finitely bounded.

Then there are only finitely many non-variant variable indecomposable sets of terms, and consequently:

Theorem 5.20 Let T be any condensed set of terms in the path logic in which finitely many constant symbols occur and in which the maximal term depth is finite. Then T is bounded in size.

6 Resolution for path logics and its application

It follows that $\mathcal{R}_{\text{COND}}$ with syntactic unification terminates for any finite input set $c(\varphi)$ of path clauses.

Theorem 6.1 Resolution together with condensing is a decision procedure for the satisfiability problem of a finite set of finite clauses in the basic path logic (with empty theory).

More generally:

Theorem 6.2 Any complete resolution procedure with condensing is a decision procedure for the satisfiability problem of a finite set of finite clauses in path logics, provided

(i) a term depth bound exists, and

(ii) unification is decidable.

These are the main theorems, and in what follows we discuss their applications and significance.

Theorem 6.1 is a special case of Theorem 6.2 and bears decision procedures for modal systems with empty theory $\Upsilon\Pi_f(\Sigma)$. It is immediate that *resolution and condensing provides a decision procedure for the satisfiability problem of modal formulae in K and KD.*

The functional encodings of multi-modal K and KD formulae are formulae inside a path logic extended with sorts. The sorts are used for associating path variables and constants with the different modal operators, for details see Ohlbach and Schmidt (1995). Extending the boundedness proofs for basic path logic with sorts is not difficult. Consequently, *resolution and condensing* *provides a decision procedure for the satisfiability problem of modal formulae in multi-modal K and KD.*

Standard resolution combined with condensing can be made more efficient. There are numerous refinement strategies that reduce the search space of resolution. Theorem 6.1 implies *resolution and condensing combined with* any *refinements that render a complete procedure will also be a decision procedure for basic path logic and uni- as well as multi-modal K and KD.*

Theorem 6.2 defines exactly the class of path logics (in general, with a theory) for which ordinary resolution plus condensing is a decision procedure. The conditions require an upper bound exists for the term depth of all resolvents and unification modulo the theory is decidable. Both conditions are true for path logics associated with the axioms T or B, or both. It remains to be investigated exactly which modal logics meet the conditions of the theorem. For transitive modal logics the first condition is not true, in particular, not for unrefined resolution. However, the theorem provides a partial proof of a resolution decision result for transitive modal logics, since it gives a size bound for resolvents. The proof is complete when we can exhibit a refinement strategy that fulfills the term depth bound condition.

The theorems are powerful for several reasons. One, as we have just explained, they render resolution based decision procedures for many uni-modal and multi-modal non-transitive extensions of K and KD.

Two, the theorems can be applied in many areas. They are applicable in all areas of computer science in which modal logics are used. A rich field of application for modal logic is knowledge representation based on a system called KL-ONE. *Description logics* like $\mathcal{ALC}$ (Schmidt-Schauß and Smolka 1991) are in essence multi-modal logics (Schild 1991, van der Hoek and de Rijke 1995). Hence, *resolution and condensing provides a decision procedure for the consistency problem of sentences in* $\mathcal{ALC}$.

Resolution and condensing may also be used for doing arithmetic inside the algebraic counterparts of modal logics. The algebraic versions of K and multi-modal K are Boolean algebras with unary operators (Jónsson and Tarski 1951 & 1952).

The third and very practical consequence of the theorems (in particular Theorem 6.1) is that any existing first-order theorem prover using the resolution approach is a suitable inference machine for modal reasoning, subsumption testing in description logics and arithmetic with algebraic identities. Condensing is not an operation that is explicitly present in standard theorem provers, which use subsumption deletion for eliminating redundancy.

