Modal Logics, Description Logics and Arithmetic Reasoning

Hans Jürgen Ohlbach

Dept. of Computer Science, King's College London, Strand, London WC2R 2LS, email: ohlbach@dcs.kcl.ac.uk

Jana Koehler

Institut für Informatik, Albert-Ludwigs-Universität, Am Flughafen 17, 79110 Freiburg, Germany, email: koehler@informatik.uni-freiburg.de

Abstract

We introduce mathematical programming and atomic decomposition as the basic modal (T-Box) inference techniques for a large class of modal and description logics. The class of description logics suitable for the proposed methods is strong on the arithmetical side. In particular there may be complex arithmetical conditions on sets of accessible worlds (role fillers).

The atomic decomposition technique can deal with set constructors for modal parameters (role terms) and parameter (role) hierarchies specified in full propositional logic. Besides the standard modal operators, a number of other constructors can be added in a relatively straightforward way. Examples are graded modalities (qualified number restrictions) and also generalized quantifiers like 'most', 'n%', 'more' and 'many'.

Key words: modal logic, description logic, combination of inference systems, mathematical programming

1 Introduction and Overview

Classical propositional logic can not only be used for reasoning about truth values. It can also be used as a set description language. Predicate symbols are

[1] This work was supported by the EPSRC Research Grant GR/L91818 'data driven logic algorithms'

mapped to sets and the Boolean connectives $\wedge$ (conjunction) $\vee$ (disjunction), $\neg$ (negation), $\Rightarrow$ (implication) are mapped to the set functions *intersection, union, complement* and the *subset* relationship.

Both modal logics and description logics take this set theoretic interpretation further by introducing new operators on the syntactic side and new structures on the semantic side. The extra features on the semantic side are binary relations, called *accessibility relations* for modal logics, and *roles* for description logics. The extra features on the syntactic side are quantifiers (*modal operators*) which quantify over accessible worlds (*role fillers*).

Although many modal logics and description logics are syntactic variants of each other – the description logic $\mathcal{ALC}$, for example, corresponds exactly to the multi-modal logic K_m [22] – their origin is completely different. Modal logics were introduced to distinguish between formulae which are true just by chance, and formulae which are *necessarily* true. Therefore the modal $\Box$-operator was called the *necessitation* operator [11,12].

Description logics, on the other hand, are late descendants of Minski's frames [18] and Brachman's KL-ONE [6]. They come in a variety of different versions, e.g. $\mathcal{ALC}$ [24], CLASSIC [7], KRIS [2], LOOM [17] and even in class-based object oriented formalisms [10]. Common to most of them is the separation of a description logic database into a so-called *T-Box* (terminological box) and a so-called *A-Box* (assertional box). The T-Box contains specifications of *concepts* and *roles*. For example a T-Box formula

$$parent \stackrel{\text{def}}{=} person \wedge |has\text{-}child| \geq 1 \tag{1}$$

specifies the concept *parent* as the set of all persons who have at least one child. The (multi)-modal logic notation for this formula would be

$$parent \Leftrightarrow (person \wedge \langle has\text{-}child\rangle \top)$$

($\top$ stands for 'truth'. $\langle has\text{-}child\rangle$ is the parameterized diamond operator.) The parameter *has-child* for the modal operator denotes the accessibility relation ('role' in the description logic jargon).

The A-Box in a description logic database, on the other hand, contains information about instances of the T-Box concepts. For example, from the A-Box entries Henry: person, and Henry: *has-child* Mary, one can conclude that *Henry* is an instance of the concept *parent*. From a modal logic point of view, A-Box instances are names for worlds. An A-Box consistent with a T-Box describes a partial model for the formulae in the T-Box.

On the T-Box level there are two major reasoning problems. First of all, one wants to know whether a newly introduced concept definition is consistent with the previously introduced ones. For example, if the T-Box contains the two definitions

$$male \stackrel{\text{def}}{=} person \wedge |has\text{-}y\text{-}chromosome| \geq 1 \tag{2}$$
$$female \stackrel{\text{def}}{=} person \wedge |has\text{-}y\text{-}chromosome| = 0 \tag{3}$$

(males are persons with at least one y-chromosome, and females are persons with no y-chromosome) and we add the new definition

$$hermaphrodite \stackrel{\text{def}}{=} male \wedge female$$

there is no non-empty extension of *hermaphrodite*, which usually indicates errors or misconceptions in the axiomatization of a given domain.

The second inference problem is *subsumption* (implication[2]). If we have (1) in our database and we add

$$grandparent \stackrel{\text{def}}{=} person \wedge atleast\ 1\ has\text{-}child.parent \tag{4}$$

(grandparents are persons who have at least one child who is a parent) then we can, of course, conclude that all grandparents are parents as well, i.e. *grandparent* $\Rightarrow$ *parent*. Subsumption relations are very useful for structuring a knowledge base. Finding out all subsumption relations between all concepts is called *classification*, and this is the basic operation of all T-Boxes. If the description logic language has the full classical negation (not all of them have it) then the subsumption problem $\varphi_1 \Rightarrow \varphi_2$ can be reduced to the consistency problem for $\varphi_1 \wedge \neg\varphi_2$.

In this paper we investigate problems which have been discussed more in the description logic context than in the modal logic context. Therefore we prefer using the description logic notions. Table 2 compares the different notions used by the modal logic community with the corresponding notions used by the description logic community.

The standard semantics of modal and description logics allows one to translate all T-Box and A-Box information into first-order predicate logic (FOL). Therefore description logics, as well as most modal logic, are essentially fragments of FOL. Since most of them are decidable, they represent proper fragments of FOL, but they are usually more expressive than propositional logic. Much effort has been invested in recent years to explore the borderline between propositional logic and FOL by investigating various versions of description logics, see [14] for a good summary of recent results.

[2] In the Description Logic literature one usually finds subsumption to be defined as the converse of implication. This contradicts the common understanding of 'subsumes' and the definition in classical logic. Therefore we prefer the 'right' meaning: subsumption = implication.

description logic	modal logic
$\mathcal{ALC}$	multi-modal logic K_m
concept formula	modal formula
concept definition	modal formula of the kind *concept-name* $\Leftrightarrow$ *formula*
concept name	predicate symbol
concept	extension of a predicate symbol
role name	parameter of a parameterized modal operator
role	accessibility relation
role term	complex parameter of a modal operator
role fillers	set of accessible worlds
T-Box	set of concept definitions
A-Box entry	name of a world
A-Box	description of a partial Kripke structure
domain	set of worlds
object	world
consistency of a concept formula	satisfiability of a modal formula
subsumption between concept formula	entailment between modal formulae
existential quantifier $\exists\, r.\varphi$	diamond operator $\langle r\rangle\varphi$
universal quantifier $\forall\, r.\varphi$	box operator $[r]\varphi$
number restriction $\lvert r\rvert \geq n$	simple graded modal operator $\langle r\rangle_n\top$ restriction on the number of accessible worlds
qualified number restriction *atmost* $r\ n.\varphi$	graded modal operator $\langle r\rangle_n\varphi$
arithmetic constraint for the role fillers	(not well investigated)

Table 1
Corresponding notions for description logics and modal logics.

Most methods for checking consistency of concept formulae and subsumption between concept formulae are tableau algorithms. Starting with a tableau entry $a:\varphi$ (the object with name a is an element of the set described by φ), tableau rules are applied to make the information explicit which is implicitly contained in the input formulae. Conjunctive rules just extend the list of derived information, whereas disjunctive rules start a case analysis by splitting the tableau into different branches.

If the consistency problem for the logic is decidable and the tableau algorithm is well designed then the application of the tableau rules eventually terminates with obvious contradictions or with open branches representing a model for the initial formula φ. The method is well suited for languages containing mainly logical operators. As soon as arithmetics comes into play, tableau approaches become very difficult to use. For example in a concept definition

$$\textit{parents-of-many-boys} = \textit{parent} \wedge |\textit{has-son}| \geq 2|\textit{has-daughter}|$$

(parents-of-many-boys are parents having more than twice as many sons than daughters) the consistency problem amounts to checking whether $x \geq 2y$ has a non-negative integer valued solution. This is a trivial check for integer programming algorithms, but almost impossible for a tableau method. Therefore the arithmetic part of most modal and description logics is very weak. They usually only allow for *number restrictions* of the kind $|r| \geq n$ or $|r| \leq n$ where n is a number. *Qualified number restrictions* $\textit{atmost}\, n\, r.\varphi$ (set of objects with at most n r-role fillers in φ) and $\textit{atleast}\, n\, r.\varphi$ (set of objects with at at least n r-role fillers in φ) are also being used.

In this paper we propose using arithmetic equation solving instead of tableau systems as the basic inference algorithm. It is, however, not the purpose of this paper to investigate arithmetic equation solving itself; we assume suitable algorithms are available (they can actually be downloaded from the internet). Therefore we do not specify a particular arithmetic language. The language depends on the available arithmetic equation solver. Most of them can solve systems of *linear* equations and in-equations. In this case only addition, subtraction and multiplication with numbers is allowed. More advanced systems also allow for certain non-linear terms. The general non-linear Diophantine equation problem, however, is undecidable (Hilbert's 10th problem [13]). Therefore the arithmetic language should not be too expressive.

There are only a few requirements about the arithmetic system, which are important for the purposes of this paper.

- In the basic mode the arithmetic system must accept conjunctions of equations and in-equations and check whether there is a solution or not. The solutions themselves are not needed.
- For the subsumption test the arithmetic system must check whether all solutions of a given (in)equation system E_1 are also solutions of another system E_2. If the arithmetic system can deal with dis-equations then this problem can be reduced to a consistency problem for $E_1 \wedge \neg E_2$.
- If the description logic allows for disjunctions in the concept definitions then the arithmetic system also should be able to deal with disjunctions of equations and in-equations.

1.1 Atomic Decomposition

In the main part of the paper we show how the consistency and the subsumption problem of concept formulae can be mapped to equation solving problems.

The *atomic decomposition technique* [21] plays a key role in this process. Since the technique is not widely known, we give a brief overview.

