Description Logics with Symbolic Number Restrictions

Franz Baader and **Ulrike Sattler** [1]

Abstract. Motivated by a chemical engineering application, we introduce an extension of the concept description language $\mathcal{ALCN}$ by symbolic number restrictions. This first extension turns out to have an undecidable concept satisfiability problem. For a restricted language—whose expressive power is sufficient for our application—we show that concept satisfiability is decidable.

1 Introduction

In recent years, a great variety of different concept description languages has been used in description logic systems and investigated in the literature. First, there was a trend of restricting the expressive power of the description language [4], to avoid the undecidability of subsumption in early systems [14, 12] and the high worst-case complexity of some of the decidable languages [9, 11]. Driven by demands from applications and facilitated by the realization that worst-case intractable languages may behave quite well in practice [1], this trend was reversed, however, by adding expressive application-relevant operators that do not cause undecidability, but may increase the worst-case complexity.

Motivated by a chemical engineering application [10, 3], we consider an extension of the expressive power of so-called number restrictions, which are present in almost all implemented systems. In its simplest form, this construct just restricts the number of role-successors w.r.t. a given role. For example, assume that `device` is a concept (unary predicate) and `input` is a role (binary predicate). Then we can describe all devices (e.g., a reactor in a chemical plant) having exactly 5 inputs and at most 6 outputs by the concept $\texttt{device} \sqcap (= 5\ \texttt{input}) \sqcap (\leq 6\ \texttt{output})$. A more expressive variant of number restrictions considered in the literature [6], but usually not available in implemented systems, are so-called qualifying number restrictions. They can be used to restrict the number of role-successors belonging to a particular concept, as for example in the concept $\texttt{device} \sqcap (\geq 5\ \texttt{input}\,.\,\texttt{pipe})$, which describes those devices having at least 5 inputs that are pipes.

A very severe restriction of the expressive power of number restrictions is the fact that we must always give exact numbers.[2] In order to express by traditional number restrictions that the number of inputs and outputs of a device must be the same, we need to know the exact number of inputs and outputs. If disjunction of concepts is available, it is sufficient to have an upper bound for the allowed number of inputs and outputs, but it is still not possible to allow for an arbitrary finite number. Symbolic number restrictions, which are introduced in the present paper, overcome this problem by allowing for variables ranging over the nonnegative integers in place of the fixed numbers in ordinary number restrictions. Devices where the number of inputs and outputs agree can thus be described by

$$\texttt{device} \sqcap (= \alpha\ \texttt{input}) \sqcap (= \alpha\ \texttt{output}).$$

The expressive power of this construct can further be increased by introducing explicit quantifiers for the numerical variables. In fact, using explicit quantification we can, on one hand, describe devices such that for every of its components the number of inputs and outputs agree:

$$\texttt{device} \sqcap \forall \texttt{component}\,.\,(\downarrow\alpha.(= \alpha\ \texttt{input}) \sqcap (= \alpha\ \texttt{output})).$$

Here, $\downarrow\alpha$ stands for an existential quantification of α. It is important that this quantifier comes after the value restriction on `component`, because we want to allow for devices where different components have a different number of inputs. On the other hand, it is sometimes important to introduce a quantified numerical variable before the value restriction in which it is used:

$$\texttt{device} \sqcap \downarrow\alpha.\forall \texttt{component}\,.\,(= \alpha\ \texttt{input})$$

makes sure that all components of the device have the same number of inputs.

If we use a concept language allowing for full negation of concepts, the existential quantifier for numerical variables induces a universal quantifier, which we will abbreviate as $\uparrow\alpha$. The expressive power of the universal quantifier is demonstrated by the fact that it can force the number of role successors of an object to be infinite: devices belonging to the concept

$$\texttt{device} \sqcap \uparrow\alpha.(\geq \alpha\ \texttt{input})$$

must have an unbounded (and thus infinite) number of inputs. This is in contrast to most of the description languages considered until now, in which every concept can be satisfied by a finite model. It turns out that universal quantification of numerical variables can even cause undecidability. In Section 2.2, we show that an extension of the concept description language $\mathcal{ALCN}$ (which allows for full negation) by symbolic number restrictions leads to a language with an undecidable satisfiability problem for concepts. However, if we restrict ourselves to atomic negation (i.e., extend $\mathcal{ALUEN}$ rather than $\mathcal{ALCN}$), and disallow universal quantification of numerical variables, we end up with a decidable satisfiability problem. It should be noted that all the reasonable examples from our application can be expressed using existentially quantified variables only. Unfortunately, the subsumption problem for this restricted language is still undecidable.

