A Multi-Dimensional Terminological Knowledge Representation Language*

Hans Jiirgen Ohlbach
Max-Planck-Institut fur Informatik
Im Stadtwald
6600 Saarbrucken 11, Germany
email: ohlbach@mpi-sb.mpg.de

Franz Baader
German Research Center for AI (DFKI)
Stuhlsatzenhausweg 3
6600 Saarbriicken 11, Germany
email: baader@dfki.uni-sb.de

Abstract

An extension of the concept description language *ACC* 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.

1 Introduction

Knowledge representation languages in the style of KL-ONE [3], 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, 4, 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:
$\textit{new-car} = \textit{car} \sqcap \exists \textit{has-part}: \textit{catalytic-converter}$

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

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 [*belief-*...][1] that satisfy appropriate modal axioms allow for more natural definitions:

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

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

$$\textit{pot-customer} = \textit{adult} \sqcap \exists \textit{eventually-wants-own}: \textit{car},$$

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 *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

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

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 [want] means that one takes only those objects that are role-fillers in all accessible intentional worlds, and the prefix (future) 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

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

[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 *(p)* 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.

to modify concepts in this way. In the definition

$$\begin{aligned}&\textit{environment-freak} = \textit{person} \sqcap \\ &\quad \forall([\textit{want}]\textit{own}){:}[\textit{belief}]\textit{environment-friendly}\end{aligned}$$

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 [belief]-operator modifies the concept en *vironmen t-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 *ACC* is just a syntactic variant of the multi-modal logic K_m [5], where m is the number of different box operators. The reason is that quantifications over roles can just be seen as applications of parametrized 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 parametrized 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 *environrncnt-fre&k* we thus get the modal prefix [[want]own] where the complex (role) term [want]own occurs inside a box-operator.

Our new approach for integrating modalities into terminological KR languages, called *M-ACC* in the following, thus extends the prototypical terminological language *ACC* 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]Satisfiability modulo concept equations is known to be exp-time complete; this is an easy consequence of a result by Fischer and Ladner [7].

2 Syntax and Semantics of *M-ACC*

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 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 $[\ldots]^+$ that combines the the box- and diamond-operator. In many cases, $[\ldots]^+$ makes more sense than the usual box-operator $[\ldots]$. 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 $[\ldots]^*$ will be used for the $[\ldots]^+$-operator as well as for the normal $[\ldots]$-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 *woman* $\wedge$ $\langle$[*has-child*]*likes*$\rangle$*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 is interpreted as a function $\Im_\Sigma(p) : \vec{D} \rightarrow 2^{D_i}$, *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 $\Im(q) : \vec{D} \rightarrow 2^{D_i}$ *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.*

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

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.*

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

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 $c_1 = d_1[c_2], \ldots, c_n = d_n[c]$ 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 $[\ldots]$-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 $[[p]q]c \wedge [p]\perp$. The term $[p]\perp$ (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 equation 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 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:

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

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, a concept equation or a role term with an argument. The expression $\vec{l} : p : a$ for example means that a is a p-successor of $l_{dim(p)}$. 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. A* role constraint *is a triple $\vec{l} : p : a$ consisting of a label $\vec{l}$, a role term p and a point a. 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 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 constraints. Since we must avoid $[\ldots]$-operators in the role terms, the rules for building negation normal form of role terms is slightly modified:

$$\begin{array}{lll}
\overline{[p]^{+}q} & \rightarrow & \langle p\rangle\overline{q}\vee[p]\perp \\
\overline{\langle p\rangle q} & \rightarrow & [p]\overline{q} \;\rightarrow\; [p]^{+}\overline{q}\vee[p]\perp
\end{array}$$

After building negation normal form, only atomic role names occur negated. Note that $[p]\perp$ is an allowed concept term. The intended interpretation of $\overline{p}$ is simply as complement: $\Im(\overline{p})(\vec{d}) = D_{dim(p)} \setminus \Im(p)(\vec{d})$. Since negated role terms occur only in a very restricted setting, this does not mean that we introduce general role negation.

The algorithm depends on a function *depth* 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 $\vec{l} : p : x$ (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 $l_0 = (l_{01}, \ldots, l_{0n})$ we define*

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

It can be shown 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 in negation normal form. It then constructs an initial label $\vec{l}_0$ and builds the constraint system $\{\vec{l}_0 : c\}$. Then it calls the function apply-rules with the singleton set $\Delta_0 = \{\{\vec{l}_0 : c\}\}$. 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

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 '$\&\Gamma$' stands for the unmodified part of the constraint system under consideration, and '$\& \Delta$' 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.

$(\vee)$ $\{\vec{l} : \phi \vee \psi \,\&\, \Gamma\} \,\&\, \Delta \quad \rightarrow \quad \{\vec{l} : \phi \vee \psi, \vec{l} : \phi \,\&\, \Gamma\}, \{\vec{l} : \phi \vee \psi, \vec{l} : \psi \,\&\, \Gamma\} \,\&\, \Delta$
if neither $\vec{l} : \phi$ nor $\vec{l} : \psi$ 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.

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

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

$(\top)$ $\Gamma \,\&\, \Delta \quad \rightarrow$ '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 *M-ACC.*

$[\langle p\rangle q]c$ 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 consists of the two constraints

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

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 $depth(b, \Gamma) = 2 = depth(\Gamma, l_0) + |\langle p\rangle q|$. 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 (7) $l_0 : [p]^{+}\overline{q} : b$. (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).

In [2] it is 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 M(c), which depends linearly on the size of the input term *c.* It should be noted that both properties strongly depend on our restriction that [...]-operators are not allowed at the role term level. The fact that labels are of bounded depth plays an important role in the proof of termination.

Theorem 3.4 *The satisfiability algorithm described above terminates for all restricted serial M -ACC concept terms. If tt returns unsatisfiable then the input term is in fact unsatisfiable.*

A proof can be found in [2]. Unfortunately, we did not succeed in showing the opposite direction of the second statement, 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 *M-ACC.*

The semantics of *M-ACC* 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 *ACC* where concept terms can be translated into formulae of the Godel class. Decidability of the satisfiability problem in *M-ACC* is still an open problem.

4 Summary and Open Problems

The present paper is a first investigation of a new kind of multi-dimensional modal logic. The logic M-ACC 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 M-ACC terms. If the answer is no, decidability remains an open question.

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 M-ACC terms is decidable[7]

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 com plete inference algorithms. In Michael M. Richter and Harold Boley, editors, Proc. of PDK 91. LNAI 567, 1991.

[2] Franz Baader and Hans Jurgen Ohlbach. A multi-dimensional terminological knowledge representation language. Technical Report MPI-I-93-212, Max Planck Institut fiir Informatik, Saarbrucken, 1993.

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

[4] 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.

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

[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 Jiirgen 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 Saarbrucken, 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. Blasius, U. Hedtstiick, 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.