Tableau Algorithms for Description Logics

Franz Baader and Ulrike Sattler

Theoretical Computer Science, RWTH Aachen, Germany

Abstract. Description logics are a family of knowledge representation formalisms that are descended from semantic networks and frames via the system KL-ONE. During the last decade, it has been shown that the important reasoning problems (like subsumption and satisfiability) in a great variety of description logics can be decided using tableau-like algorithms. This is not very surprising since description logics have turned out to be closely related to propositional modal logics and logics of programs (such as propositional dynamic logic), for which tableau procedures have been quite successful.
Nevertheless, due to different underlying intuitions and applications, most description logics differ significantly from run-of-the-mill modal and program logics. Consequently, the research on tableau algorithms in description logics led to new techniques and results, which are, however, also of interest for modal logicians. In this article, we will focus on three features that play an important rôle in description logics (number restrictions, terminological axioms, and role constructors), and show how they can be taken into account by tableau algorithms.

1 Introduction

Description logics (DLs) are a family of knowledge representation languages which can be used to represent the terminological knowledge of an application domain in a structured and formally well-understood way. The name *description logics* is motivated by the fact that, on the one hand, the important notions of the domain are described by *concept descriptions*, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL. On the other hand, DLs differ from their predecessors, such as semantic networks and frames [44, 37], in that they are equipped with a formal, *logic*-based semantics, which can, e.g., be given by a translation into first-order predicate logic.

Knowledge representation systems based on description logics (DL systems) provide their users with various inference capabilities that allow them to deduce implicit knowledge from the explicitly represented knowledge. For instance, the *subsumption* algorithm allows one to determine subconcept-superconcept relationships: C is subsumed by D iff all instances of C are also instances of D, i.e., the first description is always interpreted as a subset of the second description. In order to ensure a reasonable and predictable behaviour of a DL system, the subsumption problem for the DL employed by the system should at least be decidable, and preferably of low complexity. Consequently, the expressive power of the DL in question must be restricted in an appropriate way. If the imposed restrictions are too severe, however, then the important notions of the application domain can no longer be expressed. Investigating this trade-off between the expressivity of DLs and the complexity of their inference problems has been one of the most important issues in DL research. Roughly, this research can be classified into the following four phases.

Phase 1: First system implementations. The original KL-ONE system [12] as well as its early successor systems (such as BACK [43], K-REP [36], and LOOM [35]) employ so-called structural subsumption algorithms, which first normalise the concept descriptions, and then recursively compare the syntactic structure of the normalised descriptions (see, e.g., [38] for the description of such an algorithm). These algorithms are usually very efficient (polynomial), but they have the disadvantage that they are complete only for very inexpressive DLs, i.e., for more expressive DLs they cannot detect all the existing subsumption relationships (though this fact was not necessarily known to the designers of the early systems).

Phase 2: First complexity and undecidability results. Partially in parallel with the first phase, the first formal investigations of the subsumption problem in DLs were carried out. It turned out that (under the assumption $\mathrm{P} \neq \mathrm{NP}$) already quite inexpressive DLs cannot have polynomial subsumption algorithms [10, 39], and that the DL used by the KL-ONE system even has an undecidable subsumption problem [49]. In particular, these results showed the incompleteness of the (polynomial) structural subsumption algorithms. One reaction to these results (e.g., by the designers of BACK and LOOM) was to call the incompleteness of the subsumption algorithm a feature rather than a bug of the DL system. The designers of the CLASSIC system [42, 9] followed another approach: they carefully chose a restricted DL that still allowed for an (almost) complete polynomial structural subsumption algorithm [8].

Phase 3: Tableau algorithms for expressive DLs and thorough complexity analysis. For expressive DLs (in particular, DLs allowing for disjunction and/or negation), for which the structural approach does not lead to complete subsumption algorithms, tableau algorithms have turned out to be quite useful: they are complete and often of optimal (worst-case) complexity. The first such algorithm was proposed by Schmidt-Schauß and Smolka [50] for a DL that they called $\mathcal{ALC}$ (for "attributive concept description language with complements").[1] It quickly turned out that this approach for deciding subsumption could be extended to various other DLs [28, 26, 4, 1, 23] and also to other inference problems such as the instance problem [24]. Early on, DL researchers started to call the algorithms obtained this way "tableau-based algorithms" since they observed that the original algorithm by Schmidt-Schauß and Smolka for $\mathcal{ALC}$, as well as subsequent algorithms for more expressive DL, could be seen as specialisations of the tab-

[1] Actually, at that time the authors were not aware of the close connection between their rule-based algorithm working on constraint systems and tableau procedures for modal and first-order predicate logics.

leau calculus for first-order predicate logic (the main problem to solve was to find a specialisation that always terminates, and thus yields a decision procedure). After Schild [47] showed that $\mathcal{ALC}$ is just a syntactic variant of multi-modal K, it turned out that the algorithm by Schmidt-Schauß and Smolka was actually a re-invention of the known tableau algorithm for K.

At the same time, the (worst-case) complexity of a various DLs (in particular also DLs that are not propositionally closed) was investigated in detail [20, 21, 19].

The first DL systems employing tableau algorithms (Kris [5] and Crack [13]) demonstrated that (in spite of their high worst-case complexity) these algorithms lead to acceptable behaviour in practice [6]. Highly optimised systems such as FaCT [30] have an even better behaviour, also for benchmark problems in modal logics [29, 31].

Phase 4: Algorithms and efficient systems for very expressive DLs. Motivated by applications (e.g., in the database area), DL researchers started to investigate DLs whose expressive power goes far beyond the one of $\mathcal{ALC}$ (e.g., DLs that do not have the finite model property). First decidability and complexity results for such DLs could be obtained from the connection between propositional dynamic logic (PDL) and DLs [47]. The idea of this approach, which was perfected by DeGiacomo and Lenzerini, is to translate the DL in question into PDL. If the translation is polynomial and preserves satisfiability, then the known EXPTIME-algorithms for PDL can be employed to decide subsumption in exponential time. Though this approach has produced very strong complexity results [16–18] it turned out to be less satisfactory from a practical point of view. In fact, first tests in a database application [33] showed that the PDL formulae obtained by the translation technique could not be handled by existing efficient implementations of satisfiability algorithms for PDL [41]. To overcome this problem, DL researchers have started to design "practical" tableau algorithms for *very* expressive DLs [32, 33].