Atomic decomposition exploits the possibility to decompose finite sets of sets into mutually disjoint *atomic* components. These are the atoms of the Boolean algebra consisting of the closure of the sets under union, intersection and complement. To illustrate this idea, suppose the two roles *has-son* and *has-daughter* are specified as sub-roles of *has-child*. From

$$|\mathit{has\text{-}son}| \geq 2 \wedge |\mathit{has\text{-}daughter}| \geq 3 \tag{5}$$

one can deduce $|\mathit{has\text{-}child}| \geq 5$. For each object in the domain the role fillers of *has-son*, *has-daughter* and *has-child* form three sets, which can overlap in the most general way as depicted in Figure 1. There are seven different areas

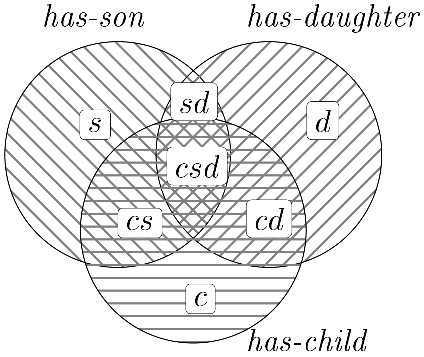


Fig. 1. A general set structure

(together with the complement of the hatched areas there are in fact $2^3 = 8$ different areas) named c, s, d, cs, cd, sd, and csd. The informal meaning is

$$\begin{aligned}
c &= \text{children, not sons, not daughters.} \\
s &= \text{sons, not children, not daughters.} \\
d &= \text{daughters, not children, not sons.} \\
cs &= \text{children, which are sons, not daughters.} \\
cd &= \text{children, which are daughters, not sons.} \\
sd &= \text{sons, which are daughters, not children.} \\
csd &= \text{children, which are both sons and daughters.}
\end{aligned}$$

The original sets can now be obtained from their 'atomic' components:

$$\begin{aligned}
\mathit{has\text{-}child} &= c \cup cs \cup cd \cup csd \\
\mathit{has\text{-}son} &= s \cup cs \cup sd \cup csd \\
\mathit{has\text{-}daughter} &= d \cup cd \cup sd \cup csd.
\end{aligned}$$

Moreover, since this decomposition is mutually disjoint and exhaustive, the cardinalities of the sets just add up:

$$\begin{aligned} |\textit{has-child}| &= |c| + |cs| + |cd| + |csd| \\ |\textit{has-son}| &= |s| + |cs| + |sd| + |csd| \\ |\textit{has-daughter}| &= |d| + |cd| + |sd| + |csd|. \end{aligned}$$

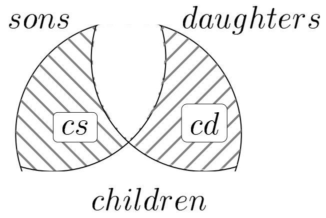
The relationships between *has-child*, *has-son* and *has-daughter* can actually be specified in propositional logic:

$$\begin{array}{ll} \textit{has-son} \Rightarrow \textit{has-child} & \text{sons are children} \\ \textit{has-daughter} \Rightarrow \textit{has-child} & \text{daughters are children} \\ \textit{has-son} \wedge \textit{has-daughter} \Rightarrow \bot & \text{there are no hermaphrodites} \\ \textit{has-child} \Rightarrow \textit{has-son} \vee \textit{has-daughter} & \text{children consist only} \\ & \text{of sons and daughters} \end{array} \tag{6}$$

These formulae have three propositional models:

$$\begin{array}{l} \textit{has-child}, \textit{has-son}, \neg\textit{has-daughter} \\ \textit{has-child}, \textit{has-daughter}, \neg\textit{has-son} \\ \neg\textit{has-child}, \neg\textit{has-son}, \neg\textit{has-daughter}. \end{array}$$

They correspond to the two non-empty sets *cs* and *cd* in the figure below, together with the surrounding area.



The fact that these are the only models means $|c| = 0$, $|s| = 0$, $|d| = 0$, $|sd| = 0$, $|cd| = 0$ and $|csd| = 0$. The problem of determining whether there are at least 5 children can now be reformulated

$$|cs| \geq 2 \wedge |cd| \geq 3 \Rightarrow |cs| + |cd| \geq 5. \tag{7}$$

Since the sets are mutually disjoint, the internal structure of 'cardinality terms' like $|cs|$ is no longer relevant, and $|cs|$ can be replaced with a non-negative integer-valued variable x_{cs}. We obtain

$$x_{cs} \geq 2 \wedge x_{cd} \geq 3 \Rightarrow x_{cs} + x_{cd} \geq 5. \tag{8}$$

which is trivial to check.

In [21] this idea was developed into a general methodology for augmenting formal systems with a Boolean algebra component. The general methodology works for formal systems whose language has a notion of (existentially quantified) variables. A typical example is a mathematical programming system for solving equations and in-equations. On the syntactic side, this formal system can be extended with set terms embedded in *bridging functions* at variable positions. The bridging functions map objects of one logic to objects of another logic. For example, if the basic system allows for equations like

$$2 \cdot x + 3 \cdot y = 5$$

then the extended system would allow for equations like

$$2 \cdot |\mathit{sons} \cup \mathit{friends}| + 3 \cdot \mathit{max\text{-}age}(\mathit{friends} \setminus \mathit{sons}) = 5.$$

$\mathit{sons} \cup \mathit{friends}$ and $\mathit{friends} \setminus \mathit{sons}$ are Boolean set terms. The cardinality function $|..|$ and the *max-age* function are bridging functions. Both map sets to numbers such that multiplication with 2 and 3 is defined. In the general setting, bridging functions map the sets to objects in the basic system which make sense there.

The relationships between the sets can be axiomatized in propositional logic. (6) is an example for such a propositional axiomatization. It exploits the fact that the elements and connectives of Boolean algebras can always be interpreted as sets and the corresponding set operations (Stone's representation theorem [25]). With some elementary Boolean algebra theory one can show that the models of the propositional axiomatization correspond to the atoms of the Boolean algebra generated by the closure of the sets under union, intersection and complement. [3]

This correspondence can be turned into an algorithm for eliminating the Boolean terms and the bridging functions. In the first step we compute a syntactic representation of the models of the propositional axiomatization $\mathcal{A}_{\mathcal{R}}$. The Boolean terms t can now be decomposed into the atomic components $\{m_1, \ldots, m_n\} \stackrel{\text{def}}{=} \{m \mid m \text{ is a model for } \mathcal{A}_{\mathcal{R}} \text{ and } m \text{ satisfies } t\}$. This way all

[3] A Boolean algebra is a non-empty set equipped with the functions $\sqcap$ (meet), $\sqcup$ (join) and $'$ (inverse), a smallest element $\perp$ and a largest element $\top$. A $\leq$-relation is definable as: $x \leq y$ iff $x \sqcap y = x$. Set algebras where $\sqcap$ is intersection, $\sqcup$ is union and $'$ is complement and $\leq$ is the subset relation, is one particular kind of Boolean algebra. Every Boolean algebra, however, is equivalent to a set algebra. A Boolean algebra is *complete* iff all (finite and infinite) joins belong to it. It is *atomic* iff every element can be obtained as the join of a set of smallest elements above $\perp$, the *atoms*. The atoms in set algebras are the singleton sets. Finite Boolean algebras are always complete and atomic (cf. any textbook on Boolean algebras).

Boolean terms t embedded in a bridging function $f(t)$ can be rewritten into $f(\{m_1, \ldots, m_n\})$.[4]

In the next step of the decomposition method we must use an extra assumption about bridging functions. They must be *additive*. This means, if two sets x and y are disjoint then it must be possible to compute $f(x \cup y)$ by first computing $f(x)$ and $f(y)$ and then joining the results with some *combination function*. Examples where this is fulfilled are:

$$x \cap y = \emptyset \Rightarrow |x \cup y| = |x| + |y|$$

$$x \cap y = \emptyset \Rightarrow \mathit{max\text{-}age}(x \cup y) = \mathit{max}(\mathit{max\text{-}age}(x), \mathit{max\text{-}age}(y))$$

$$x \cap y = \emptyset \Rightarrow \mathit{average\text{-}age}(x \cup y) = \frac{|x|\,\mathit{average\text{-}age}(x) + |y|\,\mathit{average\text{-}age}(y)}{|x| + |y|}.$$

The additivity of the bridging functions and the fact that the atoms m_i all denote disjoint sets, allows us to rewrite terms $f(\{m_1, \ldots, m_n\})$ into $g(f(m_1), \ldots, f(m_n))$ where g is the composition function.

In the last step we replace terms $f(m_i)$ with new variables $x_{f(m_i)}$ of the basic system. (7) $\rightarrow$ (8) is such a replacement.

The transformations are sound and complete. This means that the original problem in the mixed language has a solution if and only if the transformed problem has a solution in the basic system.

1.2 Atomic Decomposition and Role Terms

The atomic decomposition method can be applied to the role part of description logics. On the semantic level the sets which get decomposed are the sets of role fillers of a given object. On the syntactic side we start by using combinations of arithmetic formulae and set terms to specify constraints on role fillers. Examples are

$$\mathit{young\text{-}family} \stackrel{\mathrm{def}}{=} \mathit{average\text{-}age}(\mathit{has\text{-}child}) \leq 10$$

(*has-child* is a role, *average-age* is a bridging function.)

$$\mathit{poor\text{-}family} \stackrel{\mathrm{def}}{=} \mathit{max\text{-}income}(\mathit{member}) \leq 100$$

(*member* is a role, *max-income* is a bridging function.)

$$\mathit{dog\text{-}lovers} \stackrel{\mathrm{def}}{=} |\mathit{has\text{-}dog}| \geq 2 \cdot |\mathit{has\text{-}child}|$$

[4] The m_i are conjunctions (meets) of positive or negative Boolean variables. But for most purposes it is sufficient to take the m_i as *names* for the models.

(dog lovers have more than twice as many dogs than children.)

Relationships between different role terms can be expressed as propositional axioms. With the atomic decomposition technique we can then reduce the consistency and subsumption problems to arithmetic equation solving problems.

1.3 Predicate Symbols

The language of constraints on role fillers is, in general, not expressive enough. Therefore we investigate how various other logical constructs fit into this framework, and what extra mechanisms are needed to obtain a sound and complete decision procedure for the consistency and subsumption problem.

In the description logic context predicate symbols (also called *concept names*) are names for sets of objects. In the definition

$$\mathit{parent} \stackrel{\text{def}}{=} \mathit{person} \wedge |\mathit{has\text{-}child}| \geq 1$$

for example, the predicate symbol *person* may be an undefined symbol. In this case no particular assumptions about the set of *persons* are made. If it is defined elsewhere then the term *person* has to be replaced by its definition before any consistency and subsumption test is tried.

Predicate symbols do not interact with the arithmetic expressions. Therefore the arithmetic algorithms and the algorithms for the predicate symbols (usually some kind of propositional reasoning) are independent of each other.

1.4 Universal Quantification

In the description logic context, quantification means quantification over role fillers. For example:

$$\mathit{wooden\text{-}toy} \stackrel{\text{def}}{=} \mathit{toy} \wedge \forall\, \mathit{has\text{-}part.wooden}$$

defines a wooden toy as a toy whose parts (role fillers for the *has-part* relation) are all wooden. $\forall\, \mathit{has\text{-}part.wooden}$ denotes the set of objects, whose parts are all in the set of wooden objects. The modal logic version of this definition would be

$$\mathit{wooden\text{-}toy} \Leftrightarrow (\mathit{toy} \wedge [\mathit{has\text{-}part}]\mathit{wooden}).$$

The universal quantifiers over role terms with set constructors, or role names, which are related to other role names via some propositional axioms, must be decomposed into their atomic components. For example if *has-child* is decomposed into *has-son* and *has-daughter* then $\forall\, has\text{-}child.\varphi$ is decomposed into $\forall\, has\text{-}son.\varphi \wedge \forall\, has\text{-}daughter.\varphi$. Decomposed quantifications over the same roles can be comprised into one single quantification. $((\forall\, r.\varphi \wedge \forall\, r.\psi) \Leftrightarrow \forall\, r.(\varphi \wedge \psi))$ This way, all interactions between different universal quantifications are eliminated.

Universal quantification over role fillers interact in a relatively simple way with the arithmetic expressions over role terms. If φ is inconsistent then $\forall\, r.\varphi$ can only represent a non-empty set if there are no r-role fillers at all. Thus, $\forall\, r.\bot \Leftrightarrow |r| = 0$. The consistency check therefore first checks the arguments of universal quantifiers, and adds $|r| = 0$, if necessary.

