Practical Reasoning for Description Logics with Functional Restrictions, Inverse and Transitive Roles, and Role Hierarchies[*]

Ian Horrocks
Department of Computer Science,
University of Manchester

Ulrike Sattler and **Stephan Tobies**
LuFG Theoretical Computer Science,
RWTH Aachen

Abstract

Description Logics (DLs) are a family of knowledge representation formalisms mainly characterised by constructors to build complex concepts and roles from atomic ones. Expressive role constructors are important in many applications, but can be computationally problematical. We present an algorithm that decides satisfiability of the DL $\mathcal{ALC}$ extended with transitive and inverse roles, role hierarchies, and functional restrictions; early experiments indicate that this algorithm is well-suited for implementation. Additionally, we show that $\mathcal{ALC}$ extended with just transitive and inverse roles is still in PSPACE. Finally, we investigate the limits of decidability for this family of DLs, showing that relaxing the constraints placed on the kinds of roles used in number restrictions leads to the undecidability of all inference problems.

1 Motivation

Description Logics (DLs) are a well-known family of knowledge representation formalisms [Donini *et al.*, 1996]. They are based on the notion of concepts (unary predicates, classes) and roles (binary relations), and are mainly characterised by constructors that allow complex concepts and roles to be built from atomic ones. Sound and complete algorithms for the interesting inference problems such as subsumption and satisfiability of concepts are known for a wide variety of DLs.

Transitive and inverse roles play an important rôle not only in the adequate representation of complex, aggregated objects [Horrocks & Sattler, 1999], but also for reasoning with conceptual data models [Calvanese *et al.*, 1994]. Moreover, reasoning with respect to cyclic terminologies seems natural when using inverse roles, and is crucial with database schemata.

The relevant inference problems for (extensions of) these DLs are known to be decidable [De Giacomo & Lenzerini, 1996], and appropriate inference algorithms have been described [De Giacomo & Massacci, 1998], but their high degree of non-determinism appears to prohibit their use in realistic applications. This is mainly due to the fact that these algorithms can handle not just transitive roles but also the transitive closure of roles. It has been shown [Sattler, 1996] that restricting the DL to transitive roles can lead to a lower complexity, and that transitive roles, even when combined with role hierarchies, allow for algorithms that behave quite well in realistic applications [Horrocks, 1998]. However, it remained to show that this is still true when inverse roles are also present.

This paper extends our understanding of these issues in several directions. Firstly, we present an algorithm that decides satisfiability of $\mathcal{ALC}$ extended with transitive and inverse roles, role hierarchies, and functional restrictions. This algorithm can also be used for checking satisfiability and subsumption with respect to general concept inclusion axioms (and thus cyclic terminologies) because these axioms can be "internalised". The absence of transitive closure leads to a lower degree of non-determinism, and experiments indicate that the algorithm is well-suited for implementation.

Secondly, we show that $\mathcal{ALC}$ extended with both transitive *and* inverse roles is still in PSPACE. The algorithm used to prove this rather surprising result introduces an enhanced blocking technique that should also provide useful efficiency gains in implementations of more expressive DLs.

Finally, we investigate the limits of decidability for this family of DLs, showing that relaxing the constraints placed on the kinds of roles used in number restrictions leads to the undecidability of all inference problems.

2 Preliminaries

In this section, we present the syntax and semantics of the various DLs that are investigated in subsequent sections. This includes the definition of inference problems (concept subsumption and satisfiability, and both of these problems with respect to terminologies) and how they are interrelated.

The logics we will discuss are all based on an extension of the well known DL $\mathcal{ALC}$ [Schmidt-Schauß & Smolka, 1991] to include transitively closed primitive roles [Sattler, 1996]; we will call this logic $\mathcal{S}$ due to its relationship with the proposition (multi) modal logic $\mathbf{S4_{(m)}}$ [Schild, 1991].[1] This basic

[*]This work was partially supported by EPSRC, Grant GR/L54516, the Esprit Project 22469 – DWQ and by the DFG, Project No. GR 1324/3–1

[1]This logic has previously been called $\mathcal{ALC}_{R^+}$, but this becomes too cumbersome when adding letters to represent additional features.

DL is then extended in a variety of ways—see Figure 1 for an overview.

Definition 1 Let N_C be a set of *concept names* and $\mathbf{R}$ a set of *role names* with transitive role names $\mathbf{R}_+ \subseteq \mathbf{R}$. The set of $\mathcal{SI}$*-roles* is $\mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$. The set of $\mathcal{SI}$*-concepts* is the smallest set such that every concept name is a concept, and, if C and D are concepts and R is an $\mathcal{SI}$-role, then $(C \sqcap D)$, $(C \sqcup D)$, $(\neg C)$, $(\forall R.C)$, and $(\exists R.C)$ are also concepts.

