Number Restrictions on Complex Roles in Description Logics

Franz Baader and Ulrike Sattler

LTCS-Report 96-02

This is an extended version of the article in: Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning, 1996, Cambridge, Massachusetts.

Number Restrictions on Complex Roles in Description Logics

Franz Baader and Ulrike Sattler

Abstract

Number restrictions are concept constructors that are available in almost all implemented description logic systems. However, even though there has lately been considerable effort on integrating expressive role constructors into description logics, the roles that may occur in number restrictions are usually of a very restricted type. Until now, only languages with number restrictions on atomic roles and inversion of atomic roles, or with number restrictions on intersection of atomic roles have been investigated in detail.

In the present paper, we increase the expressive power of description languages by allowing for more complex roles in number restrictions. As role constructors, we consider composition of roles (which will be present in all our languages), and intersection, union and inversion of roles in different combinations. We will present one decidability result (for the basic language that extends $\mathcal{ALC}$ by number restrictions on roles with composition), and three undecidability results for three different extensions of the basic language.

1 Motivation and introduction

Description logics is a field of knowledge representation in which there is a rather close interaction between theory and practice. One the one hand, there are various implemented systems based on description logics, which offer a palette of description formalisms with differing expressive power [32, 9, 27, 28, 2, 10]. On the other hand, the computational properties (like decidability, complexity) of various description formalisms have thoroughly been investigated [29, 35, 31, 17, 18]. These investigation were often motivated by the use of certain constructors in systems or the need for these constructors in specific applications [4, 21], and the results have influenced the design of new systems.

The terminological formalisms of knowledge representation systems based on description logics provide constructors that can be used to build complex concepts and roles out of atomic concepts (unary predicates) and roles (binary predicates). Until recently, the main emphasis, both in implemented systems and in theoretical research, was on constructors for building complex concepts. The need for rich role constructors in certain application domains (such as representing rich schema languages for databases [13, 12], or domains that require the appropriate modeling of part-whole relations [30, 1, 33]) has triggered research on description languages that also provide for expressive role constructors [3, 16]. These investigations were facilitated by the observation that the formalisms considered in description logics are very similar to certain modal logics [34, 15]. In particular, well-known modal logics, such as propositional dynamic logics (PDL) and its extensions [20, 5, 23], provide for role constructors like composition, union, transitive closure, and inversion.

Number restrictions are concept constructors that are available in almost all implemented description logic systems. They allow to restrict the number of role-successors of an individual w.r.t. a given role. For example, if `has-child` is an atomic role and `person` is an atomic concept, then we can describe all persons having at most 2 children by the concept $\texttt{person} \sqcap (\leq 2\ \texttt{has-child})$. In contrast to the rather prominent rôle that number restrictions play in description logics, the corresponding constructors in modal logic—so-called "graded modalities" [19, 37]—have been studied only recently, and thus there are not many results available that could be transferred to description logics. In [15], the problem of adding number restrictions to PDL and various of its extensions has been investigated in detail. However, even though the description languages considered in this work have very expressive formalisms for constructing complex roles, the roles that may occur in number restrictions are restricted to atomic roles and their inverse. To the best of our knowledge, the only other well-investigated concept description language with number restrictions on non-atomic roles is $\mathcal{ALCNR}$, which allows for intersection of roles in number restrictions.

The present paper is a first attempt to overcome this research deficit. It considers description languages that extend $\mathcal{ALC}$ or $\mathcal{ALC}^+$ (the description logic equivalent to PDL) with number restrictions on complex roles. As role constructors in number restriction, we will allow for composition (which will be present in all our languages), and intersection, union, and inversion of roles in different combinations. Number restrictions on roles with composition are particularly interesting from a practical point of view since they allow to impose restrictions on role successors for a composed role without explicitly stating restrictions on its atomic components. For example, the restriction $\texttt{person} \sqcap (= 17\ \texttt{has-child} \circ \texttt{has-child})$ describes persons that have 17 grandchildren without explicitly saying anything about the number of children, and the number of children of each child. From a theoretical point of view, number restrictions on roles with composition introduce a new level of complexity: The tree model property (which most of the modal logics and description logics investigated in the literature have) is no longer satisfied (see Section 3). By adding inversion of roles, we can express that a person has at least 5 siblings: $\texttt{person} \sqcap (\geq 6\ \texttt{has-child}^{-1} \circ \texttt{has-child})$; intersection of roles can prohibit that a parent marries his/her own child: $(\leq 0\ \texttt{has-child} \sqcap \texttt{is-married-to})$; union and composition can be used to describe that all children have the same name as their parent: $(= 1\ \texttt{has-name} \sqcup (\texttt{has-child} \circ \texttt{has-name}))$.

The subsumption and satisfiability problem for the language $\mathcal{ALCN}(\circ)$, which extends $\mathcal{ALC}$ with number restrictions on roles built with composition, will be shown to be decidable. On the other hand, three extensions of this language turn out to be undecidable: $\mathcal{ALC}^+$ with number restrictions on roles built with composition and union; $\mathcal{ALC}$ with number restrictions on roles built with composition and intersection; and $\mathcal{ALC}$ with number restrictions on roles built with composition, union, and inversion.

In the next section, we introduce syntax and semantics of the concept and role constructors that will be considered. Section 3 describes the algorithm that decides satisfiability of $\mathcal{ALCN}(\circ)$-concepts. The subsequent section sketches the undecidability proofs, which all use a reduction of the domino problem. In Section 5, we mention related decidability and undecidability results from modal and description logics.

2 Concept and role constructors

We define syntax and semantics of all the constructors considered in the present paper, and introduce the description languages that will be investigated in more detail.

