An Overview of Tableau Algorithms for Description Logics *

Franz Baader (baader@cs.rwth-aachen.de) and Ulrike Sattler
(sattler@cs.rwth-aachen.de)
LuFG Theoretical Computer Science, RWTH Aachen, Germany

May 19, 2000

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.

Keywords: Description Logics, 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 (Quillian, 1967; Minsky, 1981), 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 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

* This is an extended version of a paper published in the proceedings of Tableaux 2000 (Baader and Sattler, 2000).

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, the research related to this issue can be classified into the following four phases.

Phase 1: First system implementations. The original KL-ONE system (Brachman and Schmolze, 1985) as well as its early successor systems (such as BACK (Peltason, 1991), K-REP (Mays et al., 1991), and LOOM (MacGregor, 1991)) 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., (Nebel, 1990a) 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 (Brachman and Levesque, 1984; Nebel, 1990b), and that the DL used by the KL-ONE system even has an undecidable subsumption problem (Schmidt-Schauß, 1989). 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 a DL system. The designers of the CLASSIC system (Patel-Schneider et al., 1991; Brachman, 1992) followed another approach: they carefully chose a restricted DL that still allowed for an (almost[1]) complete polynomial structural subsumption algorithm (Borgida and Patel-Schneider, 1994).

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

[1] The incompleteness is caused by individuals introduced by the one-of constructor; however, the algorithm is complete w.r.t. a non-standard semantics.

useful: they are complete and often of optimal (worst-case) complexity. The first such algorithm was proposed by Schmidt-Schauß and Smolka (1991) for a DL that they called $\mathcal{ALC}$ (for "attributive concept description language with complements").[2] It quickly turned out that this approach for deciding subsumption can be extended to various other DLs (Hollunder et al., 1990; Hollunder and Baader, 1991; Baader and Hanschke, 1991; Baader, 1991; Hanschke, 1992) and also to other inference problems such as the instance problem (Hollunder, 1990). 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 DLs, could be seen as specialisations of the tableau 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 (1991) showed that $\mathcal{ALC}$ is 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 various DLs (in particular also DLs that are not propositionally closed) was investigated in detail (Donini et al., 1991a; Donini et al., 1991b; Donini et al., 1992).

The first DL systems employing tableau algorithms (KRIS (Baader and Hollunder, 1991) and CRACK (Bresciani et al., 1995)) demonstrated that (in spite of their high worst-case complexity) these algorithms lead to acceptable behaviour in practice (Baader et al., 1994). Highly optimised systems such as FaCT (Horrocks, 1998b), DLP (Patel-Schneider, 1999), and Race (Haarslev and Möller, 1999) have an even better behaviour, also for benchmark problems in modal logics (Horrocks, 1998a; Horrocks and Patel-Schneider, 1999; Haarslev and Möller, 2000a; Horrocks, 2000; Patel-Schneider, 2000).

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 (Schild, 1991). The idea of this approach, which was perfected by De Giacomo 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 (De Giacomo and Lenzerini,

[2] 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.

1994; De Giacomo, 1995; De Giacomo and Lenzerini, 1996), it turned out to be less satisfactory from a practical point of view. In fact, first tests in a database application (Horrocks et al., 1999) showed that the PDL formulae obtained by the translation technique could not be handled by existing efficient implementations of satisfiability algorithms for PDL (Patel-Schneider, 1999). To overcome this problem, DL researchers have started to design "practical" tableau algorithms for *very* expressive DLs (Horrocks and Sattler, 1999; Horrocks et al., 1999).[3]

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 sections 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.

An overview of reasoning techniques in description logics with more emphasis on complexity results and on results for less expressive DLs can be found in (Donini et al., 1996). Reasoning in very expressive DLs with an emphasis on results obtained via the translation approach is treated in (Calvanese et al., 2001).

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* 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 the next two sections, we consider concept descriptions built from the constructors shown in Table I, where C, D stand for concept descriptions, r for a role name, and n for a nonnegative 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*.

[3] In contrast to PDL, these DLs allow for transitive roles, but not for the transitive closure operator.

Table I. Syntax and semantics of concept descriptions.

Constructor	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	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
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	$(\leqslant nr.C)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$

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 I.

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 K 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 (1991), 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 (Van der Hoek and De Rijke, 1995), which are, however, not as well-investigated as the modal logic K.

One of the most important inference services of 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 (written $C \sqsubseteq D$) 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.[4] Vice versa, satisfiability can be reduced to subsumption: C is satisfiable iff not $C \sqsubseteq P \sqcap \neg P$, where P is an arbitrary concept name.

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 (1991), 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

[4] This was the reason why Schmidt-Schauß and Smolka (1991) introduced a DL with negation in the first place.

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.

The $\rightarrow_\sqcap$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but not 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: $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$.

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

Let C_0 be 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 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 be 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 sub-description 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 (Baader and Hanschke, 1991).

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

1. The domain $\Delta^{\mathcal{I}_\mathcal{A}}$ of $\mathcal{I}_\mathcal{A}$ consists of 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.*

3.1. 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 (Savitch, 1970), 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 (Schmidt-Schauß and Smolka, 1991; Halpern and Moses, 1992).

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

3.2. 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 $\mathcal{A}_0$ 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 (Hollunder, 1996) for details).

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

Since $\mathcal{ALC}$ is closed under negation, this also implies that the instance problem is PSPACE-complete in $\mathcal{ALC}$. The consistency and the instance problem for DLs not allowing for negation has been investigated in (Schaerf, 1993; Donini et al., 1994).

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 $\mathcal{ALCN}$ (Hollunder et al., 1990; Donini et al., 1991a).

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 $x \neq y$ for individual names x, y, with the obvious semantics that an interpretation $\mathcal{I}$ satisfies $x \neq y$ iff $x^{\mathcal{I}} \neq y^{\mathcal{I}}$. These assertions are assumed to be symmetric, i.e., saying that $x \neq y$ belongs to an ABox $\mathcal{A}$ is the same as saying that $y \neq x$ 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*:

– $\{(\leqslant nr)(x)\} \cup \{r(x, y_i) \mid 1 \leq i \leq n+1\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n+1\} \subseteq \mathcal{A}$ 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. (Baader and Sattler, 1999), where this is proved for a more expressive DL). This shows that satisfiability (and thus also subsumption) of $\mathcal{ALCN}$-concept descriptions is decidable.

4.1.1. *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 (called unary notation in the following). Here, the size

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 $z_i \not\doteq z_j$ $(1 \leq i < j \leq n)$ are in $\mathcal{A}$.

Action: $\mathcal{A}' := \mathcal{A} \cup \{r(x, y_i) \mid 1 \leq i \leq n\} \cup \{y_i \not\doteq y_j \mid 1 \leq i < j \leq n\}$, 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 $y_i \not\doteq y_j$ is not in $\mathcal{A}$ for some $i, j, 1 \leq i < j \leq n+1$.

Action: For each pair y_i, y_j such that $1 \leq i < j \leq n+1$ and $y_i \not\doteq y_j$ 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.

Figure 2. The transformation rules handling unqualified number restrictions.

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, 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 not written in unary 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 be 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, even if numbers are not represented in unary notation.*

4.1.2. *The consistency problem for $\mathcal{ALCN}$-ABoxes*

Just as with $\mathcal{ALC}$, 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

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

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 $\mathcal{ALCN}$ (Hollunder, 1996).