2 Extending $\mathcal{ALCN}$ by symbolic number restrictions

2.1 Basic definitions

We extend the concept language $\mathcal{ALCN}$, as introduced in [7, 5], by allowing for numerical variables α (ranging over the nonnegative in-

[1] RWTH-Aachen, LuFG Theoretical Computer Science, Ahornstr. 55, 52074 Aachen, Germany. This work was supported by the Deutsche Forschungsgemeinschaft under Grant No. Sp 230\ 6–6.

[2] As pointed out in [9], an important aspect of expressiveness is, however, "what can be left unsaid" in a representation.

ECAI 96. *12th European Conference on Artificial Intelligence*
Edited by W. Wahlster
Published in 1996 by John Wiley & Sons, Ltd.

tegers) in number restrictions, and by adding existential quantification of numerical variables (written as $\downarrow\alpha$).

Definition 1 Let N_C be a set of *concept names*, N_R a set of *role names*, N_V a set of *numerical variables*, and let rel $\in \{=, <, >, \leq, \geq\}$. The set of *concepts* of $\mathcal{ALCN}^S$ is the smallest set such that

- every concept name is a concept.
- if C and D are concepts, R is a role name, α is a variable and $n \in \mathbf{N}$ a nonnegative integer, then $(C \sqcap D)$, $(C \sqcup D)$, $(\neg C)$, $(\forall R.C)$, $(\exists R.C)$, $(\downarrow\alpha.C)$, $(\text{rel}\,\alpha\ R)$, $(\text{rel}\,n\ R)$ are concepts.

Concepts of the form $(\text{rel}\,\alpha\ R)$ or $(\text{rel}\,n\ R)$ are called number restrictions. $\mathcal{ALCN}$ is obtained from $\mathcal{ALCN}^S$ by disallowing both quantification of numerical variables and symbolic number restrictions, i.e., number restrictions of the form $(\text{rel}\,\alpha\ R)$.

Additional Boolean operators, such as implication, will be used as abbreviations: for example, $A \Rightarrow B$ stands for $\neg A \sqcup B$. Since $\mathcal{ALCN}^S$ allows for full negation of concepts, universal quantification of numerical variables can be expressed: In the following, we us $(\uparrow\alpha.C)$ as shorthand for $\neg(\downarrow\alpha.\neg C)$. Before formally defining the semantics of $\mathcal{ALCN}^S$-concepts, we illustrate the expressiveness of the new language by three examples from our chemical engineering application. In $\mathcal{ALCN}^S$, we can describe by a concept aggregates having more connections than devices as parts:

$$\downarrow\alpha.((= \alpha\ \texttt{has\_device}) \sqcap (> \alpha\ \texttt{has\_connection})).$$

In $\mathcal{ALCN}$, one could express by a large disjunction that an aggregate has 0 devices and at least 1 connection, or 1 device and at least 2 connections, or ..., but to obtain a finite disjunction, we need a fixed upper bound for the number of devices.

The next concept describes aggregates all of whose components have the same number of inputs and the same number of outputs:

$$\downarrow\alpha\downarrow\beta.(\forall\texttt{component}.((= \alpha\ \texttt{input}) \sqcap (= \beta\ \texttt{output}))).$$

Since variables are explicitly quantified, their scope varies depending on where this quantification occurs: commuting $\downarrow\beta$ and $\forall\texttt{component}$ in the above concept would yield a concept that only expresses that each component must have some finite number of outputs, without requiring any agreements of the number of outputs between different components.

The third concept illustrates how variables can be used to express restrictions on different role-levels of a nested concept. In our application, each device may have several context-dependent realizations, each of which must have sufficiently many parameters to describe all relevant properties of the device:

$$\downarrow\alpha.((= \alpha\ \texttt{property}) \sqcap (\forall\texttt{realization}.(\geq \alpha\ \texttt{param}))).$$

Definition 2 The occurrence of a variable $\alpha \in N_V$ is said to be *bound in* C iff α occurs in the scope C' of a quantified subterm $(\downarrow\alpha.C')$ of C. Otherwise, the occurrence is said to be *free*. Note that, as usual, a variable can occur free and bound in a concept. The set free$(C) \subseteq N_V$ denotes the set of variables that occur free in C. A concept C is *closed* iff free$(C) = \emptyset$. The concept $C[\frac{n}{\alpha}]$ is obtained from a concept C by substituting all free occurrences of α by n.

Using this notation, we can define the semantics of $\mathcal{ALCN}^S$-concepts.

