MAX-PLANCK-INSTITUT FÜR INFORMATIK

A Multi-Dimensional Terminological Knowledge Representation Language

Preliminary Version

Franz Baader & Hans Jürgen Ohlbach

MPI–I–93–212 August 1993

Im Stadtwald
W 66123 Saarbrücken
Germany

Author's Address

Franz Baader
Lehr- und Forschungsgebiet Theoretische Informatik,
RWTH Aachen,
Ahornstraße 55,
D-52074 Aachen,
email:baader@informatik.rwth-aachen.de

Hans Jürgen Ohlbach
Max–Planck–Institut für Informatik
Im Stadtwald
D-6600 Saarbrücken 11
F. R. Germany
email:ohlbach@mpi-sb.mpg.de

Publication Notes

A short version of this report also appears in the proceedings of the International Conference on Artificial Intelligence (IJCAI) 1993 in Chambery, France.

Acknowledgements

This work was supported by the ESPRIT project 3125 MEDLAR, and the BMFT Projects Logo (ITS 9102) and TACOS (ITW 9201).

Abstract

An extension of the concept description language $\mathcal{ALC}$ used in KL-ONE-like terminological reasoning is presented. The extension includes multi-modal operators that can either stand for the usual role quantifications or for modalities such as belief, time etc. The modal operators can be used at all levels of the concept terms, and they can be used to modify both concepts and roles. This is an instance of a new kind of combination of modal logics where the modal operators of one logic may operate directly on the operators of the other logic.

Contents

1 Introduction

Knowledge representation languages in the style of KL-ONE [2], so-called terminological KR languages, can be used to define the relevant concepts of a problem domain and the interaction between these concepts. To this purpose, complex concepts are constructed out of atomic concepts (i.e., unary predicates) and roles (i.e., binary predicates) with the help of the language constructs provided by the particular terminological language. Various such languages have been considered in the literature and are used in KR systems (see, e.g., [12, 14, 13, 15, 6, 1, 3, 17]).

They have in common that they are only suitable for representing objective, time-independent facts about the world. Notions like belief, intentions, time—which are essential for systems that model aspects of intelligent agents—can only be represented in a very limited way. Suppose that a terminological system should represent that the agent John believes that new cars have catalytic converters whereas Tom believes that they don't. One possibility—which has, e.g., been used in SB-ONE [10]—is that the system keeps two separate terminologies, one for John's belief context and one for Tom's belief context:

John's T-Box:
$\mathit{new\text{-}car} = \mathit{car} \sqcap \exists \mathit{has\text{-}part}\colon \mathit{catalytic\text{-}converter}$

Tom's T-Box:
$\mathit{new\text{-}car} = \mathit{car} \sqcap \neg\exists \mathit{has\text{-}part}\colon \mathit{catalytic\text{-}converter}$

This does not work, however, when interactions between different beliefs, e.g., in the sense of a (modal) theory of belief are needed in the application. Modal operators of the form $[\mathit{belief}\text{-}...]$[1] that satisfy appropriate modal axioms allow for more natural definitions:

$$[\mathit{belief\text{-}John}](\mathit{new\text{-}car} = \mathit{car} \sqcap \exists \mathit{has\text{-}part}\colon \mathit{catalytic\text{-}converter}),$$
$$[\mathit{belief\text{-}Tom}](\mathit{new\text{-}car} = \mathit{car} \sqcap \neg\exists \mathit{has\text{-}part}\colon \mathit{catalytic\text{-}converter}).$$

Things become more complex if the application requires the use of modalities inside of concept expressions as well. Assume that we want to express that a potential customer (for a car salesman) is an adult who eventually wants to own a car. In a traditional terminological language a definition of this concept could be

$$\mathit{potential\text{-}customer} = \mathit{adult} \sqcap \exists \mathit{eventually\text{-}wants\text{-}own}\colon \mathit{car},$$

where *eventually-wants-own* is a new role different to the roles *own* and *wants-own*. But then there would be no interaction between these roles, whereas one would expect that *wants-own* implies

[1] The standard modal operators are usually written $\Box$ and $\Diamond$. In a multi-modal logic with different modal operators referring to different accessibility relations we write $[p]$ and $\langle p \rangle$ for the parameterized box and diamond operators. These operators can be interpreted as 'believes,' 'knows,' 'wants,' 'always' (in the future or past) and the like.

eventually-wants-own. Again, modal operators with an appropriate modal theory of time and belief can be used to capture such interactions. In our example, we get the definition

$$\textit{potential-customer} = \textit{adult} \sqcap \exists(\langle \textit{future} \rangle [\textit{want}] \textit{own}) \colon \textit{car}.$$

Intuitively, the role-fillers for *own* now also depend on the point in time and on the intentional world, and not only on the object. The prefix $[\textit{want}]$ means that one takes only those objects that are role-fillers in all accessible intentional worlds, and the prefix $\langle \textit{future} \rangle$ says that this has to be evaluated at some future time point.

In this example, the modal operators modify the *own*-role. Of course, there are also cases where one would like to modify concepts in this way. In the definition

$$\textit{environment-freak} = \textit{person} \ \sqcap \forall([\textit{want}]\textit{own}) \colon [\textit{belief}]\textit{environment-friendly},$$

an environment freak is defined as a person that wants to own only things that are believed to be environment friendly. In this case the $[\textit{belief}]$-operator modifies the concept *environment-friendly*.

We have motivated by examples that it is desirable to extend terminological languages by various types of modal operators (for time, belief, want, etc.), which should be applicable to definitions as well as inside of concept terms, and there to modify both concepts and roles. Our approach to achieve this goal is based on an observation by Schild [15] that the terminological language $\mathcal{ALC}$ is just a syntactic variant of the multi-modal logic K_m [4], where m is the number of different box operators. The reason is that quantifications over roles can just be seen as applications of parameterized modal operators to concepts. Thus we propose to treat roles and modalities in a symmetric way by using a multi-modal logic where both role names and modalities such as belief can be used as parameters in boxes and diamonds. To distinguish between roles, which operate on objects, and modalities operating, e.g., on time points or intentional worlds, we shall equip each modal operator with a type (or dimension) such as 'object,''time-point' etc.

However, the requirement that it should be possible to modify roles by modal operators is not yet captured by this approach. Until now, the parameterized modal operators are atomic in the sense that the boxes and diamonds may only contain parameter names like *own*, *future*, or *belief*. Applying modal operators to roles means that one obtains complex terms inside boxes and diamonds. For example, in the definition of *environment-freak* we thus get the modal prefix $[[\textit{want}]\textit{own}]$ where the complex (role) term $[\textit{want}]\textit{own}$ occurs inside a box-operator.

Our new approach for integrating modalities into terminological KR languages, called $\mathcal{M}$-$\mathcal{ALC}$ in the following, thus extends the prototypical terminological language $\mathcal{ALC}$ in two respects. First, 'roles' may have different types that express in what dimension (e.g., object-dimension, time-point-dimension) they operate. In addition, one can apply role quantification not only to concepts but also to roles, which provides for a very expressive language for building role terms. The expressive power of this language is, for example, demonstrated by the fact that general concept equations (see, e.g., [15]) can be expressed, even if one has only one dimension. This shows that the important inference problems (such as satisfiability of concept terms) must be of very high complexity for our language.[2] For this reason we shall impose additional restrictions on the syntactic form of certain role terms to get a practical algorithm for satisfiability of concept terms.

2 Syntax and Semantics of $\mathcal{M}$-$\mathcal{ALC}$

As for $\mathcal{ALC}$, the elementary syntax elements in $\mathcal{M}$-$\mathcal{ALC}$ are concept and role names. However, with each role name we associate a type that expresses in what dimension (e.g., object-dimension, time-point-dimension) this role operates. To simplify things, we assume there are n different dimensions, and we count them from 1 to n. Each dimension corresponds to a particular set (domain, universe). For example, the *object* dimension corresponds to the set of all objects (as used in $\mathcal{ALC}$), the *time* dimension corresponds to the set of all time points, and the *belief* dimension corresponds to the set of all belief worlds. In the present paper, however, we do not yet consider structures on these sets; for example, time points are not assumed to be linearly ordered, and the belief worlds are not

[2] Satisfiability modulo concept equations is known to be exp-time complete; this is an easy consequence of a result by Fischer and Ladner [7].

assumed to satisfy certain belief axioms. This means that the underlying logic is simply the basic modal logic K.

The syntactic form of a modal operator in $\mathcal{M}$-$\mathcal{ALC}$ is $[p]$ (box) or $\langle p\rangle$ (diamond) where p may be an atomic role name or a compound role term. In addition to the usual box-operator of modal logic we shall consider a modified box-operator $[...]^+$ that combines the the box- and diamond-operator. In many cases, $[...]^+$ makes more sense than the usual box-operator $[...]$. For example, a sentence 'All her friends are wealthy' is usually understood in the sense that the person in question really has friends. Thus it is better modelled by the expression $[\textit{has-friend}]^+\textit{wealthy}$ than by $[\textit{has-friend}]\textit{wealthy}$.

Definition 2.1 *The* signature Σ *of an* $\mathcal{M}$-$\mathcal{ALC}$ *language* $\mathcal{L}_n$ *of dimension* $n > 0$ *consists of a set* Σ_P *of* role names *and a disjoint set* Σ_C *of* concept names. *The concept names include the distinguished symbols* $\top$ *(for 'truth') and* $\bot$ *(for 'falsity'). Each role name* p *has a* dimension $\dim(p)$, *which is a positive integer* $\leq n$.