$\mathcal{SHI}$ is obtained from $\mathcal{SI}$ by allowing, additionally, for a set of role inclusion axioms of the form $R \sqsubseteq S$, where R and S are two (possibly inverse) roles.

$\mathcal{SHIN}$ is obtained from $\mathcal{SHI}$ by allowing, additionally, for number restrictions, i.e., for concepts of the form $\leqslant nR$ and $\leqslant nR$, where R is a *simple* role and $n \in \mathbb{N}$. A role is called *simple* iff it is neither transitive nor has transitive sub-roles. $\mathcal{SHIF}$ is the restriction of $\mathcal{SHIN}$, where instead of arbitrary number restrictions, only *functional restrictions* of the form $\leqslant 1R$ and their negation $\geqslant 2R$ may occur.

An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a set $\Delta^\mathcal{I}$, called the *domain* of $\mathcal{I}$, and a function $\cdot^\mathcal{I}$ which maps every concept to a subset of $\Delta^\mathcal{I}$ and every role to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that, for all concepts C, D, roles R, S, and non-negative integers n, the properties in Figure 1 are satisfied.

A concept C is called *satisfiable* iff there is some interpretation $\mathcal{I}$ such that $C^\mathcal{I} \neq \emptyset$. Such an interpretation is called a *model of* C. A concept D *subsumes* a concept C (written $C \sqsubseteq D$) iff $C^\mathcal{I} \subseteq D^\mathcal{I}$ holds for each interpretation $\mathcal{I}$. For an interpretation $\mathcal{I}$, an individual $x \in \Delta^\mathcal{I}$ is called an *instance* of a concept C iff $x \in C^\mathcal{I}$.

All DLs considered here are closed under negation, hence subsumption and (un)satisfiability can be reduced to each other: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable, and C is unsatisfiable iff $C \sqsubseteq A \sqcap \neg A$ for some concept name A.

In order to make the following considerations easier, we introduce the following expressions:

1. To avoid considering roles such as R^{--}, we define a function Inv on roles such that $\mathsf{Inv}(R) = R^-$ if R is a role name, and $\mathsf{Inv}(R) = S$ if $R = S^-$.

2. Obviously, a role R is transitive iff $\mathsf{Inv}(R)$ is transitive. We therefore define a function Trans which returns true iff R is a transitive role. More precisely, $\mathsf{Trans}(R) = \mathrm{true}$ iff $R \in \mathbf{R}_+$ or $\mathsf{Inv}(R) \in \mathbf{R}_+$.

3. For a set of role inclusion axioms $\mathcal{R}$, $\mathcal{R}^+ := (\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}, \sqsubseteq^*)$ is called a *role hierarchy*, where $\sqsubseteq^*$ is the transitive-reflexive closure of $\sqsubseteq$ over $\mathcal{R}^+$.

In [Baader, 1990; Schild, 1991; Baader *et al.*, 1993], the *internalisation* of terminological axioms is introduced, a technique that reduces reasoning with respect to a (possibly cyclic) *terminology* to satisfiability of concepts. In [Horrocks, 1998], we saw how role hierarchies can be used for this reduction. In the presence of inverse roles, this reduction must be slightly modified.

Definition 2 A *terminology* $\mathcal{T}$ is a finite set of general concept inclusion axioms, $\mathcal{T} = \{C_1 \sqsubseteq D_1, \dots, C_n \sqsubseteq D_n\}$, where C_i, D_i are arbitrary $\mathcal{SHIF}$-concepts. An interpretation $\mathcal{I}$ is said to be a *model* of $\mathcal{T}$ iff $C_i^\mathcal{I} \subseteq D_i^\mathcal{I}$ holds for all $C_i \sqsubseteq D_i \in \mathcal{T}$. C is *satisfiable* with respect to $\mathcal{T}$ iff there is a model $\mathcal{I}$ of $\mathcal{T}$ with $C^\mathcal{I} \neq \emptyset$. Finally, D *subsumes* C with respect to $\mathcal{T}$ iff for each model $\mathcal{I}$ of $\mathcal{T}$ we have $C^\mathcal{I} \subseteq D^\mathcal{I}$.