Definition 3 An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a set $\Delta^\mathcal{I}$, called the *domain* of $\mathcal{I}$, and a function $\cdot^\mathcal{I}$ that maps every concept to a subset of $\Delta^\mathcal{I}$, and every role to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that

$$\begin{aligned}
(C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\
(C \sqcup D)^\mathcal{I} &= C^\mathcal{I} \cup D^\mathcal{I} \\
(\neg C)^\mathcal{I} &= \Delta^\mathcal{I} \setminus C^\mathcal{I} \\
(\exists R.C)^\mathcal{I} &= \{d \in \Delta^\mathcal{I} \mid \exists e \in \Delta^\mathcal{I} : (d,e) \in R^\mathcal{I} \wedge e \in C^\mathcal{I}\} \\
(\forall R.C)^\mathcal{I} &= \{d \in \Delta^\mathcal{I} \mid \forall e \in \Delta^\mathcal{I} : (d,e) \in R^\mathcal{I} \Rightarrow e \in C^\mathcal{I}\} \\
(\downarrow\alpha.C)^\mathcal{I} &= \bigcup_{n \in \mathbf{N}} (C[\frac{n}{\alpha}])^\mathcal{I} \\
(\text{rel}\,n\ R)^\mathcal{I} &= \{d \in \Delta^\mathcal{I} \mid \#\{e \in \Delta^\mathcal{I} \mid (d,e) \in R^\mathcal{I}\}\ \text{rel}\ n\}
\end{aligned}$$

Here $\#X$ denotes the cardinality of the set X. If C is not closed and free$(C) = \{\alpha_1, \ldots, \alpha_n\}$ for $n \geq 1$ then $C^\mathcal{I} := (\downarrow\alpha_1. \ldots \downarrow\alpha_n.C)^\mathcal{I}$. For a role name R, an interpretation $\mathcal{I}$ and some $x \in \Delta^\mathcal{I}$, we define

$$x_R{}^\mathcal{I} = \#\{y \in \Delta^\mathcal{I} \mid (x,y) \in R^\mathcal{I}\}.$$

Since $(\uparrow\alpha.C)$ is an abbreviation for $\neg(\downarrow\alpha.\neg C)$, we can describe its semantics directly as

$$(\uparrow\alpha.C)^\mathcal{I} = \bigcap_{n \in \mathbf{N}} (C[\frac{n}{\alpha}])^\mathcal{I}.$$

A concept C is called *satisfiable* iff there is some interpretation $\mathcal{I}$ such that $C^\mathcal{I} \neq \emptyset$. We call such an interpretation a *model of* C. A concept D *subsumes* a concept C (written $C \sqsubseteq D$) iff for each interpretation $\mathcal{I}$ we have $C^\mathcal{I} \subseteq D^\mathcal{I}$. Two concepts C, D are said to be *equivalent* iff $C \sqsubseteq D$ and $D \sqsubseteq C$.

2.2 Satisfiability of $\mathcal{ALCN}^S$-concepts is undecidable

In order to show this undecidability result, we reduce a variant of the well-known domino problem to satisfiability of $\mathcal{ALCN}^S$-concepts.

Definition 4 A *tiling system* is given by a non-empty set $D = \{D_1, \ldots, D_m\}$ of *domino types*, and by horizontal and vertical *matching pairs* $H \subseteq D \times D$, $V \subseteq D \times D$. Our variant of the *domino problem* asks for a compatible tiling of the "second eighth" $(\mathbf{N} \times \mathbf{N})_\leq := \{(a,b) \mid a, b \in \mathbf{N} \text{ and } a \leq b\}$ of the plane, i.e., a mapping $t : (\mathbf{N} \times \mathbf{N})_\leq \to D$ such that

$$\begin{aligned}
(t(a,b), t(a+1,b)) &\in H \qquad \text{for all } a < b \text{ in } \mathbf{N}, \\
(t(a,b), t(a,b+1)) &\in V \qquad \text{for all } a \leq b \text{ in } \mathbf{N}.
\end{aligned}$$

The standard domino problem asks for a compatible tiling of the whole plane. However, a compatible tiling of the second eighth yields compatible tilings of arbitrarily large finite rectangles, which in turn yield a compatible tiling of the plane [8]. Thus, the undecidability result for the standard problem [2] carries over to our variant.

Intuitively, our reduction works as follows: First, we define an $\mathcal{ALCN}^S$-concept $C_\mathbf{N}$ such that for each model of $C_\mathbf{N}$ there is a natural relationship between tuples $(a,b) \in (\mathbf{N} \times \mathbf{N})_\leq$ and certain elements $y_{a,b}$ of the model. Second, for a given tiling system $\mathcal{D}$, we construct a concept $C_\mathcal{D}$ that (1) is subsumed by $C_\mathbf{N}$, (2) ensures that every $y_{a,b}$ has exactly one domino type, and (3) encodes the compatibility conditions of the matching pairs.