The purpose of this article is to give an impression of the work on tableau algorithms done in the DL community, with an emphasis on features that, though they may also occur in modal logics, are of special interest to description logics. After introducing some basic notions of description logics in Section 2, we will describe a tableau algorithm for $\mathcal{ALC}$ in Section 3. Although, from the modal logic point of view, this is just the well-known algorithm for multi-modal K, this section will introduce the notations and techniques used in description logics, and thus set the stage for extensions to more interesting DLs. In the subsequent three section we will show how the basic algorithm can be extended to one that treats number restrictions, terminological axioms, and role constructors of different expressiveness, respectively.

2 Description logics: basic definitions

The main expressive means of description logics are so-called concept descriptions, which describe sets of individuals or objects. Formally, *concept descriptions*

Table 1. Syntax and semantics of concept descriptions.

Construct name	Syntax	Semantics
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \to y \in C^{\mathcal{I}}\}$
at-least restriction	$(\geqslant nr.C)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$
at-most restriction	$(\leqslant nr.C)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$

are inductively defined with the help of a set of *concept constructors*, starting with a set N_C of *concept names* and a set N_R of *role names*. The available constructors determine the expressive power of the DL in question. In this paper, we consider concept descriptions built from the constructors shown in Table 1, where C, D stand for concept descriptions, r for a role name, and n for a non-negative integer. In the description logic $\mathcal{ALC}$, concept descriptions are formed using the constructors negation, conjunction, disjunction, value restriction, and existential restriction. The description logic $\mathcal{ALCQ}$ additionally provides us with (qualified) at-least and at-most *number restrictions.*

The semantics of concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $P \in N_C$ to a set $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1.

From the modal logic point of view, roles are simply names for accessibility relations, and existential (value) restrictions correspond to diamonds (boxes) indexed by the respective accessibility relation. Thus, any $\mathcal{ALC}$ description can be translated into a multi-modal formula and vice versa. For example, the description $P \sqcap \exists r.P \sqcap \forall r.\neg P$ corresponds to the formula $p \wedge \langle r \rangle p \wedge [r]\neg p$, where p is an atomic proposition corresponding to the concept name P. As pointed out by Schild [47], there is an obvious correspondence between the semantics of $\mathcal{ALC}$ and the Kripke semantics for multi-modal K, which satisfies $d \in C^{\mathcal{I}}$ iff the world d satisfies the formula ϕ_C corresponding to C in the Kripke structure corresponding to $\mathcal{I}$. Number restrictions also have a corresponding construct in modal logics, so-called graded modalities [53], but these are not as well-investigated as the modal logic K.

One of the most important inference services provided by DL systems is computing the subsumption hierarchy of a given finite set of concept descriptions.

Definition 1. *The concept description D* subsumes *the concept description C ($C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$; C is* satisfiable *iff there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$; and C and D are equivalent iff $C \sqsubseteq D$ and $D \sqsubseteq C$.*

In the presence of negation, subsumption can obviously be reduced to satisfiability: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable.[2]

Given concept descriptions that define the important notions of an application domain, one can then describe a concrete situation with the help of the assertional formalism of description logics.

Definition 2. *Let N_I be a set of* individual names. *An* ABox *is a finite set of assertions of the form $C(a)$ (*concept assertion*) or $r(a,b)$ (*role assertion*), where C is a concept description, r a role name, and a, b are individual names.*

An interpretation $\mathcal{I}$, which additionally assigns elements $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to individual names a, is a model *of an ABox $\mathcal{A}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ $((a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}})$ holds for all assertions $C(a)$ $(r(a,b))$ in $\mathcal{A}$.*

The Abox $\mathcal{A}$ is consistent *iff it has a model. The individual a is an* instance *of the description C w.r.t. $\mathcal{A}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{A}$.*

Satisfiability (and thus also subsumption) of concept descriptions as well as the instance problem can be reduced to the consistency problem for ABoxes: (i) C is satisfiable iff the ABox $\{C(a)\}$ for some $a \in N_I$ is consistent; and (ii) a is an instance of C w.r.t. $\mathcal{A}$ iff $\mathcal{A} \cup \{\neg C(a)\}$ is inconsistent.

Usually, one imposes the unique name assumption on ABoxes, i.e., requires the mapping from individual names to elements of $\Delta^{\mathcal{I}}$ to be injective. Here, we dispense with this requirement since it has no effect for $\mathcal{ALC}$, and for DLs with number restrictions we will explicitly introduce inequality assertions, which can be used to express the unique name assumption.

3 A tableau algorithm for $\mathcal{ALC}$

Given an $\mathcal{ALC}$-concept description C_0, the tableau algorithm for satisfiability tries to construct a finite interpretation $\mathcal{I}$ that satisfies C_0, i.e., contains an element x_0 such that $x_0 \in C_0^{\mathcal{I}}$. Before we can describe the algorithm more formally, we need to introduce an appropriate data structure in which to represent (partial descriptions of) finite interpretations. The original paper by Schmidt-Schauß and Smolka [50], and also many other papers on tableau algorithms for DLs, introduce the new notion of a constraint system for this purpose. However, if we look at the information that must be expressed (namely, the elements of the interpretation, the concept descriptions they belong to, and their role relationships), we see that ABox assertions are sufficient for this purpose.

It will be convenient to assume that all concept descriptions are in *negation normal form* (NNF), i.e., that negation occurs only directly in front of concept names. Using de Morgan's rules and the usual rules for quantifiers, any $\mathcal{ALC}$-concept description can be transformed (in linear time) into an equivalent description in NNF.

Let C_0 by an $\mathcal{ALC}$-concept in NNF. In order to test satisfiability of C_0, the algorithm starts with $\mathcal{A}_0 := \{C_0(x_0)\}$, and applies consistency preserving

[2] This was the reason why Schmidt-Schauß and Smolka [50] added negation to their DL in the first place.

The $\rightarrow_\sqcap$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but it does not contain both $C_1(x)$ and $C_2(x)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x), C_2(x)\}$.

The $\rightarrow_\sqcup$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{C_2(x)\}$.

The $\rightarrow_\exists$-rule
Condition: $\mathcal{A}$ contains $(\exists R.C)(x)$, but there is no individual name z such that $C(z)$ and $R(x, z)$ are in $\mathcal{A}$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y), R(x, y)\}$ where y is an individual name not occurring in $\mathcal{A}$.

The $\rightarrow_\forall$-rule
Condition: $\mathcal{A}$ contains $(\forall R.C)(x)$ and $R(x, y)$, but it does not contain $C(y)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$.