As another example, consider

$$\forall\, has\text{-}child.teacher \wedge \forall\, has\text{-}daughter.\neg teacher.$$

The decomposition yields

$$\forall\, has\text{-}son.teacher \wedge \forall\, has\text{-}daughter.teacher \wedge \forall\, has\text{-}daughter.\neg teacher$$

which is comprised to

$$\forall\, has\text{-}son.(teacher \wedge \neg teacher) \wedge \forall\, has\text{-}daughter.\neg teacher$$

and then simplified to

$$|has\text{-}son| = 0 \wedge \forall\, has\text{-}daughter.\neg teacher.$$

1.5 Existential Quantification

It turns out that the existential quantifier over role fillers becomes definable in the language providing role hierarchies, restrictions on the number of role fillers and the universal quantifier:

$$\exists\, r.\psi = \exists r'(r' \Rightarrow r) \wedge |r'| \geq 1 \wedge \forall\, r'.\psi$$

That means for each occurrence $\exists\, r.\psi$ of an existential quantifier, one can introduce a new (Skolemized) role name r' (which relates a subset of those r role-fillers lying in ψ) and add the axiom $r' \Rightarrow r$ to the role hierarchy. The actual occurrence of $\exists\, r.\psi$ is replaced with $|r'| \geq 1 \wedge \forall\, r'.\psi$.

1.6 Disjunction and Negation

The algorithms presented below are organized in such a way that disjunctions can be treated by putting the concept formulae into disjunctive normal form and treating each disjunct separately.

In the presence of conjunction and disjunction together with both quantifiers, negation can be moved down to the propositional level (negation normal form). Therefore no special treatment is necessary for general negation of concept formulae.

1.7 Defined Operators

The arithmetic language together with role hierarchies and the standard connectives and quantifiers are expressive enough to define other useful operators.

1.7.1 Qualified Number Restrictions

$atleast n r.\varphi$ (the set of objects with at least n role fillers in φ) and $atmost n r.\varphi$ (the set of all objects with at most n role fillers in φ) are the *qualified number restrictions*. In our system they can be treated as defined operators:

$$atleast\, n\, r.\varphi \Leftrightarrow \exists r'\; (r' \Rightarrow r) \wedge |r'| \geq n \wedge \forall\, r'.\varphi$$

$$atmost\, n\, r.\varphi \Leftrightarrow \exists r'\; (r' \Rightarrow r) \wedge |r'| \leq n \wedge \forall\, r'.\varphi \wedge \forall\, (r \setminus r').\neg\varphi.$$

Again, the new (Skolemized) role names r' together with the sub-role definition $r' \Rightarrow r$ are added to the role hierarchy. The occurrences of the *atleast* and *atmost* formulae are replaced with the numeric constraints and the universal quantifications only.

1.7.2 Percentage Operators

Operators like $\geq n\% r\varphi$ (set of objects with more than n% of the r-role fillers in φ) become also definable:

$$\geq\; n\%\, r\varphi \Leftrightarrow \exists r'\; (r' \Rightarrow r) \wedge 100|r'| \geq n|r| \wedge \forall\, r'.\varphi.$$

In the same way one can define a '$\leq\; n\,\%$' operator or a '$n\,\%$' operator or a '*most* $r\ \varphi$' operator (more than 50%).

1.7.3 The 'More' Operator

more $r\varphi$ $s\psi$ denotes the set of objects with more r-role fillers in φ than s role-fillers in ψ. For example *more has-daughter blonde has-son brown* denotes the set of objects with more blonde daughters than brown sons. A definition for this operator is:

$$\begin{aligned} \mathit{more}\, r\varphi\; s\psi \Leftrightarrow\ & \exists r', s'\ (r' \Rightarrow r) \wedge (s' \Rightarrow s) \wedge |r'| \geq |s'| \wedge \\ & \forall\, r'.\varphi \wedge \forall\, s'.\psi \wedge \forall\, (s \setminus s').\neg\psi. \end{aligned}$$

In the above example, r' would be the blonde daughters and s' would be the brown sons. $|r'| \geq |r|$ requires that there are more of the blonde daughters than brown sons.

1.7.4 The 'Many' Operator

The meaning of the operator *many* in, for example, *many has-child.teacher* (set of objects with many children which are teacher) is not clear. If there is just one child, then *many* certainly should be 100%. If there are some hundred children, then *many* might only mean a small fraction. Our language is expressive enough that we need not assume a fixed meaning of *many*, but can leave it to the user to define her version of *many*. A possible definition might be

$$\begin{aligned} \mathit{many}\, r.\varphi \Leftrightarrow\ & \exists r'\ (r' \Rightarrow r) \wedge \forall\, r'.\varphi \wedge \\ & \begin{array}{lll} |r| \leq 2 & \Rightarrow |r'| = |r| & \wedge \\ 3 \leq |r| \leq 10 & \Rightarrow |r'| \geq 0.9|r| & \wedge \\ 11 \leq |r| \leq 100 & \Rightarrow |r'| \geq 0.5|r| & \wedge \\ 101 \leq |r| & \Rightarrow |r'| \geq 90. & \end{array} \end{aligned}$$

1.8 Other Operators

There are quite a number of other operators discussed in the description logic literature. We have not yet investigated in detail how these fit into our framework. For example the role composition operator extends the Boolean language of role terms to the language or relation algebras. Since the whole approach relies on the decomposition method, and this relies on Stone's representation theorem for Boolean algebras, an extension of the decomposition method to relation algebras is by no means straightforward and yet has to be done.

In the following sections we work out the technical details of the method and we prove soundness and completeness of the algorithms.

2 Atomic Decomposition

We list the basic definitions and results. The details can be found in [21]. The presentation of the method is independent of any particular application in description logics.

2.1 Syntax of the Languages Involved

We need 3 components in the syntax. The first component of our syntax is the language $\mathcal{L}_E$ of some basic system E which we want to augment with a Boolean component. In our case E are systems of arithmetical equations and in-equations, but E may be any other suitable formal system.

The second component is the Boolean algebra component. *Boolean terms* $\mathcal{L}_B(\mathcal{R})$ are set terms over a set $\mathcal{R}$ of Boolean variables, constructed with the usual set connectives $\cup$ (union), $\cap$ (intersection), $'$ (complement), $\setminus$ (set difference), etc. In the description logic case, $\mathcal{R}$ is the set of role names.

As a bridge between the two languages $\mathcal{L}_E$ and $\mathcal{L}_B(\mathcal{R})$ we need a distinguished set $\mathcal{B}$ of *bridging functions*, different from all other symbols involved. A typical example for a bridging function is the cardinality function mapping sets to integers. A bridging function symbol may have any finite arity. Each argument position, however, can take either a Boolean term as argument, or an $\mathcal{L}_E$-term. For convenience, we assume that a bridging function of arity $n + k$ reserves the first n arguments for Boolean terms and the remaining k arguments for $\mathcal{L}_E$-terms.

The combined language is defined as follows:

Definition 1 (The combined language $\mathcal{L}_{BE}(\mathcal{R}, \mathcal{B})$)
If $s[x] \in \mathcal{L}_E$ where x is some term occurring at some positions in s, $f \in \mathcal{B}$ with arity $n + k$, $t_1, \ldots, t_n \in \mathcal{L}_B(\mathcal{R})$, $s_1, \ldots, s_k \in \mathcal{L}_{BE}(\mathcal{R}, \mathcal{B})$ then $s[x/f(t_1, \ldots, t_n, s_1, \ldots, s_k)] \in \mathcal{L}_{BE}(\mathcal{R}, \mathcal{B})$. No other terms are in $\mathcal{L}_{BE}(\mathcal{R}, \mathcal{B})$. $\square$

$\mathcal{L}_{BE}(\mathcal{R}, \mathcal{B})$ is essentially like $\mathcal{L}_E$, but $\mathcal{L}_E$-term positions can be occupied by $\mathcal{L}_B(\mathcal{R})$-terms embedded in a bridging function. The combined language is such that the Boolean parts and the $\mathcal{L}_E$-parts are separated by the functions in $\mathcal{B}$.

Example 2 *E = arithmetic, $\mathcal{B} = \{|.|, f, a, c\}$ with the informal meaning: $|.|$ is the set cardinality function, f means 'combined fortune', a means 'average income' and the binary function c means 'consumption of'. Well formed $\mathcal{L}_{BE}(\mathcal{R}, \mathcal{B})$ axioms are now:*

$|sons \cup friends| \geq 5$
(There are more than 5 sons and friends.)

$f(children) \geq 100000$
(The combined fortune of the children exceeds 100000.)

$a(daughters) = 10000$
(The average income of the daughters is 10000.)

$c(children \cup friends, cigarettes) = 0$
(The children and friends do not smoke cigarettes.) □

2.2 Semantics of the Languages Involved

The language $\mathcal{L}_E$ comes with its natural semantics. The only feature we need is that an interpretation $\Im_E$ for E maps the free variables and constant symbols to elements of E's domain and interprets function symbols as functions in the usual way. In an arithmetical language, $\Im_E$ may, for example, represent a solution of an equation system.

The language $\mathcal{L}_B(\mathcal{R})$ is to be interpreted as a complete and atomic Boolean algebra usually, but not necessarily, as a set algebra.

The interpretation is therefore a homomorphism $\Im_B : \mathcal{L}_B(\mathcal{R}) \mapsto A$ where A is a complete and atomic Boolean algebra. [5]

Since the language $\mathcal{L}_B(\mathcal{R})$ and $\mathcal{L}_E$ do not share any symbols, we can define a combined interpretation $\Im_{BE}$ as the union of the interpretations $\Im_B$ and $\Im_E$. The interpretation of the bridging function symbols also becomes part of $\Im_{BE}$.

The interpretation of the bridging function symbols in $\mathcal{B}$ can, but need not be fixed. It must, however, satisfy the additivity axioms (Def. 3) and it must be type conform. That means for a bridging function symbol f with n Boolean arguments and m $\mathcal{L}_E$-arguments, a combined interpretation $\Im_{BE}(f)$ must map tuples consisting of n elements of the Boolean algebra and m elements of E's domain to an element of E's domain.

[5] In many practical applications, A is even finite. We have not investigated the case where A is not complete or not atomic.

Definition 3 (Additivity axioms)
The additivity axioms for a bridging function $f \in F$ with arity $n+k$ are:

$$\begin{aligned} x \cap y = \emptyset \rightarrow\ & f(\ldots, t_{i-1}, x \cup y, t_{i+1}, \ldots) \\ & = g_i(f(\ldots, t_{i-1}, x, t_{i+1}, \ldots), f(\ldots, t_{i-1}, y, t_{i+1}, \ldots)) \end{aligned}$$

for each $i \in \{1, \ldots, n\}$ of the Boolean argument positions, where $g_i(x, y)$ is some term in $\mathcal{L}_E$. □

Definition 4 (Problem specification)
A problem specification $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B}, \varphi)$ *now consists of 3 parts*

(1) a set $\mathcal{A}_\mathcal{R}$ of propositional axioms over the Boolean symbols $\mathcal{R}$,
(2) the bridging function additivity axioms $\mathcal{A}_\mathcal{B}$ (Def. 3),
(3) a $\mathcal{L}_{BE}(\mathcal{R}, \mathcal{B})$ formula φ.