Definition 1 *Starting with* atomic roles *from a set N_R of role names,* complex roles *are built using the role constructors composition $(R \circ S)$, union $(R \sqcup S)$, intersection $(R \sqcap S)$, inversion (R^{-1}), and transitive closure (R^+).*

*The set of $\mathcal{ALC}$-*concepts *is built from a set N_C of concept names using the concept constructors disjunction $(C \sqcup D)$, conjunction $(C \sqcap D)$, negation $(\neg C)$, value restriction $(\forall R.C)$, and existential restriction $(\exists R.C)$, where the roles occurring in value restrictions and existential restrictions are atomic roles. In $\mathcal{ALC}^+$-*concepts*, the roles occurring in value restrictions and existential restrictions may be complex roles that are built using the constructors composition, union, and transitive closure.*

Number restrictions *are concepts of the form $(\geq\ n\ R)$ or $(\leq\ n\ R)$, where $n \in \mathbb{N}$ is a nonnegative integer and R is a complex role. For a set $M \subseteq \{\sqcup, \sqcap, \circ, {}^{-1}, {}^{+}\}$ of role constructors, we call such a number restriction an M-*number restrictions *iff R is built using only constructors from M. The set of $\mathcal{ALCN}(M)$-*concepts *(resp. $\mathcal{ALC}^+\mathcal{N}(M)$-*concepts*) is obtained from $\mathcal{ALC}$-concepts (resp. $\mathcal{ALC}^+$-concepts) by additionally allowing for M-number restrictions in concepts.*

As usual in description logics, the extensions of concepts and roles involving the constructors introduced above are defined inductively on the structure of complex concepts and roles.

Definition 2 *An* interpretation *$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$, called the* domain *of $\mathcal{I}$, and an extension function $\cdot^{\mathcal{I}}$ that maps every concept to a subset of $\Delta^{\mathcal{I}}$, and every (complex) role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that the followings equalities are satisfied:*

$$\begin{aligned}
(R_1 \sqcap R_2)^{\mathcal{I}} &= {R_1}^{\mathcal{I}} \cap {R_2}^{\mathcal{I}}, \quad (R_1 \sqcup R_2)^{\mathcal{I}} \ = \ {R_1}^{\mathcal{I}} \cup {R_2}^{\mathcal{I}}, \\
(R_1 \circ R_2)^{\mathcal{I}} &= \{(d,f) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d,e) \in R_1^{\mathcal{I}} \wedge (e,f) \in R_2^{\mathcal{I}}\}, \\
(R^{-1})^{\mathcal{I}} &= \{(e,d) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}}\},
\end{aligned}$$

$$\begin{array}{rcl}
(R^+)^{\mathcal{I}} & = & \cup_{i\geq 1}(R^{\mathcal{I}})^i,\\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}},\ \ (C \sqcup D)^{\mathcal{I}} \ = \ C^{\mathcal{I}} \cup D^{\mathcal{I}},\ \ \neg C^{\mathcal{I}} \ = \ \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},\\
(\exists R.C)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d,e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\},\\
(\forall R.C)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \forall e \in \Delta^{\mathcal{I}} : (d,e) \in R^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\},\\
(\geq n\ R)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \#\{e \in \Delta^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}}\} \geq n\},\\
(\leq n\ R)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \#\{e \in \Delta^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}}\} \leq n\}.
\end{array}$$

Here $\#X$ denotes the size of a set X. If $d \in C^{\mathcal{I}}$, we say that d is an instance *of C in $\mathcal{I}$. If $(d,e) \in R^{\mathcal{I}}$, we say that d is an R-predecessor of e, and e is an R-successor of d in $\mathcal{I}$.*

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 all interpretations $\mathcal{I}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Since all the languages considered in the present paper allow for negation and conjunction of concepts, subsumption and (un)satisfiability can be reduced to each other:

- $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable,
- C is unsatisfiable iff $C \sqsubseteq A \sqcap \neg A$ (for a concept name A).

For this reason, we may restrict our attention to the satisfiability problem, both in the decidability and in the undecidability proofs.

3 $\mathcal{ALCN}(\circ)$ is decidable

We present a tableau-like algorithm for deciding satisfiability of $\mathcal{ALCN}(\circ)$-concepts. The algorithm and the proof of its correctness are very similar to existing algorithms and proofs for languages with number restrictions on atomic roles [25, 24]. It should be noted, however, that the presence of number restrictions on role chains of the form $R_1 \circ R_2 \circ \ldots \circ R_n$ with $n > 1$ has as consequence that the finite models generated by the algorithm need no longer be tree models. A *tree model* of a concept C is an interpretation such that *(1)* every element of the model can be reached from an initial (root) element, which is an instance of C, via role chains, *(2)* the root does not have a role predecessor, and *(3)* every other element has exactly one role predecessor. The following $\mathcal{ALCN}(\circ)$-concept is satisfiable, but it obviously does not have a tree model:

$$(\exists R.A) \sqcap (\exists R.\neg A) \sqcap (\forall R.(\exists S.B)) \sqcap (\leq 1\ R{\circ}S).$$

Nevertheless, the models that will be generated by our algorithm are very similar to tree models in that properties (1) and (2) are still satisfied, and every role chain from the root to an element has the same length (even though there may exist more than one such chain). This fact will become important in the proof of termination.

As usual, we assume without loss of generality that all concepts are in negation normal form (NNF), i.e., negation occurs only immediately in front of concept names. The basic data structure our algorithm works on are constraints:

Definition 3 *Let $\tau = \{x, y, z, \dots\}$ be a countably infinite set of individual variables. A* constraint *is of the form*

$$xRy, \quad x : D, \quad or \quad x \neq y,$$

where R is a role name, x, y are individual variables, and D is an $\mathcal{ALCN}(\circ)$-concept in NNF. A constraint system *is a set of constraints. For a constraint system S, let $\tau_S \subseteq \tau$ denote the individual variables occuring in S.*

An interpretation $\mathcal{I}$ is a model of a constraint system *S iff there is a mapping $\pi : \tau_S \to \Delta^{\mathcal{I}}$ such that $\mathcal{I}, \pi$ satisfies each constraint in S, i.e.,*

$$\begin{array}{rl} (\pi(x), \pi(y)) \in R^{\mathcal{I}} & \textit{for all } xRy \in S, \\ \pi(x) \neq \pi(y) & \textit{for all } (x \neq y) \in S, \\ \pi(x) \in D^{\mathcal{I}} & \textit{for all } x : D \in S. \end{array}$$