The formal definition of $C_\mathbf{N}$ is given in Figure 1. Assume that $\mathcal{I}$ is an interpretation and $x \in \Delta^\mathcal{I}$ such that $x \in C_\mathbf{N}^\mathcal{I}$. Now, C_1 expresses that for every nonnegative integer a, x has an S-successor having exactly a L-successors. The precondition of C_2 makes sure that a is smaller than b, and thus the whole implication says that for each pair

$$
\begin{array}{lll}
C_{\mathbf{N}} := (\uparrow\alpha.\uparrow\beta.(C_1 \sqcap C_2 \sqcap C_3)) \text{ where } & C_1 & := \quad (\exists S.(= \alpha\ L)) \\
& C_2 & := \quad ((\exists S.(= \alpha\ L) \sqcap (\leq \beta\ L)) \Rightarrow (\exists S.(= \alpha\ L) \sqcap (= \beta\ R))) \\
& C_3 & := \quad (\forall S.((= \alpha\ L) \sqcap (= \beta\ R)) \Rightarrow (\leq \beta\ L))
\end{array}
$$

Given a tiling system $\mathcal{D} = (\{D_1, \ldots, D_m\}, H, V)$ and the subconcepts C_1, C_2, C_3 of $C_{\mathbf{N}}$ defined above, let

$$
\begin{array}{l}
C_{\mathcal{D}} := (\forall S.(\bigsqcup_{1\leq i\leq m}(D_i \sqcap (\bigsqcap_{\substack{1\leq j\leq m \\ i\neq j}} \neg D_j)))) \sqcap C_{\mathbf{N}} \sqcap \\
\qquad (\uparrow\alpha.\uparrow\beta.\ \bigsqcap_{1\leq i\leq m}(\exists S.((= \alpha\ L) \sqcap (= \beta\ R) \sqcap D_i)) \Rightarrow \\
\qquad\qquad ((\forall S.((\neq \alpha\ L) \sqcup (\neq \beta\ R) \sqcup D_i)) \sqcap \qquad (1) \\
\qquad\qquad (\uparrow\gamma.(<(\alpha,\beta) \sqcap =(\alpha+1,\gamma)) \Rightarrow (\forall S.(((= \gamma\ L) \sqcap (= \beta\ R)) \Rightarrow \bigsqcup_{(D_i,D_j)\in H} D_j))) \sqcap \qquad (2) \\
\qquad\qquad (\uparrow\gamma.(=(\beta+1,\gamma) \Rightarrow (\forall S.(((= \alpha\ L) \sqcap (= \gamma\ R)) \Rightarrow \bigsqcup_{(D_i,D_j)\in V} D_j)))))))), \qquad (3)
\end{array}
$$

Figure 1. Definition of the concepts $C_{\mathbf{N}}$ and $C_{\mathcal{D}}$ used for the reduction of the domino problem to the $\mathcal{ALCN}^S$ satisfiability problem

$a \leq b$ of nonnegative integers, x has an S-successor having exactly a L-successors and b R-successors (there can be more than one such S-successor). Finally, C_3 says that whenever an S-successor of x has a L-successors and b R-successors, we have $a \leq b$. Thus, there is an obvious correspondence between S-successors of x and points in the second eighth of the plane: every S-successor corresponds to a point in $(\mathbf{N} \times \mathbf{N})_{\leq}$ and vice versa.

Obviously, the following "canonical" model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $C_{\mathbf{N}}$ satisfies $x \in C_{\mathbf{N}}^{\mathcal{I}}$:

$$
\begin{array}{l}
\Delta^{\mathcal{I}} := \{x\} \uplus \{y_{a,b} \mid a, b \in \mathbf{N} \text{ and } a \leq b\} \uplus \\
\qquad\quad \{l_a, r_b \mid a, b \in \mathbf{N}\}, \\
S^{\mathcal{I}} := \{(x, y_{a,b}) \mid a, b \in \mathbf{N} \text{ and } a \leq b\}, \\
L^{\mathcal{I}} := \{(y_{a,b}, l_{a'}) \mid a, a', b \in \mathbf{N} \text{ and } a' < a \leq b\}, \\
R^{\mathcal{I}} := \{(y_{a,b}, r_{b'}) \mid a, b, b' \in \mathbf{N} \text{ and } a \leq b \text{ and } b' < b\}.
\end{array}
$$

The definition of the concept $C_{\mathcal{D}}$ associated with a tiling system $\mathcal{D}$ is given in Figure 1, where the following abbreviations are employed

$$
\begin{array}{rcl}
<(\alpha,\beta) & := & (\exists S.((= \alpha\ L) \sqcap (= \beta\ R) \sqcap \neg(= \beta\ L))), \\
=(\alpha+1,\beta) & := & <(\alpha,\beta) \sqcap (\forall S.((\leq \alpha\ L) \sqcup (\geq \beta\ L))).
\end{array}
$$