The satisfiability *problem is to find out whether such a specification is consistent, i.e. whether it has an interpretation satisfying all 3 parts.* □

The propositional axioms $\mathcal{A}_\mathcal{R}$ can have an ordinary propositional interpretation where the Boolean variables in $\mathcal{R}$ are mapped to binary truth values, or they can have a more general Boolean algebra interpretation, where the Boolean variables in $\mathcal{R}$ are mapped to the elements of the Boolean algebra. In the description logic case the desired interpretation is set theoretic, where the Boolean variables in $\mathcal{R}$ are mapped to sets of role fillers. To explain the exact correlation between these different kinds of interpretations, some basic Boolean algebra theory (ultrafilters) is necessary. One can prove that for any kind of atomic Boolean algebra interpretation of $\mathcal{A}_\mathcal{R}$, there is a one-to-one correspondence between the propositional models m of $\mathcal{A}_\mathcal{R}$ ($m \models \mathcal{A}_\mathcal{R}$), and the atoms of the Boolean algebra. That means a syntactic representation (the *syntactic atoms*) of $\mathcal{A}_\mathcal{R}$'s models can be used to represent the atoms of the Boolean algebra.

Therefore for every Boolean term t and set theoretic interpretation $\Im$:

$$\varphi^\Im = \bigcup_{m \models (\varphi \wedge \mathcal{A}_\mathcal{R})} m^\Im \tag{9}$$

where $m^\Im$ means the set theoretic interpretation of the atom corresponding to the propositional model m. This is the basis for a sequence of transformations which eliminate the Boolean terms and the bridging functions from a problem specification $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B}, \varphi)$ and reduce the satisfiability problem to a problem in the basic language $\mathcal{L}_E$.

Definition 5 (Atomic decomposition) *Given a problem specification* $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B}, \varphi)$ *we define the following sequence of transformations.*

(1) Atomic decomposition of Boolean terms:

$$\begin{array}{lll}
\alpha_{\mathcal{A}_\mathcal{R}}(p) & = \{m_1, \ldots, m_n\} & \text{where } p \text{ is a Boolean variable} \\
\alpha_{\mathcal{A}_\mathcal{R}}(x \cup y) & \stackrel{\text{def}}{=} \alpha_{\mathcal{A}_\mathcal{R}}(x) \cup \alpha_{\mathcal{A}_\mathcal{R}}(y) & \text{and } m_i \models p \wedge \mathcal{A}_\mathcal{R} \\
\alpha_{\mathcal{A}_\mathcal{R}}(x \cap y) & \stackrel{\text{def}}{=} \alpha_{\mathcal{A}_\mathcal{R}}(x) \cap \alpha_{\mathcal{A}_\mathcal{R}}(y) & \\
\alpha_{\mathcal{A}_\mathcal{R}}(x') & \stackrel{\text{def}}{=} \alpha_{\mathcal{A}_\mathcal{R}}(\top) \setminus \alpha_{\mathcal{A}_\mathcal{R}}(x) & \\
 & \alpha_{\mathcal{A}_\mathcal{R}}(\top) \text{ is the set of all models of } \mathcal{A}_\mathcal{R}. &
\end{array}$$

Notice that the set connectives at the left-hand side are just term building functions, whereas at the right-hand side, the real set operations are meant.

For a $\mathcal{L}_{BE}(\mathcal{R}, \mathcal{B})$*-formula* φ *let* $\alpha_{\mathcal{A}_\mathcal{R}}(\varphi)$ *be the result of applying the decomposition function* $\alpha_{\mathcal{A}_\mathcal{R}}$ *to all Boolean terms occurring in* φ.

(2) Elimination of additivity axioms:
For a decomposed $\mathcal{L}_{BE}(\mathcal{R}, \mathcal{B})$*-formula* $\alpha_{\mathcal{A}_\mathcal{R}}(\varphi)$ *let* $\alpha_{\mathcal{A}_\mathcal{B}}(\alpha_{\mathcal{A}_\mathcal{R}}(\varphi))$ *be the result of an exhaustive left to right application of the equations in* $\mathcal{A}_\mathcal{B}$ *to* $\alpha_{\mathcal{A}_\mathcal{R}}(\varphi)$*, taking the sets* $\{m_1, \ldots, m_k\}$ *as union terms* $m_1 \cup \ldots \cup m_k$.

(3) Elimination of the bridging functions:
Finally we define a replacement operation on $\alpha_{\mathcal{A}_\mathcal{B}}(\alpha_{\mathcal{A}_\mathcal{R}}(\varphi))$ *which replaces all bridging function symbols* f *with* n *Boolean and* k $\mathcal{L}_E$*-arguments by corresponding* $\mathcal{L}_E$*-terms.* β *introduces for each term* $f(m_1, \ldots, m_n, \ldots)$ *a new* $\mathcal{L}_E$*-function (or constant) symbol* $f'_{m_1,\ldots,m_n}$ *and replaces terms* $f(m_1, \ldots, m_n, s_1, \ldots, s_k)$ *with* $f'_{m_1,\ldots,m_n}(s_1, \ldots, s_k)$.

Let $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi)$ *be the result of this replacement to* $\alpha_{\mathcal{A}_\mathcal{B}}(\alpha_{\mathcal{A}_\mathcal{R}}(\varphi))$. □

Theorem 6 (Soundness and completeness)
A problem specification $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B}, \varphi)$ *is satisfiable (falsifiable) if and only if i)* $\mathcal{A}_\mathcal{R}$ *is satisfiable and ii) the transformed formula* $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi)$ *is satisfiable (falsifiable) in the basic system* $\mathcal{L}_E$. □

The inference procedure derived from this theorem comprises the following steps: in order to check satisfiability of a combined specification $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B}, \varphi)$, first compute the *syntactic atoms* derived from the propositional models of $\mathcal{A}_\mathcal{R}$. If there are no models then the specification is unsatisfiable. If there are models, decompose the Boolean terms occurring in φ into sets of syntactic atoms. Use the additivity axioms in $\mathcal{A}_\mathcal{B}$ to push the bridging functions down to the level of single atoms. Then replace the resulting 'bridging terms' with variables or composed $\mathcal{L}_E$-terms, and check the result with an E-satisfiability checker.

If satisfiability in E is decidable we get a decision procedure for the combination with the Boolean language. Satisfiability for this combination is then decidable as well.

2.3 Optimizations

A formula with l Boolean variables may, in the worst case, have 2^l models. For all of them one has to generate syntactic atoms. This makes the whole approach questionable. Fortunately there are some optimizations which can reduce the number of syntactic atoms considerably.

Relevancy Principle

A Boolean variable p occurring in the Boolean axioms $\mathcal{A}_\mathcal{R}$ of some problem specification $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B}, \varphi)$, but not in φ does not contribute much to the problem solution. Boolean variables are implicitly existentially quantified. That means $\mathcal{A}_\mathcal{R}$ is in fact short for $\exists p\ \mathcal{A}_\mathcal{R}$ if p does not occur in φ. $\mathcal{A}_\mathcal{R}$ is a conjunction of propositional formulae, and therefore the existentially quantified p can be eliminated using a quantifier elimination procedure [15]. The result is some formula $\mathcal{A}_\mathcal{R}{}'$ which is equivalent to $\exists p\ \mathcal{A}_\mathcal{R}$, but does not contain p. In the propositional case, elimination of the existentially quantified p amounts to generating all resolvents with p in the clause form of $\mathcal{A}_\mathcal{R}$. The resolvents represent all consequences of p and therefore p is no longer necessary [1].

This way one can have large databases of Boolean axioms, but for the actual problem at hand, the atomic decomposition takes into account only the relevant Boolean variables.

Factoring Principle

A Boolean axiomatization $\mathcal{A}_\mathcal{R}$ which can be split into separate parts $\mathcal{A}_{\mathcal{R}1}, \ldots,$ $\mathcal{A}_{\mathcal{R}r}$ such that the parts mutually do not share Boolean variables, simplifies the atomic decomposition as well. The set of propositional models for $\mathcal{A}_\mathcal{R}$ can be factored into the product $M_1 \times \ldots \times M_r$ of the set of models for the $\mathcal{A}_{\mathcal{R}i}$. This means that the algebra A, which is the image of the set theoretic interpretation of $\mathcal{A}_\mathcal{R}$ can be factored into the product $A_1 \times \ldots \times A_r$ of algebras. The atoms of such a product have the form $(\ldots, \perp_{i-1}, a_i, \perp_{i+1}, \ldots)$ where a_i is an atom of A_i and all other components are the bottom elements of the other algebras.

If there is no other information about intersections or unions of Boolean terms from different factors of the product, then this can be exploited for the representation of the syntactic atoms. They can have the form $(\ldots, \perp, m_i, \perp, \ldots)$ where m_i is a syntactic atom of the component M_i. A further simplification is possible by just storing m_i and labelling it with the information 'belongs to M_i'.

This reduces the overall number of syntactic atoms from $|M_1| \cdot \ldots \cdot |M_r|$ to $|M_1| + \ldots + |M_r|$, which is an exponential improvement.

The meaning of this simplification also makes sense from an application point of view. As an illustration, consider some $\mathcal{A}_{\mathcal{R}}$ axiomatizing, say family relationships, and in addition relationships between the makes of cars. If there are no axioms saying something about the intersection of people and cars, then the factoring operation implicitly imposes that there is no object which is at the same time a person and a car. Therefore the whole Boolean algebra is split into the part with sets of people and the part with sets of cars. People and cars together are represented by tuples in the product algebra. On the calculus side we therefore get syntactic atoms which represent either people or cars, but none for the intersection of both.

3 Arithmetic Constraints for Role Fillers

We define different Description Logic languages, starting with a purely arithmetical part, and then including more operators. With all of them one can define *concepts*

$$c \stackrel{\text{def}}{=} \varphi$$

where c is a *concept name* and φ a *concept formula* in the corresponding language (cf. (1) or (2) or (3)).

One important restriction on concept definitions is that the equations $c \stackrel{\text{def}}{=} \varphi$ can be used as rewrite rules from left to right such that the rewriting operation terminates. The rewritten concept definitions are in *expanded normal form*. For example the expanded normal form for the two concept definitions

$$\begin{aligned}
&\mathit{parent} && \stackrel{\text{def}}{=} \mathit{person} \wedge |\mathit{has\text{-}child}| \geq 1 \\
&\mathit{grandparent} && \stackrel{\text{def}}{=} \mathit{parent} \wedge \exists\, \mathit{has\text{-}child.parent}
\end{aligned}$$

is

$$\begin{aligned}
&\mathit{parent} && \stackrel{\text{def}}{=} \mathit{person} \wedge |\mathit{has\text{-}child}| \geq 1 \\
&\mathit{grandparent} && \stackrel{\text{def}}{=} \mathit{person} \wedge |\mathit{has\text{-}child}| \geq 1\ \wedge \\
& && \quad\ \exists\, \mathit{has\text{-}child.}(\mathit{person} \wedge |\mathit{has\text{-}child}| \geq 1).
\end{aligned}$$

Truly recursive concept definitions, where the rewriting does not terminate, are possible, but they require a different approach to the one presented in this paper [20,23]. Therefore we always assume that the concept formulae are in expanded normal form. (Since the expanded normal form may be exponential, a clever implementation needs to avoid the expansion.)