The sets of role terms *and* modal operators *is defined to be the least set such that*

- *Each role name* p *is a role term (of dimension* $\dim(p)$*). In addition,* $[p]$, $[p]^+$ *and* $\langle p\rangle$ *are modal operators of dimension* $\dim(p)$.
- *If* p *is a role name of dimension* i *and* m *is a sequence of modal operators then* $m\ p$ *is a role term of dimension* i, *and* $[m\ p]$, $[m\ p]^+$ *as well as* $\langle m\ p\rangle$ *are modal operators of dimension* i.

The notation $[...]^*$ *will be used for the* $[...]^+$*-operator as well as for the normal* $[...]$*-operator.*

The set of concept terms *is defined to be the least set such that*

- *Each concept name* c *is a concept term.*
- *If* c *and* d *are concept terms then* $\neg c$, $c \vee d$ *and* $c \wedge d$ *are concept terms.*
- *If* p *is a role term (of arbitrary dimension) and* c *is a concept term then* $[p]c$, $[p]^+c$ *and* $\langle p\rangle c$ *are concept terms.*

A concept equation *is a formula* $m\ (c = d)$ *where* c *is a concept name,* d *a concept term, and* m *a modal prefix, i.e., a (possibly empty) sequence of box and diamond operators. A* T-Box *is a set of concept equations.*

A concept term is called restricted serial *if role terms do not contain normal* $[...]$*-operators.*

Even for a single dimension, $\mathcal{M}$-$\mathcal{ALC}$ is an extension of $\mathcal{ALC}$ since one can build complex role terms. This allows one to express interactions between roles that cannot be captured in $\mathcal{ALC}$. For example, the concept of a 'woman all of whose children have a common favourite meal' is expressible by $\textit{woman} \wedge \langle[\textit{has-child}]\textit{likes}\rangle\textit{meal}$.

The semantics of $\mathcal{M}$-$\mathcal{ALC}$ is similar to the Kripke style possible worlds semantics for many-dimensional modal logic [16]. For each dimension i we introduce a non-empty set D_i. The elements of $D_1 \times \ldots \times D_n$ correspond to worlds in the modal logic sense. As in modal logic, there is always an 'actual world tuple' $\vec{d} = (d_1, \ldots, d_n)$ that determines the interpretation of the syntax elements.

Since the domain consists of n-tuples, concept terms correspond to n-ary predicates. If, for example, there are the two dimensions *object* and *time* then the extension of the concept term *car* is a set of tuples (o, t). Another way of looking at this is that *car* corresponds to a subset of *objects*, but depending on the time point, i.e., the set of things that are cars changes over the time. Conversely one may also see *car* as a set of time points, and this set depends on the objects. This yields the time points (lifespan) for which each object is considered as a car.

Accordingly, roles in $\mathcal{M}$-$\mathcal{ALC}$ correspond to $n + 1$-ary predicates. For example, the role *own* of dimension object is not simply a binary relation between *objects*. For a given object o, the set of role-fillers for this object will depend not only on o but also, say, on the actual belief world and time point.

In the definition of the semantics of $\mathcal{M}$-$\mathcal{ALC}$ we shall use the following notational conventions. For a fixed number of dimensions n, the Cartesian product of the sets $D_1, \ldots, D_n$ is denoted by $\vec{D}$. An element $(d_1, \ldots, d_n)$ of $\vec{D}$ is denoted by $\vec{d}$, and $(d_1, \ldots, d_{i-1}, x, d_{i+1}, \ldots, d_n)$ by $\vec{d}[i/x]$.

Definition 2.2 *An interpretation $\Im = (\vec{D}, \Im_\Sigma)$ for an n-dimensional $\mathcal{M}$-$\mathcal{ALC}$ language $\mathcal{L}_n$ consists of the Cartesian product $\vec{D} = D_1 \times \ldots \times D_n$ of n non-empty carrier sets (domains), and a signature interpretation $\Im_\Sigma$.*

The signature interpretation assigns successor functions to role names and n-place relations to concept names. To be more precise, a role name p of dimension i is interpreted as a function $\Im_\Sigma(p) : \vec{D} \to 2^{D_i}$, and a concept name c as a set $\Im_\Sigma(c) \subseteq \vec{D}$. The concept name $\top$ is interpreted as $\vec{D}$ and $\bot$ as the empty set.

The interpretation of a role term q of dimension i is also a function $\Im(q) : \vec{D} \to 2^{D_i}$ that is inductively defined as follows. Let p be a role name, q, r be role terms, and let j be the dimension of q.

$$\begin{array}{lcl} \Im(p)(\vec{d}) & = & \Im_\Sigma(p)(\vec{d}), \\ \Im([q]r)(\vec{d}) & = & \bigcap_{x \in \Im(q)(\vec{d})} \Im(r)(\vec{d}[j/x]), \\ \Im(\langle q \rangle r)(\vec{d}) & = & \bigcup_{x \in \Im(q)(\vec{d})} \Im(r)(\vec{d}[j/x]), \\ \Im([q]^+ r)(\vec{d}) & = & \Im([q]r)(\vec{d}) \cap \Im(\langle q \rangle r)(\vec{d}). \end{array}$$

A satisfiability relation $\models$ between an interpretation $\Im$ with actual point $\vec{d}$ and concept terms and concept equations is defined as follows. Let c be a concept name, e, f be concept terms, p be a role term of dimension i, and ϕ be a concept term or concept equation.

$$\begin{array}{ll} \Im, \vec{d} \models c & \textit{iff } \vec{d} \in \Im_\Sigma(c) \\ \Im, \vec{d} \models c = e & \textit{iff } (\Im, \vec{d} \models c \textit{ iff } \Im, \vec{d} \models e) \\ \Im, \vec{d} \models \neg e & \textit{iff not } \Im, \vec{d} \models e \\ \Im, \vec{d} \models e \vee f & \textit{iff } \Im, \vec{d} \models e \textit{ or } \Im, \vec{d} \models f \\ \Im, \vec{d} \models e \wedge f & \textit{iff } \Im, \vec{d} \models e \textit{ and } \Im, \vec{d} \models f \\ \Im, \vec{d} \models [p]\phi & \textit{iff for } \text{all } x \in \Im(p)(\vec{d}) : \Im, \vec{d}[i/x] \models \phi \\ \Im, \vec{d} \models \langle p \rangle \phi & \textit{iff for } \text{some } x \in \Im(p)(\vec{d}) : \Im, \vec{d}[i/x] \models \phi \\ \Im, \vec{d} \models [p]^+ \phi & \textit{iff } \Im, \vec{d} \models [p]\phi \textit{ and } \Im, \vec{d} \models \langle p \rangle \phi \end{array}$$

An interpretation $\Im$ is a model of a T-Box iff for all *actual points $\vec{d}$ and all equations ϕ of the T-Box one has $\Im, \vec{d} \models \phi$.*

It should be noted that with respect to this semantics, concept equations are treated in the same way as in terminological languages, i.e., they are required to hold for all actual points. This differs from the usual definition of models in modal logics where a formula is only required to hold at one point (world). Only the characteristic axioms of the particular modal system are required to hold at all points. To really capture both cases one would need a more flexible definition of a model where the elements of some dimensions (e.g., objects) are treated as universally quantified, whereas the objects of the remaining dimensions (e.g., belief worlds) are (implicitly) assumed to be existentially quantified. For the case of two dimensions (objects and intentional worlds for a know-operator), equations are treated in this way in [11]. However, there the modal operator for 'know' may only occur in front of equations, but not inside of concept terms. Since our concept and role terms already have a very complex structure we shall concentrate on the concept term level, and leave the treatment of T-Boxes—with a possibly more flexible definition of a model—as a topic of further research.

The basic reasoning services every KL-ONE system provides are to decide whether a given concept term denotes the empty set or not (*satisfiability problem*) and whether one concept term always denotes a subset of another concept term (*subsumption problem*).

Definition 2.3 *The concept term e subsumes the concept term f iff, for all interpretations $\Im$ and actual points $\vec{d}$, $\Im, \vec{d} \models f$ implies $\Im, \vec{d} \models e$.*

The concept term e is satisfiable iff there is an interpretation $\Im$ and an actual point $\vec{d}$ such that $\Im, \vec{d} \models e$.

Since e subsumes f iff $f \wedge \neg e$ is unsatisfiable, it is sufficient to have an algorithm for the satisfiability problem.

3 The Satisfiability Test

In $\mathcal{ALC}$ a subsumption algorithm for concept terms is fully sufficient for computing the concept hierarchy in T-Boxes. The reason is that the concept equations are interpreted universally, and that the T-Boxes are deterministic and cycle free. That means no pairs $c = d$ and $c = d'$ with $d \neq d'$ are in the T-Box, and no chains $c_1 = d_1[c_2], \ldots, c_n = d_n[c_1]$ are in the T-Box. With this restriction, all defined concepts in a concept term can be expanded with their definition prior to the subsumption test.

In the general case where modal concept equations $m(c = d)$ either hold everywhere, or at some point only, or at one point in some dimensions and everywhere in others, this is no longer possible. A quite complex specification and control mechanism for the application of concept equations would be necessary, which is out of the scope of this paper. Therefore we only present a satisfiability algorithm for concept terms (which is in fact a general theorem prover for the modal logic $\mathcal{M}$-$\mathcal{ALC}$).