THEOREM 5. *Consistency of $\mathcal{ALCN}$-ABoxes is PSPACE-complete, even if numbers are not represented in unary notation.*

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 (Van der Hoek and De Rijke, 1995)):

- 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 of $(\geqslant nr.C)$ by adding the assertions $C(y_i)$ to $\mathcal{A}'$.
- The $\rightarrow_\leq$-rule only applies to $(\geqslant nr.C)$ if $\mathcal{A}$ also contains the assertions $C(y_i)$ $(1 \leq i \leq n+1)$.

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: $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{\neg C(y)\}$.

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

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 individual a and concept description 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 (Hollunder and Baader, 1991). 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 $\mathcal{ALCQ}$ (Hollunder and Baader, 1991).

4.2.1. *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 unary notation. For $\mathcal{ALCQ}$, the idea that leads to a PSPACE-algorithm for $\mathcal{ALCN}$ with non-unary 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. non-unary notation of numbers (Tobies, 1999). 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 (Hollunder, 1996) 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, even if numbers are not represented in unary notation.*

5. Terminological axioms

DL 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 $\mathcal{T}$ *(written* $C \sqsubseteq_{\mathcal{T}} D$*) 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) (see, e.g., (Buchheit et al., 1993)).

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 (1990b)) 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 (1990b) 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 (Lutz, 1999). 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 to use 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 defined name and C its defining concept, then it adds the assertion $C(x)$. However, it does not further unfold C at this stage. It can be shown that this really yields a PSPACE-algorithm for satisfiability (and thus also for subsumption) of concepts w.r.t. acyclic terminologies in $\mathcal{ALC}$ (Lutz, 1999).

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 $\mathcal{ALC}$ with functional roles as well as agreements and disagreements on chains of functional roles.

More precisely, in $\mathcal{ALCF}$, a set $N_F \subseteq N_R$ of *feature names* is fixed, and a *feature chain* $u = f_1 \cdots f_n$ is defined to be a non-empty sequence of feature names $f_i \in N_F$. An interpretation $\mathcal{I}$ maps each $f \in N_F$ to a functional role $f^{\mathcal{I}}$, i.e., $(x, y), (x, z) \in f^{\mathcal{I}}$ implies $y = z$. The interpretation of a feature name can thus also be viewed as a partial function $f^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{I}}$. For this reason, we will usually write $f^{\mathcal{I}}(x) = y$ instead of $(x, y) \in f^{\mathcal{I}}$. The feature chain $u = f_1 \cdots f_n$ is interpreted as the composition of its features, i.e., $u^{\mathcal{I}}(x) := f_n^{\mathcal{I}}(\cdots f_1^{\mathcal{I}}(x) \cdots)$.

The DL $\mathcal{ALCF}$ is obtained from $\mathcal{ALC}$ by allowing for feature names in value and existential restrictions, and for additional concept descriptions of the form $u \downarrow v$ (agreement) and $u \uparrow v$ (disagreement), where u, v are feature chains. These new descriptions are interpreted as follows:

$$
\begin{aligned}
(u \downarrow v)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \text{there is a } y \in \Delta^{\mathcal{I}} \text{ with } u^{\mathcal{I}}(x) = y = v^{\mathcal{I}}(x)\} \\
(u \uparrow v)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \text{there are } y_1, y_2 \in \Delta^{\mathcal{I}} \text{ with } y_1 \neq y_2, \\
&\qquad u^{\mathcal{I}}(x) = y_1 \text{ and } v^{\mathcal{I}}(x) = y_2\}
\end{aligned}
$$

The tableau-based satisfiability algorithm for $\mathcal{ALC}$ can easily be extended to $\mathcal{ALCF}$ (Hollunder and Nutt, 1990). Both agreements and disagreements are handled by rules that generate the feature successors required by the semantics. To ensure that features are interpreted as functional roles in the canonical interpretation, one uses an identification rule (similar to the $\rightarrow_{\leq}$-rule): if $f(x, y), f(x, z)$ occurs in the ABox, then the rule replaces every occurrence of y by z, unless the ABox also contains an inequality assertion $y \not\doteq z$. This second case leads to a new type of clashes. Inequality assertions are introduced by the rule that handles disagreements: the final individuals reached by the feature chains are explicitly asserted to be distinct.

It can easily be seen that this algorithm can again be realised within polynomial space. There is, however, a significant difference between the PSPACE-algorithm for $\mathcal{ALC}$ and the one for $\mathcal{ALCF}$. Due to identifications caused by agreements, the canonical interpretation built by the algorithm need no longer have tree shape. For example, an application of the agreement rule to $(f_1 f_2 \downarrow g_1 g_2)(x)$ leads to assertions $g_1(x, y_1), g_2(y_1, z), f_1(x, y_2), f_2(y_2, z)$. In particular, this means that the successors y_1 and y_2 of x cannot be handled independently since they lead to a common successor. However, this problem is restricted to individuals connected by feature chains. It is easy to show that each such *feature-connected component* is polynomial in the size of the concept description to be tested for satisfiability (if identification of feature successors is done eagerly). Thus, it is unproblematic to generate the whole feature-connected component issuing from a given individual.

In the presence of acyclic terminologies, this is no longer true. In fact, by using a sequence of terminological axioms of the form $C_{n+1} \doteq \exists f.C_n \sqcap \exists g.C_n$, one can enforce feature-connected components of size exponential in the size of the given terminology and concept description. In (Lutz, 1999), this fact is used to show that satisfiability of $\mathcal{ALCF}$-concept descriptions w.r.t. acyclic terminologies is NEXPTIME-complete.

THEOREM 8. *Satisfiability of $\mathcal{ALCF}$-concept descriptions is PSPACE-complete, but satisfiability w.r.t. acyclic terminologies is NEXPTIME-complete in $\mathcal{ALCF}$.*

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 axioms $C_1 \doteq D_1, \ldots, C_n \doteq D_n$, 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 $\mathcal{A}_0 := \{(\exists r.P)(x_0)\}$ w.r.t. 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 algorithm has run into a cycle.

Termination can be regained by using a mechanism that detects 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 $\exists r.P \doteq \top$. Since the set of concepts asserted for the blocked individual is a subset of the set of those asserted for the blocking individual, we call this blocking condition *subset blocking*.

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 (Buchheit et al., 1993; Baader et al., 1996) for details).

In the algorithm we have just described, we do not impose any order or strategy on the application of the transformation rules. This leads to what is called *dynamic blocking* (Horrocks and Sattler, 1999), where blocks can be established and then broken. For example, suppose an individual x is blocked by an individual y. Then, the application of the $\rightarrow_\forall$-rule to x's predecessor may add $C(x)$ to $\mathcal{A}$. If $C(y)$ is not present in $\mathcal{A}$, then x is no longer blocked by y. However, using a strategy that (basically) applies generating rules only if no non-generating ones can be applied, blocks that are established once will never be broken again. Thus, when employing this strategy, we can block *statically*. Note that implementations of tableau-based algorithms usually employ this strategy anyway.

It should be noted that the algorithm we have described above is no longer in PSPACE since it may generate role paths of exponential length before

Table II. Syntax and semantics of role constructors and restrictions.