The atomic decomposition technique is already a framework for a first version of a description logic. The basis is an arithmetic equation solving or a mathematical programming system. This system is combined with Boolean role terms. Let us call it DL_{ar} (Description Logic with arithmetics and role terms).

Definition 7 (DL_{ar}-syntax) *The language primitives consist of a set $\mathcal{R}$ of* role names*, a set $\mathcal{C}$ of* concept names *and a set $\mathcal{B}$ of* bridging functions*. $\mathcal{B}$ contains the set cardinality function $|\ldots|$.*

A DL_{ar}-basis $(\mathcal{A}_{\mathcal{R}}, \mathcal{A}_{\mathcal{B}})$ consists of

(1) a finite set $\mathcal{A}_{\mathcal{R}}$ of propositional axioms for the role names in $\mathcal{R}$, [6]
(2) a finite set $\mathcal{A}_{\mathcal{B}}$ of additivity axioms for the bridging functions in $\mathcal{B}$, [7] □

The DL_{ar}-semantics semantics interprets role terms as binary relations and concept formulae as sets of objects in some domain.

Definition 8 (DL_{ar}-semantics) *We assume an interpretation $\Im$ over some non-empty domain $\mathcal{D}_\Im$ for a DL_{ar}-basis to interpret the arithmetic parts of the language in the natural way (see Section 2.2), and to interpret the bridging functions also in a natural way, such that the bridging axioms are satisfied. The definitions specific to DL_{ar} are:*

- $r^\Im \subseteq \mathcal{D}_\Im \times \mathcal{D}_\Im$ *for every role name* $r \in \mathcal{R}$,
- $r^{\Im_x} \stackrel{\text{def}}{=} \{y \mid (x, y) \in r^\Im\}$ *for every* $x \in \mathcal{D}_\Im$ *and* $r \in \mathcal{R}$,
- *if f is a bridging function of arity $n + k$ and $r_1, \ldots, r_n$ are role terms then* $(f(r_1, \ldots, r_n, s_1, \ldots, s_k))^{\Im_x} \stackrel{\text{def}}{=} f^\Im(r_1^{\Im_x}, \ldots r_n^{\Im_x}, s_1^\Im, \ldots, s_k^\Im)$.

A concept formula φ is consistent *(or* satisfiable*) if $\varphi^\Im \neq \emptyset$ for some interpretation $\Im$.*

A concept formula φ_1 subsumes *(entails) a concept formula φ_2 iff $\varphi_1^\Im \subseteq \varphi_2^\Im$ for all interpretations $\Im$.* □

[6] $\mathcal{A}_{\mathcal{R}}$ specifies the set theoretic relationships between the roles, usually, but not necessarily a subset hierarchy (the 'role hierarchy').
[7] It might be necessary to include further axioms in $\mathcal{A}_{\mathcal{B}}$. For example the correlation between the cardinality function and, say, a bridging function *average-income* may be $\forall x\ |x| = 0 \Rightarrow$ *average-income*$(x) = 0$. It is quite straightforward to translate axioms like this into the language of the underlying arithmetic system.

The problem of checking consistency of a concept formula φ can be solved by checking the problem specification $(\mathcal{A}_{\mathcal{R}}, \mathcal{A}_{\mathcal{B}}, \varphi)$ (Def. 4) for consistency with the atomic decomposition method.

The problem of checking subsumption between φ_1 and φ_2 can be solved by checking the problem specification $(\mathcal{A}_{\mathcal{R}}, \mathcal{A}_{\mathcal{B}}, \varphi_1 \wedge \neg\varphi_2,)$ for consistency. (We assume the underlying arithmetic algorithms can deal with negated formulae.)

Atomic decomposition has previously been used to develop inference algorithms for description logics, for example in [9,8]. In their approach, the concepts themselves are decomposed, not the roles. Since the technique was applied to a different logic (with inverse roles and arbitrary terminological axioms), one cannot compare the two approaches directly.

4 Concept Formulae with Concept Names

The first extension of the language DL_{ar} is DL_{arc}, where we allow for propositional formulae over concept names to occur in concept terms.

Definition 9 (The language DL_{arc}) *The language $DL_{arc}$is like DL_{ar}, but concept formulae φ may be of the form $\varphi_a \wedge \varphi_c$ where φ_a is an arithmetical concept term of the language DL_{ar} and φ_c is a propositional formula over concept names.*

The DL_{ar} semantics works for DL_{arc} as well if concept names are mapped to subsets of the domain and $\varphi_c^{\Im}$ is the set theoretic interpretation of φ_c. □

An example is

$$\textit{busy-academic} \stackrel{\text{def}}{=} |\textit{has-courses}| + |\textit{has-projects}| \geq 2 \wedge (\textit{staff-member} \vee \textit{researcher})$$

$|\textit{has-courses}| + |\textit{has-projects}| \geq 2$ is the arithmetic part (φ_a) and $(\textit{staff-member} \vee \textit{researcher})$ the purely propositional part (φ_c).

Since the arithmetic part and the concept name part do not share any symbols, consistency and subsumption can be checked separately.

Proposition 10 (Consistency and subsumption check for DL_{arc})
A DL_{arc}-concept formula $\varphi_a \wedge \varphi_c$ is consistent iff φ_a is consistent, which can be checked with the DL_{ar} consistency check, and φ_c is consistent, which can be checked with a propositional satisfiability checker.

A DL_{arc}-concept formula $\varphi_a \wedge \varphi_c$ subsumes a concept formula $\psi_a \wedge \psi_c$ iff φ_a subsumes ψ_a, which can be checked with the DL_{ar}-subsumption algorithm, and

φ_c entails ψ_c as propositional formulae, which can be checked with a propositional satisfiability checker. $\square$

Since φ_a and φ_c do not share any non-logical symbols, the proofs are straightforward.

5 Concept Formulae with Universal Quantifiers

Universal quantification $\forall\, r.\varphi$ expresses properties of role fillers (all r-role fillers of a given object x must lie in the concept φ).

Definition 11 (The language $DL_{arc\forall}$) *The language $DL_{arc\forall}$ is like the language DL_{arc}, but concept formulae φ may be of the kind $\varphi_a \wedge \varphi_c \wedge \varphi_\forall$ where $\varphi_a \wedge \varphi_c$ is a DL_{arc}-concept formulae and $\varphi_\forall$ is a conjunction of quantifications $\forall\, r.\psi$ where r is a role term and ψ is a $DL_{arc\forall}$-concept formula.*

The semantics of $\forall\, r.\varphi$ is

$$(\forall\, r.\psi)^{\Im} \stackrel{\text{def}}{=} \{x \in \mathcal{D}_{\Im} \mid \forall y\ r^{\Im}(x,y) \Rightarrow y \in \psi^{\Im}\}.$$

($\forall\, r.\psi$ denotes the set of objects all whose r-role fillers are in ψ.) $\square$

Notice that if φ_a consists of conjunctions of expressions of the form $|r| \geq n$ or $|r| \leq n$, where n is an integer, φ_c is a conjunction of concept names, and role terms consist of role names only, and the role hierarchy $\mathcal{A}_{\mathcal{R}}$ is empty then this language is the logic $\mathcal{TF}$ [19,14].

If the atomic decomposition of the role term r is $\{a_1, \ldots, a_n\}$ then

$$\forall\, r.\psi \Leftrightarrow (\forall\, a_1.\psi \wedge \ldots \wedge \forall\, a_n.\psi). \tag{10}$$

Therefore the $\varphi_\forall$-part of a $DL_{arc\forall}$-concept formula can be normalized such that

$$\varphi_\forall = (\forall\, a_1.\varphi_1 \wedge \ldots \wedge \forall\, a_n.\varphi_n)$$

where the a_i are symbolic atoms. For example

$$\forall\, \textit{has-child.teacher} \wedge \forall\, \textit{has-son.male}$$

is normalized to

$$\forall\, s.(\textit{teacher} \wedge \textit{male}) \wedge \forall\, d.\textit{teacher}$$

if *has-child* is decomposed into $\{s, d\}$.

Concept formulae φ which denote the whole domain in all interpretations, i.e. $\varphi^{\Im} = \top^{\Im}$ for all interpretations $\Im$ are useless tautologies and should be eliminated. We give a necessary and sufficient criterion for recognizing them.

Theorem 12 (Tautology) *A $DL_{arc\forall}$-concept formula $\varphi = \varphi_a \wedge \varphi_c \wedge \varphi_\forall$ where φ_a is the arithmetic part, φ_c is the propositional part and $\varphi_\forall$ are the universal quantifications, is a tautology over a $DL_{arc\forall}$-Basis $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B})$, (written $\models \varphi$ [8]), iff*

i) $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a)$ is a tautology in the basic system $\mathcal{L}_E$ (i.e. all possible assignments to the variables are solutions),
ii) φ_c is a propositional tautology, and
iii) for all '$\forall\, r.\psi$' in $\varphi_\forall$, ψ is a tautology.

PROOF. If all three conditions are satisfied then φ is certainly a tautology. We show that if one of them is not satisfied then φ can be interpreted as a proper subset of the domain of some interpretation.

If $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a)$ is not a tautology, there is a falsifying model for $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a)$. By Theorem 6 there is a falsifying interpretation $\Im'$ for φ_a as well. $\Im'$ maps the symbolic atoms m to subsets of a domain. We now construct an interpretation $\Im$ for φ where $\mathcal{D}_\Im = \{x\} \cup \bigcup_{m\in\alpha_{\mathcal{A}_\mathcal{R}}(\top)} m^{\Im'}$ for some arbitrary element x and for all atoms m: $m^\Im \stackrel{\text{def}}{=} \{(x, b) \mid b \in m^{\Im'}\}$. This means the m-successors of x are just those elements assigned to m by $\Im'$. This way we get an interpretation where the role fillers of x falsify φ_a. Thus, φ is not a tautology.

If φ_c is not a propositional tautology we can certainly find an interpretation where $\varphi^\Im$ is a proper subset of the domain.

If for some '$\forall\, r.\psi$' in $\varphi_\forall$: ψ is not a tautology, there is an interpretation $\Im$ and some domain element $b \notin \psi^\Im$. In the same way as in the first case we construct an interpretation $\Im'$ with b as r-successor of some new element x. Then it is not the case that for all r-successors of x, ψ holds, which means that φ is not a tautology. □

Definition 13 (Decomposition of universal quantifications)
If $\varphi = \bigwedge_{r\in R} \forall\, r.\psi_r$ is a concept formula *over a DL_{ar}-basis $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B})$, where R is a set of role terms, we define*

$$\alpha_{\mathcal{A}_\mathcal{R}}(\varphi) \stackrel{\text{def}}{=} \bigwedge_{m\in\bigcup_{r\in R}\alpha_{\mathcal{A}_\mathcal{R}}(r)} \forall\, m.\alpha_{\mathcal{A}_\mathcal{R}}(\varphi, m).$$

where

$$\alpha_{\mathcal{A}_\mathcal{R}}(\varphi, m) \stackrel{\text{def}}{=} \bigwedge_{r\in R: m\in\alpha_{\mathcal{A}_\mathcal{R}}(r)} \psi_r.$$

□

[8] We use the symbol $\models$ as a binary relation $\Im \models \varphi$ (the interpretation $\Im$ satisfies φ) and as a predicate $\models \varphi$ (φ is true in all models).