The algorithm we shall present works only for the restricted serial case where no [...]-operators occur in the role terms. The following example demonstrates the expressive power the unrestricted language has. Assume that we have only one dimension, and that the object o is an element of the concept term $[[p]q]c \wedge [p]\bot$. The term $[p]\bot$ (which is also allowed in the restricted case) forces the set of p-fillers of o to be empty. This in turn means that o is connected with all objects of the domain by the role term $[p]q$ (which is not allowed in the restricted case). Because o is also an element of $[[p]q]c$ this implies that all objects have to be in c. This shows that concept terms of $\mathcal{M}$-$\mathcal{ALC}$ can be used to simulate general concept equations of the form $c = \top$ where c is a complex concept term. As mentioned in the introduction, general concept equations are very hard to handle algorithmically. If $\mathcal{M}$-$\mathcal{ALC}$ terms are assumed to satisfy the seriality restriction concept equations can no longer be expressed. In the following we assume that *all concept terms we consider are restricted serial.*

In the satisfiability algorithm we assume that all concept terms are in negation normal form, i.e., negation symbols occur only in front of the atomic names. The following rules transform a given concept term into an equivalent term in negation normal form:

$$
\begin{array}{llll}
\neg\top & \rightarrow \bot & \neg\bot & \rightarrow \top \\
\neg\neg\phi & \rightarrow \phi & \neg[p]\phi & \rightarrow \langle p\rangle\neg\phi \\
\neg(\phi \vee \psi) & \rightarrow \neg\phi \wedge \neg\psi & \neg\langle p\rangle\phi & \rightarrow [p]\neg\phi \\
\neg(\phi \wedge \psi) & \rightarrow \neg\phi \vee \neg\psi & \neg[p]^{+}\phi & \rightarrow \langle p\rangle\neg\phi \vee [p]\bot
\end{array}
$$

In order to be as close to the semantics as possible, we write the satisfiability test as a labelled deductive system [8]. The control structure, however, is a tableau expansion. Each data item is a pair $\vec{l} : \phi$ where $\vec{l} = (l_1, \ldots, l_n)$ consists of constant symbols generated during the expansion of the tableau. The label $\vec{l}$ is the syntactic counterpart of the 'actual world tuple' $\vec{d}$ in an interpretation. That means $\vec{l} : \phi$ describes ϕ in the context $\vec{l}$. Here ϕ may be a concept term or a role term with an argument. The expression $\vec{l} : p : a$ for example means that a is a p-successor of $l_{dim(p)}$. The constraint systems that are generated by our satisfiability algorithm will always have a specified initial label $\vec{l_0}$. The algorithm calls itself recursively with 'negated' role terms $\overline{p}$ in role constraints. The intended interpretation of $\overline{p}$ is simply as complement: $\Im(\overline{p})(\vec{d}) = D_{dim(p)} \setminus \Im(p)(\vec{d})$. Since we must avoid [...]-operators in the role terms, the rules for building negation normal form are more complex for role terms than they are for concept terms. These rules are not applied in a preprocessing step. Instead, they are integrated into the satisfiability algorithm.

The rules for building negation normal forms of role terms generate so-called *role terms with negation*, which are slightly more general than simple negated role terms: if p and q are role

terms (without negation) then $\overline{p}$, $[p]^+\overline{q}$, and $\langle p\rangle\overline{q}$ are role terms with negation. The dimension of a role term with negation is just the dimension of the corresponding role term without negation: $dim(\overline{p}) = dim(p)$, $dim([p]^+\overline{q}) = dim([p]^+q)$, and $dim(\langle p\rangle\overline{q}) = dim(\langle p\rangle q)$. In the following, the expression 'role term' will always mean a role term with or without negation. Since the negation occurs only in a very restricted setting in such terms (in particular, not inside of box or diamond operators), allowing for such terms does not mean that we introduce general role negation.

More formally, the rules of the satisfiability algorithm work on so-called constraint systems, which are defined as follows.

Definition 3.1 *For an n-dimensional $\mathcal{M}$-$\mathcal{ALC}$ language, constraints are built from constant symbols (*points*), n-tuples of constant symbols (*labels*), and role and concept terms. Each constant symbol has a dimension between* 1 *and* n*, and it may occur in a label as* i*-th component only if its dimension is* i*.*

A role constraint *is a triple $\vec{l} : p : a$, where $\vec{l}$ is a label, p is a role term (with or without negation), and a is a point of dimension $dim(p)$. A* concept constraint *is a tuple $\vec{l} : c$ consisting of a label $\vec{l}$ and a concept term c. A* constraint system *is a set of role constraints and concept constraints.*

The constraints in one system will be interpreted conjunctively (i.e., all of them must be satisfied to satisfy the whole system), whereas sets of constraint systems will be interpreted disjunctively (i.e., only one of the systems must be satisfied to satisfy the whole set of systems).

The algorithm depends on a function d that, for a given point in a constraint system, measures its distance from the initial label $\vec{l}_0$, i.e., it counts with how many atomic steps $\vec{l} : p : x$ or $\vec{l} : \overline{p} : x$ (where p is a role name) it can be reached from the initial label. For general constraint systems, the notion of depth may depend on the path chosen to reach the point, and it may even be undefined if the point cannot be reached from the initial label.

Definition 3.2 *For a role term p we define $|p|$ to be the number of occurrences of role names in p. For a constraint set Γ with initial label $\vec{l}_0 = (l_{01}, \ldots, l_{0n})$ we define*

- $d(l_{0i}, \Gamma) = 0$ *for* $i = 1, \ldots, n$.
- $d(\vec{l}, \Gamma) = \max_{1 \le i \le n} d(l_i, \Gamma)$.
- *If* $n = d(\vec{l}, \Gamma)$ *is already defined, and* $\vec{l} : p : b \in \Gamma$ *is selected by some selection function for* b *then* $d(b, \Gamma) = n + |p|$.

It will be shown (see Lemma 4.2 below) that for the constraint systems generated by the satisfiability algorithm, this definition determines for every point and label in the system a unique depth (i.e., one that is independent of the selection function). We are now ready to formulate the algorithm. Because of the presence of disjunction in our language we will actually have to consider finite sets of constraint systems.

Definition 3.3 *The satisfiability algorithm takes as input a restricted serial $\mathcal{M}$-$\mathcal{ALC}$ concept term c_0 in negation normal form. It then constructs an initial label $\vec{l}_0$ and builds the constraint system $\{\vec{l}_0 : c_0\}$. Then it calls the function* apply-rules *with the singleton set $\Delta_0 = \{\{\vec{l}_0 : c_0\}\}$. This function iteratively applies the rules of Figure 1 to the constraint systems already obtained. It returns 'unsatisfiable' if the $\emptyset$-rule applies, and 'satisfiable' if the $\top$-rule applies.*

A small example shall illustrate how the algorithm works. In the one-dimensional case, the concept terms $[\langle p\rangle q]c$ and $[p][q]c$ are equivalent. We prove that the second term subsumes the first by applying the satisfiability algorithm to $[\langle p\rangle q]c \wedge \neg[p][q]c$.

The initial constraint set for this input term contains the constraint $l_0 : [\langle p\rangle q]c \wedge \langle p\rangle\langle q\rangle\neg c$. By an application of the $\wedge$-rule one obtains the two constraints

$$(1)\ l_0 : [\langle p\rangle q]c \quad \text{and} \quad (2)\ l_0 : \langle p\rangle\langle q\rangle\neg c.$$

In the rules below, c and d are concept terms, ϕ is either a concept term or a role term like $p : a$, and $i = dim(p)$. The suffix '&Γ' stands for the unmodified part of the constraint system under consideration, and '& Δ' stands for the other constraint systems currently not under consideration.

$(\wedge)$ $\{\vec{l} : c \wedge d \;\&\; \Gamma\} \;\&\; \Delta \quad \rightarrow \quad \{\vec{l} : c \wedge d, \vec{l} : c, \vec{l} : d \;\&\; \Gamma\} \;\&\; \Delta$
if not both $\vec{l} : c$ and $\vec{l} : d$ are in Γ.

$(\vee)$ $\{\vec{l} : c \vee d \;\&\; \Gamma\} \;\&\; \Delta \quad \rightarrow \quad \{\vec{l} : c \vee d, \vec{l} : c \;\&\; \Gamma\}, \{\vec{l} : c \vee d, \vec{l} : d \;\&\; \Gamma\} \;\&\; \Delta$
if neither $\vec{l} : c$ nor $\vec{l} : d$ is in Γ.

$(\overline{[]^+})$ $\{\vec{l} : \overline{[p]^+ q} : a \;\&\; \Gamma\} \;\&\; \Delta \quad \rightarrow \quad \{\vec{l} : \overline{[p]^+ q} : a, \vec{l} : \langle p \rangle \overline{q} : a \;\&\; \Gamma\},$
$\{\vec{l} : \overline{[p]^+ q} : a, \vec{l} : [p]\bot \;\&\; \Gamma\} \;\&\; \Delta$
if neither $\vec{l} : \langle p \rangle \overline{q} : a$ nor $\vec{l} : [p]\bot$ is in Γ.

$(\overline{\langle\rangle})$ $\{\vec{l} : \overline{\langle p \rangle q} : a \;\&\; \Gamma\} \;\&\; \Delta \quad \rightarrow \quad \{\vec{l} : \overline{\langle p \rangle q} : a, \vec{l} : [p]^+ \overline{q} : a \;\&\; \Gamma\},$
$\{\vec{l} : \overline{\langle p \rangle q} : a, \vec{l} : [p]\bot \;\&\; \Gamma\} \;\&\; \Delta$
if neither $\vec{l} : [p]^+ \overline{q} : a$ nor $\vec{l} : [p]\bot$ is in Γ.