Fig. 1. Transformation rules of the satisfiability algorithm for $\mathcal{ALC}$.

transformation rules (see Fig. 1) to this ABox. The transformation rule that handles disjunction is *nondeterministic* in the sense that a given ABox is transformed into two new ABoxes such that the original ABox is consistent iff *one of* the new ABoxes is so. For this reason we will consider finite sets of ABoxes $\mathcal{S} = \{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ instead of single ABoxes. Such a set is *consistent* iff there is some i, $1 \leq i \leq k$, such that $\mathcal{A}_i$ is consistent. A rule of Fig. 1 is applied to a given finite set of ABoxes $\mathcal{S}$ as follows: it takes an element $\mathcal{A}$ of $\mathcal{S}$, and replaces it by one ABox $\mathcal{A}'$ or by two ABoxes $\mathcal{A}'$ and $\mathcal{A}''$.

Definition 3. *An ABox $\mathcal{A}$ is called* complete *iff none of the transformation rules of Fig. 1 applies to it. The ABox $\mathcal{A}$ contains a* clash *iff $\{P(x), \neg P(x)\} \subseteq \mathcal{A}$ for some individual name x and some concept name P. An ABox is called* closed *if it contains a clash, and* open *otherwise.*

The *satisfiability algorithm for $\mathcal{ALC}$* works as follows. It starts with the singleton set of ABoxes $\{\{C_0(x_0)\}\}$, and applies the rules of Fig. 1 (in arbitrary order) until no more rules apply. It answers "satisfiable" if the set $\widehat{\mathcal{S}}$ of ABoxes obtained this way contains an open ABox, and "unsatisfiable" otherwise. Correctness of this algorithm is an easy consequence of the following lemma.

Lemma 1. *Let C_0 by an $\mathcal{ALC}$-concept in negation normal form.*

1. *There cannot be an infinite sequence of rule applications*

$$\{\{C_0(x_0)\}\} \rightarrow \mathcal{S}_1 \rightarrow \mathcal{S}_2 \rightarrow \cdots .$$

2. *Assume that $\mathcal{S}'$ is obtained from the finite set of ABoxes $\mathcal{S}$ by application of a transformation rule. Then $\mathcal{S}$ is consistent iff $\mathcal{S}'$ is consistent.*

3. Any closed ABox $\mathcal{A}$ is inconsistent.
4. Any complete and open ABox $\mathcal{A}$ is consistent.

The first part of this lemma (termination) is an easy consequence of the facts that (i) all concept assertions occurring in an ABox in one of the sets $\mathcal{S}_i$ are of the form $C(x)$ were C is a subdescription of C_0; and (ii) if an ABox in $\mathcal{S}_i$ contains the role assertion $r(x, y)$, then the maximal role depth (i.e., nesting of value and existential restrictions) of concept descriptions occurring in concept assertions for y is strictly smaller than the maximal role depth of concept descriptions occurring in concept assertions for x. A detailed proof of termination (using an explicit mapping into a well-founded ordering) for a set of rules extending the one of Fig. 1 can, e.g., be found in [4].

The second and third part of the lemma are quite obvious, and the fourth part can be proved by defining the *canonical interpretation* $\mathcal{I}_\mathcal{A}$ induced by $\mathcal{A}$:

1. The domain $\Delta^{\mathcal{I}_\mathcal{A}}$ of $\mathcal{I}_\mathcal{A}$ consists of all the individual names occurring in $\mathcal{A}$.
2. For all concept names P we define $P^{\mathcal{I}_\mathcal{A}} := \{x \mid P(x) \in \mathcal{A}\}$.
3. For all role names r we define $r^{\mathcal{I}_\mathcal{A}} := \{(x, y) \mid r(x, y) \in \mathcal{A}\}$.

By definition, $\mathcal{I}_\mathcal{A}$ satisfies all the role assertions in $\mathcal{A}$. By induction on the structure of concept descriptions, it is easy to show that it satisfies the concept assertions as well, provided that $\mathcal{A}$ is complete and open.

It is also easy to show that the canonical interpretation has the shape of a finite tree whose depth is linearly bounded by the size of C_0 and whose branching factor is bounded by the number of different existential restrictions in C_0. Consequently, $\mathcal{ALC}$ has the *finite tree model property*, i.e., any satisfiable concept C_0 is satisfiable in a finite interpretation $\mathcal{I}$ that has the shape of a tree whose root belongs to C_0.

To sum up, we have seen that the transformation rules of Fig. 1 reduce satisfiability of an $\mathcal{ALC}$-concept C_0 (in NNF) to consistency of a finite set $\widehat{\mathcal{S}}$ of complete ABoxes. In addition, consistency of $\widehat{\mathcal{S}}$ can be decided by looking for obvious contradictions (clashes).

Theorem 1. *It is decidable whether or not an $\mathcal{ALC}$-concept is satisfiable.*

Complexity issues. The satisfiability algorithm for $\mathcal{ALC}$ presented above may need exponential time and space. In fact, the size of the complete and open ABox (and thus of the canonical interpretation) built by the algorithm may be exponential in the size of the concept description. For example, consider the descriptions C_n ($n \geq 1$) that are inductively defined as follows:

$$\begin{aligned} C_1 &:= \exists R.A \sqcap \exists R.B, \\ C_{n+1} &:= \exists R.A \sqcap \exists R.B \sqcap \forall R.C_n. \end{aligned}$$

Obviously, the size of C_n grows linearly in n. However, given the input description C_n, the satisfiability algorithm generates a complete and open ABox whose canonical interpretation is a binary tree of depth n, and thus consists of $2^{n+1}-1$ individuals.

Nevertheless, the algorithm can be modified such that it needs only polynomial space. The main reason is that different branches of the tree model to be generated by the algorithm can be investigated separately, and thus the tree can be built and searched in a depth-first manner. Since the complexity class NPSPACE coincides with PSPACE [46], it is sufficient to describe a nondeterministic algorithm using only polynomial space, i.e., for the nondeterministic $\rightarrow_\sqcup$-rule, we may simply assume that the algorithm chooses the correct alternative. In principle, the modified algorithm works as follows: it starts with $\{C_0(x_0)\}$ and

1. applies the $\rightarrow_\sqcap$- and $\rightarrow_\sqcup$-rules as long as possible and checks for clashes;
2. generates all the necessary direct successors of x_0 using the $\rightarrow_\exists$-rule and exhaustively applies the $\rightarrow_\forall$-rule to the corresponding role assertions;
3. successively handles the successors in the same way.