Lemma 14 *The decomposition of the universal quantifications (Def. 13) is equivalence preserving. That means $\varphi^{\Im} = (\alpha_{\mathcal{A}_\mathcal{R}}(\varphi))^{\Im}$ for all interpretations satisfying $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B})$.* □

PROOF. The lemma is a consequence of (9) and (10) and $(\forall\, r.\varphi \wedge \forall\, r.\psi) \Leftrightarrow \forall\, r.(\varphi \wedge \psi)$:

First the universal quantifications in $\varphi = \forall\, r_1.\psi_1 \wedge \ldots \wedge \forall\, r_n.\psi_n$ are decomposed into their atomic parts:

$\forall\, m_{r_1 1}.\psi_1 \wedge \ldots \wedge \forall\, m_{r_1 k_1}.\psi_1 \ldots \wedge \forall\, m_{r_n 1}.\psi_n \wedge \ldots \wedge \forall\, m_{r_n k_n}.\psi_n$

where $m_{r_i 1}, \ldots, m_{r_i k_i}$ are the atomic components of r_i for $1 \leq i \leq n$. This is a equivalence preserving transformation (10). Then all quantifications $\forall\, m.\psi_i$ with the same role m are collected in one single quantification with a conjunction of all the relevant ψ_i. This is also an equivalence preserving transformation. And that is the result of the normal form. □

Universal quantification over an empty set is a tautology, therefore $\forall\, r.\varphi$ is satisfiable even if φ is inconsistent. Consequently, if φ is inconsistent then the set or r-role fillers must be empty. $|r| = 0$ can be added in this case. These observations lead to the following normal form for $DL_{arc\forall}$-concept formulae.

Definition 15 ($DL_{arc\forall}$-normal form) *Let $\varphi = \varphi_a \wedge \varphi_c \wedge \varphi_\forall$ be a $DL_{arc\forall}$-concept formula over a $DL_{arc\forall}$-basis $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B})$, where φ_a is the arithmetic part, φ_c the concept name part and $\varphi_\forall$ the universal quantifications (all parts may be empty).*

The normal form $NF_{arc\forall}(\varphi)$ of φ is

$$NF_{arc\forall}(\varphi) \stackrel{\text{def}}{=} \begin{cases} \alpha'_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a, \varphi_\forall) \wedge \varphi_c \wedge \alpha'_{\mathcal{A}_\mathcal{R}}(\varphi_\forall) & \text{if } \alpha'_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi) \text{ and } \varphi_c \\ & \text{are consistent} \\ \bot & \text{otherwise} \end{cases}$$

where

$$\alpha'_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a, \varphi_\forall) \stackrel{\text{def}}{=} \alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a) \wedge \bigwedge_{`\forall m.\psi'\text{'} \in \alpha_{\mathcal{A}_\mathcal{R}}(\varphi_\forall, m), NF_{arc\forall}(\psi') = \bot} x_m = 0$$

and

$$\alpha'_{\mathcal{A}_\mathcal{R}}(\varphi_\forall) \stackrel{\text{def}}{=} \bigwedge_{`\forall m.\psi'\text{'} \in \alpha_{\mathcal{A}_\mathcal{R}}(\varphi_\forall, m), \psi = NF_{arc\forall}(\psi') \neq \bot, \alpha'_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi) \not\models (x_m = 0), \not\models \psi} \forall\, m.\psi.$$ □

$\alpha'_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a, \varphi_\forall)$ in the normal form $NF_{arc\forall}(\varphi)$ is the arithmetic part. It consists of the original decomposed arithmetic part $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a)$ (Def. 5,3) where the role terms have been replaced by the corresponding arithmetic terms, plus some equations $x_m = 0$ where x_m is the variable introduced for $|m|$. These equations come from quantifications $\forall\, a.\psi$ with inconsistent ψ, and therefore there cannot be any m-successors. $\alpha'_{\mathcal{A}_\mathcal{R}}(\varphi_\forall)$ is the decomposed and reduced quantification part where all quantifications over empty atomic role components and all tautologies have been eliminated.

Theorem 16 (Soundness of the $NF_{arc\forall}$-normal form) *If the normal form $NF_{arc\forall}(\varphi) = \bot$ for a $DL_{arc\forall}$-concept formula $\varphi = \varphi_a \wedge \varphi_c \wedge \varphi_\forall$ over a DL_{ar}-basis $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B})$, then φ is inconsistent.*

PROOF. By induction on the number of nested universal quantifications: in the base case, $\varphi = \varphi_a \wedge \varphi_c$, either $\alpha'_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a)$ is inconsistent or φ_c is inconsistent. In the first case, φ_a must be inconsistent (Theorem 6), and in the second case φ is inconsistent anyway.

Induction step: If φ_c is inconsistent, then so is φ. If $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a)$ is inconsistent then φ_a is inconsistent (Theorem 6). If $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a)$ is consistent, but $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a) \wedge \bigwedge_{\text{`}\forall m.\psi'\text{'} \in \alpha_{\mathcal{A}_\mathcal{R}}(\varphi_\forall, m), NF_{arc\forall}(\psi') = \bot} x_m = 0$ is inconsistent, then $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a)$ forces the existence of an m-role filler for which the corresponding $NF_{arc\forall}(\psi') = \bot$, and therefore, by the induction hypothesis, ψ is inconsistent. Thus, we find some $\forall\, m.\psi$ in the decomposition of $\varphi_\forall$ with $\psi \Leftrightarrow \bot$ and non-empty m-role fillers. This makes $\varphi_\forall$ inconsistent. □

Theorem 17 (Completeness of the $NF_{arc\forall}$-normal form) *If the normal form $NF_{arc\forall}(\varphi) \neq \bot$ for a $DL_{arc\forall}$-concept formula $\varphi = \varphi_a \wedge \varphi_c \wedge \varphi_\forall$ over a DL_{ar}-basis $(\mathcal{A}_\mathcal{R}, \mathcal{A}_\mathcal{B})$, then for every non-empty set D there is an interpretation $\Im$ with $\varphi^\Im = D$ (which means in particular that φ is consistent).*

PROOF. By induction on the number of nested universal quantifications: for technical reasons, we assume that every propositional part φ_c at all levels of the formula φ has the form $p \wedge \varphi'_c$ where p is a single unique concept name, not occurring anywhere else in φ. This does not change the consistency or inconsistency of φ.

The base case of the induction is again $\varphi = \varphi_a \wedge \varphi_c$. Since $\alpha_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a)$ is consistent, there is an interpretation $\Im'$ for φ_a (Theorem 6). We define an interpretation $\Im$ where the atoms m in the atomic decomposition are interpreted as binary relations such that the m-role fillers of the elements in D are just $m^{\Im'}$. For all $x \in D$ let $m^{\Im_x} \stackrel{\text{def}}{=} m^{\Im'}$. For $p \wedge \varphi'_c$ we define $p^\Im \stackrel{\text{def}}{=} D$ and for each concept name q in φ_c with $q^{\Im'} = 1$ let $p^{\Im} \stackrel{\text{def}}{=} D$ and if $q^{\Im'} = 0$ let $p^{\Im} \stackrel{\text{def}}{=} \emptyset$. This guarantees $(\varphi_a \wedge \varphi_c)^{\Im} = D$.

Induction Step: Since $NF_{arc\forall}(\varphi) \neq \bot$, $\alpha'_{\mathcal{A_R},\mathcal{A_B}}(\varphi_a, \varphi_\forall) \wedge \varphi_c$ must be consistent. In the same way as in the base case we construct an interpretation $\Im_1$ with $(\varphi_a \wedge \bigwedge_{`\forall m.\psi'' \in \alpha_{\mathcal{A_R}}(\varphi_\forall, m), NF_{arc\forall}(\psi')=\bot} x_m = 0)^{\Im_1} = D$. Because of Lemma 14 it is sufficient to show that $\Im_1$ can be extended to some interpretation $\Im_k$ $\forall m.\psi'^{\Im_k} = D$ for all '$\forall m.\psi'$' in $\alpha_{\mathcal{A_R}}(\varphi_\forall)$.

If $m^{\Im_{1x}} = \emptyset$ for some (or all) $x \in D$ then $x \in (\forall m.\psi')^{\Im_1}$ because quantifications over empty sets are always true.

If $m^{\Im_{1x}} \neq \emptyset$ for some $x \in D$ then $NF_{arc\forall}(\psi') \neq \bot$ for '$\forall m.\psi'$' $\in \alpha_{\mathcal{A_R}}(\varphi_\forall, m)$ (because otherwise $x_m = 0$ would be in the arithmetic part). By the induction hypothesis there are some interpretations $\Im_x$ with $\psi'^{\Im_x} = m^{\Im_{1x}}$. We define a joint interpretation $\Im_2$ by requiring $m^{\Im_2} = m^{\Im_1} \cup \bigcup_x m^{\Im_x}$ for each atom m and $q^{\Im_2} = \bigcup_x q^{\Im_1} \cup q^{\Im_x}$ for each concept name q occurring in both φ_c and ψ.

$m^{\Im_1}$ and $m^{\Im_2}$ just contribute different levels to the binary relation associated with m. Since $\varphi_c = p \wedge \varphi'_c$ and p does not occur in ψ, $\varphi_c^{\Im_2} = \varphi_c^{\Im_1} = D$, and a similar statement is true for the propositional parts in ψ.

Repeating this process for each $\forall m.\psi'$ we end up with an interpretation $\Im_k$ with $\varphi^{\Im_k} = D$. $\square$

5.1 The Subsumption Test

Testing subsumption means figuring out whether $\varphi_1^{\Im} \subseteq \varphi_2^{\Im}$ holds for two concept formulae φ_1 and φ_2 and for *all* interpretations $\Im$. In our case we make use of our normal form for concept formulae where the arithmetic information is comprised in the arithmetic constraint part and the quantifications are decomposed into their atomic components. The structure of the normalized φ is $\varphi = \varphi_a \wedge \varphi_c \wedge \varphi_\forall$ where φ_a is the decomposed arithmetic part, φ_c is a propositional formula and $\varphi_\forall$ contains the decomposed and reduced universal quantifications.

In order to verify that φ_1 subsumes φ_2 we have to prove each conjunctive part in φ_2 from φ_1. The normal form allows us to separate the problem. The arithmetical part φ_{2a} can only follow from the arithmetical part φ_{1a}. This, we assume, can be checked with a corresponding arithmetic algorithm. The propositional part φ_{2c} can only follow from the propositional part φ_{1c}, which can be tested with a propositional satisfiability checker.

Finally, the atomic components $\forall m.\psi_2$ of φ_2 can follow from corresponding

$\forall m.\psi_1$ components in $\varphi_{1\forall}$, if ψ_1 subsumes ψ_2. Here the algorithm becomes recursive. There is, however, one other possibility where $\forall m.\psi_2$ is also a consequence of φ_1, namely if φ_{a1} forces $x_m = 0$. Then there are no m-successors, and $\forall m.\psi_2$ is vacuously true.