$(\langle\rangle)$ $\{\vec{l} : \langle p \rangle \phi \;\&\; \Gamma\} \;\&\; \Delta \quad \rightarrow \quad \{\vec{l} : \langle p \rangle \phi, \vec{l} : p : a, \vec{l}[i/a] : \phi \;\&\; \Gamma\} \;\&\; \Delta$
$\{\vec{l} : [p]^+ \phi \;\&\; \Gamma\} \;\&\; \Delta \quad \rightarrow \quad \{\vec{l} : [p]^+ \phi, \vec{l} : p : a, \vec{l}[i/a] : \phi \;\&\; \Gamma\} \;\&\; \Delta$
if $\vec{l} : p : b, \vec{l}[i/b] : \phi$ is not in Γ for some b.
Here a is assumed to be a new constant of dimension i.

$([]^*)$ $\{\vec{l} : [p]^* \phi, \;\&\; \Gamma\} \;\&\; \Delta \quad \rightarrow \quad \{\vec{l} : [p]^* \phi, \vec{l}[i/a] : \phi \;\&\; \Gamma\} \;\&\; \Delta$
if a is a constant of dimension i, $d(\vec{l}, \Gamma) + |p| = d(a, \Gamma) =: n$, $\vec{l}[i/a] : \phi \notin \Gamma$,
and *apply-rules*$(\{\{\vec{l'} : q : b \mid d(b, \Gamma) \leq n\} \cup \{\vec{l} : \overline{p} : a)\}\})$ = 'unsatisfiable'

$(\bot)$ $\{\vec{l} : c, \vec{l} : \neg c \;\&\; \Gamma\} \;\&\; \Delta \quad \rightarrow \quad \Delta$
$\{\vec{l} : p : a, \vec{l} : \overline{p} : a \;\&\; \Gamma\} \;\&\; \Delta \quad \rightarrow \quad \Delta$
$\{\vec{l} : \bot \;\&\; \Gamma\} \;\&\; \Delta \quad \rightarrow \quad \Delta$

$(\top)$ $\Gamma \;\&\; \Delta \quad \rightarrow$ 'satisfiable'
if none of the other rules is applicable to Γ

$(\emptyset)$ $\emptyset \quad \rightarrow$ 'unsatisfiable'

Figure 1: Transformation rules of the satisfiability algorithm for $\mathcal{M}$-$\mathcal{ALC}$.

The $\langle\rangle$-rule, applied to (2), adds the constraints

$$(3)\ l_0 : p : a \quad \text{and} \quad (4)\ a : \langle q \rangle \neg c.$$

The $\langle\rangle$-rule, applied to (4), adds

$$(5)\ a : q : b \quad \text{and} \quad (6)\ b : \neg c.$$

Now $d(b, \Gamma) = 2 = d(l_0, \Gamma) + |\langle p \rangle q|$. For this reason, we have to make a recursive call of the function *apply-rules* to determine whether the $[]^*$-rule fires for (1) and b.

In this recursive call we consider the constraints (3) and (5) together with the new negated role constraint $l_0 : \overline{\langle p \rangle q} : b$. An application of the $\overline{\langle\rangle}$-rule yields two new systems, one with the additional constraint

$$(7)\ l_0 : [p]^+ \overline{q} : b,$$

and the other with the additional constraint $l_0 : [p]\bot$. In the following we restrict our attention to the first system. (The alternative $l_0 : [p]\bot$ also leads to 'unsatisfiable.') Now the $[]^*$-rule becomes applicable for (7) and a. In fact, the recursive call of *apply-rules* with the constraint (3) and the new negated role constraint (8) $l_0 : \overline{p} : a$ immediately returns unsatisfiable. Application of the

$[]^*$-rule for (7) and a yields the new constraint (9) $a : \overline{q} : b$, which clashes with (5). Thus the first recursive call also yields unsatisfiable.

This shows that in our original system the $[]^*$-rule can fire for (1) and b. From this we get (10) $b : c$, and thus a clash with (6).

4 Proof of Termination and Soundness

First, it will be shown that, for constraint systems generated by the algorithm, the depth function is always uniquely defined (i.e., independent of the selection function). In addition, the depth of all points and labels is bounded by a positive integer, which depends linearly on the size of the input term.

To define an appropriate bound on the depth of all points and labels occurring in a constraint system generated by the algorithm, we extend the notion of length of a role term to concept terms and expressions of the form $p : a$.

Definition 4.1 *For concept names c and role names p we define $|c| := |p| := 1$. Now assume that a is a point, q is a role term without negation, e, f are concept terms, and ϕ is a concept term or an expression of the form $q : a$.*

1. $|\neg c| := |c|$ *and* $|\overline{q}| := |q|$.
2. $|e \vee f| := |e \wedge f| := \max\{|e|, |f|\}$.
3. $|[q]^*\phi| := |\langle q\rangle\phi| := |p| + |\phi|$.
4. $|q : a| := |q|$.

For a constraint system Γ and a label $\vec{l}$ in Γ we define

$$m(\vec{l}, \Gamma) := \max\{|\phi| \mid \vec{l} : \phi \in \Gamma\}$$

as the length of $\vec{l}$ in Γ.

Lemma 4.2 *Let c_0 be a concept term of length $|c_0| = n_0$, and assume that the function apply-rules is called with the singleton set $\{\Gamma_0\}$, where $\Gamma_0 = \{\vec{l_0} : c_0\}$.*

1. *The depths of points and labels is uniquely defined, and it remains invariant during the execution of the function* apply-rules.
2. *For any constraint system Γ generated during the execution of* apply-rules *and any label $\vec{l}$ in Γ, we have $d(\vec{l}, \Gamma) + m(\vec{l}, \Gamma) \leq n_0$.*

Proof: by induction on the number of rule applications.

In the initial state there is only the label $\vec{l_0}$, and no points other than the ones occurring in $\vec{l_0}$. For these the first part is obviously true: their depth is uniquely defined to be zero, and thus $d(\vec{l_0}, \Gamma_0) = 0$. In addition, we have $m(\vec{l_0}, \Gamma_0) = |c_0| = n_0$, which shows that the second part of the lemma is true as well.

Obviously, an application of the $\wedge$- or $\vee$-rule does not change the depth of points and labels, and it is easy to see that this also holds for the length of labels.

The two rules dealing with negated role terms are also unproblematic. We restrict our attention to the $\overline{[]^+}$-rule. (The other rule can be treated analogously.) In the first alternative, the $\overline{[]^+}$-rule introduces a new role constraint $\vec{l} : \langle p\rangle\overline{q} : a$. Since the system already contains the constraint $\vec{l} : \overline{[p]^+q} : a$, which satisfies $|\overline{[p]^+q}| = |\langle p\rangle\overline{q}|$, this additional constraint neither changes the depth of a nor the length of $\vec{l}$. In the second alternative, the rule adds the constraint $\vec{l} : [p]\bot$. This new

constraint does not change the length of $\vec{l}$. The reason is that the system already contains the constraint $\vec{l} : \overline{[p]^+q} : a$, which satisfies $|\overline{[p]^+q}| \geq |p| + 1 = |[p]\bot|$.

Thus the only remaining cases in the induction step are the $\langle\rangle$- and the $[]^*$-rules. Assume that Γ' is obtained from Γ by application of such a rule.

(1) First, let us consider the $\langle\rangle$-rule. In this case, we may without loss of generality assume that Γ contains $\vec{l} : \langle p\rangle\phi$ and $\Gamma' = \Gamma \cup \{\vec{l} : p : a, \vec{l}[i/a] : \phi\}$ where $i = dim(p)$. (The case where Γ contains $\vec{l} : [p]^+\phi$ can be treated in exactly the same way.) The interesting case is where ϕ is a role constraint, i.e., ϕ is of the form $q : a'$. (The case where ϕ is a concept constraint is similar, but easier.)

(1.1) To prove the first part of the lemma, we show that in Γ' the old constants (i.e., constants different from a) have a unique depth, which is identical to their depth in Γ. This is done by induction on the depth of these constants in Γ.

The only constants of depth 0 in Γ are the components of the tuple $\vec{l_0}$. In fact, any other constant b must have been introduced by a $\langle\rangle$-rule. Thus there exists a role constraint of the form $\vec{g} : r : b$ in Γ, which shows that $d(b, \Gamma) \geq |r| \geq 1$. For the components of $\vec{l_0}$ we have the unique depth $d(\vec{l_{0i}}, \Gamma') = 0 = d(\vec{l_{0i}}, \Gamma')$ by definition, and this coincides with the depth they have in Γ.

Let b be a constant of depth greater than 0. First, assume that $b \neq a'$. We know that there exists a role constraint $\vec{g} : r : b$, and

$$(*) \quad d(b, \Gamma) = d(\vec{g}, \Gamma) + |r|.$$

In addition, for any other role constraint $\vec{g'} : r' : b \in \Gamma$, we also have

$$(**) \quad d(b, \Gamma) = d(\vec{g'}, \Gamma) + |r'|,$$

by our induction assumption on Γ. From the equations $(*)$ and $(**)$ one can deduce that the components of $\vec{g}$ and $\vec{g'}$ have a depth (in Γ) that is smaller than the one of b. For this reason, we know $d(\vec{g}, \Gamma) = d(\vec{g}, \Gamma')$ and $d(\vec{g'}, \Gamma) = d(\vec{g'}, \Gamma')$ by induction. This, together with the equations $(*)$ and $(**)$ shows that in Γ' both role constraints give us the same depth for b, namely the one b already had in Γ. Since we have assumed $b \neq a'$, the system Γ' does not contain new role constraints for b.

Now assume that $b = a'$. As above, one can show that the old role constraints for a' provide us with the same depth for a' as in Γ. Note that the constraint $\vec{l} : \langle p\rangle q : a' \in \Gamma$ shows that $d(a', \Gamma) = d(\vec{l}, \Gamma) + |p| + |q|$.