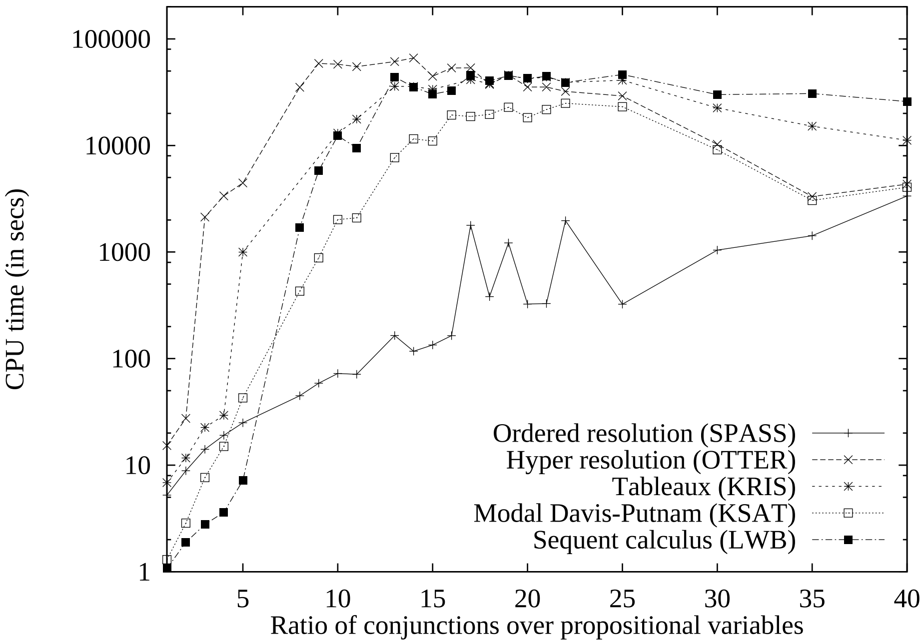


Figure 6: A comparison of the performance of different theorem provers

But, as a condensation of a clause C is a factor of C that subsumes it, condensing can be simulated by any fair implementation of factoring and subsumption and is implicit in all 'fair theorem provers'. For many modal systems no special refinement strategies of resolution (with or without a theory) are required. All we have to implement, is a front end for transforming modal formulae to their path encodings and then we can use one of many resolution based theorem provers that are available. This gives users complete freedom to fine tune their theorem prover for improved performance with any modifications of the standard resolution calculus (for example, hyper-resolution) or additional reduction rules (for example, tautology deletion) or refinement strategies (for example, ordering strategies) that are compatible with resolution and condensing, or subsumption deletion.

In practice, performance depends very much on the refinements used and the sophistication of the implementation. This is demonstrated in the experiments by Hustadt (1997). Figures 6 and 7 give the relative performances of implementations of different decision methods. The systems used are OTTER Version 3.0 (McCune 1994) with the hyper resolution setting, the prototype