Since the successors of a given individual can be treated separately, the algorithm needs to store only one path of the tree model to be generated, together with the *direct* successors of the individuals on this path and the information which of these successors must be investigated next. Since the length of the path is linear in the size of the input description C_0, and the number of successors is bounded by the number of different existential restrictions in C_0, the necessary information can obviously be stored within polynomial space.

This shows that the satisfiability problem for $\mathcal{ALC}$-concept descriptions is in PSPACE. PSPACE-hardness can be shown by a reduction from validity of Quantified Boolean Formulae [50].

Theorem 2. *Satisfiability of $\mathcal{ALC}$-concept descriptions is PSPACE-complete.*

The consistency problem for $\mathcal{ALC}$-ABoxes. The satisfiability algorithm described above can also be used to decide consistency of $\mathcal{ALC}$-ABoxes. Let $\mathcal{A}_0$ be an $\mathcal{ALC}$-ABox such that (w.l.o.g.) all concept descriptions in $\mathcal{A}$ are in NNF. To test $\mathcal{A}_0$ for consistency, we simply apply the rules of Fig. 1 to the singleton set $\{\mathcal{A}_0\}$. It is easy to show that Lemma 1 still holds. Indeed, the only point that needs additional consideration is the first one (termination). Thus, the rules of Fig. 1 yield a decision procedure for consistency of $\mathcal{ALC}$-ABoxes.

Since now the canonical interpretation obtained from a complete and open ABox need no longer be of tree shape, the argument used to show that the satisfiability problem is in PSPACE cannot directly be applied to the consistency problem. In order to show that the consistency problem is in PSPACE, one can, however, proceed as follows: In a *pre-completion* step, one applies the transformation rules only to *old* individuals (i.e., individuals present in the original ABox $\mathcal{A}_0$). Subsequently, one can forget about the role assertions, i.e., for each individual name in the pre-completed ABox, the satisfiability algorithm is applied to the conjunction of its concept assertions (see [25] for details).

Theorem 3. *Consistency of $\mathcal{ALC}$-ABoxes is PSPACE-complete.*

4 Number restrictions

Before treating the qualified number restrictions introduced in Section 2, we consider a restricted form of number restrictions, which is the form present in most DL systems. In *unqualified* number restrictions, the qualifying concept is the top concept $\top$, where $\top$ is an abbreviation for $P \sqcup \neg P$, i.e., a concept that is always interpreted by the whole interpretation domain. Instead of $(\geqslant nr.\top)$ and $(\leqslant nr.\top)$, we write unqualified number restrictions simply as $(\geqslant nr)$ and $(\leqslant nr)$. The DL that extends $\mathcal{ALC}$ by unqualified number restrictions is denoted by $\mathcal{ALCN}$.

Obviously, $\mathcal{ALCN}$- and $\mathcal{ALCQ}$-concept descriptions can also be transformed into NNF in linear time.

4.1 A tableau algorithm for $\mathcal{ALCN}$

The main idea underlying the extension of the tableau algorithm for $\mathcal{ALC}$ to $\mathcal{ALCN}$ is quite simple. At-least restrictions are treated by generating the required role successors as new individuals. At-most restrictions that are currently violated are treated by (nondeterministically) identifying some of the role successors. To avoid running into a generate-identify cycle, we introduce explicit inequality assertions that prohibit the identification of individuals that were introduced to satisfy an at-least restriction.

Inequality assertions are of the form $x \not\doteq y$ for individual names x, y, with the obvious semantics that an interpretation $\mathcal{I}$ satisfies $x \not\doteq y$ iff $x^{\mathcal{I}} \neq y^{\mathcal{I}}$. These assertions are assumed to be symmetric, i.e., saying that $x \not\doteq y$ belongs to an ABox $\mathcal{A}$ is the same as saying that $y \not\doteq x$ belongs to $\mathcal{A}$.

The *satisfiability algorithm* for $\mathcal{ALCN}$ is obtained from the one for $\mathcal{ALC}$ by adding the rules in Fig. 2, and by considering a second type of *clashes*:

– $\{(\leqslant nr)(x)\} \cup \{r(x, y_i) \mid 1 \leq i \leq n+1\} \cup \{y_i \not\doteq y_j \mid 1 \leq i < j \leq n+1\} \subseteq \mathcal{A}$ for $x, y_1, \ldots, y_{n+1} \in N_I$, $r \in N_R$, and a nonnegative integer n.

The nondeterministic $\rightarrow_{\leq}$-rule replaces the ABox $\mathcal{A}$ by finitely many new ABoxes $\mathcal{A}_{i,j}$. Lemma 1 still holds for the extended algorithm (see e.g. [7], where this is proved for a more expressive DL). This shows that satisfiability (and thus also subsumption) of $\mathcal{ALCN}$-concept descriptions is decidable.

Complexity issues. The ideas that lead to a PSPACE algorithm for $\mathcal{ALC}$ can be applied to the extended algorithm as well. The only difference is that, before handling the successors of an individual (introduced by at-least and existential restrictions), one must check for clashes of the second type and generate the necessary identifications. However, this simple extension only leads to a PSPACE algorithm if we assume the numbers in at-least restrictions to be written in base 1 representation (where the size of the representation coincides with the number represented). For bases larger than 1 (e.g., numbers in decimal notation), the number represented may be exponential in the size of the representation. Thus,

The $\rightarrow_{\geq}$-rule
Condition: $\mathcal{A}$ contains $(\geqslant nr)(x)$, and there are no individual names $z_1, \ldots, z_n$ such that $r(x, z_i)$ $(1 \leq i \leq n)$ and $z_i \neq z_j$ $(1 \leq i < j \leq n)$ are contained in $\mathcal{A}$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{r(x, y_i) \mid 1 \leq i \leq n\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n\}$, where $y_1, \ldots, y_n$ are distinct individual names not occurring in $\mathcal{A}$.

The $\rightarrow_{\leq}$-rule
Condition: $\mathcal{A}$ contains distinct individual names $y_1, \ldots, y_{n+1}$ such that $(\leqslant nr)(x)$ and $r(x, y_1), \ldots, r(x, y_{n+1})$ are in $\mathcal{A}$, and $y_i \neq y_j$ is not in $\mathcal{A}$ for some $i \neq j$.
Action: For each pair y_i, y_j such that $i < j$ and $y_i \neq y_j$ is not in $\mathcal{A}$, the ABox $\mathcal{A}_{i,j} := [y_i/y_j]\mathcal{A}$ is obtained from $\mathcal{A}$ by replacing each occurrence of y_i by y_j.