It remains to be shown that the new constraint $\vec{l}[i/a] : q : a'$ does not give a different result. To determine the depth induced by this constraint, we have to find out what the depth of the new constant a is in Γ'. There is exactly one role constraint for a in Γ', namely $\vec{l} : p : a$. Because $\vec{l} : \langle p\rangle q : a' \in \Gamma$, we know that the components of $\vec{l}$ have a smaller depth than a' in Γ. By induction, we thus know that $d(\vec{l}, \Gamma') = d(\vec{l}, \Gamma)$ is uniquely defined. But then $d(a, \Gamma') = d(\vec{l}, \Gamma') + |p| = d(\vec{l}, \Gamma) + |p|$ is also uniquely defined. Obviously, this implies that $d(\vec{l}[i/a], \Gamma') = d(a, \Gamma')$ since a is the component of maximal depth in $\vec{l}[i/a]$.

The constraint $\vec{l}[i/a] : q : a'$ induces for a' in Γ' the depth $d(\vec{l}[i/a], \Gamma') + |q|$. However, we know that this is equal to $d(a, \Gamma') + |q| = d(\vec{l}, \Gamma') + |p| + |q| = d(\vec{l}, \Gamma) + |p| + |q| = d(a', \Gamma)$, as was to be shown.

(1.2) Now let us turn to the second part of the lemma. First, we consider the length of old labels, i.e., labels not containing a. Obviously, $\vec{l}$ is the only such label having an additional constraint in Γ'. But this new constraint, $\vec{l} : p : a$, cannot increase the length of $\vec{l}$ since Γ already contains $\vec{l} : \langle p\rangle\phi$, and $|\langle p\rangle\phi| \geq |p| = |p : a|$. Thus, for all old labels $\vec{g}$ we have $m(\vec{g}, \Gamma') = m(\vec{g}, \Gamma)$, and since the depth of these labels also coincides in Γ and Γ' (by (1.1)), the second part of the lemma is satisfied for $\vec{g}$ by induction.

The only new label in Γ' is $\vec{l}[i/a]$, and its only constraint is $\vec{l}[i/a] : \phi$. By induction, we know

$$n_0 \geq m(\vec{l}, \Gamma) + d(\vec{l}, \Gamma),$$

and in (1.1) we have seen that

$$d(\vec{l}[i/a], \Gamma') = d(a, \Gamma') = d(\vec{l}, \Gamma') + |p| = d(\vec{l}, \Gamma) + |p|.$$

Since Γ contains the constraint $\vec{l} : \langle p \rangle \phi$, we obtain

$$m(\vec{l}, \Gamma) \geq |\langle p \rangle \phi| = |p| + |\phi|,$$

and since $\vec{l}[i/a] : \phi$ is the only constraint for $\vec{l}[i/a]$, we also have

$$m(\vec{l}[i/a], \Gamma') = |\phi|.$$

These facts imply that $n_0 \geq m(\vec{l}, \Gamma) + d(\vec{l}, \Gamma) \geq |p| + |\phi| + d(\vec{l}, \Gamma) = d(\vec{l}[i/a], \Gamma') + m(\vec{l}[i/a], \Gamma')$, as was to be shown.

(2) Second, we consider an application of the $[]^*$-rule. In this case, Γ contains the constraint $\vec{l} : [p]^* \phi$ and $\Gamma' = \Gamma \cup \{\vec{l}[i/a] : \phi\}$. Since the rule applies to a, we know that $d(a, \Gamma) = d(\vec{l}, \Gamma) + |p|$.

(2.1) To prove the first part of the lemma, we show by induction on the depth of constants in Γ that all constants have the same depth in Γ' as they had in Γ. Again, the constants of depth 0 in Γ and Γ' are exactly the components of the tuple $\vec{l}_0$.

The only new constraint in Γ' is $\vec{l}[i/a] : \phi$. If ϕ is not a role constraint, the depth of labels and constants is obviously not changed by its introduction. Thus assume that $\phi = q : a'$. Since Γ contains the constraint $\vec{l} : [p]^* q : a'$ we have $d(a', \Gamma) = d(\vec{l}, \Gamma) + |p| + |q|$. Since we also know that $d(a, \Gamma) = d(\vec{l}, \Gamma) + |p|$ this shows that a is of smaller depth than a' in Γ. For this reason, we already know by induction that $d(a, \Gamma') = d(a, \Gamma) = d(\vec{l}, \Gamma) + |p| = d(\vec{l}, \Gamma') + |p|$. In particular, this means that $d(\vec{l}[i/a], \Gamma') = d(\vec{l}, \Gamma) + |p|$.

The constraint $\vec{l}[i/a] : q : a'$ thus gives us $d(\vec{l}, \Gamma) + |p| + |q|$ as depth of a' in Γ'. This is just the depth of a' in Γ. It is easy to see that the role constraints for a' that were already present in Γ yield the same depth for a' in Γ' as they did in Γ.

(2.2) Finally, we show that the second part of the lemma holds in this case as well. Since the only new constraint in Γ' is $\vec{l}[i/a] : \phi$, the only label for which the length can change is $\vec{l}[i/a]$. If this does not happen, i.e., if $m(\vec{l}[i/a], \Gamma) = m(\vec{l}[i/a], \Gamma')$, then the second part of the lemma follows by induction and (2.1).

Thus assume that $m(\vec{l}[i/a], \Gamma) < m(\vec{l}[i/a], \Gamma')$. This means that the length $m(\vec{l}[i/a], \Gamma')$ must be equal to $|\phi|$. From (2.1) we know that $d(\vec{l}[i/a], \Gamma') = d(\vec{l}, \Gamma) + |p|$. Together, these to facts yield $m(\vec{l}[i/a], \Gamma') + d(\vec{l}[i/a], \Gamma') = |\phi| + |p| + d(\vec{l}, \Gamma)$. Since $\vec{l} : [p]^* \phi$ is a constraint in Γ, we also know that $m(\vec{l}, \Gamma) \geq |\phi| + |p|$. Thus, we know $m(\vec{l}[i/a], \Gamma') + d(\vec{l}[i/a], \Gamma') \leq m(\vec{l}, \Gamma) + d(\vec{l}, \Gamma)$, and by induction, this is smaller or equal n_0, which concludes the proof of the lemma. □

We have to show that the system Γ'' for which the satisfiability algorithm is recursively called in the $[]^*$-rule satisfies the lemma as well.

Lemma 4.3 *Let c_0 be a concept term of length $|c_0| = n_0$, and assume that the function apply-rules is called with the singleton set $\{\Gamma_0\}$ where $\Gamma_0 = \{\vec{l}_0 : c_0\}$. Let Γ be a constraint system generated during the execution of apply-rules. Let $\vec{l} : [p]^* \phi \in \Gamma$, and a be a constant in Γ with $d(a, \Gamma) = n = d(\vec{l}, \Gamma) + |p|$. We consider the system*

$$\Gamma'' := \{\vec{l}' : q : b \mid d(b, \Gamma) \leq n\} \cup \{\vec{l} : \overline{p} : a)\}.$$

1. *The depth of all constants in Γ'' is uniquely defined, and it coincides with their depth in Γ.*
2. *$d(\vec{l}, \Gamma'') + m(\vec{l}, \Gamma'') \leq n_0$.*

Proof: by induction on the depth of the constants in Γ. The components of $\vec{l}_0$ are the only constants of depth 0 in Γ. By definition, they also have depth 0 in Γ''.

Let b be a constant of depth greater than 0 in Γ. There is a constraint $\vec{g} : q : b$ in Γ, and $d(b, \Gamma) = d(\vec{g}, \Gamma) + |q|$. If b occurs in Γ'' then this depth is not larger than n. For this reason, the components of $\vec{g}$ are of depth smaller than n, and thus also occur in Γ''. By induction, we know $d(\vec{g}, \Gamma'') = d(\vec{g}, \Gamma)$. This shows that the constraint $\vec{g} : q : b$ yields the same depth for b in Γ'' as it did in Γ.

If $b \neq a$, we are finished (since the above argument applies to any constraint of the form $\vec{g} : q : b$ that is both in Γ and Γ''). Otherwise, we have to take into account that for a we have the new constraint $\vec{l} : \overline{p} : a$. But this gives us the depth $d(\vec{l}, \Gamma'') + |p|$ for a. By induction, we know that $d(\vec{l}, \Gamma'') = d(\vec{l}, \Gamma)$, and by our assumption on a, $d(\vec{l}, \Gamma) + |p|$ is the depth of a in Γ. Thus we have proved the first part of the lemma.

For the second part, we note that for $\vec{g} \neq \vec{l}$ we have $m(\vec{g}, \Gamma'') \leq m(\vec{g}, \Gamma)$. This is so because $\vec{g}$ does not have more constraints in Γ than it has in Γ''. For $\vec{l}$, we also have $m(\vec{l}, \Gamma'') \leq m(\vec{l}, \Gamma)$. In fact, the only new constraint is $\vec{l} : \overline{p} : a$, and we know $m(\vec{l}, \Gamma) \geq |p|$ (since Γ contains the constraint $\vec{l} : [p]^* \phi$). □

The fact that labels are of bounded depth plays an important role in the proof of termination.

Theorem 4.4 *The satisfiability algorithm described above terminates.*

Before we can prove the theorem we have to introduce some notation. We say that a constraint system Γ' is a *descendant* of the constraint system Γ iff one of the following two conditions holds:

- Γ' is obtained from Γ by applying the $\wedge$-, $\vee$-, $\overline{[]^+}$-, $\overline{\langle\rangle}$-, $\langle\rangle$-, or $[]^*$-rule.
- Γ is a system for which applicability of the $[]^*$-rule for a constraint $\vec{l} : [p]^* \phi$ is tested by recursively calling the function *apply-rules* with input Γ'.