Constructor/Restriction	Syntax	Semantics
intersection	$r \sqcap s$	$(r \sqcap s)^{\mathcal{I}} = r^{\mathcal{I}} \cap s^{\mathcal{I}}$
union	$r \sqcap s$	$(r \sqcup s)^{\mathcal{I}} = r^{\mathcal{I}} \cup s^{\mathcal{I}}$
complement	$\neg r$	$(\neg r)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus r^{\mathcal{I}}$
composition	$r \circ s$	$(r \circ s)^{\mathcal{I}} = \{(x,z) \mid$ there is a y such that $(x,y) \in r^{\mathcal{I}}$ and $(y,z) \in s^{\mathcal{I}}\}$
transitive closure	R^+	$(R^+)^{\mathcal{I}} = (R^{\mathcal{I}})^+$
inverse	R^-	$(R^-)^{\mathcal{I}} = \{(y,x) \mid (x,y) \in R^{\mathcal{I}}\}$
transitive roles	$R \in N_R^+$	$R^{\mathcal{I}}$ is transitive
role hierarchy	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

blocking occurs. In fact, even for the language $\mathcal{ALC}$, satisfiability w.r.t. general terminological axioms is known to be EXPTIME-complete (Schild, 1994). The tableau-based algorithm sketched above is a NEXPTIME algorithm. However, using the translation technique mentioned in the introduction, it can be shown (De Giacomo and Lenzerini, 1994) that $\mathcal{ALCN}$-ABoxes and TBoxes can be translated into PDL.

THEOREM 9. *Consistency of $\mathcal{ALCN}$-ABoxes w.r.t. TBoxes is EXPTIME-complete.*

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 (Nebel, 1991; Baader et al., 1993).

6. Expressive roles

The DLs considered so far allowed for atomic roles only. There are two ways of extending the expressivity of DLs w.r.t. roles: adding role constructors and constraining the interpretation of roles. An overview of the syntax and semantics of both are given in Table II, where the first part refers to role constructors and the second to role constraints. *Role constructors* can be used to build complex roles from atomic ones. In the following, we will mostly restrict our attention to the inverse constructor, which makes it possible to "use a role in both directions". For example, using inverse roles, we can describe both parents of nice children by $\forall \mathsf{has\_child}.\mathsf{Nice}$ as well as children of nice parents by $\forall \mathsf{has\_child}^-.\mathsf{Nice}$. The other role constructors have also been considered in the literature (e.g., Boolean operators in (De Giacomo, 1995; Lutz and Sattler, 2000), and composition, union, and transitive closure in (Baader, 1991; Schild, 1991)).

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 roles $f \in N_F \subseteq N_R$ 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 (Sattler, 1996). Elements of N_R^+ must be interpreted by transitive binary relations. (This corresponds to the frame condition for the modal logic $\mathsf{K}4$.) 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 $r \sqsubseteq s \in \mathcal{H}$. For example, we can use the role inclusion axiom $\mathsf{has\_son} \sqsubseteq \mathsf{has\_child}$ to express that every son of a person is also her child.

6.1. Expressive roles in number restrictions

In DLs with expressive roles and number restrictions, the roles that are allowed to occur in the number restrictions are usually of a restricted form (see, e.g., (De Giacomo and Lenzerini, 1994; De Giacomo and Lenzerini, 1996; Horrocks et al., 1999; Haarslev and Möller, 2000b)). Whereas tableau-based algorithms that respectively handle number restrictions on conjunctions of roles (Donini et al., 1991a), on compositions of roles (Baader and Sattler, 1999), on inverse roles (see Section 6.2.3), and on roles occurring in a role hierarchy (Horrocks and Sattler, 1999; Haarslev and Möller, 2000b) are known from the literature, other role constructors and restrictions appear to be more problematic when used within number restrictions.

Let us illustrate this with two examples. First, transitive closure of roles, transitive roles, or roles having a transitive sub-role (with respect to a role hierarchy) are usually not allowed inside number restrictions. In fact, a tableau-based algorithm for a DL containing such number restrictions would need to differ significantly from the algorithms we have described until now. Intuitively, this is due to the fact that transitivity (in one of the forms mentioned above) can yield situations where, for a transitive role r, a long r-path starting at an individual x would need to be collapsed into a single r-successor of x, due to the presence of an assertion $(\leq 1r)(x)$. This destroys the tree shape of the canonical interpretation to be generated, which (for example) means that the usual arguments for showing termination can no longer be applied. At least in the case where roles having transitive sub-roles are allowed to occur in number restrictions, these problems cannot be overcome: the extension of $\mathcal{ALCN}$ that allows roles having transitive sub-roles to occur in number restrictions has an undecidable subsumption problem (Horrocks et al., 1999).

Second, the combination of role composition with Boolean role constructors and inverse roles in number restrictions usually causes undecidability. In (Baader and Sattler, 1999), the tableau-based algorithm for $\mathcal{ALCN}$ is first extended to composition of roles in number restrictions, and then to union and intersection of role compositions of the *same length*. It is also shown that most of the other combinations lead to an undecidable DL.

6.2. ROLE HIERARCHIES, INVERSE ROLES, AND TRANSITIVE ROLES

Before considering different extensions of $\mathcal{ALC}$ and $\mathcal{ALCN}$ by these role constructors, a general remark is in order. For most of the DLs considered in this subsection, satisfiability and subsumption of concept descriptions are EXPTIME-complete problems. The reason for these DLs to be EXPTIME-hard is that they can simulate general TBoxes within concept descriptions (see below). The fact that they are in EXPTIME follows from results for PDL and converse-PDL (Pratt, 1979; Harel, 1984). The tableau-based algorithms that will be sketched below are NEXPTIME-algorithms. The point in designing these algorithms was not to prove worst-case complexity results, but rather to obtain "practical" algorithms, i.e., algorithms that are easy to implement and optimise, and which behave well on realistic knowledge bases. Nevertheless, the fact that "natural" tableau algorithms for such EXPTIME-complete logics are usually NEXPTIME-algorithms is an unpleasant phenomenon. In contrast, automata-based algorithms (Vardi and Wolper, 1986) often yield optimal worst-case complexity results, but do not behave well in practice (since they are also best-case exponential). Attempts to design EXPTIME-tableaux for such logics (De Giacomo et al., 1996; De Giacomo and Massacci, 1996; Donini and Massacci, 1999) usually lead to rather complicated (and thus not easy to implement) algorithms, which (to the best of our knowledge) have not been implemented yet.

6.2.1. *DLs with transitive roles and role hierarchies*

In the DL $\mathcal{SH}$, i.e., the extension of $\mathcal{ALC}$ with transitive roles and role hierarchies, reasoning w.r.t. (general) TBoxes can be reduced to reasoning without TBoxes using a standard technique from modal logics, which is called *internalisation* in the DL literature (Schild, 1991; Baader et al., 1993). As mentioned in Section 5.2, we may assume that TBoxes consist of a single axiom of the form $\widehat{C} \doteq \top$. Internalisation of this axiom introduces a new *transitive* role u, and asserts 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. $\{\widehat{C} \doteq \top\}$ iff $C_0 \sqcap \widehat{C} \sqcap \forall u.\widehat{C}$ is satisfiable with respect to the role hierarchy.