*For a constraint system S, individual variables x, y, and role names R_i, we say that y is a $R_1{\circ}\dots{\circ}R_m$-*successor *of x in S iff there are $y_0, \dots, y_m \in \tau$ such that $x = y_0, y = y_m$, and $\{y_i R_{i+1} y_{i+1} \mid 0 \leq i \leq m-1\} \subseteq S$. S contains a* clash *iff $\{x : A, x : \neg A\} \subseteq S$ for some concept name A and some variable $x \in \tau_S$, or $x :(\leq n\ R) \in S$ and x has $\ell > n$ R-successors $y_1, \dots, y_\ell$ in S such that for all $i \neq j$ we have $y_i \neq y_j \in S$. A constraint system S is called* complete *iff none of the completion rules given in Figure 1 can be applied to S. In these rules, the constraint system $S[y_1/y_2]$ is obtained from S by substituting each occurrence of y_2 in S by y_1.*

Figure 1 introduces the *completion rules* that are used to test $\mathcal{ALCN}(\circ)$-concepts for satisfiability. The *completion algorithm* works on a tree where each node is labelled with a constraint system. It starts with the tree consisting of a root labelled with $S = \{x_0 : C_0\}$, where C_0 is the $\mathcal{ALCN}(\circ)$-concept in NNF to be tested for satisfiability. A rule can only be applied to a leaf labelled with a clash-free constraint system. Applying a rule $S \rightarrow S_i$, for $1 \leq i \leq n$, to such a leaf leads to the creation of n new successors of this node, each labelled with one of the constraint systems S_i. The algorithm terminates if none of the rules can be applied to any of the leaves. In this situation, it answers with "C_0 is satisfiable" iff one of the leaves is labelled with a clash-free constraint system.

1. Conjunction: If $x :(C_1 \sqcap C_2) \in S$ and $x : C_1 \notin S$ or $x : C_2 \notin S$, then
$S \rightarrow \quad S \cup \{x : C_1, x : C_2\}$

2. Disjunction: If $x :(C_1 \sqcup C_2) \in S$ and $x : C_1 \notin S$ and $x : C_2 \notin S$, then
$S \rightarrow \quad S_1 = S \cup \{x : C_1\}$
$S \rightarrow \quad S_2 = S \cup \{x : C_2\}$

3. Value restriction: If $x :(\forall R.C) \in S$ for a role name R, y is an R-successor of x in S and $y : C \notin S$, then
$S \rightarrow \quad S \cup \{y : C\}$

4. Existential restriction: If $x :(\exists R.C) \in S$ for a role name R and there is no R-successor y of x in S with $y : C \in S$, then
$S \rightarrow \quad S \cup \{xRz, z : C\}$ for a new variable $z \in \tau \setminus \tau_S$.

5. Number restriction: If $x :(\geq n\ R_1 \circ \ldots \circ R_m) \in S$ for role names $R_1, \ldots, R_m$ and x has less than $n\ R_1 \circ \ldots \circ R_m$-successors in S, then
$S \rightarrow \quad S \cup \{xR_1y_2, y_mR_mz\} \cup \{y_iR_iy_{i+1} \mid 2 \leq i \leq m-1\} \cup$
$\{z \neq w \mid w$ is an $R_1 \circ \ldots \circ R_m$-successor of x in $S\}$
where z, y_i are new variables in $\tau \setminus \tau_S$.

6. Number restriction: If $x :(\leq n\ R_1 \circ \ldots \circ R_m) \in S$, x has more than $n\ R_1 \circ \ldots \circ R_m$-successors in S, and there are $R_1 \circ \ldots \circ R_m$ successors y_1, y_2 of x in S with $(y_1 \neq y_2) \notin S$, then
$S \rightarrow S_{y_1,y_2} = S[y_1/y_2]$
for all pairs y_1, y_2 of $R_1 \circ \ldots \circ R_m$-successors of x with $(y_1 \neq y_2) \notin S$.

Figure 1: The completion rules for $\mathcal{ALCN}(\circ)$

Correctness of this algorithm is an immediate consequence of the following facts:

Lemma 4 *Let C_0 be an $\mathcal{ALCN}(\circ)$-concept, and let S be a constraint system obtained by applying the completion rules to $\{x_0 : C_0\}$. Then*

1. *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 $\mathcal{I}$ is a model of one of the systems S_i obtained by applying $\mathcal{R}$.*
2. *If S is a complete and clash-free constraint system, then S has a model.*
3. *If S contains a clash, then S does not have a model.*
4. *The completion algorithm terminates when applied to $\{x_0 : C_0\}$.*

Indeed, termination shows that after finitely many steps we obtain a tree such that all its leaf nodes are labelled with complete constraint systems. If C_0 is satisfiable, then $\{x_0 : C_0\}$ is also satisfiable, and thus one of the complete constraint systems is satisfiable by (1). By (3), this system must be clash-free. Conversely, if one of the complete constraint systems is clash-free, then it is satisfiable by (2), and because of (1) this implies that $\{x_0 : C_0\}$ is satisfiable. Consequently, the algorithm is a decision procedure for satisfiability of $\mathcal{ALCN}(\circ)$-concepts:

Theorem 5 *Subsumption and satisfiability of $\mathcal{ALCN}(\circ)$-concepts is decidable.*

Proof of Part 1 of Lemma 4: We consider only the rules concerned with number restrictions, since the proof for Rules 1–4 is just as for $\mathcal{ALC}$.

5. Number restriction: Assume that the rule is applied to the constraint $x : (\geq n\ R_1 \circ \ldots \circ R_m)$, and that its application yields

$$\begin{aligned} S' = S \ & \cup \ \{xR_1y_2, y_mR_mz\} \cup \{y_iR_iy_{i+1} \mid 2 \leq i \leq m-1\} \\ & \cup \ \{z \neq w \mid w \text{ is an } R_1 \circ \ldots \circ R_m\text{-successor of } x \text{ in } S\}. \end{aligned}$$

Since S is a subset of S', any model of S' is also a model of S. Conversely, assume that $\mathcal{I}$ is a model of S, and let $\pi : \tau_S \to \Delta^{\mathcal{I}}$ be the corresponding mapping of individual variables to elements of $\Delta^{\mathcal{I}}$. On the one hand, since $\mathcal{I}$ satisfies $x : (\geq n\ R_1 \circ \ldots \circ R_m)$, $\pi(x)$ has at least n $R_1 \circ \ldots \circ R_m$-successors in $\mathcal{I}$. On the other hand, since Rule 5 is applicable to $x : (\geq n\ R_1 \circ \ldots \circ R_m)$, x has less than n $R_1 \circ \ldots \circ R_m$-successors in S. Thus, there exists $b \in \Delta^{\mathcal{I}}$ such that $b \neq \pi(w)$ for all

$R_1 \circ \ldots \circ R_m$-successors w of x in S. Let $b_2, \ldots, b_m \in \Delta^{\mathcal{I}}$ be such that $(\pi(x), b_2) \in R_1^{\mathcal{I}}, (b_2, b_3) \in R_2^{\mathcal{I}}, \ldots, (b_m, b) \in R_m^{\mathcal{I}}$. We define $\pi' : \tau_{S'} \to \Delta^{\mathcal{I}}$ by $\pi'(y) := \pi(y)$ for all $y \in \tau_S$, $\pi'(y_i) := b_i$ for all $i, 2 \leq i \leq m$, and $\pi'(z) := b$. Obviously, $\mathcal{I}, \pi'$ satisfy S'.