Assume that the algorithm does not terminate. It is easy to see that this implies the existence of an infinite chain $\Gamma_0, \Gamma_1, \ldots$ of constraint systems such that Γ_{i+1} is a descendant of Γ_i. To prove that this leads to a contradiction, we define a mapping Ψ of constraint systems into a set Q, which will be equipped with a well-founded strict partial ordering $\gg$. Since the ordering is well-founded, there cannot be an infinitely decreasing chain $\Psi(\Gamma_0) \gg \Psi(\Gamma_1) \gg \cdots$ of constraint systems. Thus it will be sufficient to show that $\Psi(\Gamma) \gg \Psi(\Gamma')$ whenever Γ' is a descendant of Γ.

The elements of the set Q will have a rather complex structure. They are 2-tuples where the first component is a nonnegative integer. The second component is a finite multiset of 4-tuples. Each component of these 4-tuples is either a finite multiset of nonnegative integers (for the third and fourth component) or a nonnegative integer (for the first and second component). Multisets are like sets, but allow for multiple occurrences of identical elements. For example, $\{2, 2, 2\}$ is a multiset that is distinct from the multiset $\{2\}$. A given ordering on a set T can be used to define an ordering on the finite multisets over T. In this ordering, a finite multiset M is larger than a finite multiset M' iff M' can be obtained from M by replacing one or more elements in M by any finite number of elements taken from T, each of which is smaller than one of the replaced elements. For example, $\{2, 2, 2\}$ is larger than $\{2\}$ and $\{2, 2, 1, 1, 0\}$. In [5] it is shown that the induced ordering on finite multisets over T is well-founded if the original ordering on T is so.

The nonnegative integer components of our 4-tuples are compared with respect to the usual ordering on integers, and the finite multiset components by the multiset ordering induced by this ordering. The whole 4-tuples are ordered lexicographically from left to right, i.e., $(c_1, ..., c_4)$ is larger than $(c'_1, ..., c'_4)$ iff there exists $i, 1 \leq i \leq 4$, such that $c_1 = c'_1, ..., c_{i-1} = c'_{i-1}$, and c_i is larger than c'_i. Since the orderings on the components are well-founded, the lexicographical ordering on the tuples is well-founded as well. Finite multisets of these tuples are now compared with respect to the multiset ordering induced by this lexicographical ordering. Finally, the 2-tuples consisting of a nonnegative integer in the first component, and a multiset of 4-tuples in the second component are again ordered lexicographically from left to right. This is the well-founded ordering $\gg$ on Q mentioned above.

Before we can define the mapping Ψ from constraint systems to elements of Q, we need some more notation. For a concept term c in negation normal form we denote the number of ' $\wedge$ ' and

'$\vee$' operators occurring in c by $oas(c)$. For a role term p with negation, we denote the length of the over-lined part of p by $nol(p)$.

Definition 4.5 *Let c_0 be a concept term of length $|c_0| = n_0$, and assume that the function apply-rules is called with the singleton set $\{\{\vec{l}_0 : c_0\}\}$. Let Γ be a constraint system generated during the execution of apply-rules. Then $\Psi(\Gamma) := (\Psi_1(\Gamma), \Psi_2(\Gamma))$. The first component of this 2-tuple is defined as*

$$\Psi_1(\Gamma) := \max\{d(\vec{l}, \Gamma) + m(\vec{l}, \Gamma) \mid \vec{l} \text{ is a label occurring in } \Gamma\}.$$

The second component, $\Psi_2(\Gamma)$, is the multiset that contains for each label $\vec{l}$ occurring in Γ a 4-tuple $\psi(\vec{l}, \Gamma)$ defined as follows:

1. *The first component of $\psi(\vec{l}, \Gamma)$ is the integer $k_1 := n_0 - d(\vec{l}, \Gamma)$. By Lemma 4.2, k_1 is well-defined and nonnegative.*
2. *The second component of $\psi(\vec{l}, \Gamma)$ is the number of constraints $\vec{g} : [p]^*\phi$ in Γ that satisfy*
   - $\vec{l} = \vec{g}[i/a]$,
   - $d(a, \Gamma) = d(\vec{g}, \Gamma) + |p|$, *and*
   - $\vec{l} : \phi \notin \Gamma$.

   Note that in this case $d(\vec{l}, \Gamma) = d(a, \Gamma) > d(\vec{g}, \Gamma)$.
3. *The third component of $\psi(\vec{l}, \Gamma)$ is the multiset that consists of all numbers $oas(c \wedge d)$ (resp. $oas(e \vee f)$, $nol(\overline{p})$) such that $\vec{l} : c \wedge d$ (resp. $\vec{l} : e \vee f$, $\vec{l} : \overline{p} : a$) is in Γ, and the $\wedge$-rule (resp. $\vee$-rule, $\overline{\langle\rangle}$- or $\overline{[]^+}$) is still applicable to this constraint.*
4. *The fourth component of $\psi(\vec{l}, \Gamma)$ is the multiset that consists of all numbers $|p|$ such that $\vec{l} : \langle p \rangle \phi$ or $\vec{l} : [p]^+\phi$ is in Γ, and the $\langle\rangle$-rule is still applicable to this constraint.*

Lemma 4.6 *Assume that Γ is a system for which applicability of the $[]^*$-rule for a constraint $\vec{l} : [p]^*\phi$ is tested by recursively calling the function apply-rules with input Γ'. Then $\Psi_1(\Gamma) \gg \Psi_1(\Gamma')$, and thus $\Psi(\Gamma) \gg \Psi(\Gamma')$.*

Proof: We have $\vec{l} : [p]^*\phi \in \Gamma$, and a constant a in Γ such that $d(a, \Gamma) = n = d(\vec{l}, \Gamma) + |p|$. The system Γ' is defined as

$$\Gamma' = \{\vec{g} : q : b \mid d(b, \Gamma) \leq n\} \cup \{\vec{l} : \overline{p} : a\}.$$

First, we show that for any label $\vec{g}$ in Γ' we have $d(\vec{g}, \Gamma') + m(\vec{g}, \Gamma') \leq n$. Let $\vec{g} : q : b$ be the constraint for which $|q : b| = m(\vec{g}, \Gamma')$.

Assume that $\vec{g}$ is different from $\vec{l}$. By definition of Γ' we know $d(b, \Gamma) \leq n$, and in the proof of Lemma 4.3 we have seen that $d(b, \Gamma) = d(b, \Gamma')$. In addition, the presence of $\vec{g} : q : b$ in Γ' shows that $d(b, \Gamma') = d(\vec{g}, \Gamma') + |q|$. Thus $d(\vec{g}, \Gamma') + m(\vec{g}, \Gamma') = d(\vec{g}, \Gamma') + |q : b| = d(\vec{g}, \Gamma') + |q| = d(b, \Gamma') = d(b, \Gamma) \leq n$.

If $\vec{g} = \vec{l}$, then the additional constraint $\vec{l} : \overline{p} : a$ must be taken into account. Since $d(a, \Gamma) = n = d(\vec{l}, \Gamma) + |p|$ and $|\overline{p}| = |p|$, however, the same argument as above can be used.

Second, we show that $d(\vec{l}, \Gamma) + m(\vec{l}, \Gamma) > n$, which obviously concludes the proof of the lemma. We know that $\vec{l} : [p]^*\phi$ is in Γ, and that $d(\vec{l}, \Gamma) = n - |p|$. But the first fact implies $m(\vec{l}, \Gamma) \geq |[p]^*\phi| > |p|$ since $|\phi|$ must be at least 1. □

The proof of the second part of Lemma 4.2 shows that $\Psi_1(\Gamma)$ is not changed by applying an $\wedge$-, $\vee$-, $\overline{\langle\rangle}$-, $\overline{[]^+}$-, $\langle\rangle$-, or $[]^*$-rule. Thus, if Γ' is a descendant of Γ obtained by applying one of these rules it is sufficient to show that the second component of the tuple $\Psi(\Gamma) = (\Psi_1(\Gamma), \Psi_2(\Gamma))$ decreases.

Lemma 4.7 *If Γ' is obtained from Γ by application of the $\wedge$-rule then $\Psi_2(\Gamma) \gg \Psi_2(\Gamma')$.*

Proof: Assume that the $\wedge$-rule is applied to the constraint $\vec{l} : c \wedge d$.

(1) Consider the tuple corresponding to $\vec{l}$. As shown in Lemma 4.2 we have $d(\vec{l}, \Gamma') = d(\vec{l}, \Gamma)$, and thus the first component of $\psi(\vec{l}, \Gamma')$ coincides with the first component of $\psi(\vec{l}, \Gamma)$.

The second component of the tuple cannot increase. In fact, this could only happen if a constraint of the form $\vec{g} : [p]^*\phi$ is added for a label $\vec{g}$ with $d(\vec{l}, \Gamma) > d(\vec{g}, \Gamma)$. But the only label for which constraints are added is $\vec{l}$.

The third component of $\psi(\vec{l}, \Gamma)$ decreases: $oas(c \wedge d)$ is removed from this multiset, and possible replaced by the smaller elements $oas(c)$ and $oas(d)$ (if these terms have a top-level conjunction or disjunction).