With respect to expressive power, this is a nice property of $\mathcal{SH}$. However, it also shows that satisfiability and subsumption of concept descriptions in $\mathcal{SH}$ is EXPTIME-hard.[5] The tableau algorithm for $\mathcal{SH}$ presented in (Horrocks, 1998b) handles role hierarchies by an appropriate definition of

[5] More precisely, reasoning in $\mathcal{SH}$ is EXPTIME-complete (Horrocks et al., 2000a).

r-successors: an individual y is called an *r-successor* of an individual x in an ABox $\mathcal{A}$ iff $s(x, y) \in \mathcal{A}$ for some sub-role s of r. Then, the condition $r(x, y) \in \mathcal{A}$ in the $\rightarrow_\exists$- and the $\rightarrow_\forall$-rule is replaced by the condition "if y is an r-successor of x in $\mathcal{A}$". Transitive roles are taken care of by a new rule, the $\rightarrow_\forall^+$-rule, which, basically, adds $(\forall r.C)(y)$ to $\mathcal{A}$ iff y is an r-successor of x such that $(\forall s.C)(x) \in \mathcal{A}$ and r is a transitive sub-role of s. (Note that this corresponds to the treatment of $\mathsf{K4}$-modalities in tableau algorithms from modal logics (Halpern and Moses, 1992).) Obviously, this shifting of value restrictions along transitive roles makes for a non-terminating algorithm, unless one employs an appropriate blocking technique. The blocking strategy used in (Horrocks, 1998b) coincides with the one we have presented in Section 5.2, i.e., subset-blocking.

6.2.2. *DLs with transitive and inverse roles, and role hierarchies*

The extension of $\mathcal{SH}$ with inverse roles is called $\mathcal{SHI}$. In this DL, TBoxes can be internalised in a way similar to the one we have described for $\mathcal{SH}$. The only difference is that now u is not only specified as a (transitive) super-role of all roles occurring in the input concept and the TBox, but also of the inverses of these roles (Horrocks and Sattler, 1999).

In (Horrocks and Sattler, 1999), a tableau algorithm for $\mathcal{SHI}$ is obtained from the one for $\mathcal{SH}$ sketched above by extending the notion of r-successors to r-neighbours, and modifying the transformation rules accordingly. Modulo some technical details, an individual y is called an r-neighbour of an individual x in $\mathcal{A}$ iff $s(x, y) \in \mathcal{A}$ or $s^-(y, x) \in \mathcal{A}$ for some sub-role s of r. Obviously, using r-neighbours instead of r-successors in the new $\rightarrow_\forall$-rule implies that the rule can now be applied in both direction. For example, if $r^-(x, y), (\forall r.C)(y) \in \mathcal{A}$, then the rule adds $C(x)$. The main technical problem is to find an appropriate blocking condition, i.e., a condition that still ensures termination, but does not compromise correctness of the algorithm. The blocking strategy introduced in (Horrocks and Sattler, 1999) differs in two points from blocking for $\mathcal{SH}$.

First, one can no longer use subset blocking as described in Section 5.2. Consider the example shown in Fig. 4 (where, for the sake of legibility, not all concepts necessary for generating this situation are explicitly given). If subset blocking is used, then y is blocked by x. However, when building the canonical interpretation $\mathcal{I}$, the r-successor x_1 of x is used to satisfy $(\exists r.A)(y)$, i.e., $(y, x_1) \in r^\mathcal{I}$. This violates the value restriction for x_1, which shows that the interpretation obtained this way is not a model of the complete and open ABox. This problem can be overcome by using *equality* blocking, i.e., an individual y is blocked by an individual x iff $\{D \mid D(x) \in \mathcal{A}\} = \{D' \mid D'(y) \in \mathcal{A}\}$.

Second, blocking is now dynamic, even if rules are applied according to the strategy that applies non-generating rules with higher priority. This is

$$\begin{tikzcd}
x \circ \; (\forall s^{-}.A)(x),\ (\exists r.A)(x), \ldots \arrow[d, "r"] \\
x_1 \circ \; A(x_1),\ (\forall r^{-}.\forall s^{-}.A)(x_1), \ldots \arrow[d, "r"] \\
x_2 \circ \; (\neg A)(x_2), \ldots \arrow[d, "s"] \\
y \circ \; (\exists r.A)(y)
\end{tikzcd}$$

Figure 4. A situation where subset blocking fails for $\mathcal{SHI}$.

due to the fact that the $\rightarrow_{\forall}$-rule can be applied back and forth on a chain of individuals.

Alternatively to the approach described until now, which goes back and forth in the interpretation to be generated, one could have chosen to guess (nondeterministically) all those assertions $C(x)$ that could be propagated "back" from an r-successor y of x due to value restrictions $(\forall r^{-}.C)(y)$. In the case of a wrong guess, one has a new type of clashes. The *analytic cut* rule in (De Giacomo and Massacci, 1996) does this for a well-chosen, relatively small set of sub-descriptions of the input description. In this setting, blocking would again become static. However, in an actual implementation it is preferable to avoid this "blind" guessing. For $\mathcal{SHI}$ (and its extensions treated in the following subsections), avoiding this source of nondeterminism is indeed possible. This does not appear to be the case for the extension of $\mathcal{ALC}$ with transitive closure and inverse of roles. This DL is closely related to converse-PDL, for which a tableau algorithm is presented in (De Giacomo and Massacci, 1996) using the analytic cut rule. (In Section 6.2.4, we will comment in more detail on tableau algorithms for DLs with transitive closure of roles.)

By dropping role hierarchies from $\mathcal{SHI}$, we obtain the logic $\mathcal{SI}$. Obviously, the internalisation of TBoxes sketched above does no longer work since we cannot specify super-roles of roles. It can be shown that $\mathcal{SI}$ is indeed less complex than $\mathcal{SH}$ or $\mathcal{SHI}$. Using a rather sophisticated blocking technique, a tableau algorithm can be designed that decides satisfiability of concept descriptions in $\mathcal{SI}$ using space polynomial in the size of the input description (Spaan, 1993; Horrocks et al., 1999). This implies that satisfiability of concept descriptions in $\mathcal{SI}$ is PSPACE-complete, i.e., of the same worst-case complexity as $\mathcal{ALC}$.

6.2.3. *DLs with transitive and inverse roles, role hierarchies, and number restrictions*

Things become even more complicated for the DL $\mathcal{SHIN}$, which extends $\mathcal{SHI}$ with unqualified number restrictions on simple roles. A role r is called *simple* iff r is an atomic role or its inverse such that r does not have a transitive sub-role (Horrocks and Sattler, 1999).

In contrast to $\mathcal{SHI}$, the DL $\mathcal{SHIN}$ no longer has the finite model property. For example, if the role hierarchy contains the axiom $s \sqsubseteq r$ for a transitive role $r \in N_R^+$, then the following concept is obviously satisfiable, but each of its models has an infinite s-path: $\neg A \sqcap \exists s.A \sqcap \forall r.(\exists s.A \sqcap (\leqslant 1 s^-))$.

Thus, instead of directly trying to construct a (possibly infinite) interpretation that satisfies C_0, the tableau algorithm for $\mathcal{SHIN}$ introduced in (Horrocks and Sattler, 1999; Horrocks et al., 1999) first tries to construct a so-called *pre-model*, i.e., a structure that can be "unravelled" to a (possibly infinite) canonical (tree) interpretation. In principle, this algorithm is obtained by extending the algorithm for $\mathcal{SHI}$ with the rules that handle number restrictions. The main technical problem to be solved is again to design the appropriate blocking condition.