Fig. 2. The transformation rules handling unqualified number restrictions.

we cannot introduce all the successors required by at-least restrictions while only using space polynomial in the size of the concept description if the numbers in this description are written in decimal notation.

It is not hard to see, however, that most of the successors required by the at-least restrictions need not be introduced at all. If an individual x obtains at least one r-successor due to the application of the $\rightarrow_{\exists}$-rule, then the $\rightarrow_{\geq}$-rule need not be applied to x for the role r. Otherwise, we simply introduce *one* r-successor as representative. In order to detect inconsistencies due to conflicting number restrictions, we need to add *another type of clashes:* $\{(\leqslant nr)(x), (\geqslant mr)(x)\} \subseteq \mathcal{A}$ for nonnegative integers $n < m$. The canonical interpretation obtained by this modified algorithm need not satisfy the at-least restrictions in C_0. However, it can easily by modified to an interpretation that does, by duplicating r-successors (more precisely, the whole subtrees starting at these successors).

Theorem 4. *Satisfiability of $\mathcal{ALCN}$-concept descriptions is PSPACE-complete.*

The consistency problem for $\mathcal{ALCN}$-ABoxes. Just as for $\mathcal{ALC}$, the extended rule set for $\mathcal{ALCN}$ can also be applied to arbitrary ABoxes. Unfortunately, the algorithm obtained this way need *not terminate*, unless one imposes a specific strategy on the order of rule applications. For example, consider the ABox

$$\mathcal{A}_0 := \{r(a, a),\ (\exists R.P)(a),\ (\leqslant 1r)(a),\ (\forall r.\exists r.P)(a)\}.$$

By applying the $\rightarrow_{\exists}$-rule to a, we can introduce a new r-successor x of a:

$$\mathcal{A}_1 := \mathcal{A}_0 \cup \{r(a, x),\ P(x)\}.$$

The $\rightarrow_{\forall}$-rule adds the assertion $(\exists r.P)(x)$, which triggers an application of the $\rightarrow_{\exists}$-rule to x. Thus, we obtain the new ABox

$$\mathcal{A}_2 := \mathcal{A}_1 \cup \{(\exists r.P)(x),\ r(x, y),\ P(y)\}.$$

The $\rightarrow_{\textbf{choose}}$-rule
Condition: $\mathcal{A}$ contains $(\leqslant nr.C)(x)$ and $r(x,y)$, but neither $C(y)$ nor $\neg C(y)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{\neg C(y)\}$.

Fig. 3. The $\rightarrow_{\text{choose}}$-rule for qualified number restrictions.

Since a has two r-successors in $\mathcal{A}_2$, the $\rightarrow_{\leq}$-rule is applicable to a. By replacing every occurrence of x by a, we obtain the ABox

$$\mathcal{A}_3 := \mathcal{A}_0 \cup \{P(a),\, r(a,y),\, P(y)\}.$$

Except for the individual names (and the assertion $P(a)$, which is, however, irrelevant), $\mathcal{A}_3$ is identical to $\mathcal{A}_1$. For this reason, we can continue as above to obtain an infinite chain of rule applications.

We can easily regain termination by requiring that *generating rules* (i.e., the rules $\rightarrow_{\exists}$ and $\rightarrow_{\geq}$) may only be applied if none of the other rules is applicable. In the above example, this strategy would prevent the application of the $\rightarrow_{\exists}$-rule to x in the ABox $\mathcal{A}_1 \cup \{(\exists r.P)(x)\}$ since the $\rightarrow_{\leq}$-rule is also applicable. After applying the $\rightarrow_{\leq}$-rule (which replaces x by a), the $\rightarrow_{\exists}$-rule is no longer applicable since a already has an r-successor that belongs to P.

In order to obtain a PSPACE algorithm for consistency of $\mathcal{ALCN}$-ABoxes, the pre-completion technique sketched above for $\mathcal{ALC}$ can also be applied to $\mathcal{ALCN}$ [25].

Theorem 5. *Consistency of $\mathcal{ALCN}$-ABoxes is PSPACE-complete.*

4.2 A tableau algorithm for $\mathcal{ALCQ}$

An obvious idea when attempting to extend the satisfiability algorithm for $\mathcal{ALCN}$ to one that can handle $\mathcal{ALCQ}$ is the following (see [53]):

- Instead of simply generating n new r-successors $y_1, \ldots, y_n$ in the $\rightarrow_{\geq}$-rule, one also asserts that these individuals must belong to the qualifying concept C by adding the assertions $C(y_i)$ to $\mathcal{A}'$.
- The $\rightarrow_{\leq}$-rule only applies if $\mathcal{A}$ also contains the assertions $C(y_i)$ $(1 \leq i \leq n+1)$.

Unfortunately, this does not yield a correct algorithm for satisfiability in $\mathcal{ALCQ}$. In fact, this simple algorithm would not detect that the concept description $(\geqslant 3r) \sqcap (\leqslant 1r.P) \sqcap (\leqslant 1r.\neg P)$ is unsatisfiable. The (obvious) problem is that, for some individuals a and concept descriptions C, the ABox may neither contain $C(a)$ nor $\neg C(a)$, whereas in the canonical interpretation constructed from the ABox, one of the two must hold. In order to overcome this problem, the nondeterministic $\rightarrow_{\text{choose}}$-rule of Fig. 3 must be added [26]. Together with the $\rightarrow_{\text{choose}}$-rule, the simple modification of the $\rightarrow_{\geq}$- and $\rightarrow_{\leq}$-rule described above yields a correct algorithm for satisfiability in $\mathcal{ALCQ}$ [26].

Complexity issues. The approach that leads to a PSPACE-algorithm for $\mathcal{ALC}$ can be applied to the algorithm for $\mathcal{ALCQ}$ as well. However, as with $\mathcal{ALCN}$, this yields a PSPACE-algorithm only if the numbers in number restrictions are assumed to be written in base 1 representation. For $\mathcal{ALCQ}$, the idea that leads to a PSPACE-algorithm for $\mathcal{ALCN}$ with decimal notation does no longer work: it is not sufficient to introduce just one successor as representative for the role successors required by at-least restrictions. Nevertheless, it is possible to design a PSPACE-algorithm for $\mathcal{ALCQ}$ also w.r.t. decimal notation of numbers [52]. Like the PSPACE-algorithm for $\mathcal{ALC}$, this algorithm treats the successors separately. It uses appropriate counters (and a new type of clashes) to check whether qualified number restrictions are satisfied. By combining the pre-completion approach of [25] with this algorithm, we also obtain a PSPACE-result for consistency of $\mathcal{ALCQ}$-ABoxes.