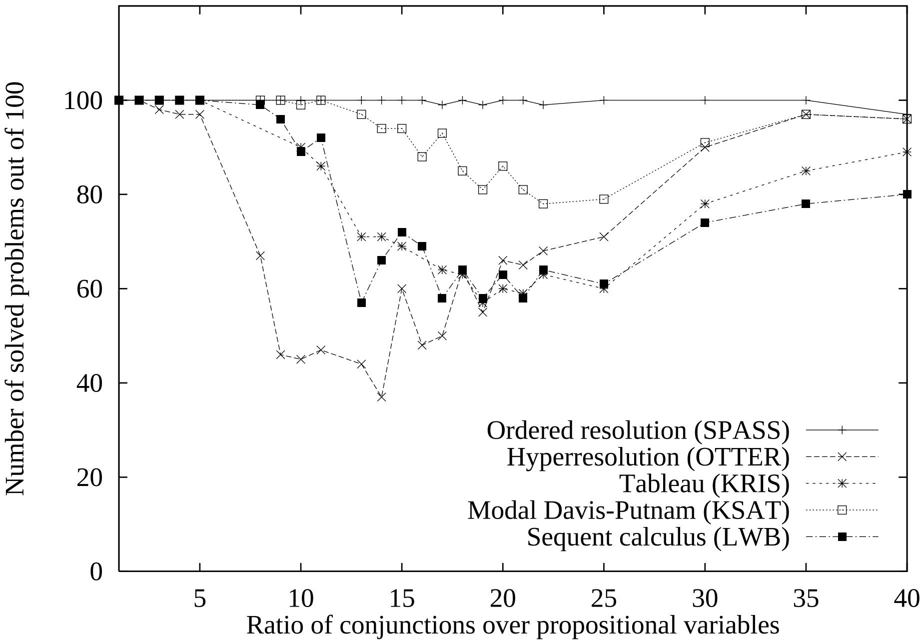


Figure 7: The number of problems solved in the cut-off time of 1000 seconds

SPASS Version 0.42 (Weidenbach, Gaede and Rock 1996) implementing standard resolution restricted by an extension of the Knuth-Bendix ordering, a system called KSAT (Giunchiglia and Sebastiani 1996a) based on a Davis-Putnam algorithm for propositional modal logic, the $\mathcal{KRIS}$ system (Baader and Hollunder 1991) developed for description logics including $\mathcal{ALC}$ which uses a tableaux method, and the Logics Workbench (LWB) (Heuerding and Schwendimann 1996) which is based on a sequent calculus.

Each point in the graph of Figure 6 gives the accumulated CPU runtime for testing the satisfiability of one hundred K-formulae of the same complexity. The formulae are in a normal form generated at random according to the scheme of Giunchiglia and Sebastiani (1996a, 1996b). The fixed parameters are: five propositional variables, three disjuncts in any clause and modal degree two. The number of conjunctions vary which is reflected along the horizontal axis in the ratio of the number of conjunctions over the number of propositional variables. This ration provides a measure for the complexity of the different sets of formulae.

The runtime values are plotted in logarithmic scale along the vertical axis and give the total CPU time for the reasoning process on one hundred formulae. Excluded from the measurement are: the transformation of the modal formulae to the respective languages of the theorem provers, doing obvious simplifications that throw out trivial tautologies, and the conversion of the first-order formulations to clausal form for OTTER and SPASS. Problems not solved within the time-limit of 1000 seconds are not penalised and contribute 1000 seconds to the total runtime. No theorem prover completes all problems within the time-limit. Some do better than others as we see in Figure 7.

The benchmarks indicate the superiority of a state of the art theorem prover, like SPASS, utilising the optimised functional translation method over all special purpose modal theorem provers for K that we are aware of.

7 Concluding remarks

There are a number of open ends. A pertinent question is, which logics in the hierarchy of propositional normal modal logics that are decidable can be embedded in the class of path logics and satisfy the conditions of Theorem 6.2.

An issue not addressed in this paper is computational complexity. The space bound for condensed clauses implicit in the encoding by f_s is non-elementary. Whether an improved space bound can be found that is exponential or even polynomial is open. A starting point for investigations in this direction is the result by Sperner (1928) on the maximal size of antichains in finite power set algebras. Alternatively, a more focussed study of the deduction steps of resolution and their effect on variables may yield a space bound that is closer to the optimal polynomial space bound. The complexity of refined resolution procedures which are more efficient is of particular interest.

There are interesting decidable extensions of propositional modal logic, which have not been explored in the functional framework, for example, extensions with many-dimensional modal operators like the operators of Tarskian set constraints and extensions with nominals in modal logic, or constants (ABox elements) in description logics.

References

Auffray, Y. and Enjalbert, P. (1992), Modal theorem proving: An equational viewpoint, *J. Logic Computat.* **2**(3), 247–297.

Baader, F. and Hollunder, B. (1991), $\mathcal{KRIS}$: Knowledge representation and inference system: System description, *Technical Memo TM-90-03*, DFKI, Kaiserslautern.