Unravelling is also known in modal logic (see, for example, (Stirling, 1992)), and works as follows. To construct a model from a pre-model, we define elements of the model's domain to be *paths* in the pre-model that follow edges $r(x, y)$ where, instead of going to a blocked individual, the path goes to its blocking individual. Thus, if blocking occurs, we may obtain an infinite model (e.g., if the blocking individual is a predecessor of the blocked individual)—even though the input concept might have a finite one.

Before describing the blocking condition introduced in (Horrocks and Sattler, 1999; Horrocks et al., 1999), let us point out a new phenomenon that can occur when running the tableau algorithm for $\mathcal{SHIN}$. Due to the interaction of role hierarchies and number restrictions, the algorithm can generate an ABox $\mathcal{A}$ with $\{r(x, y), s(x, y)\} \subseteq \mathcal{A}$ where r, s are not sub-roles of each other. This situation can be caused by an assertion $(\leqslant 1 t)(x)$, where t is a common super-role of r and s, and x already has an r- and an s-successor. These two successors are then merged into the single successor y. Note, however, that each individual generated by the algorithm still has a unique predecessor, though it may be related with more than one role to this predecessor.

The new blocking condition for $\mathcal{SHIN}$ is called *pair-wise* blocking. It extends the one for $\mathcal{SHI}$ as follows. In order for an individual y to be blocked by an individual x, the predecessors of x and y must also have identical assertion attached to them, and x and y must be related by the same roles to their respective predecessors. More precisely, assume that x, y are individuals in $\mathcal{A}$ that respectively have the predecessors x', y' in $\mathcal{A}$. For y to be blocked by x, the following conditions must be satisfied: (i) for each role r, x is an

$$\begin{tikzcd}[row sep=small]
x \arrow[d, "s"] & C_0(x),\ (\neg A)(x),\ (\exists s.D)(x),\ (\forall r.\exists s.D)(x) \\
y \arrow[d, "s"] & D(y),\ A(y),\ (\leq 1\ s^-)(y),\ (\exists s^-.\neg A)(y),\ (\exists s.D)(y),\ (\forall r.\exists s.D)(y) \\
z & D(z),\ A(z),\ (\leq 1\ s^-)(z),\ (\exists s^-.\neg A)(z),\ (\exists s.D)(z),\ (\forall r.\exists s.D)(z)
\end{tikzcd}$$

Figure 5. A situation where pair-wise blocking is crucial.

r-successor of x' iff y is an r-successor of y'; (ii) $\{D \mid D(x) \in \mathcal{A}\} = \{D' \mid D'(y) \in \mathcal{A}\}$; and (iii) $\{D \mid D(x') \in \mathcal{A}\} = \{D' \mid D'(y') \in \mathcal{A}\}$. The following example should give a better intuition for why this complex blocking condition is needed. In Fig. 5, we show relevant parts of an ABox that was generated to decide the satisfiability of the concept C_0, where

$$C_0 := \neg A \sqcap (\exists s.D) \sqcap (\forall r.\exists s.D),$$

s is a sub-role of the transitive role r, and $D := A \sqcap (\leqslant 1 s^-) \sqcap (\exists s^-.\neg A)$. Using equality blocking, z would be blocked by y. When constructing the canonical interpretation, we cannot re-use y's s-successor as z's successor: this would make z an s-successor of itself, and thus z would have itself and y as s^--successors, contradicting the assertion $(\leqslant 1 s^-)(z)$. Thus, unravelling is really necessary in this example. As explained above, unravelling the ABox to an interpretation would generate as elements of the interpretation the path $[x]$ (corresponding to x), the path $[x, y]$ (corresponding to y), the path $[x, y, y]$ (which is used instead of the blocked individual z), the path $[x, y, y, y]$ etc. However, in this interpretation the element $[x, y, y]$ and its successors do not belong to the concept description $\exists s^-.\neg A$, which shows that this interpretation does not satisfy C_0. With respect to pair-wise blocking, z is not blocked by y since the predecessor x of y has a concept assertion $(\neg A)(x)$ that the predecessor y of z does not have. Hence the tableau algorithm generates an s-successor to satisfy $(\exists s.D)(z)$ and an s^--successor to satisfy $(\exists s^-.\neg A)(z)$.

It should be noted that the problems that lead to the need for pair-wise blocking do not depend on the presence of "large" numbers in number restrictions. In fact, the above example used only *functional restrictions*, i.e., number restrictions of the form $(\leq 1\ r)$.

The tableau-based satisfiability algorithm for $\mathcal{SHIN}$ described until now can also be extended to decide the consistency of ABoxes (Horrocks et al., 2000b). Recall that, for $\mathcal{ALCN}$, the naive extension of the satisfiability algorithm to a consistency algorithm ran into termination problems. This problem can be overcome by applying the pre-completion technique, which reduces ABox consistency to satisfiability of concept descriptions (see Section 4.1.2). Pre-completion does not work for $\mathcal{SHIN}$ due to the presence of inverse roles. For example, the inconsistency of the ABox $\{r(x, y), A(x)$

$(\exists s.\forall s^-.\forall r^-.\neg A)(y)\}$ cannot be detected if, after the application of non-generating rules only, x and y are treated in unrelated ABoxes. However, the termination problem pointed out for $\mathcal{ALCN}$ is not relevant for $\mathcal{SHIN}$ since the algorithm employs blocking to ensure termination. Basically, the only difference between the satisfiability and the consistency algorithm for $\mathcal{SHIN}$ is that one must be a bit more careful when the block involves old individuals, i.e., individuals present in the input ABox.

6.2.4. *DLs with the transitive closure of roles*

Finally, let us briefly comment on the difference between transitive roles and transitive closure of roles. The transitive closure of roles 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 (Schild, 1991)). One reason could be that the known tableau algorithm for converse-PDL (De Giacomo and Massacci, 1996) requires an analytic cut rule (see Section 6.2.2), which is massively nondeterministic, and thus very hard to implement efficiently.

Another problem with transitive closure is that a blocked individual need no longer indicate "success", as is the case in DLs with transitive roles. In the presence of transitive closure, when blocking occurs, one must check whether this block is due to a harmless, cyclic repetition of the same assertions (as is always the case for $\mathcal{SHIN}$), or whether the block is caused by the repeated unsuccessful attempt to satisfy an assertion of the form $(\exists r^+.C)(x)$, where C is unsatisfiable or in conflict with an assertion $(\forall r^+.D)(x)$. The former case is called a "good" cycle and the latter a "bad" cycle in (Baader, 1991). To satisfy an assertion of the form $(\exists r^+.C)(x)$ (often called "eventuality" in the modal or temporal logic literature), one has two possibilities: (i) satisfy it now, i.e., generate an r-successor that belongs to C; or (ii) defer it till later on, i.e., generate an r-successor that belongs to $\exists r^+.C$. However, one must ensure that the $(\exists r^+.C)(x)$ is satisfied eventually, i.e., one does not always choose the second alternative. To ensure termination, the algorithm in (Baader, 1991) basically uses equality blocking, together with a rather strict strategy on the application of rules. A block (called cycle in (Baader, 1991)) can now indicate two things: either it is good, which corresponds to the situation encountered in logics like $\mathcal{SHIN}$, or it is bad, which corresponds to infinitely deferring to satisfy an eventuality. Since good cycles can be distinguished from bad cycles, the algorithm can stop with success in the first case, and it must backtrack in the second. Note that the algorithm in (Baader, 1991) is very similar to the satisfiability algorithm for DPDL sketched in Section 5.3 of

(Ben-Ari et al., 1982). The main difference is that Ben-Ari et al. (1982) first treat all cycles as good, but then detect bad cycles by checking whether the generated interpretation really satisfies the input formula.