6. Number restriction: Assume that the rule can be applied to $x : (\leq n\ R_1 \circ \ldots \circ R_m) \in S$, and let $\mathcal{I}$ together with the valuation $\pi : \tau_S \to \Delta^{\mathcal{I}}$ be a model of S. On the one hand, since the rule is applicable, x has more than $n\ R_1 \circ \ldots \circ R_m$-successors in S. On the other hand, $\mathcal{I}, \pi$ satisfy $x : (\leq m\ R_1 \circ \ldots \circ R_m) \in S$, and thus there are two different $R_1 \circ \ldots \circ R_m$-successors y_1, y_2 of x in S such that $\pi(y_1) = \pi(y_2)$. Obviously, this implies that $(y_1 \neq y_2) \notin S$, which shows that $S_{y_1,y_2} = S[y_1/y_2]$ is one of the constraint systems obtained by applying Rule 6 to $x : (\leq n\ R_1 \circ \ldots \circ R_m)$. In addition, since $\pi(y_1) = \pi(y_2)$, $\mathcal{I}, \pi$ satisfy S.
Conversely, assume that $S_{y_1,y_2} = S[y_1/y_2]$ is obtained from S by applying Rule 6, and let $\mathcal{I}$ together with the valuation π be a model of S_{y_1,y_2}. If we take a valuation π' that coincides with π on the variables in $\tau_{S_{y_1,y_2}}$ and satisfies $\pi'(y_2) = \pi(y_1)$, then $\mathcal{I}, \pi'$ obviously satisfy S.

■

Proof of Part 2 of Lemma 4: Let S be a complete and clash-free constraint system that is obtained by applying the completion rules to $\{x_0 : C_0\}$. We define a canonical model $\mathcal{I}$ of S as follows:

$$\begin{array}{llll} \Delta^{\mathcal{I}} := \tau_S \quad \text{and} & \text{for all } A \in N_C : & x \in A^{\mathcal{I}} & \text{iff } x : A \in S, \\ & \text{for all } R \in N_R : & (x,y) \in R^{\mathcal{I}} & \text{iff } xRy \in S. \end{array}$$

In addition, let $\pi : \tau_S \to \Delta^{\mathcal{I}}$ be the identity on τ_S. We show that $\mathcal{I}, \pi$ satisfy every constraint in S.

By definition of $\mathcal{I}$, a role constraint of the form xRy is satisfied by $\mathcal{I}, \pi$ iff $xRy \in S$. More generally, y is an $R_1 \circ \ldots \circ R_m$-successor of x in S iff y is an $R_1 \circ \ldots \circ R_m$-successor of x in $\mathcal{I}$. We show by induction on the structure of the concept C, that every concept constraint $x : C \in S$ is satisfied by $\mathcal{I}, \pi$. Again, we restrict our attention to number restrictions since the induction base and the treatment of the other constructors is just as for $\mathcal{ALC}$.

- Consider $x : (\geq n\ R_1 \circ \ldots \circ R_m) \in S$. Since S is complete, Rule 5 cannot be applied to $x : (\geq n\ R_1 \circ \ldots \circ R_m)$, and thus x has at least $n\ R_1 \circ \ldots \circ R_m$-successors in S, which are also $R_1 \circ \ldots \circ R_m$-successors of x in $\mathcal{I}$. This shows that $\mathcal{I}, \pi$ satisfy $x : (\geq n\ R_1 \circ \ldots \circ R_m)$.

- Constraints of the form $x:(\leq n\ R_1\circ\ldots\circ R_m) \in S$ are satisfied because S is clash-free and complete. In fact, assume that x has more than n $R_1\circ\ldots\circ R_m$-successors in $\mathcal{I}$. Then x also has more than n $R_1\circ\ldots\circ R_m$-successors in S. If S contained inequality constraints $y_i \neq y_j$ for all these successors, then we would have a clash. Otherwise, Rule 6 could be applied.

■

Proof of Part 3 of Lemma 4: Assume that S contains a clash. If $\{x:A, x:\neg A\} \subseteq S$, then it is clear that no interpretation can satisfy both constraints. Thus assume that $x:(\leq n\ R) \in S$ and x has $\ell > n$ R-successors $y_1, \ldots, y_\ell$ in S with $(y_i \neq y_j) \in S$ for all $i \neq j$. Obviously, this implies that in any model $\mathcal{I}, \pi$ of S, $\pi(x)$ has $\ell > n$ distinct R-successors $\pi(y_1), \ldots, \pi(y_\ell)$ in $\mathcal{I}$, which shows that $\mathcal{I}, \pi$ cannot satisfy $x:(\leq n\ R)$. ■

Proof of Part 4 of Lemma 4: In the following, we consider only constraint systems S that are obtained by applying the completion rules to $\{x_0:C_0\}$. For a concept C, we define its and/or-size $|C|_{\sqcap,\sqcup}$ as the number of occurrences of conjunction and disjunction constructors in C. The maximal role depth depth(C) of C is defined as follows:

$$\begin{array}{lclcl}
\mathrm{depth}(A) & := & \mathrm{depth}(\neg A) & := & 0, \\
\mathrm{depth}(C_1 \sqcap C_2) & := & \mathrm{depth}(C_1 \sqcup C_2) & := & \max\{\mathrm{depth}(C_1), \mathrm{depth}(C_2)\}, \\
\mathrm{depth}(\forall R_1.C_1) & := & \mathrm{depth}(\exists R_1.C_1) & := & 1 + \mathrm{depth}(C_1),
\end{array}$$
$$\mathrm{depth}(\geq n\ R_1\circ\ldots\circ R_m) := \mathrm{depth}(\leq n\ R_1\circ\ldots\circ R_m) := m.$$

For the termination proof, the following observations, which are an easy consequence of the definition of the completion rules, are important:

1. Every variable $x \neq x_0$ that occurs in S is an $R_1\circ\ldots\circ R_m$-successor of x_0 for some role chain of length $m \geq 1$. In addition, every other role chain that connects x_0 with x has the same length.