Bull, R. A. and Segerberg, K. (1984), Basic modal logic, *in* D. Gabbay and F. Guenther (eds), *Handbook of Philosophical Logic*, Vol. II, Reidel, Dordrecht, pp. 1–88.

Chellas, B. F. (1980), *Modal Logic: An Introduction*, Cambridge Univ. Press.

Fariñas Del Cerro, L. and Herzig, A. (1988), Linear modal deductions, *in* E. Lusk and R. Overbeek (eds), *Proc. CADE'88*, Vol. 310 of *Lecture Notes in Computer Science*, Springer, pp. 487–499.

Fariñas Del Cerro, L. and Herzig, A. (1995), Modal deduction with applications in epistemic and temporal logics, *in* D. M. Gabbay, C. J. Hogger and J. A. Robinson (eds), *Handbook of Logic in Artificial Intelligence and Logic Programming: Epistemic and Temporal Reasoning*, Vol. 4, Clarendon Press, Oxford, pp. 499–594.

Fermüller, C., Leitsch, A., Tammet, T. and Zamov, N. (1993), *Resolution Method for the Decicion Problem*, Vol. 679 of *Lecture Notes in Computer Science*, Springer.

Fine, K. (1975), Normal forms in modal logic, *Notre Dame J. Formal Logic* **16**, 229–237.

Fitting, M. (1983), *Proof Methods for Modal and Intuitionistic Logics*, Vol. 169 of *Synthese Library*, Reidel, Dordrecht.

Giunchiglia, F. and Sebastiani, R. (1996a), Building decision procedures for modal logics from propositional decision procedures: The case study of modal K, *in* M. A. McRobbie and J. K. Slaney (eds), *Automated Deduction: CADE-13*, Vol. 1104 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 583–597.

Giunchiglia, F. and Sebastiani, R. (1996b), A SAT-based decision procedure for alc, *in* L. C. Aiello, J. Doyle and S. Shapiro (eds), *Proc. KR'96*, Morgan-Kaufmann, pp. 304–314.

Gottlob, G. and Fermüller, C. G. (1993), Removing redundancy from a clause, *Artificial Intelligence* **61**, 263–289.

Herzig, A. (1989), *Raisonnement automatique en logique modale et algorithmes d'unification.*, PhD thesis, Univ. Paul-Sabatier, Toulouse.

Heuerding, A. and Schwendimann, S. (1996), On the modal logic K plus theories, *in* H. Kleine Büning (ed.), *Proc. CSL'95*, Vol. 1092 of *Lecture Notes in Computer Science*, Springer, pp. 308–319.

Hughes, G. E. and Cresswell, M. J. (1984), *A Companion to Modal Logic*, Methuen, London.

Hustadt, U. (1997), Resolution-based decisison procedures for subclasses of first-order logic. Unpublished PhD thesis, Univ. Saarlandes, Germany.

Jónsson, B. and Tarski, A. (1951 & 1952), Boolean algebras with operators, Part I & II, *Amer. J. Math.* **73 & 74**, 891–939 & 127–162.

Joyner Jr., W. H. (1976), Resolution strategies as decision procedures, *J. ACM* **23**(3), 398–417.

McCune, W. (1994), OTTER 3.0 reference manual and guide, *Technical Report ANL-94/6*, Argonne National Lab., Argonne, IL.

Mints, G. (1986), A resolution method for non-classical logics, *Semiotika and Informatika* **25**, 120–135.

Mints, G. (1989), Resolution calculi for modal logics, *Amer. Math. Soc. Transl.* **143**, 1–14.

Mints, G. (1990), Gentzen-type systems and resolution rules. Part I: Propositional logic, *Proc. COLOG-88*, Vol. 417 of *Lecture Notes in Computer Science*, Springer, pp. 198–231.

Ohlbach, H. J. (1988a), *A Resolution Calculus for Modal Logics*, PhD thesis, Univ. Kaiserslautern, Germany.