Automata-based methods (Vardi and Wolper, 1986) elegantly treat the problems caused by eventualities by employing appropriate acceptance conditions (e.g., Büchi acceptance). However, as mentioned above, a direct implementation of these methods is also best-case exponential. To the best of our knowledge, there is no efficient implementation of these methods, and we conjecture that an attempt to optimise them would lead to an algorithm that is very similar to a tableau algorithm.

7. Conclusion

Though many of the tableau-based algorithms sketched in this paper are of optimal worst-case complexity, and thus provide complexity results for subsumption and satisfiability in DLs, theoretical complexity results never were the main focus of this line of DL research. The design of these algorithms was strongly motivated by the goal to obtain practical algorithms, i.e., algorithms that are easy to implement and optimise, and which behave well on realistic knowledge bases. In particular, for the logics treated in Section 6.2, the exact worst-case complexity (EXPTIME) was known before the (NEXPTIME) tableau algorithms sketched above were designed. The claim that these algorithms really are "practical" must still be supported by more empirical evaluations, but the first results are rather encouraging (see below).

The notion of what is thought to be a practical subsumption algorithm in description logics has gone through a remarkable evolution in the last 15 years. Throughout the eighties and up to the early nineties, anything non-polynomial was deemed to be impractical. Consequently, when the first complexity results showed that all of the DLs used in systems had subsumption problems of a higher complexity, the proposed solution was either to restrict the expressive power or to employ incomplete algorithms. The first tableau algorithms for more expressive DLs (with PSPACE-complete subsumption problems) were widely considered to be of (complexity) theoretic interest only, though not by their designers. In fact, it turned out that implementations of these algorithms were amenable to optimisation techniques and behaved quite well in practice (Baader et al., 1994; Bresciani et al., 1995).

Following this lead, Ian Horrocks implemented the first system, FaCT, based on an EXPTIME-complete DL. The satisfiability algorithm of FaCT is a highly optimised implementation of the tableau algorithm for $\mathcal{SH}$ sketched above. FaCT was originally designed to represent medical terminology (where the whole expressive power of $\mathcal{SH}$ is needed), and it has behaved very well on the large medical knowledge base it was designed for (Horrocks, 1998b).

In addition, FaCT performed equally well on randomly generated benchmarks for formulae in (PSPACE) modal logics designed for system comparisons (Horrocks, 1998a; Patel-Schneider and Horrocks, 1999; Horrocks, 2000). These formulae do not use the whole expressive power of $\mathcal{SH}$, but to the best of our knowledge there are no benchmark formulae available for EXPTIME-complete logics. Encouraged by these experiences, other DL systems were designed that use (optimised) implementations of the tableau algorithms described in Section 6.2, and they also proved to behave quite well, both in realistic applications, and on the available (PSPACE) benchmarks (Horrocks and Patel-Schneider, 1999; Haarslev and Möller, 2000b; Horrocks et al., 2000a). This shows that, at the beginning of the new millennium, even an EXPTIME-algorithm is no longer automatically considered to be impractical in the DL community.

Databases have turned out to be a very interesting application area for DLs, which needs the expressive power offered by logics such as $\mathcal{SHIN}$. Indeed, such expressive DLs can be viewed as a unifying formalism for class-based representation systems such as object-oriented or frame-based systems, and they capture the semantics of conceptual modelling formalisms such as Entity-Relationship diagrams (Calvanese et al., 1999b). DL systems can be used to support the design and evolution of database schemata or to optimise queries (Calvanese et al., 1998a; Calvanese et al., 1998c); to support the integration of sources in heterogeneous databases/data warehouses (Calvanese et al., 1998b; Calvanese et al., 1999a); and to support the conceptual modelling of multidimensional aggregation (Franconi and Sattler, 1999).

A first tool that provides an interface for the above mentioned database applications is i•com (Franconi and Ng, 2000). Its graphical user interface supports the design of conceptual models using enhanced Entity-Relationship diagrams. The underlying inference engine is the new version of the DL system FaCT which implements $\mathcal{SHIQ}$, i.e., the extension of $\mathcal{SHIN}$ with qualified number restrictions. Once the user has finished a modelling step, she can ask the system to translate the conceptual model into a $\mathcal{SHIQ}$ knowledge base. This knowledge base is then given to FaCT, which checks for implicit IS-A (i.e., subsumption) relationships between entities/relations and tests entities/relations for inconsistencies. In case of an inconsistency or an unexpected IS-A link, the user can then modify her conceptual model appropriately.

References

Baader, F.: 1991, 'Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles'. In: *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*. A long version appeared as DFKI-Research-Report RR-90-13, Kaiserslautern, Germany.

Baader, F., M. Buchheit, and B. Hollunder: 1996, 'Cardinality Restrictions on Concepts'. *Artificial Intelligence Journal* **88**(1–2), 195–213.

Baader, F., H.-J. Bürckert, B. Nebel, W. Nutt, and G. Smolka: 1993, 'On the Expressivity of Feature Logics with Negation, Functional Uncertainty, and Sort Equations'. *Journal of Logic, Language and Information* **2**, 1–18.

Baader, F., E. Franconi, B. Hollunder, B. Nebel, and H. Profitlich: 1994, 'An Empirical Analysis of Optimization Techniques for Terminological Representation Systems or: Making KRIS get a move on'. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management* **4**, 109–132.

Baader, F. and P. Hanschke: 1991, 'A Schema for Integrating Concrete Domains into Concept Languages'. Technical Report RR-91-10, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany. An abridged version appeared in *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*.

Baader, F. and B. Hollunder: 1991, 'A Terminological Knowledge Representation System with Complete Inference Algorithm'. In: *Proc. of PDK'91*, Vol. 567 of *Lecture Notes In Artificial Intelligence*. pp. 67–86, Springer-Verlag.

Baader, F. and U. Sattler: 1999, 'Expressive Number Restrictions in Description Logics'. *Journal of Logic and Computation* **9**(3), 319–350.

Baader, F. and U. Sattler: 2000, 'Tableau Algorithms for Description Logics'. In: R. Dyckhoff (ed.): *Proc. of the Int. Conf. on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, Vol. 1847 of *Lecture Notes in Artificial Intelligence*. pp. 1–18, Springer-Verlag.

Ben-Ari, M., J. Y. Halpern, and A. Pnueli: 1982, 'Deterministic Propositional Dynamic Logic: Finite Models, Complexity, and Completeness'. *Journal of Computer and System Science* **25**, 402–417.

Borgida, A. and P. F. Patel-Schneider: 1994, 'A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic'. *Journal of Artificial Intelligence Research* **1**, 277–308.

Brachman, R. J.: 1992, '"Reducing" CLASSIC to Practice: Knowledge Representation Meets Reality'. In: *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-92)*. pp. 247–258, Morgan Kaufmann.

Brachman, R. J. and H. J. Levesque: 1984, 'The Tractability of Subsumption in Frame-Based Description Languages'. In: *Proc. of the 4th Nat. Conf. on Artificial Intelligence (AAAI-84)*, pp. 34–37, AAAI Press.