Theorem 6. *Satisfiability of $\mathcal{ALCQ}$-concept descriptions as well as consistency of $\mathcal{ALCQ}$-ABoxes are PSPACE-complete problems.*

5 Terminological axioms

DLs systems usually provide their users also with a terminological formalism. In its simplest form, this formalism can be used to introduce names for complex concept descriptions. More general terminological formalisms can be used to state connections between complex concept descriptions.

Definition 4. *A* TBox *is a finite set of terminological axioms of the form $C \doteq D$, where C, D are concept descriptions. The terminological axiom $C \doteq D$ is called* concept definition *iff C is a concept name.*

An interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds for all terminological axioms $C \doteq D$ in $\mathcal{T}$.

The concept description D subsumes *the concept description C* w.r.t. the TBox *$\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$; C is* satisfiable w.r.t. *$\mathcal{T}$ iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. The Abox $\mathcal{A}$ is* consistent w.r.t. *$\mathcal{T}$ iff it has a model that is also a model of $\mathcal{T}$. The individual a is an* instance of *C* w.r.t. *$\mathcal{A}$* and *$\mathcal{T}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds for each model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$.*

In the following, we restrict our attention to terminological reasoning (i.e., the satisfiability and subsumption problem) w.r.t. TBoxes; however, the methods and results also apply to assertional reasoning (i.e., the instance and the consistency problem for ABoxes).

5.1 Acyclic terminologies

The early DL systems provided TBoxes only for introducing names as abbreviations for complex descriptions. This is possible with the help of acyclic terminologies.

Definition 5. *A TBox is an* acyclic terminology *iff it is a set of concept definitions that neither contains multiple definitions nor cyclic definitions.* Multiple definitions *are of the form* $A \doteq C, A \doteq D$ *for distinct concept descriptions* C, D, *and* cyclic definitions *are of the form* $A_1 \doteq C_1, \ldots, A_n \doteq C_n$, *where* A_i *occurs in* C_{i-1} $(1 < i \leq n)$ *and* A_1 *occurs in* C_n. *If the acyclic terminology* $\mathcal{T}$ *contains a concept definition* $A \doteq C$, *then* A *is called* defined name *and* C *its* defining concept.

Reasoning w.r.t. *acyclic terminologies* can be reduced to reasoning without TBoxes by *unfolding* the definitions: this is achieved by repeatedly replacing defined names by their defining concepts until no more defined names occur. Unfortunately, unfolding may lead to an exponential blow-up, as the following acyclic terminology (due to Nebel [39]) demonstrates:

$$\{A_0 \doteq \forall r.A_1 \sqcap \forall s.A_1, \ldots, A_{n-1} \doteq \forall r.A_n \sqcap \forall s.A_n\}.$$

This terminology is of size linear in n, but unfolding applied to A_0 results in a concept description containing the name A_n 2^n times. Nebel [39] also shows that this complexity can, in general, not be avoided: for the DL $\mathcal{FL}_0$, which allows for conjunction and value restriction only, subsumption between concept descriptions can be tested in polynomial time, whereas subsumption w.r.t. acyclic terminologies is coNP-complete.

For more expressive languages, the presence of acyclic TBoxes may or may not increase the complexity of the subsumption problem. For example, subsumption of concept descriptions in the language $\mathcal{ALC}$ is PSPACE-complete, and so is subsumption w.r.t. acyclic terminologies [34]. Of course, in order to obtain a PSPACE-algorithm for subsumption in $\mathcal{ALC}$ w.r.t. acyclic terminologies, one cannot first apply unfolding to the concept descriptions to be tested for subsumption since this may need exponential space. The main idea is that one uses a tableau algorithm like the one described in Section 3, with the difference that it receives concept descriptions containing defined names as input. *Unfolding* is then done *on demand*: if the tableau algorithm encounters an assertion of the form $A(x)$, where A is a name occurring on the left-hand side of a definition $A \doteq C$ in the terminology, then it adds the assertion $C(x)$. However, it does not further unfold C at this stage. It is not hard to show that this really yields a PSPACE-algorithm for satisfiability (and thus also for subsumption) of concepts w.r.t. acyclic terminologies in $\mathcal{ALC}$ [34].

Theorem 7. *Satisfiability w.r.t. acyclic terminologies is PSPACE-complete in* $\mathcal{ALC}$.

Although this technique also works for many extensions of $\mathcal{ALC}$ (such as $\mathcal{ALCN}$ and $\mathcal{ALCQ}$), there are extensions for which it fails. One such example is the language $\mathcal{ALCF}$, which extends $\mathcal{ALC}$ by functional roles as well as agreements and disagreements on chains of functional roles (see, e.g., [34] for the definition of these constructors). Satisfiability of concept descriptions is PSPACE-complete for this DL [27], but satisfiability of concept descriptions w.r.t. acyclic terminologies is NEXPTIME-complete [34].

5.2 General TBoxes

For general terminological axioms of the form $C \doteq D$, where C may also be a complex description, unfolding is obviously no longer possible. Instead of considering finitely many such axiom $C_1 \doteq D_1, \ldots, C_n \doteq D_n$, it is sufficient to consider the single axiom $\widehat{C} \doteq \top$, where

$$\widehat{C} := (\neg C_1 \sqcup D_1) \sqcap (C_1 \sqcup \neg D_1) \sqcap \cdots \sqcap (\neg C_n \sqcup D_n) \sqcap (C_n \sqcup \neg D_n)$$

and $\top$ is an abbreviation for $P \sqcup \neg P$.

The axiom $\widehat{C} \doteq \top$ just says that any individual must belong to the concept $\widehat{C}$. The tableau algorithm for $\mathcal{ALC}$ introduced in Section 3 can easily be modified such that it takes this axiom into account: all individuals are simply asserted to belong to $\widehat{C}$. However, this modification may obviously lead to nontermination of the algorithm.