2. If x can be reached in S by a role chain of length m from x_0, then for each constraint $x:C$ in S, the maximal role depth of C is bounded by the maximal role depth of C_0 minus m. Consequently, m is bounded by the maximal role depth of C_0.

Let m_0 be the maximal role depth of C_0. Because of the first fact, every individual x in a constraint system S (reached from $\{x_0:C_0\}$ by applying completion rules) has a unique role level level(x), which is its distance from the root node x_0, i.e., the unique length of the role chains that connects x_0 with x. Because of the second fact, the level of each individual is an integer between 0 and m_0.

In the following, we define a mapping κ of constraint systems S to $5m_0$-tuples of nonnegative integers such that $S \to S'$ implies $\kappa(S) \succ \kappa(S')$, where $\succ$ denotes the lexicographic ordering on $5m_0$-tuples. Since the lexicographic ordering is well-founded, this implies termination of our algorithm. In fact, if the algorithm did not terminate, then there would exist an infinite sequence $S_0 \to S_1 \to \ldots$, and this would yield an infinite descending $\succ$-chain of tuples.

Thus, let S be a constraint system that can be reached from $\{x_0 : C_0\}$ by applying completion rules. We define

$$\kappa(S) := (\kappa_0, \kappa_1, \ldots, \kappa_{m_0-1}, \kappa_{m_0}),$$

where $\kappa_\ell := (k_{\ell,1}, k_{\ell,2}, k_{\ell,3}, k_{\ell,4}, k_{\ell,5})$ and the components $k_{\ell,i}$ are obtained as follows:

- $k_{\ell,1}$ is the number of individual variables x in S with level$(x) = \ell$.
- $k_{\ell,2}$ is the sum of the and/or-sizes $|C|_{\sqcap,\sqcup}$ of all constraints $x : C \in S$ such that level$(x) = \ell$ and the conjunction or disjunction rule is applicable to $x : C$ in S.
- For a constraint $x :(\geq n\ R_1 \circ \ldots \circ R_m)$, let k be the maximal cardinality of all sets M of $R_1 \circ \ldots \circ R_m$-successors of x for which $y_i \neq y_j \in S$ for all pairs of distinct elements y_i, y_j of M. We associate with $x :(\geq n\ R_1 \circ \ldots \circ R_m)$ the number $r := n - k$, if $n \geq k$, and $r := 0$ otherwise. $k_{\ell,3}$ sums up all the numbers r associated with constraints of the form $x :(\geq n\ R_1 \circ \ldots \circ R_m)$ for variables x with level$(x) = \ell$.
- $k_{\ell,4}$ is the number of all constraints $x :(\exists R.C) \in S$ such that level$(x) = \ell$ and the existential restriction rule is applicable to $x :(\exists R.C)$ in S.
- $k_{\ell,5}$ is the number of all pairs of constraints $x :(\forall R.C)$, $xRy \in S$ such that level$(x) = \ell$ and the value restriction rule is applicable to $x :(\forall R.C)$, xRy in S.

In the following, we show for each of the rules of Figure 1 that $S \to S'$ implies $\kappa(S) \succ \kappa(S')$.

1. **Conjunction:** Assume that the rule is applied to the constraint $x : C_1 \sqcap C_2$, and let S' be the system obtained from S by its application. Let $\ell :=$ level(x).
   First, we compare κ_ℓ and κ'_ℓ, the tuples respectively associated with level ℓ in S and S'. Obviously, the *first components* of κ_ℓ and κ'_ℓ agree since the number of individuals and their levels are not changed. The *second component* of κ'_ℓ is *smaller* than the second component of κ_ℓ: $|C_1 \sqcap C_2|_{\sqcap,\sqcup}$ is removed from the sum, and replaced a number that is not larger than $|C_1|_{\sqcap,\sqcup} + |C_2|_{\sqcap,\sqcup}$ (depending on whether the top constructor of C_1 and C_2 is disjunction or conjunction or some other constructor). Since tuples are compared with the lexicographic ordering, a decrease in this component makes sure that it is irrelevant what happens in later components.
   For the same reason, we need not consider tuples κ_m for $m > \ell$. Thus, assume that $m < \ell$. In such a tuple, the first three components are not changed by application of the rule, whereas the remaining two components remain unchanged or decrease. Such a decrease can happen if level$(y) = m$ and S contains constraints yRx, $y :(\forall R.C_i)$ (or $y :(\exists R.C_i)$).

2. **Disjunction:** This rule can be treated like the conjunction rule.

3. **Value restriction:** Assume that the rule is applied to the constraints $x :(\forall R.C), xRy$, and let S' be the system obtained from S by its application. Let $\ell :=$ level(x). Obviously, this implies that level$(y) =$ level$(x) + 1 > \ell$.
   On level ℓ, the first three components of κ_ℓ remain unchanged; the fourth remains the same, or decreases (if S contains constraints zSy and $z :(\exists S.C)$ for an individual z with level$(z) = \ell$); and the fifth decreases by at least one since the constraints $x :(\forall R.C), xRy$ are no longer counted. It may decrease by more than one if S contains constraints zSy and $z :(\forall S.C)$ for an individual z with level$(z) = \ell$.
   Because of this decrease at level ℓ, the tuples at larger levels (in particular, the one for level level$(x) + 1$, where there might be an increase), need not be considered.
   The tuples of levels smaller than ℓ are not changed by application of the rule. In particular, the third component of such a tuple does not change since no role constraints or inequality constraints are added or

removed.