Brachman, R. J. and H. J. Levesque (eds.): 1985, *Readings in Knowledge Representation*. Morgan Kaufmann.

Brachman, R. J. and J. G. Schmolze: 1985, 'An Overview of the KL-ONE Knowledge Representation System'. *Cognitive Science* **9**(2), 171–216.

Bresciani, P., E. Franconi, and S. Tessaris: 1995, 'Implementing and Testing Expressive Description Logics: Preliminary Report'. In: A. Borgida, M. Lenzerini, D. Nardi, and B. Nebel (eds.): *Working Notes of the 1995 Description Logics Workshop*. pp. 131–139.

Buchheit, M., F. M. Donini, and A. Schaerf: 1993, 'Decidable Reasoning in Terminological Knowledge Representation Systems'. *Journal of Artificial Intelligence Research* **1**, 109–138.

Calvanese, D., G. De Giacomo, and M. Lenzerini: 1998a, 'On the Decidability of Query Containment under Constraints'. In: *Proc. of the Seventeenth ACM SIGACT SIGMOD Sym. on Principles of Database Systems (PODS-98)*. pp. 149–158.

Calvanese, D., G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati: 1998b, 'Description Logic Framework for Information Integration'. In: *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*. pp. 2–13, Morgan Kaufmann.

Calvanese, D., G. De Giacomo, and R. Rosati: 1999a, 'Data Integration and Reconciliation in Data Warehousing: Conceptual Modeling and Reasoning Support'. *Network and Information Systems* **4**(2).

Calvanese, D., G. D. Giacomo, M. Lenzerini, and D. Nardi: 2001, 'Reasoning in Expressive Description Logics'. In: A. Robinson and A. Voronkov (eds.): *Handbook of Automated Reasoning*. Amsterdam, NL: Elsevier Science Publishers.

Calvanese, D., M. Lenzerini, and D. Nardi: 1998c, 'Description Logics for Conceptual Data Modeling'. In: J. Chomicki and G. Saake (eds.): *Logics for Databases and Information Systems*. Kluwer Academic Publisher, pp. 229–264.

Calvanese, D., M. Lenzerini, and D. Nardi: 1999b, 'Unifying Class-Based Representation Formalisms'. *Journal of Artificial Intelligence Research* **11**, 199–240.

De Giacomo, G.: 1995, 'Decidability of Class-Based Knowledge Representation Formalisms'. Ph.D. thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza".

De Giacomo, G., F. Donini, and F. Massacci: 1996, 'EXPTIME tableaux for $\mathcal{ALC}$'. In: *Proc. of the 1996 Description Logic Workshop (DL'96)*.

De Giacomo, G. and M. Lenzerini: 1994, 'Boosting the Correspondence between Description Logics and Propositional Dynamic Logics'. In: *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI-94)*. pp. 205–212, AAAI Press/The MIT Press.

De Giacomo, G. and M. Lenzerini: 1996, 'TBox and ABox Reasoning in Expressive Description Logics'. In: L. C. Aiello, J. Doyle, and S. C. Shapiro (eds.): *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-96)*. pp. 316–327, Morgan Kaufmann.

De Giacomo, G. and F. Massacci: 1996, 'Tableaux and algorithms for propositional dynamic logic with converse'. In: *Proc. of the 13th Conf. on Automated Deduction (CADE-96)*, Vol. 1104 of *Lecture Notes In Artificial Intelligence*. pp. 613–628, Springer-Verlag. A long versioned will appear in *Information and Computation*.

Donini, F. M., B. Hollunder, M. Lenzerini, A. M. Spaccamela, D. Nardi, and W. Nutt: 1992, 'The Complexity of Existential Quantification in Concept Languages'. *Artificial Intelligence Journal* **2–3**, 309–327.

Donini, F. M., M. Lenzerini, D. Nardi, and W. Nutt: 1991a, 'The Complexity of Concept Languages'. In: J. Allen, R. Fikes, and E. Sandewall (eds.): *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-91)*. pp. 151–162, Morgan Kaufmann.

Donini, F. M., M. Lenzerini, D. Nardi, and W. Nutt: 1991b, 'Tractable Concept Languages'. In: *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*. pp. 458–463.

Donini, F. M., M. Lenzerini, D. Nardi, and A. Schaerf: 1994, 'Deduction in Concept Languages: From Subsumption to Instance Checking'. *Journal of Logic and Computation* **4**(4), 423–452.

Donini, F. M., M. Lenzerini, D. Nardi, and A. Schaerf: 1996, 'Reasoning in Description Logics'. In: G. Brewka (ed.): *Principles of Knowledge Representation*, Studies in Logic, Language and Information. CSLI Publications, pp. 193–238.

Donini, F. M. and F. Massacci: 1999, 'EXPTIME Tableaux for ALC'. Technical Report 32/99, Dipartimento di Ingegenria dell'Informazione, Universita degli studi di Siena, Italy. To appear in Artificial Intelligence.

Franconi, E. and G. Ng: 2000, 'The i•com Tool for Intelligent Conceptual Modelling'. In: *Working Notes of the ECAI2000 Workshop on Knowledge Representation Meets Databases (KRDB2000)*, CEUR Electronic Workshop Proceedings.

Franconi, E. and U. Sattler: 1999, 'A Data Warehouse Conceptual Data Model for Multidimensional Aggregation: a preliminary report'. *Italian Association for Artificial Intelligence AI*IA Notizie* **1**, 9–21.

Haarslev, V. and R. Möller: 1999, 'RACE System Description'. In: *Proc. of the 1999 Description Logic Workshop (DL'99)*. pp. 130–132, CEUR Electronic Workshop Proceedings.

Haarslev, V. and R. Möller: 2000a, 'Consistency Testing: The RACE Experience'. In: R. Dyckhoff (ed.): *Proc. of the Int. Conf. on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, Vol. 1847 of *Lecture Notes in Artificial Intelligence*. pp. 57–61, Springer-Verlag.

Haarslev, V. and R. Möller: 2000b, 'Expressive ABox Reasoning with Number Restrictions, Role Hierarchies, and Transitively Closed Roles'. In: *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-00)*. Morgan Kaufmann.

Halpern, J. Y. and Y. Moses: 1992, 'A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief'. *Artificial Intelligence Journal* **54**, 319–379.

Hanschke, P.: 1992, 'Specifying Role Interaction in Concept Languages'. In: *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-92)*. pp. 318–329, Morgan Kaufmann.

Harel, D.: 1984, 'Dynamic Logic'. In: *Handbook of Philosophical Logic*, Vol. 2. Dordrecht, Holland: D. Reidel, pp. 497–640.

Hollunder, B.: 1990, 'Hybrid Inferences in KL-ONE-based Knowledge Representation Systems'. In: *Proc. of GWAI'90*, Vol. 251 of *Informatik-Fachberichte*. pp. 38–47, Springer-Verlag.

Hollunder, B.: 1996, 'Consistency Checking Reduced to Satisfiability of Concepts in Terminological Systems'. *Annals of Mathematics and Artificial Intelligence* **18**(2–4), 133–157.

Hollunder, B. and F. Baader: 1991, 'Qualifying Number Restrictions in Concept Languages'. In: *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-91)*. pp. 335–346, Morgan Kaufmann.

Hollunder, B. and W. Nutt: 1990, 'Subsumption Algorithms for Concept Languages'. Technical Report RR-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany.