(2) Consider the tuple corresponding to a label $\vec{g}$ different from $\vec{l}$. Since Γ and Γ' contain the same constraints for $\vec{g}$, the only component that may change is the second one. This may happen if Γ' contains a new constraint of the form $\vec{l} : [p]^*\phi$. But this new constraint can only increase the tuple of $\vec{g}$ if $d(\vec{l}, \Gamma) < d(\vec{g}, \Gamma)$. In this case, the first component of $\psi(\vec{l}, \Gamma)$ is larger than the first component of $\psi(\vec{g}, \Gamma')$. Assume that $\vec{g}_1, \ldots, \vec{g}_k$ are the labels for which such an increase of the tuple takes place. Then the multiset $\Psi_2(\Gamma')$ can be obtained from $\Psi_2(\Gamma)$ by first removing all tuples $\psi(\vec{g}_i, \Gamma)$, and then replacing $\psi(\vec{l}, \Gamma)$ by the smaller tuples $\psi(\vec{l}, \Gamma'), \psi(\vec{g}_1, \Gamma'), \ldots, \psi(\vec{g}_k, \Gamma')$. Obviously, this shows that $\Psi_2(\Gamma) \gg \Psi_2(\Gamma')$. □

Lemma 4.8 *If Γ' is obtained from Γ by application of the $\vee$-rule then $\Psi_2(\Gamma) \gg \Psi_2(\Gamma')$.*

Proof: very similar to the one for the $\wedge$-rule. □

Lemma 4.9 *If Γ' is obtained from Γ by application of the $\overline{[]^+}$-rule then $\Psi_2(\Gamma) \gg \Psi_2(\Gamma')$.*

Proof: Assume that the $\overline{[]^+}$-rule is applied to the constraint $\vec{l} : \overline{[p]^+ q} : a$. We have to consider two cases: Γ' may contain the additional constraint $\vec{l} : \langle p \rangle \overline{q} : a$ or $\vec{l} : [p]\bot$.

(1) First, we consider the case where $\Gamma' = \Gamma \cup \{\vec{l} : \langle p \rangle \overline{q} : a\}$.

(1.1) Consider the tuple corresponding to $\vec{l}$. The proof of Lemma 4.2 shows that $\psi(\vec{l}, \Gamma)$ and $\psi(\vec{l}, \Gamma')$ coincide in the first component. The second component of the tuple does not increase since there are no constraints added to labels of depth smaller than $\vec{l}$. The third component of $\psi(\vec{l}, \Gamma)$ decreases: $nol(\overline{[p]^+ q})$ is removed from this multiset, and no new tuple added (since the new role constraint does not have the negation on top level).

(1.2) Consider the tuple corresponding to a label $\vec{g}$ different from $\vec{l}$. Since Γ and Γ' contain the same constraints for $\vec{g}$, we can apply the same argument as in the corresponding case in the proof of Lemma 4.7.

(2) Second, we consider the case where $\Gamma' = \Gamma \cup \{\vec{l} : [p]\bot\}$. Here the same arguments as in the first case can be employed. □

Lemma 4.10 *If Γ' is obtained from Γ by application of the $\overline{\langle\rangle}$-rule then $\Psi_2(\Gamma) \gg \Psi_2(\Gamma')$.*

Proof: very similar to the one for the $\overline{[]^+}$-rule. □

Lemma 4.11 *If Γ' is obtained from Γ by application of the $\langle\rangle$-rule then we have $\Psi_2(\Gamma) \gg \Psi_2(\Gamma')$.*

Proof: Without loss of generality we consider the case where Γ contains a constraint of the form $\vec{l} : \langle p \rangle \phi$, and $\Gamma' = \Gamma \cup \{\vec{l} : p : a, \vec{l}[i/a] : \phi\}$.

(1) Consider the tuple corresponding to $\vec{l}$. Again, the proof of Lemma 4.2 shows that the first components of $\psi(\vec{l}, \Gamma')$ and $\psi(\vec{l}, \Gamma)$ are the identical.

The second component of the tuple does not increase. The reason is that the only labels for which constraints are added are $\vec{l}$ and $\vec{l}[i/a]$. But these labels have a depth that is not smaller than the depth of $\vec{l}$. The second component of the tuple, however, can only increase if a constraint of the form $\vec{g} : [p']^*\phi'$ is added for a label $\vec{g}$ of smaller depth.

The third component is not changed since the only new constraint for $\vec{l}$ is a role constraint without negation.[3] Thus no concept constraints with top-level ' $\vee$ ' or ' $\wedge$ ' are added for $\vec{l}$, and also no role constraints with negation.

The fourth component of the tuple decreases. In fact, $|p|$ is removed from this multiset. If p is of the form $\langle q \rangle r$ or $[q]^+ r$ then $|q|$ is added to the multiset, but obviously $|q| < |p|$.

(2) Next, we consider the tuple of $\vec{l}[i/a]$. Since a is a new constant, this is a new tuple in $\Psi_2(\Gamma')$ (i.e., one that was not present in $\Psi_2(\Gamma)$). We know, however that $d(\vec{l}[i/a], \Gamma') = d(\vec{l}, \Gamma') + |p| = d(\vec{l}, \Gamma) + |p|$, and thus the first component of this tuple is smaller than the one of $\psi(\vec{l}, \Gamma)$.

(3) Finally, consider the tuple corresponding to a label $\vec{g}$ different from $\vec{l}$ and $\vec{l}[i/a]$. Obviously, the changes for the constraints on $\vec{l}$ and $\vec{l}[i/a]$ can only increase this tuple in the second component. But then the depth of $\vec{g}$ must be larger than the one of $\vec{l}$,[4] which means that the first component of the tuple corresponding to $\vec{g}$ is smaller than the first component of $\psi(\vec{l}, \Gamma)$. □

Lemma 4.12 *If Γ' is obtained from Γ by application of the $[]^*$-rule then $\Psi_2(\Gamma) \gg \Psi_2(\Gamma')$.*

Proof: We have a constraint of the form $\vec{l} : [p]^*\phi$ in Γ, and $\Gamma' = \Gamma \cup \{\vec{l}[i/a] : \phi\}$.

(1) Consider the tuple corresponding to $\vec{l}$. Since there are no new constraints on $\vec{l}$ in Γ', the only component in which this tuple could increase is the second one. But this cannot be the case since the label $\vec{l}[i/a]$ (which is the only one with a new constraint) is of larger depth than the label $\vec{l}$.

(2) Next, we consider the tuple corresponding to $\vec{l}[i/a]$. By Lemma 4.2 its first component does not change.

The second component of the tuple decreases. In fact, the constraint $\vec{l} : [p]^*\phi$ is no longer counted since $\vec{l}[i/a] : \phi$ has been added. Since labels $\vec{g}$ of smaller depth than the one of $\vec{l}[i/a]$ do not obtain new constraints, this really results in a decrease of the second component.

(3) Finally, consider the tuple corresponding to a label $\vec{g}$ different from $\vec{l}$ and $\vec{l}[i/a]$. The additional constraint on $\vec{l}[i/a]$ can only increase this tuple in the second component. But then the depth of $\vec{g}$ must be larger than the one of $\vec{l}[i/a]$, which means that the first component of the tuple corresponding to $\vec{g}$ is smaller than the first component of $\psi(\vec{l}[i/a], \Gamma)$. □

This completes the proof of termination. When the algorithm has terminated it returns either 'unsatisfiable,' if all constraint systems have been eliminated by the $\bot$-rule since they contain clashes, or it returns 'satisfiable,' if there remains at least one system without a clash. The next theorem shows that in the first case the answer 'unsatisfiable' is correct.

Theorem 4.13 *Let c_0 be a concept term of $\mathcal{M}$-$\mathcal{ALC}$, and assume that the function apply-rules is called with the singleton set $\{\{\vec{l_0} : c_0\}\}$. If it returns 'unsatisfiable' then the input term c_0 is in fact unsatisfiable.*

In order to prove the theorem we have to extend the notion of a model to constraint systems.

Definition 4.14 (Semantics of Constraint Systems) *An interpretation for a constraint system Γ is an interpretation of the underlying $\mathcal{M}$-$\mathcal{ALC}$-language that, in addition,*

- *assigns an element of D_i to each constant of dimension i occurring in Γ,*

[3] Inside of diamond-operators we do not allow the use of negated roles.

[4] Note that the depth of $\vec{l}[i/a]$ is larger than the depth of $\vec{l}$.

- *interprets over-lined role terms* $\overline{p}$ *of dimension* i *as* $\Im(\overline{p})(\vec{d}) = D_i \setminus \Im(p)(\vec{d})$.

More general role terms with negation are interpreted according to the semantics for box- and diamond-operators on roles given in Definition 2.2. The interpretation of a label $\vec{l} = (l_1, \ldots, l_n)$ *is* $\Im(\vec{l}) = (\Im(l_1), \ldots, \Im(l_n))$.

The interpretation $\Im$ *satisfies the concept constraint* $\vec{l} : c$ *iff* $\Im(\vec{l}) \in \Im(c)$*, and it satisfies the role constraint* $\vec{l} : p : a$ *iff* $\Im(a) \in \Im(p)(\Im(\vec{l}))$*. It is a model of* Γ *iff it satisfies all constraints in* Γ*. The interpretation* $\Im$ *is a model of a set* Δ *of constraint systems iff it is a model of one of the systems in* Δ*. A constraint system (set of constraint systems) is satisfiable iff it has a model.*

The following three lemmas show that application of rules keeps the satisfiability of sets of constraint systems unchanged.

Lemma 4.15 *If the application of the* $\vee$*-,* $\overline{\langle\rangle}$*-, or* $\overline{[]^+}$*-rule replaces the system* Γ *by* Γ_1 *and* Γ_2 *then* Γ *is satisfiable iff* Γ_1 *or* Γ_2 *is satisfiable.*

Proof: (1) This is rather obvious for the $\vee$-rule.