Theorem 18 ($DL_{arc\forall}$-subsumption test) *For $i = 1, 2$ let $\varphi_i = \varphi_{ia} \wedge \varphi_{ic} \wedge \varphi_{i\forall}$ be* consistent *concept formulae with normal forms $\varphi'_{ia} \wedge \varphi_{ic} \wedge \varphi'_{i\forall}$.*

φ_1 subsumes φ_2 if and only if

i) φ'_{1a} entails φ'_{2a} in the basic arithmetic system,
ii) φ_{1c} entails φ_{2c} in propositional logic, and
iii) for all '$\forall m.\psi_2$' in $\varphi'_{2\forall}$
a) φ'_{1a} entails $x_m = 0$ or
b) there is some '$\forall m.\psi_1$' in $\varphi'_{1\forall}$ with ψ_1 subsumes ψ_2.

PROOF. The 'only-if'-part (soundness) is obvious. For the 'if'-part (completeness) we show by induction on the number of nested universal quantifications in φ_2, that if one of *i)*, *ii)*, or *iii)* is violated, there is an interpretation $\Im$ with $\varphi_1^{\Im} \not\subseteq \varphi_2^{\Im}$).

This means, if $\varphi_1^{\Im} \subseteq \varphi_2^{\Im}$ for all interpretations $\Im$, then *i)*, *ii)* and *iii)* must be true, and we can check subsumption by testing *i)*, *ii)*, and *iii)*.

In the base case of the induction, φ_2 consists of the arithmetical part and the propositional part (which are independent of each other).

If the decomposed formula φ'_{1a} does not entail φ'_{2a} then $\varphi'_{1a} \wedge \neg\varphi'_{2a}$ is consistent which, by Theorem 6 and a similar construction as in Theorem 17 means that there is an interpretation $\Im$ with $(\varphi_{1a} \wedge \neg\varphi_{2a})^{\Im} \neq \emptyset$. Thus, φ_1 cannot subsume φ_2.

If the propositional part φ_{1c} does not entail φ_{2c} then obviously φ_1 cannot subsume φ_2.

Induction Step: The arguments for the arithmetical and propositional parts are the same as for the base case. For the quantification we have to investigate the case that *iii)* is violated: s'uppose there is some '$\forall m.\psi_2$' in $\varphi'_{2\forall}$ and φ'_{1a} does not entail $x_m = 0$, i.e. there is an interpretation with some m-role filler o' for some domain object o, and

a) there is no corresponding '$\forall m.\psi_1$' in $\varphi'_{1\forall}$ or

b) there is some '$\forall m.\psi_1$' in $\varphi'_{1\forall}$, but ψ_1 does not subsume ψ_2.

In the first case there is no restriction about the m-role filler o'. Since all tautologies have been eliminated from $\varphi'_{2\forall}$, there is an interpretation $\Im$ which places o into $\varphi_1^{\Im}$, but o' into the complement of $\psi_2^{\Im}$ such that $o \notin \varphi_2^{\Im}$.

Case b) is a consequence of the induction hypothesis. $\square$

Example 19 *Let us illustrate the subsumption checking procedure with an example taken from [19], page 80. The task is to show that a concept*[9]

$$\varphi_2 \stackrel{\text{def}}{=} |r| \geq 3$$

is subsumed by a concept

$$\varphi_1 \stackrel{\text{def}}{=} \forall\,(r \cap p).S \,\wedge\, \forall\,(r \cap q).\neg S \,\wedge\, |r \cap p| \geq 2 \,\wedge\, |r \cap q| \geq 2$$

First of all we need to compute the normal forms for φ_1 and φ_2. We begin with the decomposition of the role set $R = \{r, p, q\}$ according to Definition 5. We obtain

$$\begin{aligned}
\alpha_{\mathcal{A}_\mathcal{R}}(r) &= \{r, rp, rq, rpq\}\\
\alpha_{\mathcal{A}_\mathcal{R}}(p) &= \{p, rp, pq, rpq\}\\
\alpha_{\mathcal{A}_\mathcal{R}}(q) &= \{q, rq, pq, rpq\}.
\end{aligned}$$

Now we are able to decompose the universal quantifiers in φ_1 following Definition 13 into

$$\forall\, rp.S \,\wedge\, \forall\, rpq.(S \wedge \neg S) \,\wedge\, \forall\, rq.\neg S.$$

Since $S \wedge \neg S$ is inconsistent, we obtain

$$\alpha_{\mathcal{A}_\mathcal{R}}(\varphi_{1\forall}) = \forall\, rp.S \wedge \forall\, rq.\neg S$$

and the first equation for $\alpha'_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_{1a}, \varphi_{1\forall})$ is $x_{rpq} = 0$. Using this equation to simplify the normal form $x_{rp}+x_{rpq} \geq 2 \wedge x_{rq}+x_{rpq} \geq 2$ of $|r \cap p| \geq 2 \wedge |r \cap q| \geq 2$ we obtain the two in-equations

$$\alpha'_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_a, \varphi_\forall) = x_{rp} \geq 2 \,\wedge\, x_{rq} \geq 2 \,\wedge\, x_{rpq} = 0.$$

Normalizing φ_2 leads to a single in-equation

$$NF_{arc\forall}(\varphi_2) \,=\, \alpha'_{\mathcal{A}_\mathcal{R},\mathcal{A}_\mathcal{B}}(\varphi_{2a}, \varphi_{2\forall}) = x_r + x_{rp} + x_{rq} + x_{rpq} \geq 3.$$

Since φ_{2c} and $\varphi_{2\forall}$ are empty, it remains to prove that

$$x_{rp} \geq 2 \,\wedge\, x_{rq} \geq 2 \,\Rightarrow\, x_r + x_{rp} + x_{rq} + x_{rpq} \geq 3$$

[9] The notation has been adapted to fit into our framework.

is valid, and this is obvious. □

The constraints on the number of role fillers expressible in $DL_{arc\forall}$ cannot relate role fillers at different levels of the quantification. For example the representation of 'the set of people with more grandchildren than children' requires an expression like $|\mathit{has\text{-}child}; \mathit{has\text{-}child}| \leq |\mathit{has\text{-}child}|$ or a kind of aggregation functions which can lead to undecidability [5].

6 Concept Formulae with Existential Quantifiers

The existential quantifier $\exists\, r.\psi$ postulates the existence of an r-role filler in the concept ψ. As already mentioned, this quantifier is definable in our language:

$$\exists\, r.\psi = \exists r'(r' \Rightarrow r) \wedge |r'| \geq 1 \wedge \forall\, r'.\psi.$$

Therefore a language $DL_{arc\forall\exists}$ with an existential quantification is convenient, but theoretically not necessary.

For each occurrence $\exists\, r.\psi$ of an existential quantifier, one introduces a new (Skolemized) role name r' (which relates a subset of those r role-fillers lying in ψ) and adds the axiom $r' \Rightarrow r$ to the role hierarchy. The actual occurrence of $\exists\, r.\psi$ is replaced with $|r'| \geq 1 \wedge \forall\, r'.\psi$.

It should be noted that this extension to the role hierarchy is only local to the nesting of the universal quantifiers. For each $\forall\, s.\varphi$ containing an existential quantifier in the top level of φ, one can have a local extension of the role hierarchy. This way exponentially many atoms in the atomic decomposition of the role terms can be avoided. The assumption behind this optimization is that role fillers in different quantifications have nothing in common.

Example 20 *From the information*

$$\begin{aligned}\varphi = {} & \exists\,\text{has-child}.(\mathit{male} \wedge \mathit{teacher}) \wedge \exists\,\text{has-child}.(\neg \mathit{male} \wedge \mathit{teacher}) \wedge \\ & |\text{has-child}| \leq 2\end{aligned}$$

one can conclude

$$\psi = \forall\,\text{has-child}.\mathit{teacher}$$

because there are at most (in fact exactly) two children, and one must be the male and one must be the female child, and both are teachers.

The existential quantifiers are eliminated by introducing two new roles $c_1 \Rightarrow$ has-child *and* $c_2 \Rightarrow$ has-child.

$\exists\,\text{has-child}.(\mathit{male} \wedge \mathit{teacher})$ *becomes* $|c_1| \geq 1 \wedge \forall\, c_1.(\mathit{male} \wedge \mathit{teacher})$

$\exists$ has-child.$(\neg male \wedge teacher)$ *becomes* $|c_2| \geq 1 \wedge \forall\, c_2.(\neg male \wedge teacher)$.

Since $(male \wedge teacher)$ *and* $(\neg male \wedge teacher)$ *are inconsistent, the intersection of* c_1 *and* c_2 *is empty, which will be found out during the* $DL_{arc\forall}$*-normal form computation (Def. 15). Taking this and the hierarchy axioms into account, the decomposition of* has-child *therefore yields just* $\{c_1, c_2, r\}$ *where* r *stands for 'all the rest'.*

The $DL_{arc\forall}$*-normal form of* φ *is then*

$$\begin{aligned}\varphi' = x_{c_1} \geq 1 \wedge x_{c_2} \geq 1 \wedge (x_{c_1} + x_{c_2} + x_r) \leq 2 \wedge \\ \forall\, c_1.(male \wedge teacher) \wedge \forall\, c_2.(\neg male \wedge teacher),\end{aligned}$$

which obviously implies $x_r = 0$.

The $DL_{arc\forall}$*-normal form of the first second formula* $\forall$ has-child.*teacher is*

$$\forall\, c_1.teacher \wedge \forall\, c_2.teacher \wedge \forall\, r.teacher.$$

In order to check that φ *subsumes* ψ*, one has to prove recursively every quantification in* ψ' *from the corresponding quantification in* φ'*, which is trivial for the first two ones, or to check whether* φ' *implies that there are no role fillers, which is true for the third quantification in* ψ' *because* $x_r = 0$ *is a consequence of* φ'. □

7 Concept Formulae with Disjunction and Negation

A straightforward way to handle disjunctions is to generate a disjunctive normal form and to treat the disjuncts, which are actually $DL_{arc\forall\exists}$-formulae, by the $DL_{arc\forall\exists}$ algorithms: a concept formula is consistent iff at least one disjunct in the disjunctive normal form is consistent. A concept formula φ_1 subsumes a concept formula φ_2 iff one disjunct in the disjunctive normal form of φ_2 is subsumed by all disjuncts in the disjunctive normal form of φ_1.

If conjunction, disjunction, both quantifiers, negation of arithmetic expressions and negation of concept names are available, then full negation can be treated by putting a concept formula in *negation normal form* where all negation symbols are in front of concept names or in dis-equations. This way the consistency test with full negation can be reduced to the consistency test for the language $DL_{arc\forall\exists}$ with disjunction. Moreover, the subsumption test for φ_1 and φ_2 can be reduced to the consistency test for $\varphi_1 \wedge \neg\varphi_2$.

8 Other Operators in the Language

Quite a number of other operators can be added to our language without changing the algorithms.

8.1 Number-Valued Functional Roles (Features)

These are just functions mapping objects to numbers. They can only appear in the arithmetic part of the language, and there they are treated as ordinary arithmetical variables. An example for a number-valued functional role is *cubic-capacity* in the definition

$$500er \stackrel{\text{def}}{=} car \wedge cubic\text{-}capacity = 500 \cdot |has\text{-}cylinder|. \tag{11}$$

(500er is the set of cars with cubic capacity of 500 cc per cylinder.)