In the context of the concept $C_{\mathbf{N}}$, these abbreviations really express the relation $<$ and the successor relation on natural numbers: For $x \in C_{\mathbf{N}}{}^{\mathcal{I}}$ we have

$$
\begin{array}{lll}
x \in (<(\alpha,\beta)[\frac{a}{\alpha}][\frac{b}{\beta}])^{\mathcal{I}} & \text{iff} & a < b, \\
x \in (=(\alpha+1,\beta)[\frac{a}{\alpha}][\frac{b}{\beta}])^{\mathcal{I}} & \text{iff} & a+1 = b.
\end{array}
$$

The first line in the definition of $C_{\mathcal{D}}$ makes sure that $C_{\mathbf{N}}$ subsumes $C_{\mathcal{D}}$, and that every S-successor of an instance x of $C_{\mathcal{D}}$ has exactly one domino type. In the remainder of the definition, we consider an S-successor $y_{a,b}$ with domino type D_i and a L- and b R-successors. Now, (1) ensures that every S-successor with the same number of L- and R-successors as $y_{a,b}$ has the same domino type D_i, (2) takes care of the horizontal matching condition, and (3) of the vertical matching condition. Given this intuition, it is easy to show that the following lemma holds (see [13] for a complete proof).

Lemma 5 *$C_{\mathcal{D}}$ is satisfiable iff there exists a compatible tiling of the first eighth of the plane using $\mathcal{D}$.*

Thus, undecidability of the domino problem yields undecidability of the satisfiability problem for $\mathcal{ALCN}^S$-concepts. Since C is unsatisfiable iff $C \sqsubseteq (A \sqcap \neg A)$, this implies undecidability of subsumption.

Theorem 1 *Satisfiability and subsumption of $\mathcal{ALCN}^S$-concepts are undecidable.*

3 Satisfiability of $\mathcal{ALUEN}^S$-concepts is decidable

In the following, we show that satisfiability of $\mathcal{ALCN}^S$-concepts becomes decidable if negation is restricted to concept names. The only effect of this restriction is that universal quantification of numerical variables can no longer be expressed. In fact, we still have all of $\mathcal{ALCN}$ as sublanguage because every $\mathcal{ALCN}$-concept can be transformed into an equivalent one in negation normal form (where negation is only applied to concept names).

Definition 6 *Concepts* of $\mathcal{ALUEN}^S$ are those concepts of $\mathcal{ALCN}^S$ in which negation occurs only in front of concept names.

In order to simplify our investigation of the satisfiability problem for $\mathcal{ALUEN}^S$-concepts, we will restrict our attention to concepts where each numerical variable occurs either bound or free, and where each variable is bound at most once by $\downarrow$. It is easy to see that each $\mathcal{ALUEN}^S$-concept can be transformed to an equivalent concept of this form by existentially quantifying all free variables and by renaming of bound variables.

Decidability of satisfiability of $\mathcal{ALUEN}^S$-concepts will be shown by presenting a tableau-based algorithm and showing that for each $\mathcal{ALUEN}^S$-concept C, this algorithm is sound, complete, and terminating. The basic data structure this algorithm works on are so-called constraints:

Definition 7 We assume that we have a countably infinite set $\tau = \{x, y, z, \ldots\}$ of individual variables, and for each pair $(\alpha, x) \in N_V \times \tau$ a new numerical variable α_x which may occur free in concepts. A *constraint* is either of the form

$x\,R\,y$, where R is a role name in N_R and $x, y \in \tau$, or
$x : D$ for some $\mathcal{ALUEN}^S$-concept D and some $x \in \tau$.

A *constraint system* is a set of constraints.

An interpretation $\mathcal{I}$ is a model of a constraint system S iff there is a mapping $\pi : \tau \to \Delta^{\mathcal{I}}$ and a mapping $\nu : N_V \times \tau \to \mathbf{N}$ such that $\mathcal{I}, \pi, \nu$ satisfy each constraint in S, i.e., we have

$(\pi(x), \pi(y)) \in R^{\mathcal{I}}$ for all $x\,R\,y \in S$,
$\pi(x) \in \nu(D)^{\mathcal{I}}$ for all $x : D \in S$,

where $\nu(D)$ is obtained from D by replacing each variable α_y by its ν-image $\nu(\alpha, y)$.

A constraint system S is said to contain a *clash* iff for some concept name A and some variable $x \in \tau$ we have $\{x : A, x : \neg A\} \subseteq S$.

A constraint system S is said to be *numerically consistent* iff the conjunction of all numerical constraints in S, i.e.,

$$\bigwedge_{\substack{x:(\mathrm{rel}\, n\ R)\in S \\ x\in\tau, R\in N_R, n\in\mathbf{N}}} (x_R \,\mathrm{rel}\, n) \wedge \bigwedge_{\substack{x:(\mathrm{rel}\,\alpha_y\ R)\in S \\ x,y\in\tau, R\in N_R, \alpha\in N_V}} (x_R \,\mathrm{rel}\, \alpha_y),$$

is satisfiable in $(\mathbf{N}, <)$, where x_R, α_y are interpreted as variables for nonnegative integers.