4. **Existential restriction:** Assume that the rule is applied to the constraint $x:(\exists R.C)$, and let $S' = S \cup \{xRy, y:C\}$ be the system obtained from S by its application. Let $\ell := \text{level}(x)$. Obviously, this implies that $\text{level}(y) = \text{level}(x) + 1 > \ell$.
   The first two components of κ_ℓ obviously remain unchanged, and so does the third since the new individual y is not related via inequality constraints with any of the old individuals. Since the fourth component decreases, the possible increase of the fifth component is irrelevant.
   For the same reason, the increase of the first component of $\kappa_{\ell+1}$ is irrelevant.
   Tuples of levels smaller than ℓ are not changed by application of the rule. In particular, the third component of such a tuple does not change since the new individual y is not related via inequality constraints with any of the old individuals.

5. **Number restriction:** Assume that the rule is applied to the constraint $x:(\geq n\ R_1\circ\ldots\circ R_m) \in S$, let S' be the system obtained by rule application, and let $\ell = \text{level}(x)$.
   As for Rule 4, the first two components of κ_ℓ remain the same. In addition, there is a decrease in the third component of κ_ℓ, since the new individual z can now be added to the maximal sets of explicitly distinct $R_1\circ\ldots\circ R_m$-successors of x. Note that these sets were previously smaller than n (because even the set of all $R_1\circ\ldots\circ R_m$-successors of x was smaller than n).
   For this reason, the possible increase in the fifth component of κ_ℓ and in the first components of tuples of levels larger than ℓ are irrelevant.Tuples of levels smaller than ℓ are either unchanged by application of the rule, or their third component decreases.

6. **Number restriction:** Assume that the rule is applied to the constraint $x:(\leq n\ R_1\circ\ldots\circ R_m) \in S$, let $S' = S_{y_1,y_2}$ be the system obtained by rule application, and let $\ell = \text{level}(x)$.
   On level $\ell + m$, the first component of the tuple $\kappa_{\ell+m}$ decreases. Thus, possible increases in the other components of this tuple are irrelevant. Tuples associated with smaller levels remain unchanged or decrease. In fact, since y_1 in S' has all its old constraints and the constraints of y_2 in S, some value restrictions or existential restrictions for individuals

of the level immediately above level $\ell + m$ may become satisfied (in the sense that the corresponding rule no longer applies). Since no constraints are removed, previously satisfied value restrictions or existential restrictions remain satisfied. The third component of such tuples cannot increase since the individuals y_1, y_2 that have been identified were not related by inequality constraints.

■

For languages where number restrictions may also contain union or intersection of roles, an important property used in the above termination proof is no longer satisfied: It is not possible to associate each individual generated by a tableau-like procedure with a unique role level, which is its distance to the "root" individual x_0 (i.e., the instance x_0 of C_0 to be generated by the tableau algorithm). Indeed, in the concept

$$C_0 := (\exists R.\exists R.A) \sqcap (\leq 1\ R \sqcup R \circ R),$$

the number restriction enforces that an R-successor of an instance of C_0 is also an $R \circ R$-successor of this instance. For this reason, an R-successor of the root individual must be both on level 1 and on level 2, and thus the relatively simple termination argument that was used above is not available for these larger languages. We will show below that satisfiability is in fact undecidable for $\mathcal{ALCN}(\circ, \sqcap)$. For $\mathcal{ALCN}(\circ, \sqcup)$, decidability of satisfiability is still an open problem.

4 Undecidability results

We will use a reduction of the domino problem—a well-known undecidable problem [26, 6] often used in undecidability proofs in logic—to show that concept satisfiability is undecidable for three extensions of the decidable language $\mathcal{ALCN}(\circ)$ considered in the previous section.

Definition 6 *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 $H \subseteq D \times D$, $V \subseteq D \times D$. *The* domino problem *asks for a* compatible tiling *of the first quadrant $\mathbb{N} \times \mathbb{N}$ of the plane, i.e., a mapping $t : \mathbb{N} \times \mathbb{N} \to D$ such that*

$$\forall m, n \in \mathbb{N}:\quad (t(m,n), t(m+1,n)) \in H \quad \text{and} \quad (t(m,n), t(m,n+1)) \in V.$$

In order to reduce the domino problem to satisfiability of concepts, we must show how a given tiling system $D = \{D_1, \ldots, D_m\}$ can be translated into a concept E_D (of the language under consideration) such that E_D is satisfiable iff D allows for a compatible tiling. This task can be split into three subtasks, which we will first explain on an intuitive level, before showing how they can be achieved for the three concept languages under consideration.

Task 1: It must be possible to represent a single "square" of $\mathbb{N} \times \mathbb{N}$, which consists of points $(n, m), (n, m+1), (n+1, m)$, and $(n+1, m+1)$. The idea is to introduce roles X, Y, where X goes one step into the horizontal (i.e. x-) direction, and Y goes one step into the vertical (i.e. y-) direction. The concept language must be expressive enough to describe that an individual (a point (n, m)) has exactly one X-successor (the point $(n+1, m)$), exactly one Y-successor (the point $(n, m+1)$), and that the $X \circ Y$-successor coincides with the $Y \circ X$-successor (the point $(n+1, m+1)$).

Task 2: It must be possible to express that a tiling is locally correct, i.e., that the X- and Y-successors of a point have an admissible domino type. The idea is to associate each domino type D_i with an atomic concept D_i, and to express the horizontal and vertical matching conditions via value restrictions on the roles X, Y.

Task 3: It must be possible to impose the above *local* conditions on all points in $\mathbb{N} \times \mathbb{N}$. This can be achieved by constructing a "universal" role U and a "start" individual such that every point is a U-successor of this start individual. The local conditions can then be imposed on all points via value restrictions on U for the start individual.

Task 2 is rather easy, and can be realized using the following $\mathcal{ALC}$-concept:

$$\begin{aligned} C_D := & \bigsqcup_{1 \le i \le m} (D_i \sqcap (\bigsqcap_{\substack{1 \le j \le m \\ i \ne j}} \neg D_j)) \sqcap \\ & \bigsqcap_{1 \le i \le m} (D_i \Rightarrow ((\forall X.(\bigsqcup_{(D_i, D_j) \in H} D_j)) \sqcap (\forall Y.(\bigsqcup_{(D_i, D_j) \in V} D_j)))), \end{aligned}$$

where $A \Rightarrow B$ is an abbreviation for $\neg A \sqcup B$. The first line expresses that every point has exactly one domino type, and the value restrictions in the second line express the horizontal and vertical matching conditions.