For example, consider what happens if this algorithm is applied to test consistency of the ABox $\mathcal{A}_0 := \{(\exists r.P)(x_0)\}$ modulo the axiom $\exists r.P \doteq \top$: the algorithm generates an infinite sequence of ABoxes $\mathcal{A}_1, \mathcal{A}_2, \ldots$ and individuals $x_1, x_2, \ldots$ such that $\mathcal{A}_{i+1} := \mathcal{A}_i \cup \{r(x_i, x_{i+1}),\ P(x_{i+1}),\ (\exists r.P)(x_{i+1})\}$. Since all individuals x_i $(i \geq 1)$ receive the same concept assertions as x_1, we may say that the algorithms has run into a cycle.

Termination can be regained by trying to detect such cyclic computations, and then blocking the application of generating rules: the application of the rule $\rightarrow_\exists$ to an individual x is *blocked* by an individual y in an ABox $\mathcal{A}$ iff $\{D \mid D(x) \in \mathcal{A}\} \subseteq \{D' \mid D'(y) \in \mathcal{A}\}$. The main idea underlying blocking is that the blocked individual x can use the role successors of y instead of generating new ones. For example, instead of generating a new r-successor for x_2 in the above example, one can simply use the r-successor of x_1. This yields an interpretation $\mathcal{I}$ with $\Delta^{\mathcal{I}} := \{x_0, x_1, x_2\}$, $P^{\mathcal{I}} := \{x_1, x_2\}$, and $r^{\mathcal{I}} := \{(x_0, x_1),\ (x_1, x_2), (x_2, x_2)\}$. Obviously, $\mathcal{I}$ is a model of both $\mathcal{A}_0$ and the axiom $\exists r.P \doteq \top$.

To avoid cyclic blocking (of x by y and vice versa), we consider an enumeration of all individual names, and define that an individual x may only be blocked by individuals y that occur before x in this enumeration. This, together with some other technical assumptions, makes sure that a tableau algorithm using this notion of blocking is sound and complete as well as terminating both for $\mathcal{ALC}$ and $\mathcal{ALCN}$ (see [14, 2] for details).

Theorem 8. *Consistency of $\mathcal{ALCN}$-ABoxes w.r.t. TBoxes is decidable.*

It should be noted that the algorithm is no longer in PSPACE since it may generate role paths of exponential length before blocking occurs. In fact, even for the language $\mathcal{ALC}$, satisfiability modulo general terminological axioms is known to be EXPTIME-complete [48].

Blocking does not work for all extensions of $\mathcal{ALC}$ that have a tableau-based satisfiability algorithm. An example is again the DL $\mathcal{ALCF}$, for which satisfiability is decidable, but satisfiability w.r.t. general TBoxes undecidable [40, 3].

6 Expressive roles

The DLs considered until now allowed for atomic roles only. There are two ways of extending the expressivity of DLs w.r.t. roles: adding role constructors and allowing to constrain the interpretation of roles.

Role constructors can be used to build complex roles from atomic ones. In the following, we will restrict our attention to the inverse constructor, but other interesting role constructors have been considered in the literature (e.g., Boolean operators [15] or composition and transitive closure [1, 47]). The *inverse* r^- of a role name r has the obvious semantics: $(r^-)^\mathcal{I} := \{(y,x) \mid (x,y) \in r^\mathcal{I}\}$.

Constraining the interpretation of roles is very similar to imposing frame conditions in modal logics. One possible such constraint has already been mentioned in the previous section: in $\mathcal{ALCF}$ the interpretation of some roles is required to be functional. Here, we will consider transitive roles and role hierarchies. In a DL with *transitive roles*, a subset N_R^+ of the set of all role names N_R is fixed [45]. Elements of N_R^+ must be interpreted by transitive binary relations. (This corresponds to the frame condition for the modal logic K_4.) A *role hierarchy* is given by a finite set of role inclusion axioms of the form $r \sqsubseteq s$ for roles r, s. An interpretation $\mathcal{I}$ satisfies the role hierarchy $\mathcal{H}$ iff $r^\mathcal{I} \subseteq s^\mathcal{I}$ holds for each $r \sqsubseteq s \in \mathcal{H}$.

DLs with transitive roles and role hierarchies have the nice property that reasoning w.r.t. TBoxes can be reduced to reasoning without TBoxes using a technique called internalisation [3, 30, 32]. Like in Section 5.2, we may assume that TBoxes are of the form $\mathcal{T} = \{\widehat{C} \doteq \top\}$. In $\mathcal{SH}$, the extension of $\mathcal{ALC}$ with transitive roles and role hierarchies, we introduce a new *transitive* role name u and assert in the role hierarchy that u is a super-role of all roles occurring in $\widehat{C}$ and the concept description C_0 to be tested for satisfiability. Then, C_0 is satisfiable w.r.t. $\mathcal{T}$ iff $C \sqcap \widehat{C} \sqcap \forall u.\widehat{C}$ is satisfiable. Extending this reduction to inverse roles consists simply in making u also a super-role of the inverse of each role occurring in $\widehat{C}$ or C_0 [32]. This reduction shows that a tableau algorithm for $\mathcal{SH}$ must also employ some sort of *blocking* to ensure termination (see Section 5.2).

Things become even more complex if we consider the DL $\mathcal{SHIF}$, which extends $\mathcal{SH}$ by the inverse of roles and functional roles. In fact, it is easy to show that $\mathcal{SHIF}$ no longer has the finite model property, i.e., there are satisfiable $\mathcal{SHIF}$-concept descriptions that are not satisfiable in a finite interpretation [32]. Instead of directly trying to construct an interpretation that satisfies C_0 (which might be infinite), the tableau algorithm for $\mathcal{SHIF}$ introduced in [32, 33] first tries to construct a so-called *pre-model*, i.e., a structure that can be "unravelled" to a (possibly infinite) canonical (tree) interpretation. To ensure termination (without destroying correctness), the algorithm employs blocking techniques that are more sophisticated than the one described in Section 5.2. Interestingly, an optimised implementation of this algorithm in the system I-FaCT behaves quite well in realistic applications [33]. A refinement of the blocking techniques employed for $\mathcal{SHIF}$ can be used to prove that satisfiability in $\mathcal{SI}$

(i.e., the extension of $\mathcal{ALC}$ by transitive and inverse roles) is in PSPACE [51, 33].