A constraint system S is called *complete* iff S is clash-free, numerically consistent and none of the completion rules of Figure 2 can be applied to S.

Before showing that the completion algorithm described in Figure 2 yields a decision procedure for satisfiability of $\mathcal{ALUEN}^S$-concepts, let us make some comments on the rules. First, note that each of the completion rules adds constraints when applied to a constraint system, none of the rules removes constraints, and individual variables $x \in \tau$ are never identified or substituted. With respect to this last property, our algorithm differs from the tableau-based algorithm for $\mathcal{ALCN}$ described in [5]. Unlike our Rule 4, their algorithm introduces for each constraint of the form $x:\exists R.C$ a new R-successor of x. If x also has a constraint of the form $x:(\leq\ n\ R)$, and more than n R-successors have been introduced, then some of these individuals are identified. Our Rule 4 avoids identification by "guessing" the number of allowed R-successors of x before introducing these successors. In fact, since we do not have explict numbers, and since restrictions on numerical variables α_y in constraints $x:(\leq\ \alpha_y\ R)$ can derive from different parts of the constraint system, an identification on demand is not possible here. The second new feature is Rule 3. Given a constraint $x:(\downarrow\alpha.D)$, we substitute a new numerical variable α_x for α to make sure that the semantics of the existential quantifier $\downarrow\alpha$ is obeyed, i.e., that the valuation for α depends on x. If we would just use α, the difference between $\downarrow\alpha.\forall R.D$ and $\forall R.(\downarrow\alpha.D)$ would not be captured.

Decidability of satisfiability of $\mathcal{ALUEN}^S$-concepts is an easy consequence of the following lemma.

Lemma 8 *1. For each closed input $\mathcal{ALUEN}^S$-concept C_0, the completion algorithm terminates.*

2. Let C_0 be a closed $\mathcal{ALUEN}^S$-concept and let S be a constraint system obtained by applying the completion rules to $\{x_0:C_0\}$. Then for each completion rule $\mathcal{R}$ that can be applied to S and for each interpretation $\mathcal{I}$ we have:

$\mathcal{I}$ is a model of S iff $\mathcal{R}$ yields some S_i satisfied by $\mathcal{I}$.

3. If S is a complete constraint system obtained by applying the completion rules to $\{x_0:C_0\}$ for some closed $\mathcal{ALUEN}^S$-concept C_0, then there exists an interpretation $\mathcal{I}$ satisfying S.

4. If S is a constraint system that contains either a clash or is not numerically consistent, then S is unsatisfiable.

Proof: 1. The termination proof is similar to the one for the tableau-based algorithm for $\mathcal{ALCN}$ [5].

2. The proof, which can be found in [13], is rather straightforward, but somewhat technical. Here, we only consider Rule 3. Application of this rule adds a constraint $x:C[\frac{\alpha_x}{\alpha}]$ to S, if $x:\downarrow\alpha.C$ is contained in S. If $\mathcal{I}, \pi, \nu$ satisfy S, then we know that there exists an $n \in \mathbf{N}$ such that $\pi(x) \in \nu(C[\frac{n}{\alpha}])^{\mathcal{I}}$. Since the variable α_x does not occur in S (by our assumption that every variable is bound only once in the input concept), we can assume without loss of generality that $\nu(\alpha_x) = n$, and thus $\mathcal{I}, \pi, \nu$ satisfy $x:C[\frac{\alpha_x}{\alpha}]$. The other direction is trivial.

3. As usual, we construct the canonical interpretation $\mathcal{I}_S$ induced by S: $\Delta^{\mathcal{I}_S}$ consists of the individual variables occurring in S, $(x,y) \in R^{\mathcal{I}_S}$ iff $xRy \in S$, and $x \in A^{\mathcal{I}_S}$ iff $x:A \in S$. This yields a tree-like interpretation, which need not be a model of S, since some number restrictions might not be satisfied for the following reasons: Either (1) an individual does not have any role successors, but their existence is implied by number restrictions, or (2) it has some, but not sufficiently many role successors. Note that exact numerical restrictions on the number of role successors are given by a solution in $(\mathbf{N}, <)$ of the numerical constraints (which are satisfiable since S is numerically consistent). In the first case, S does not contain any constraints on such role successors, and we can simply generate an appropriate number of them. In the second case, the idea is to add sufficiently many *copies* of some already existing role successor y. More precisely, we need to copy the whole subtree that has y as its root. Proceeding like this from the leaves to the root, we end up with a model of S. This can be shown by induction on the structure of concepts in constraints.