Task 1 can be achieved in any extension of $\mathcal{ALCN}(\circ)$ with either union or intersection of roles in number restrictions:

$$\begin{aligned} C_\sqcup &:= (= 1\ X) \sqcap (= 1\ Y) \sqcap (= 1\ X \circ Y) \sqcap (= 1\ Y \circ X) \sqcap (= 1\ Y \circ X \sqcup X \circ Y), \\ C_\sqcap &:= (= 1\ X) \sqcap (= 1\ Y) \sqcap (= 1\ X \circ Y) \sqcap (= 1\ Y \circ X) \sqcap (= 1\ Y \circ X \sqcap X \circ Y), \end{aligned}$$

where $(= n\ R)$ is an abbreviation for $(\ge n\ R) \sqcap (\le n\ R)$.

Task 3 is easy for languages that extend $\mathcal{ALC}^+$, and more difficult for languages without transitive closure. The general idea is that the start individual s is an instance of the concept E_D to be constructed. From this individual, one can reach via U the origin $(0,0)$ of $\mathbb{N}\times\mathbb{N}$, and all points that are connected with the origin via arbitrary X- and Y-chains.

(1) In extensions of $\mathcal{ALC}^+$, we can use an atomic role R to reach the origin, and the complex role $R \sqcup (R\circ(X \sqcup Y)^+)$ to reach every point. Thus, the tiling system D can be translated into the $\mathcal{ALC}^+\mathcal{N}(\circ,\sqcup)$-concept

$$E_D^{(1)} := (= 1\ R) \sqcap (\forall(R \sqcup (R\circ(X \sqcup Y)^+)).(C_\sqcup \sqcap C_D)).$$

We can even restrict the complex role in the value restriction to a simple transitive closure of an atomic roles. To achieve this, we make sure that the X- and the Y-successors of a point are also R-successors of this point. This allows us to use R^+ in place of $R \sqcup (R\circ(X \sqcup Y)^+)$ as "universal" role:

$$E_D^{(1')} := (= 1\ R) \sqcap (\forall R^+.(C_\sqcup \sqcap C_D \sqcap (\geq 2\ R) \sqcap (\leq 2\ R \sqcup X \sqcup Y)))$$

(2) In $\mathcal{ALCN}(\circ,\sqcup,^{-1})$, we explicitly introduce a role name U for the "universal" role, and use number restrictions involving composition, union, and inversion of roles to make sure that the start individual is directly connected via U with every point:

$$\begin{aligned} E_D^{(2)} := (\geq 1\ U) \sqcap (\forall U.(C_\sqcup \sqcap C_D \sqcap (= 1\ X\circ U^{-1}) \sqcap (= 1\ Y\circ U^{-1}) \sqcap \\ (\leq 1\ U^{-1} \sqcup Y\circ U^{-1} \sqcup X\circ U^{-1}))). \end{aligned}$$

The number restrictions inside the value restriction make sure that every point p that is reached via U from the start individual satisfies the following: Its X-successor and its Y-successor each have exactly one U-predecessor, which coincides with the (unique) U-predecessor of p, i.e., the start individual. Thus, the X-successor and the Y-successor of p are also U-successors of the start individual.

(3) For $\mathcal{ALCN}(\circ,\sqcap)$, a similar construction is possible if we introduce role names R and T. The intuition is that T plays the rôle of the inverse of R (except for one individual), and the "universal" role corresponds to the composition $R\circ T\circ R$:

$$\begin{aligned} E_D^{(3)} := {} & (= 1\ R \sqcap R\circ T\circ R) \sqcap \\ & (\forall R.\forall T.\forall R.\ (C_\sqcap \sqcap C_D \sqcap (\leq 1\ T) \sqcap (\forall Y.(\leq 1\ T)) \sqcap (\forall X.(\leq 1\ T)) \sqcap \\ & \qquad (= 1\ T \sqcap X\circ T \sqcap Y\circ T) \sqcap \\ & \qquad (= 1\ X \sqcap X\circ T\circ R) \sqcap (= 1\ Y \sqcap Y\circ T\circ R))) \end{aligned}$$

The start individual s (which is an instance of $E_D^{(3)}$), has exactly one R-successor $p_{(0,0)}$, which coincides with its $R \circ T \circ R$-successor. The individual $p_{(0,0)}$ corresponds to the origin of $\mathbb{N} \times \mathbb{N}$. Let s' be the $R \circ T$-successor of s. The number restrictions of $E_D^{(3)}$ make sure that $p_{(0,0)}$ satisfies the following: It has exactly one T-successor, namely s', which coincides with the (unique) T-successors of its X- and Y-successors. In addition, the (unique) X-successor of $p_{(0,0)}$ is also an $X \circ T \circ R$-successor of $p_{(0,0)}$, which makes sure that the X-successor of $p_{(0,0)}$ is an R-successor of s', and thus an $R \circ T \circ R$-successor of s. The same holds for the Y-successor. One can now continue the argument with the X-successor (resp. Y-successor) of $p_{(0,0)}$ in place of $p_{(0,0)}$.

With the intuition given above, it is not hard to show for all $i, 1 \leq i \leq 3$, that a tiling system D has a compatible tiling iff $E_D^{(i)}$ is satisfiable.

Theorem 7 *Satisfiability (and thus also subsumption) of concepts is undecidable in $\mathcal{ALC}^+\mathcal{N}(\circ, \sqcup)$, $\mathcal{ALCN}(\circ, \sqcup, ^{-1})$, and $\mathcal{ALCN}(\circ, \sqcap)$.*

The concept $E_D^{(1')}$ shows that the undecidability result for $\mathcal{ALC}^+\mathcal{N}(\circ, \sqcup)$-concepts also hold if only transitive closure of atomic roles is allowed.