If only numeric features occur in the arithmetic part then $DL_{arc\forall\exists}$ is almost like Baader and Hanschke's $\mathcal{ALC}(\mathcal{D})$, $\mathcal{ALC}$ with the *concrete domain* $\mathcal{D}$ = real numbers [3]. The difference is our treatment of the existential quantifier, which introduces a numeric constraint for the role fillers. Therefore the consistency and subsumption checking algorithms are very different.

Baader and Sattler have investigated an extension of this language in which auxiliary variables can be used to link different features at different levels of the quantifications. For example $\downarrow x\ (2x = age \wedge \forall\, has\text{-}child.x = age)$ specifies the set of objects which are twice as old as their children. x serves as an auxiliary variable and links the *age*-feature of the object with the *age*-feature of the object's children. We have not yet investigated how this language extension fits into our approach. It is certainly not straightforward because consistency is undecidable in this language [4].

8.2 Other Functional Roles

Functional roles which have exactly (or at most) one role filler can be defined using arithmetic constraints and the universal quantifier.

$$|has\text{-}name| = 1 \wedge \forall\, has\text{-}name.name$$

specifies a functional role *has-name* mapping objects to objects in the set *name*.

Thus, functional roles can be treated with the mechanisms available for universal quantifiers and arithmetics. In special cases, however, it might be more efficient to treat functional roles in a special way.

8.3 Qualified Number Restrictions

Qualified number restrictions can be introduced as defined operators:

$$\begin{aligned}
&\mathit{atleast}\, n\, r.\varphi \Leftrightarrow \exists r'\; r' \Rightarrow r \wedge |r'| \geq n \wedge \forall\, r'.\varphi \\
&\mathit{atmost}\, n\, r.\varphi \Leftrightarrow \exists r'\; r' \Rightarrow r \wedge |r'| \leq n \wedge \forall\, r'.\varphi \wedge \forall\, (r \setminus r').\neg\varphi.
\end{aligned}$$

The new roles are Skolemized and the hierarchy information $r' \Rightarrow r$ is added to the role hierarchy in the same way as for the existential quantifier.

Example 21 *Let*

$$A = \left\{ \begin{array}{l}
(1)\ \mathit{atleast}\, 20\, \top.p \\
(2)\ \mathit{atleast}\, 20\, \top.q \\
(3)\ |\top| \leq 30 \\
(4)\ \mathit{atmost}\, 9\, \top.(p \wedge q).
\end{array} \right\}$$

An intuitive interpretation of the example could be: there are atleast 20 horses with white colour (p), there are atleast 20 horses with black colour (q), and there are atmost 30 horses in all. Therefore there must be atleast 10 zebras ($p \wedge q$). The last statement is the negation of this theorem. ($\top$ denotes the universal relation).

The elimination of the number restriction operators yields

$$A' = \left\{ \begin{array}{l}
(1')\ |R| \geq 20 \wedge \forall\, R.p \\
(2')\ |S| \geq 20 \wedge \forall\, S.q \\
(3')\ |\top| \leq 30 \\
(4')\ |T| \leq 9 \wedge \forall T.(p \wedge q) \wedge \forall\, (\top \setminus T).\neg(p \wedge q)
\end{array} \right\}$$

The decomposition of R, S and T generates the 8 atoms $r, s, t, rs, rt, st, rst, c$ (c stands for the complement of $R \cup S \cup T$). We use the same names for the generated non-negative integer variables.

From $(1')$ *we obtain*
$(1'') = r + rs + rt + rst \geq 20$ *and* $\forall\, \{r, rs, rt, rst\}.p.$

From $(2')$ *we obtain*

$(2'') = s + rs + st + rst \geq 20$ *and* $\forall\,\{s, rs, st, rst\}.q$.

From $(3')$ *we obtain*
$(3'') = r + s + t + rs + rt + st + rst + c \leq 30$

From $(4')$ *we obtain*
$(4'') = t{+}rt{+}st{+}rst \leq 9$ *and* $\forall\,\{t, rt, st, rst\}.(p \wedge q)$ *and* $\forall\,\{r, s, rs, c\}.\neg(p \wedge q)$.

The rs*-part of the universal quantifications in* $(1'')$, $(2'')$ *and* $(4'')$ *are contradictory. Therefore* $rs = 0$ *must hold. Using this, we get a simplified in-equation system:*

$$\begin{aligned} r + rt + rst &\geq 20 \\ s + st + rst &\geq 20 \\ r + s + t + rt + st + rst + c &\leq 30 \\ t + rt + st + rst &\leq 9 \end{aligned}$$

which is inconsistent. □

8.4 Generalized Quantifiers

of the form $n\%r\varphi$ and $\geq n\%r\varphi$ and $\leq n\%r\varphi$ where n is a number between 0 and 100, can also be defined:

$$\begin{aligned} n\%r\varphi &= \exists r'\ r' \Rightarrow r \wedge 100|r'| = n|r| \wedge \forall\, r'.\varphi \wedge \forall\,(r' \setminus r).\neg\varphi \\ \geq n\%r\varphi &= \exists r'\ r' \Rightarrow r \wedge 100|r'| \geq n|r| \wedge \forall\, r'.\varphi \\ \leq n\%r\varphi &= \exists r'\ r' \Rightarrow r \wedge 100|r'| \leq n|r| \wedge \forall\, r'.\varphi \wedge \forall\,(r' \setminus r).\neg\varphi. \end{aligned}$$

most $r\ \varphi$ in the meaning 'at least 50%' is a special case of a percentage operator.

As we have seen in the introduction, even operators like *more* and *many* $r\ \varphi$ are definable, although not in a standard way.

9 Problematic Features of Description Logics

A number of operators and features of modal and description logics can be found in the literature, whose integration into our framework is more difficult and goes beyond the scope of this paper. Role conjunction and role inverse require role terms which are no longer Boolean. To deal with these kind of operators, the atomic decomposition technique has to be extended to more expressive algebras than Boolean algebras.

Roles with special properties (reflexivity, symmetry, transitivity) are also of great interest for description logics. The *has-part* relation, for example, which is very useful in technical domains, is transitive [16].

The algorithms presented above depend very much on the fact that the quantifiers $\forall\, r.\varphi$ and $\exists\, r.\psi$ over ordinary role terms with no special properties define levels of role fillers, which correspond to the syntactic structure of nested role terms. If the roles have special properties then the levels get mixed. Transitivity in particular reduces all levels to just one. There is no straightforward way to extend the algorithms to deal with these cases, but it does not seem impossible.

10 Summary

We presented a method for developing modal and description logics together with T-Box consistency and subsumption algorithms, where the basic inference engine is arithmetic equation solving (mathematical programming). The key technique, which allows one to reduce the consistency and subsumption problem to arithmetic equation solving, is atomic decomposition of Boolean role terms embedded in bridging functions which map role fillers to numbers. Therefore the basic language in our approach can already deal with role hierarchies specified in propositional logic, role terms with set constructors, and arithmetic constraints on numeric features of role fillers.

With a few extra mechanisms one can integrate many of the standard operators in description logics, in particular quantification over role fillers, disjunction and negation. The extended system with all these features is expressive enough to treat a number of operators as defined operators, in particular qualified number restrictions, generalized quantifiers like $n\%r\varphi$ or *most* or *more* or *many*. The decision problem for the languages with these operators is therefore reduced to the decision problem for linear Diophantine equation and in-equation systems, which is decidable.

The complexity of the algorithms depends on the expressiveness of the basic arithmetic language, which may even be undecidable, and on the structure of the atomic decomposition, which may be exponential. There are, however, various optimizations of the algorithms, which reduce the complexity enormously. For example, if there is no role hierarchy at all, then the atomic decomposition becomes trivial. Each role name is mapped to itself. On the other hand, the more axioms there are to restrict the role hierarchy, the less models they have and the less atoms are generated. More information yields less complexity in this case.

Given the basic idea of using atomic decomposition to reduce the consistency and subsumption tests to arithmetic problems, it was quite straightforward to adapt the main algorithms to the standard operators for description logics. It is not yet clear, how to extend the method to other operators, in particular to more complex role constructors. Many interesting problems with this framework still need to be solved.

Acknowledgement

We would like to thank the anonymous referee for her/his valuable suggestions. In particular, thanks to these suggestions, the treatment of the existential quantifier is now much simpler than in the original version.

References

[1] W. Ackermann. Untersuchung über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110:390–413, 1935.

[2] F. Baader and B. Hollunder. KRIS: Knowledge representation and inference system. *SIGART Bulletin*, 2(2):8–15, 1991.

[3] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In John Mylopolous and Ray Reiter, editors, *Proc. of IJCAI 91*, pages 452–457. Morgan Kaufmann, 1991.

[4] F. Baader and U. Sattler. Description logics with symbolic number restrictions. In W. Wahlster, editor, *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 283–287. John Wiley & Sons Ltd, 1996. An extended version has appeared as Technical Report LTCS-96-03.

[5] F. Baader and U. Sattler. Description logics with concrete domains and aggregation. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 336–340. John Wiley & Sons Ltd, 1998.

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

[7] 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.

[8] D. Calvanese. Finite model reasoning in description logics. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 292–303. Morgan Kaufmann, Los Alto, 1996.

[9] D. Calvanese and M. Lenzerini. Making object-oriented schemas more expressive. In *Proc. of PODS'94*, pages 243–254. ACM press and Addison Wesley, 1994.

[10] D. Calvanese, M. Lenzerini, and D. Nardi A unified framework for class based representation formalisms In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 109-120, Morgan Kaufmann, Los Altos, 1994.

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

[12] A. Chargov and M. Zakharayaschev. *Modal Logic*, volume 35 of Oxford Logic Guides. Oxford University Press, 1997.

[13] M. Davis. Hilbert's tenth problem is undecidable. *American Mathematical Monthly*, 80:233–269, 1973.

[14] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. *The complexity of concept languages.* Research Report DFKI-RR-95-07, DFKI, 1995.

[15] D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning (KR92)*, pages 425–435. Morgan Kaufmann, 1992.

[16] I. Horrocks and U. Sattler. A description logic with transitive and converse roles and role hierarchies. In *Proceedings of the International Workshop on Description Logics*, Povo - Trento, Italy, 1998. IRST.

[17] R. MacGregor. A description classifier for the predicate calculus. In *Proceedings AAAI-94*, pages 213–220. Morgan Kaufmann, San Francisco, 1994.

[18] M. Minsky. A framework for representing knowledge. In R. J. Brachman and H. J. Levesque, editors, *Reprinted in Readings in Kwowledge Representation*, pages 245–262. Morgan Kaufmann, San Francisco, 1990.

[19] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence.* Springer Verlag, 1990.

[20] B. Nebel. Terminological cycles: Semantics and computational properties. In J.F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, 1991.

[21] H. J. Ohlbach and J. Koehler. How to extend a formal system with a Boolean Algebra component. In P. H. Schmidt W. Bibel, editor, *Automated Deduction. A Basis for Applications*, volume III, pages 57–75. Kluwer Academic Publishers, 1998.

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

[23] K. Schild. Terminological cycles and the propositional μ-calculus. In *Proc. of the 1st international conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 509–520. Morgan Kaufmann, 1994.

[24] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

[25] M. H. Stone. The theory of representations for Boolean algebras. *Transactions of American Mathematical Society*, 40:37–111, 1936.