Ohlbach, H. J. (1988b), A resolution calculus for modal logics, *in* E. Lusk and R. Overbeek (eds), *Proc. CADE'88*, Vol. 310 of *Lecture Notes in Computer Science*, Springer, pp. 500–516.

Ohlbach, H. J. (1991), Semantics based translation methods for modal logics, *J. Logic Computat.* **1**(5), 691–746.

Ohlbach, H. J. and Schmidt, R. A. (1995), Functional translation and second-order frame properties of modal logics, *Technical Report MPI-I-95-2-002*, Max-Planck-Institut für Informatik, Saarbrücken, Germany. To appear in *J. Logic Computat.*

Rautenberg, W. (1983), Modal tableau calculi and interpolation, *J. Philos. Logic* **12**, 403–423.

Schild, K. (1991), A correspondence theory for terminological logics: Preliminary report, *Proc. IJCAI'91*, pp. 466–471.

Schmidt-Schauß, M. and Smolka, G. (1991), Attributive concept description with complements, *Artificial Intelligence* **48**, 1–26.

Sperner, E. (1928), Ein Satz über Untermengen einer endlichen Menge, *Math. Z.* **27**, 544–548.

van der Hoek, W. and de Rijke, M. (1995), Counting objects, *J. Logic Computat.* **5**(3), 325–345.

Weidenbach, C., Gaede, B. and Rock, G. (1996), SPASS & FLOTTER, version 0.42, *in* M. A. McRobbie and J. K. Slaney (eds), *Automated Deduction: CADE-13*, Vol. 1104 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 141–145.

Zamov, N. K. (1989), Modal resolutions, *Soviet Math.* **33**(9), 22–29. Translated from *Izv. Vyssh. Uchebn. Zaved. Mat.* **9** (328) (1989) 22–29.

Index of notation

Subject index

INFORMATIK

Below you find a list of the most recent technical reports of the research group *Logic of Programming* at the Max-Planck-Institut für Informatik. They are available by anonymous ftp from our ftp server `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Birgit Hofmann
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-97-2-001	D. Basin, S. Matthews, L. Viganò	Labelled modal logics: Quantifiers
MPI-I-96-2-010	A. Nonnengart	Strong Skolemization
MPI-I-96-2-009	D. Basin, N. Klarlund	Beyond the Finite in Automatic Hardware Verification
MPI-I-96-2-007	A. Herzig	SCAN and Systems of Conditional Logic
MPI-I-96-2-006	D. Basin, S. Matthews, L. Vigano	Natural Deduction for Non–Classical Logics
MPI-I-96-2-005	A. Nonnengart	Auxiliary Modal Operators and the Characterization of Modal Frames
MPI-I-96-2-004	G. Struth	Non-Symmetric Rewriting
MPI-I-96-2-003	H. Baumeister	Using Algebraic Specification Languages for Model-Oriented Specifications
MPI-I-96-2-002	D. Basin, S. Matthews, L. Vigano	Labelled Propositional Modal Logics: Theory and Practice
MPI-I-96-2-001	H. Ganzinger, U. Waldmann	Theorem Proving in Cancellative Abelian Monoids
MPI-I-95-2-011	P. Barth, A. Bockmayr	Modelling Mixed-Integer Optimisation Problems in Constraint Logic Programming
MPI-I-95-2-010	D. A. Plaisted	Special Cases and Substitutes for Rigid E-Unification
MPI-I-95-2-009	L. Bachmair, H. Ganzinger	Ordered Chaining Calculi for First-Order Theories of Binary Relations
MPI-I-95-2-008	H. J. Ohlbach, R. A. Schmidt, U. Hustadt	Translating Graded Modalities into Predicate Logic
MPI-I-95-2-007	A. Nonnengart, A. Szalas	A Fixpoint Approach to Second-Order Quantifier Elimination with Applications to Correspondence Theory
MPI-I-95-2-006	D. Basin, H. Ganzinger	Automated Complexity Analysis Based on Ordered Resolution