4. This is obvious. ∎

Theorem 2 *Satisfiability of $\mathcal{ALUEN}^S$-concepts is decidable.*

Proof: Lemma 8 implies that the completion algorithm always terminates. In addition, the second statement of the lemma shows that the original system $\{x_0:C_0\}$ has a model iff one of the leaves of the tree obtained by the algorithm has a model. Thus, if none of the leaves is complete, the fourth statement of the lemma shows that $\{x_0:C_0\}$ does not have a model. On the other hand, if one of the leaves is complete, the third statement shows that $\{x_0:C_0\}$ has a model. Obviously, $\{x_0:C_0\}$ has a model iff C_0 is satisfiable. It remains to be shown that it is decidable whether a constraint system contains a clash and whether a constraint system is numerically consistent. Detecting clashes is trivial. Numerical consistency can be tested using a modified cycle detection algorithm running in time cubic to the size of the formula. ∎

Unfortunately, as $\mathcal{ALUEN}^S$ is not propositionally closed, subsumption cannot be reduced to satisfiability. A closer look at the specific form of the concept $C_{\mathcal{D}}$ introduced in Figure 1 reveals that it can be written as $C_{\mathcal{D}} = D_1 \sqcap \neg D_2$ for two $\mathcal{ALUEN}^S$-concepts D_1, D_2: In fact, D_1 is the first conjunct of $C_{\mathcal{D}}$ and D_2 is the negation of the remainder of $C_{\mathcal{D}}$. Note that D_1 does not contain numerical variables. Furthermore, all numerical variables occurring in the remainder of $C_{\mathcal{D}}$ are universally quantified, which shows that D_2 contains only existential quantification of numerical variables. Since $D_1 \sqcap \neg D_2$ is unsatisfiable iff $D_1 \sqsubseteq D_2$, this implies:

Theorem 3 *Subsumption of $\mathcal{ALUEN}^S$-concepts is undecidable.*

4 Conclusion

Even though the expressiveness of $\mathcal{ALCN}^S$ is still rather restricted—for example, we did not even allow for arithmetic operations on numerical variables—the presence of universal quantification of numerical variables makes satisfiability of concepts undecidable. Since none of the concepts in our application really needs universal quantification, we have defined the restricted language $\mathcal{ALUEN}^S$, and have shown that satisfiability of $\mathcal{ALUEN}^S$-concepts is decidable. Unfortunately, subsumption of $\mathcal{ALUEN}^S$-concepts has turned out to be undecidable, which makes complete terminological reasoning in $\mathcal{ALUEN}^S$ impossible. On the other hand, the satisfiability algorithm can easily

Completion rules:

1. Intersection: If $x:(C_1 \sqcap C_2) \in S$ and $x:C_1 \notin S$ or $x:C_2 \notin S$

$S \rightarrow_{\sqcap} S \cup \{x:C_1, x:C_2\}$

2. Union: If $x:(C_1 \sqcup C_2) \in S$ and $x:C_1 \notin S$ and $x:C_2 \notin S$

$S \rightarrow_{\sqcup} S_1 = S \cup \{x:C_1\}$

$S \rightarrow_{\sqcup} S_2 = S \cup \{x:C_2\}$

3. Numerical Existential Quantification: If $x:(\downarrow\alpha.D) \in S$ and $x:D[\frac{\alpha_x}{\alpha}] \notin S$

$S \rightarrow_{\downarrow} S \cup \{x:D[\frac{\alpha_x}{\alpha}]\}$

4. New Objects

If $xRy \notin S$ for all $y \in \tau$ and $m > 0, k \geq 0$ are maximal such that $\{x:(\exists R.E_1), \ldots, x:(\exists R.E_m), x:(\forall R.D_1), \ldots, x:(\forall R.D_k)\} \subseteq S$ and Rules 1–3 cannot be applied to S, then for each n with $1 \leq n \leq m$ and for each n-Partition $P = \uplus_{1 \leq i \leq n} P_i = \{1, \ldots, m\}$ of m let S_P be defined as follows:

$S \rightarrow_R S_P = S \cup \{xRy_i, \mid 1 \leq i \leq n\} \cup \{y_i:E_j \mid 1 \leq i \leq n, j \in P_i\} \cup \{y_i:D_j \mid 1 \leq i \leq n, 1 \leq j \leq k\} \cup \{x:(\geq n\ R)\}$

where $y_i \in \tau$ are new variables (i.e., variables not occurring in S).