(2) Thus, consider the $\overline{[]^+}$-rule. This means that Γ contains the constraint $\vec{l} : \overline{p} : a$ for $p = [q]^+ r$, and $\Gamma_1 = \Gamma \cup \{\vec{l} : \langle q \rangle \overline{r} : a\}$ whereas $\Gamma_2 = \Gamma \cup \{\vec{l} : [q]\bot\}$.

The 'if' direction of the lemma is trivially satisfied since both Γ_1 and Γ_2 are supersets of Γ.

To show the 'only-if' direction of the lemma, we assume that the interpretation $\Im$ satisfies the constraint $\vec{l} : \overline{p} : a$, i.e., $\Im(a) \notin \Im(p)(\Im(\vec{l}))$. Since $p = [q]^+ r$ this means that either $\Im(q)(\Im(\vec{l})) = \emptyset$, or this set is non-empty but contains an element x such that $\Im(a) \notin \Im(r)(x)$. In the first case, $\Im$ satisfies the constraint $\vec{l} : [q]\bot$, and thus Γ_2 is satisfiable. In the second case, $\Im(a) \in \Im(\overline{r})(x)$, and thus $\Im(a) \in \Im(\langle q \rangle \overline{r})(\Im(\vec{l}))$. This shows that $\Im$ satisfies Γ_1.

(3) The $\overline{\langle\rangle}$-rule can be treated similarly. □

Lemma 4.16 *If the application of the* $\wedge$*-,* $\langle\rangle$*-, or* $[]^*$*-rule replaces the system* Γ *by* Γ' *then* Γ *is satisfiable iff* Γ' *is satisfiable.*

Proof: The proof is obvious for the $\wedge$- and the $\langle\rangle$-rule.

Thus consider the $[]^*$-rule. If Γ' is satisfiable then Γ is satisfiable as well since Γ is a subset of Γ'. To prove the other direction it obviously suffices to show the following property:

> If the recursive call of the function *apply-rules* returns 'unsatisfiable' then we have $\Im(a) \in \Im(p)(\Im(\vec{l}))$ for all models $\Im$ of Γ.

Assume to the contrary that $\Im$ is a model of Γ such that

$$(*) \quad \Im(a) \notin \Im(p)(\Im(\vec{l})).$$

Obviously, $\Im$ is a model of the subset $\{\vec{l'} : q : b \mid d(b, \Gamma) \leq n\}$ of Γ. Because of the assumption $(*)$, $\Im$ satisfies the constraint $\vec{l} : \overline{p} : a$ as well. This means that the recursive call returns 'unsatisfiable' even though the input system is satisfiable. By induction, we can assume, however, that Theorem 4.13 already holds for this smaller system. □

Lemma 4.17 *If the* $\bot$*-rule applies to a constraint system* Γ *then* Γ *is unsatisfiable.*

Proof: obvious by the semantics for concept and role negation and for $\bot$. □

Now we can prove Theorem 4.13. Assume that the concept term c_0 is satisfiable, and that we call the function *apply-rules* with input $\{\{\vec{l_0} : c_0\}\}$. Since c_0 is satisfiable, the constraint system

$\Gamma_0 = \{\vec{l_0} : c_0\}$ is satisfiable as well. Let Δ be a set of constraint systems obtained by repeatedly applying the rules of Figure 1 to this input. Since Γ_0 is satisfiable, the Lemmas 4.15, 4.16, and 4.17 obviously imply that there exists a constraint systems in Δ that is satisfiable. For this reason, Δ cannot be empty, which shows that the $\emptyset$-rule cannot be applied to Δ. This completes the proof of Theorem 4.13.

Unfortunately, we did not succeed in showing the opposite direction of the statement in Theorem 4.13, but we strongly conjecture that it holds. Since subsumption is reduced to unsatisfiability this means that we have presented a sound (but possibly incomplete) algorithm for subsumption in $\mathcal{M}$-$\mathcal{ALC}$. In order to prove the conjecture it is sufficient to show that a constraint to which the $\top$-rule applies is in fact satisfiable. The main problem in the proof is to show that the $[]^*$-rule is complete, i.e., that that the restrictions on its applicability are not to severe.

In the above proofs we have never used the restriction that role terms must not contain [...]-operators. If this restriction did not hold, however, the algorithm would obviously be incomplete. This is illustrated by the following example, which is a modification of an example given in Section 3. We consider a 1-dimensional $\mathcal{M}$-$\mathcal{ALC}$ language, and are interested in the satisfiability of the concept term

$$c_0 = [[p]q]c \wedge [p]\bot \wedge \neg c.$$

Note that the first conjunct of this term is not restricted serial. In Section 3 we have shown the following fact: If $\Im$ is an interpretation such that $\Im([[p]q]c \wedge [p]\bot) \neq \emptyset$ then any object o in the carrier set of $\Im$ is an element of $\Im(c)$. Obviously, this implies that c_0 is unsatisfiable.

Our algorithm, however, does not recognize the unsatisfiability of c_0. The algorithm starts with the constraint system $\{l_0 : c_0\}$. By applying the $\wedge$-rule twice we obtain the system $\Gamma = \{l_0 : [[p]q]c, l_0 : [p]\bot, l_0 : \neg c\}$. At this point, none of the rules other than the $\top$-rule can be applied. In fact, the only possible rule could be the $[]^*$-rule. But there is no constant a of depth $d(l_0, \Gamma) + |p|$ or $d(l_0, \Gamma) + |p| + |q|$ in Γ.

5 Summary and Open Problems

The present paper is a first investigation of a new kind of multi-dimensional modal logic. The logic $\mathcal{M}$-$\mathcal{ALC}$ is a combination of modal logics K_m, but the combination is of an unusual type. The modal operators of the component logics do not only operate on the formulae in the combined logic, but also directly on the operators of the other logics. As we have seen, this gives rise to quite complicated interactions between the component logics. This kind of logic was motivated by applications in the area of KL-ONE-like knowledge representation systems, and in particular by the need of modelling the knowledge of intelligent agents.

In this paper, we have only worked out the basic framework, and have defined a calculus based on the idea of labelled deductive systems. There are various interesting questions that remain open.

First, of course, is the question whether the algorithm is also complete for unsatisfiability. If the answer is yes, this would show decidability of the satisfiability problem for restricted serial $\mathcal{M}$-$\mathcal{ALC}$ terms. If the answer is no, decidability remains an open question. The semantics of $\mathcal{M}$-$\mathcal{ALC}$ allows for a straightforward translation of concept terms into first-order predicate logic. But the translated versions of even small concept terms may already become very complicated. The formulae one gets do not seem to fall into one of the known decidable subclasses of first-order logic. This is in contrast to $\mathcal{ALC}$ where concept terms can be translated into formulae of the Gödel class.

More generally, one can ask whether the methods described above can be adapted to the case where arbitrary [...]-operators are allowed at the role term level. Related is the question whether satisfiability for arbitrary $\mathcal{M}$-$\mathcal{ALC}$ terms is decidable?

An adequate representation of modalities such as know, belief, or time require component logics that are stronger than K_m; for example S4.3 for knowledge, linear structures for time etc. More generally, one can ask whether it is possible to modify the satisfiability algorithm such that it can take additional modal axioms into account. A possible way of attacking this problem could be to

translate the modal axiom schemas into properties of the accessibility relations (see [9]). Complex correlations between different modal operators can thus be investigated, and turned into additional rules of the satisfiability algorithm. However, without additional restrictions, the rule set one thus obtains will usually not be terminating.

As already mentioned, a flexible treatment of T-Box axioms would be desirable. Can such axioms be handled by a satisfiability algorithm? Also, the interaction with A-Boxes has not yet been considered. In this context, is it possible to parameterize the role terms with A-Box elements or concept terms, e.g., by writing [*know*(*John*)] or [*believe*(*car-salesmen*)]?

References

[1] Franz Baader and Bernhard Hollunder. A terminological knowledge representation system with complete inference algorithms. In Michael M. Richter and Harold Boley, editors, *Proc. of PDK 91*. LNAI 567, 1991.

[2] R. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[3] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, Calif., 1991.

[4] B. F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, Cambridge, 1980.

[5] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the* ACM, 8(22):465–476, 1979.

[6] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In *Proc. of KR'91, Principles of Knowledge Representation and Reasoning*, pages 151–162. Morgan Kaufmann, 1991.

[7] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Science*, 18:194–211, 1979.

[8] Dov M. Gabbay. *Labelled Deduction Systems.* Oxford University Press, 1991, in preparation.

[9] Dov M. Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second-order predicate logic. *South African Computer Journal*, 7:35–43, July 1992. Also in Proc. of KR92, Morgan Kaufmann, 1992, pp 425–436.

[10] A. Kobsa. The SB-ONE knowledge representation workbench. In *Preprints of the Workshop on Formal Aspects of Semantic Networks*, Two Habours, Calif., 1989.

[11] A. Laux. Integrating a modal logic of knowledge into terminological logics. Research Report RR-92-56, DFKI Saarbrücken, 1992.

[12] Hector J. Levesque and Ron J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[13] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems.* LNAI 422, 1990.

[14] Bernhard Nebel and Gert Smolka. Representation and reasoning with attributive descriptions. In K.H. Bläsius, U. Hedtstück, and C.-R. Rollinger, editors, *Sorts and Types in Artificial Intelligence*, pages 112–139. LNAI 418, 1990.

[15] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI 91*, pages 466–471. Morgan Kaufmann, 1991.

[16] Y. Venema. *Many-Dimensional Modal Logic.* PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 1992.

[17] William A. Woods and James G. Schmolze. The KL-ONE family. *Computers and Mathematics with Applications*, 23(2–5):133–177, 1992. Special Issue on Semantic Networks in Artificial Intelligence.