Finally, let us briefly comment on the difference between transitive roles and transitive closure of roles. Transitive closure is more expressive, but it appears that one has to pay dearly for this. In fact, whereas there exist quite efficient implementations for very expressive DLs with transitive roles, inverse roles, and role hierarchies (see above), no such implementations are known (to us) for closely related logics with transitive closure, such as converse-PDL (which is a notational variant of the extension of $\mathcal{ALC}$ by transitive closure, union, composition, and inverse of roles [47]). One reason could be that the known tableau algorithm for converse-PDL [22] requires a "cut" rule, which is massively nondeterministic, and thus very hard to implement efficiently. An other problem with transitive closure is that a blocked individual need no longer indicate "success", as is the case in DLs with transitive roles (see, e.g., the discussion of "good" and "bad" cycles in [1]).

References

1. Franz Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of IJCAI-91*, Sydney, Australia, 1991.
2. Franz Baader, Martin Buchheit, and Bernhard Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.
3. Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, and Gert Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *J. of Logic, Language and Information*, 2:1–18, 1993.
4. Franz Baader and Philipp Hanschke. A schema for integrating concrete domains into concept languages. Technical Report RR-91-10, DFKI, Kaiserslautern, Germany, 1991. An abridged version appeared in *Proc. of IJCAI-91*.
5. Franz Baader and Bernhard Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of PDK'91*, volume 567 of *LNAI*, pages 67–86. Springer-Verlag, 1991.
6. Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, and Enrico Franconi. An empirical analysis of optimization techniques for terminological representation systems. In *Proc. of KR-92*, pages 270–281. Morgan Kaufmann, 1992.
7. Franz Baader and Ulrike Sattler. Expressive number restrictions in description logics. *J. of Logic and Computation*, 9(3):319–350, 1999.
8. Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.
9. Ronald J. Brachman. "Reducing" CLASSIC to practice: Knowledge representation meets reality. In *Proc. of KR-92*, pages 247–258. Morgan Kaufmann, 1992.
10. Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of AAAI-84*, pages 34–37, 1984.
11. Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, 1985.
12. Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

13. Paolo Bresciani, Enrico Franconi, and Sergio Tessaris. Implementing and testing expressive description logics: Preliminary report. *Working Notes of the 1995 Description Logics Workshop*, Technical Report, RAP 07.95, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", pages 131–139, Rome (Italy), 1995.
14. Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.
15. Giuseppe De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms.* PhD thesis, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", 1995.
16. Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of AAAI-94*, pages 205–212. AAAI Press/The MIT Press, 1994.
17. Giuseppe De Giacomo and Maurizio Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In A. G. Cohn, editor, *Proc. of ECAI-94*, pages 411–415. John Wiley & Sons, 1994.
18. Giuseppe De Giacomo and Maurizio Lenzerini. TBox and ABox reasoning in expressive description logics. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Proc. of KR-96*, pages 316–327. Morgan Kaufmann, 1996.
19. Francesco M. Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, and Werner Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.
20. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proc. of KR-91*, pages 151–162. Morgan Kaufmann, 1991.
21. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. Tractable concept languages. In *Proc. of IJCAI-91*, pages 458–463, Sydney, 1991.
22. Giuseppe De Giacomo and Fabio Massacci. Tableaux and algorithms for propositional dynamic logic with converse. In *Proc. of CADE-96*, pages 613–628, 1996.
23. Philipp Hanschke. Specifying role interaction in concept languages. In *Proc. of KR-92*, pages 318–329. Morgan Kaufmann, 1992.
24. Bernhard Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proc. of GWAI'90*, volume 251 of *Informatik-Fachberichte*, pages 38–47. Springer-Verlag, 1990.
25. Bernhard Hollunder. Consistency checking reduced to satisfiability of concepts in terminological systems. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):133–157, 1996.
26. Bernhard Hollunder and Franz Baader. Qualifying number restrictions in concept languages. In *Proc. of KR-91*, pages 335–346, 1991.
27. Bernhard Hollunder and Werner Nutt. Subsumption algorithms for concept languages. Technical Report RR-90-04, DFKI, Kaiserslautern, Germany, 1990.
28. Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. of ECAI-90*, pages 348–353, London, 1990. Pitman.
29. Ian Horrocks. The FaCT system. In Harrie de Swart, editor, *Proc. of TABLEAUX-98*, volume 1397 of *LNAI*, pages 307–312. Springer-Verlag, 1998.
30. Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR-98*, pages 636–647, 1998.
31. Ian Horrocks and Peter F. Patel-Schneider. Optimizing description logic subsumption. *J. of Logic and Computation*, 9(3):267–293, 1999.
32. Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.

33. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, LNAI. Springer-Verlag, 1999.
34. Carsten Lutz. Complexity of terminological reasoning revisited. In *Proc. of LPAR'99*, volume 1705 of *LNAI*. Springer-Verlag, 1999.
35. Robert MacGregor. The evolving technology of classification-based knowledge representation systems. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, 1991.
36. Eric Mays, Robert Dionne, and Robert Weida. K-REP system overview. *SIGART Bulletin*, 2(3), 1991.
37. Marvin Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design*. The MIT Press, 1981. Republished in [11].
38. Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *LNAI*. Springer-Verlag, 1990.
39. Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.
40. Bernhard Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, 1991.
41. Peter F. Patel-Schneider. DLP. In *Proc. of DL'99*, pages 9–13. CEUR Electronic Workshop Proceedings, 1999. `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-22/`.
42. Peter F. Patel-Schneider, Deborah L. McGuiness, Ronald J. Brachman, Lori Alperin Resnick, and Alexander Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bulletin*, 2(3):108–113, 1991.
43. Christof Peltason. The BACK system – an overview. *SIGART Bulletin*, 2(3):114–119, 1991.
44. M. Ross Quillian. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, 12:410–430, 1967. Republished in [11].
45. Ulrike Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, editors, *Proc. of KI'96*, volume 1137 of *LNAI*. Springer-Verlag, 1996.
46. Walter J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *J. of Computer and System Science*, 4:177–192, 1970.
47. Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, Sydney, Australia, 1991.
48. Klaus Schild. Terminological cycles and the propositional μ-calculus. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of KR-94*, pages 509–520, Bonn, 1994. Morgan Kaufmann.
49. Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proc. of KR-89*, pages 421–431. Morgan Kaufmann, 1989.
50. Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
51. Edith Spaan. The complexity of propositional tense logics. In Maarten de Rijke, editor, *Diamonds and Defaults*, pages 287–307. Kluwer Academic Publishers, 1993.
52. Stephan Tobies. A PSPACE algorithm for graded modal logic. In *Proc. of CADE-99*, volume 1632 of *LNCS*. Springer-Verlag, 1999.
53. Wiebe Van der Hoek and Maarten De Rijke. Counting objects. *J. of Logic and Computation*, 5(3):325–345, 1995.