Hollunder, B., W. Nutt, and M. Schmidt-Schauß: 1990, 'Subsumption Algorithms for Concept Description Languages'. In: *Proc. of the 9th European Conf. on Artificial Intelligence (ECAI-90)*. pp. 348–353, Pitman.

Horrocks, I.: 1998a, 'The FaCT System'. In: H. de Swart (ed.): *Proc. of the Int. Conf. on Automated Reasoning with Tableaux and Related Methods (Tableaux-98)*, Vol. 1397 of *Lecture Notes in Artificial Intelligence*. pp. 307–312, Springer-Verlag.

Horrocks, I.: 1998b, 'Using an Expressive Description Logic: FaCT or Fiction?'. In: *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*. pp. 636–647, Morgan Kaufmann.

Horrocks, I.: 2000, 'Benchmark Analysis for FaCT'. In: R. Dyckhoff (ed.): *Proc. of the Int. Conf. on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, Vol. 1847 of *Lecture Notes in Artificial Intelligence*. pp. 62–66, Springer-Verlag.

Horrocks, I. and P. F. Patel-Schneider: 1999, 'Optimizing Description Logic Subsumption'. *Journal of Logic and Computation* **9**(3), 267–293.

Horrocks, I. and U. Sattler: 1999, 'A Description Logic with Transitive and Inverse Roles and Role Hierarchies'. *Journal of Logic and Computation* **9**(3), 385–410.

Horrocks, I., U. Sattler, and S. Tobies: 1999, 'Practical Reasoning for Expressive Description Logics'. In: *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, Vol. 1705 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag.

Horrocks, I., U. Sattler, and S. Tobies: 2000a, 'Practical Reasoning for Very Expressive Description Logics'. *Logic Journal of the IGPL* **8**(3), 239–264.

Horrocks, I., U. Sattler, and S. Tobies: 2000b, 'Reasoning with Individuals for the Description Logic SHIQ'. In: D. MacAllester (ed.): *Proc. of the 13th Conf. on Automated Deduction (CADE-17)*. Springer-Verlag.

Lutz, C.: 1999, 'Complexity of Terminological Reasoning Revisited'. In: *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, Vol. 1705 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag.

Lutz, C. and U. Sattler: 2000, 'The Complexity of Reasoning with Boolean Modal Logic'. In: *Advances in Modal Logic 2000 (AiML 2000)*. Leipzig, Germany.

MacGregor, R.: 1991, 'The Evolving Technology of Classification-Based Knowledge Representation Systems'. In: J. F. Sowa (ed.): *Principles of Semantic Networks*. Morgan Kaufmann, pp. 385–400.

Mays, E., R. Dionne, and R. Weida: 1991, 'K-REP System Overview'. *SIGART Bulletin* **2**(3).

Minsky, M.: 1981, 'A Framework for Representing Knowledge'. In: J. Haugeland (ed.): *Mind Design*. The MIT Press. Republished in (Brachman and Levesque, 1985).

Nebel, B.: 1990a, *Reasoning and Revision in Hybrid Representation Systems*, Vol. 422 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag.

Nebel, B.: 1990b, 'Terminological Reasoning is Inherently Intractable'. *Artificial Intelligence Journal* **43**, 235–249.

Nebel, B.: 1991, 'Terminological Cycles: Semantics and Computational Properties'. In: J. F. Sowa (ed.): *Principles of Semantic Networks*. Morgan Kaufmann, pp. 331–361.

Patel-Schneider, P.: 2000, 'TANCS-2000 Results for DLP'. In: R. Dyckhoff (ed.): *Proc. of the Int. Conf. on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, Vol. 1847 of *Lecture Notes in Artificial Intelligence*. pp. 72–76, Springer-Verlag.

Patel-Schneider, P. F.: 1999, 'DLP'. In: *Proc. of the 1999 Description Logic Workshop (DL'99)*. pp. 9–13, CEUR Electronic Workshop Proceedings.

Patel-Schneider, P. F. and I. Horrocks: 1999, 'DLP and FaCT'. In: *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-99)*, Vol. 1397 of *Lecture Notes In Artificial Intelligence*. pp. 19–23, Springer-Verlag.

Patel-Schneider, P. F., D. L. McGuiness, R. J. Brachman, L. A. Resnick, and A. Borgida: 1991, 'The CLASSIC Knowledge Representation System: Guiding Principles and Implementation Rational'. *SIGART Bulletin* **2**(3), 108–113.

Peltason, C.: 1991, 'The BACK System – an Overview'. *SIGART Bulletin* **2**(3), 114–119.

Pratt, V. R.: 1979, 'Models of Program Logic'. In: *Proc. of the 20th Annual Sym. on the Foundations of Computer Science (FOCS-79)*. pp. 115–122.

Quillian, M. R.: 1967, 'Word Concepts: A Theory and Simulation of Some Basic Capabilities'. *Behavioral Science* **12**, 410–430. Republished in (Brachman and Levesque, 1985).

Sattler, U.: 1996, 'A Concept Language Extended with Different Kinds of Transitive Roles'. In: G. Görz and S. Hölldobler (eds.): *Proc. of the 20th German Annual Conf. on Artificial Intelligence (KI'96)*, Vol. 1137 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag.

Savitch, W. J.: 1970, 'Relationship between Nondeterministic and Deterministic Tape Complexities'. *Journal of Computer and System Science* **4**, 177–192.

Schaerf, A.: 1993, 'On the Complexity of the Instance Checking Problem in Concept Languages with Existential Quantification'. *Journal of Intelligent Information Systems* **2**, 265–278.

Schild, K.: 1991, 'A Correspondence Theory for Terminological Logics: Preliminary Report'. In: *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*. pp. 466–471.

Schild, K.: 1994, 'Terminological Cycles and the Propositional μ-Calculus'. In: J. Doyle, E. Sandewall, and P. Torasso (eds.): *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*. pp. 509–520, Morgan Kaufmann.

Schmidt-Schauß, M.: 1989, 'Subsumption in KL-ONE is Undecidable'. In: R. J. Brachman, H. J. Levesque, and R. Reiter (eds.): *Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-89)*. pp. 421–431, Morgan Kaufmann.

Schmidt-Schauß, M. and G. Smolka: 1991, 'Attributive Concept Descriptions with Complements'. *Artificial Intelligence Journal* **48**(1), 1–26.

Spaan, E.: 1993, 'The Complexity of Propositional Tense Logics'. In: M. de Rijke (ed.): *Diamonds and Defaults*. Kluwer Academic Publishers, pp. 287–307.

Stirling, C.: 1992, 'Modal and Temporal Logic'. In: S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum (eds.): *Handbook of Logic in Computer Science*. Clarendon Press, pp. 477–563.

Tobies, S.: 1999, 'A PSPACE Algorithm for Graded Modal Logic'. In: *Proc. of the 13th Conf. on Automated Deduction (CADE-16)*, Vol. 1632 of *Lecture Notes in Computer Science*. Springer-Verlag.

Van der Hoek, W. and M. De Rijke: 1995, 'Counting Objects'. *Journal of Logic and Computation* **5**(3), 325–345.

Vardi, M. Y. and P. Wolper: 1986, 'Automata-theoretic Techniques for Modal Logics of Programs'. *Journal of Computer and System Science* **32**, 183–221. A preliminary version appeared in *Proc. of the 16th ACM SIGACT Symp. on Theory of Computing (STOC'84)*.