5. Prophylactic new objects

If $xRy \notin S$ for all $y \in \tau$ and $x:(= 0\ R) \notin S$ and k maximal with $x:(\forall R.D_i) \in S$ for $1 \leq i \leq k$, $x:(\text{rel}\, N\ R) \in S$ for $N \in \mathbf{N}$ or $N = \alpha_y$ for some $y \in \tau, \alpha \in N_V$ and Rules 1–4 cannot be applied to S, then S_1, S_2 are defined as follows:

$S \rightarrow_n S_1 = S \cup \{x:(= 0\ R)\}$

$S \rightarrow_n S_2 = S \cup \{xRy\} \cup \{y:D_i \mid 1 \leq i \leq k\} \cup \{x:(> 0\ R)\}$ where $y \in \tau$ is a new variables (i.e., a variable not occurring in S).

Figure 2. The *completion algorithm* works on a tree where each node is labelled with a constraint system. It starts with a tree consisting of a root labelled with $S = \{x_0:C_0\}$ for some closed concept C_0. A rule can only be applied to a leaf labelled with a clash-free constraint system. Applying a rule $S \rightarrow S_i$ $(1 \leq i \leq n)$ to such a leaf leads to the creation of n new successors of this node labelled with the constraint systems S_i. The algorithm terminates if none of the rules can be applied to any of the leaves. The algorithm answers with "C_0 is satisfiable" iff one of the leaves obtained this way is a complete constraint system.

be extended to an algorithm that decides consistency of $\mathcal{ALUEN}^S$-ABoxes. This allows us to answer instantiation queries on $\mathcal{ALUEN}^S$-ABoxes with negated $\mathcal{ALUEN}^S$-concepts, and thus in particular with $\mathcal{ALCN}$-concepts.

An open problem is the exact complexity of satisfiability in $\mathcal{ALUEN}^S$: the algorithm as presented above needs exponential time and space. For $\mathcal{ALCN}$, one can turn a similar algorithm into a PSPACE-algorithm by considering different role-successors of an individual separately in so-called traces. For $\mathcal{ALUEN}^S$, this does not seem to be possible since we must collect the numerical constraints of all traces, and test them together for satisfiability in $(\mathbf{N}, <)$.

A variable-free approach for comparing numbers of role successors of different roles R, S could be to write something like $(= R\ S)$, with the intended meaning "the number of role successors of R and S agree." If only atomic roles are allowed here, the expressive power is rather restricted (only the first example in Section 2.1 can be expressed). If we allow for complex roles (for example composition of roles), we add another source of expressiveness, which is orthogonal to the one introduced by explicit quantification of numerical variables.

REFERENCES

[1] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.J. Profitlich, 'An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on', *Applied Artificial Intelligence*, **4**, 109–132, (1994).

[2] R. Berger, 'The undecidability of the dominoe problem', *Mem. Amer. Math. Soc.*, **66**, (1966).

[3] R. Bogusch and W. Marquardt, 'A formal representation of process model equations', *Computers and Chemical Engineering*, **19**, (1995).

[4] R. J. Brachman, D. McGuinness, P. Patel-Schneider, L. Resnick, and A. Borgida, 'Living with CLASSIC: When and how to use a KL-ONE-like language', in *Principles of Semantic Networks*, ed., John F. Sowa, M. Kaufmann, Los Altos, (1991).

[5] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt, 'The complexity of concept languages', in *Proc. of KR-91*, Boston (USA), (1991).

[6] B. Hollunder and F. Baader, 'Qualifying number restrictions in concept languages', in *Proc. of KR-91*, pp. 335–346, Boston (USA), (1991).

[7] B. Hollunder, W. Nutt, and M. Schmidt-Schauss, 'Subsumption algorithms for concept description languages', in *ECAI-90*, Pitman Publishing, London, (1990).

[8] D.E. Knuth, *The Art of computer programming*, volume 1, Addison Wesley Publ. Co., Reading, Massachussetts, 1968.

[9] H. Levesque and R. J. Brachman, 'Expressiveness and tractability in knowledge representation and reasoning', *Computational Intelligence*, **3**, 78–93, (1987).

[10] W. Marquardt, 'Trends in computer-aided process modeling', in *Proc. of ICPSE '94*, pp. 1–24, Kyongju, Korea, (1994).

[11] B. Nebel, 'Computational complexity of terminological reasoning in BACK', *Artificial Intelligence*, **34**(3), 371–383, (1988).

[12] P. F. Patel-Schneider, 'Undecidability of subsumption in NIKL', *AIJ*, **39**, 263–272, (1989).

[13] U. Sattler and F. Baader, 'Description logics with symbolic number restrictions'. Technical Report, available via ftp: `cantor.informatik.rwth-aachen.de/pub/papers/Symb-Numb.ps.Z`.

[14] M. Schmidt-Schauss, 'Subsumption in KL-ONE is undecidable', in *Proc. of KR-89*, pp. 421–431, Boston (USA), (1989).