5 Related work and open problems

Propositional dynamic logic (PDL), which corresponds to our language $\mathcal{ALC}^+$, has been shown to be decidable in [20], and decidability of its extension by deterministic programs, DPDL, is shown in [5]. In principle, the use of deterministic programs corresponds to introducing a restricted form of number-restrictions, namely $(\leq 1\ R)$ for atomic roles R. Adding inversion (of atomic roles) to DPDL has a drastic consequence: the finite model property is lost, i.e., there are satisfiable formulae (concepts) that do not have a finite model. Nevertheless, satisfiability is still decidable [38] (Exp-time complete, like all the other decision problems for PDL and extensions mentioned until now). It should be noted, however, that in these languages inversion does not occur in the number-restrictions, since only atomic programs are asserted to be deterministic. In [14], general number restrictions and Boolean operators for roles are added to PDL with inversion, and (Exp-time) decidability is shown by a rather ingeneous reduction to the decision problem for PDL. In this work, atomic roles and their inverse my occur in number restrictions, but not more complex roles. If one adds number restrictions on atomic roles and their intersections to $\mathcal{ALC}$, satisfiability in the obtained language is still decidable with a PSPACE-algorithm [17].

As related undecidability results, one can mention undecidability of the extension of DPDL by intersection of roles (which does not occur in the number restrictions, however) [23]. In [22], an extension of $\mathcal{ALC}$ by so-called existential and universal agreements on role chains is shown to be undecidable. It is easy to see that existential (universal) agreements can be simulated by number restrictions involving composition and intersection (union).

In this paper, we have shown that the language $\mathcal{ALCN}(\circ)$, which adds number restrictions on roles with composition to $\mathcal{ALC}$, is still decidable. It is not clear, however, whether there exists a PSPACE algorithm for the problem. The one presented above is Exp-time, and since different role paths need to be joined together, the trace method developed in [36] cannot directly be applied. Almost all extensions of $\mathcal{ALCN}(\circ)$ by union, intersection, and inversion of roles were shown to be undecidable: only decidability for $\mathcal{ALCN}(\circ, \sqcup)$ is still open. For $\mathcal{ALC}^+$, however, the extension by composition and union could be shown to be undecidable. Another interesting question would be what happens if composition is not allowed in number restrictions: Is $\mathcal{ALCN}(\sqcup, \sqcap, \neg, ^{-1})$, the extension of $\mathcal{ALC}$ by number restrictions with inversion and Boolean operators on roles, still decidable? An interesting feature of this language is that it is contained in $\mathcal{C}_2$, i.e., predicate logic with 2 variables and counting quantifiers [11, 7], for which decidability is still open.

References

[1] A. Artale, F. Cesarini, E. Grazzini, F. Pippolini, and G. Soda. Modelling composition in a terminological language environment. In *Workshop Notes of the ECAI Workshop on Parts and Wholes: Conceptual Part-Whole Relations and Formal Mereology*, pages 93–101, Amsterdam, 1994.

[2] 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.

[3] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. Technical Report RR-

90-13, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1990. An abridged version appeared in *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence IJCAI-91*, pp. 446–451.

[4] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proc. of the 16th German AI-Conference, GWAI-92*, volume 671 of *Lecture Notes in Computer Science*, pages 132–143, Bonn, Germany, 1993. Springer-Verlag.

[5] M. Ben-Ari, J.Y. Halpern, and A. Pnueli. Deterministic propositional dynamic logic: finite models, complexity and completeness. *Journal of Computer and System Science*, 25:402–417, 1982.

[6] R. Berger. The undecidability of the dominoe problem. *Memoirs of the American Mathematical Society*, 66, 1966.

[7] A. Borgida. On the relative expressive power of Description Logics and Predicate Calculus. To appear in *Artificial Intelligence*, 1996.

[8] A. Borgida, M. Lenzerini, D. Nardi, and B. Nebel, editors. *Proceedings of the International Workshop on Description Logics*, Rome, 1995.

[9] 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 John F. Sowa, editor, *Principles of Semantic Networks.* Morgan Kaufmann, Los Altos, 1991.

[10] P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: A preliminary report. In Borgida et al. [8].

[11] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. In *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science (FOCS-89)*, pages 612–617. IEEE Computer Society Press, 1989.

[12] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proceedings of the Fourth International Conference on Deductive and Object-Oriented Databases (DOOD-95)*, volume 1013 of *Lecture Notes in Computer Science*, pages 229–246, 1995.

[13] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 109–120, Bonn, 1994. Morgan Kaufmann, Los Altos.

[14] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Università degli Studi di Roma “La Sapienza”, 1995.

[15] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics (extended abstract). In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 1994.

[16] G. De Giacomo and M. Lenzerini. What’s in an aggregate: Foundations for description logics with tuples and sets. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995.

[17] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, Boston (USA), 1991.

[18] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 458–463, Sydney, 1991.

[19] K. Fine. In so many possible worlds. *Notre Dame Journal of Formal Logics*, 13:516–520, 1972.

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

[21] E. Franconi. A treatment of plurals and plural quantifications based on a theory of collections. *Minds and Machines*, 3(4):453–474, November 1994.

[22] P. Hanschke. Specifying Role Interaction in Concept Languages. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, 1992.

[23] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Extensions of Classical Logic*, volume 2. D. Reidel, 1984.

[24] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, pages 335–346, Boston (USA), 1991.

[25] B. Hollunder, W. Nutt, and M. Schmidt-Schauss. Subsumption algorithms for concept description languages. In *ECAI-90*, Pitman Publishing, London, 1990.

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

[27] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In John F. Sowa, editor, *Principles of Semantic Networks*. Morgan Kaufmann, Los Altos, 1991.

[28] E. Mays, R. Dionne, and R. Weida. K-Rep system overview. *SIGART Bulletin*, 2(3):93–97, 1991.

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

[30] L. Padgham and P. Lambrix. A framework for part-of hierarchies in terminological logics. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 485–496, 1994.

[31] P. F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence Journal*, 39:263–272, 1989.

[32] C. Peltason. The BACK System - an Overview. *SIGART Bulletin*, 1991.

[33] U. Sattler. A concept language for an engineering application with part-whole relations. In Borgida et al. [8], pages 119–123.

[34] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, Sydney, 1991.

[35] M. Schmidt-Schauss. Subsumption in KL-ONE is undecidable. In *Proceedings of the First International Conference on the Principles of Knowledge Representation and Reasoning (KR-89)*, pages 421–431, Boston (USA), 1989.

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

[37] W. Van Der Hoek and M. De Rijke. Counting objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.

[38] M. Vardi. The Taming of Converse: Reasoning about Two-Way Computations. In *Proceedings of the Fourth Workshop on Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 413–424. Springer-Verlag, 1985.