Construct Name	Syntax	Semantics	
atomic concept	A	$A^\mathcal{I} \subseteq \Delta^\mathcal{I}$	
atomic role	R	$R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$	
transitive role	$R \in \mathbf{R}_+$	$R^\mathcal{I} = (R^\mathcal{I})^+$	
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$	
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$	$\mathcal{S}$
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$	
exists restriction	$\exists R.C$	$\{x \mid \exists y.\langle x,y\rangle \in R^\mathcal{I}$ and $y \in C^\mathcal{I}\}$	
value restriction	$\forall R.C$	$\{x \mid \forall y.\langle x,y\rangle \in R^\mathcal{I}$ implies $y \in C^\mathcal{I}\}$	
role hierarchy	$R \sqsubseteq S$	$R^\mathcal{I} \subseteq S^\mathcal{I}$	$\mathcal{H}$
inverse role	R^-	$\{\langle x,y\rangle \mid \langle y,x\rangle \in R^\mathcal{I}\}$	$\mathcal{I}$
number restrictions	$\geqslant nR$	$\{x \mid \sharp\{y.\langle x,y\rangle \in R^\mathcal{I}\} \geqslant n\}$	$\mathcal{N}$
	$\leqslant nR$	$\{x \mid \sharp\{y.\langle x,y\rangle \in R^\mathcal{I}\} \leqslant n\}$	

Figure 1: Syntax and semantics of the $\mathcal{SI}$ family of DLs

The following lemma shows how general concept inclusion axioms can be *internalised* using a "universal" role U, a transitive super-role of all roles occurring in $\mathcal{T}$ and their respective inverses.

Lemma 3 Let $\mathcal{T}$ be terminology and C, D be $\mathcal{SHIF}$-concepts and let

$$C_\mathcal{T} := \bigsqcap_{C_i \sqsubseteq D_i \in \mathcal{T}} \neg C_i \sqcup D_i.$$

Let U be a transitive role with $R \sqsubseteq U$, $\mathsf{Inv}(R) \sqsubseteq U$ for each role R that occurs in $\mathcal{T}, C$, or D.

Then C is satisfiable with respect to $\mathcal{T}$ iff $C \sqcap C_\mathcal{T} \sqcap \forall U.C_\mathcal{T}$ is satisfiable. D subsumes C with respect to $\mathcal{T}$ iff $C \sqcap \neg D \sqcap C_\mathcal{T} \sqcap \forall U.C_\mathcal{T}$ is unsatisfiable.

The proof of Lemma 3 is similar to the ones that can be found in [Schild, 1991; Baader, 1990]. Most importantly, it must be shown that, (a) if a $\mathcal{SHIF}$-concept C is satisfiable with respect to a terminology $\mathcal{T}$, then $C, \mathcal{T}$ have a *connected* model, and (b) if y is reachable from x via a role path (possibly involving inverse roles), then $\langle x,y\rangle \in U^\mathcal{I}$. These are easy consequences of the semantics and the definition of U.

Theorem 4 Satisfiability and subsumption of $\mathcal{SHIF}$-concepts (resp. $\mathcal{SHI}$-concepts) with respect to terminologies are polynomially reducible to (un)satisfiability of $\mathcal{SHIF}$-concepts (resp. $\mathcal{SHI}$-concepts).

3 Reasoning for $\mathcal{SI}$ Logics

In this section, we present two tableaux algorithms: the first decides satisfiability of $\mathcal{SHIF}$-concepts, and can be used for all $\mathcal{SHIF}$ reasoning problems (see Theorem 4); the second decides satisfiability (and hence subsumption) of $\mathcal{SI}$-concepts in PSPACE. In this paper we only sketch most of the proofs. For details on the $\mathcal{SHIF}$-algorithm, please refer to [Horrocks & Sattler, 1999], for details on the $\mathcal{SI}$- and $\mathcal{SIN}$-algorithm, please refer to [Horrocks *et al.*, 1998].

Please note that $\mathcal{SHIF}$ no longer has the finite model property: for example the following concept, where R is a transitive super-role of F, is satisfiable, but each of its models has an infinite domain.

$$\neg C \sqcap \exists F^-.C \sqcap \leqslant 1F \sqcap \forall R^-.(\exists F^-.(C \sqcap \leqslant 1F))$$

The correctness of the algorithms can be proved by showing that they create a *tableau* for a concept iff it is satisfiable. For ease of construction, we assume all concepts to be in *negation normal form* (NNF), that is, negation occurs only in front of concept names. Any $\mathcal{SHIF}$-concept can easily be transformed to an equivalent one in NNF by pushing negations inwards [Hollunder *et al.*, 1990].

Definition 5 Let D be a $\mathcal{SHIF}$-concept in NNF, $\mathbf{R}_D$ the set of roles occurring in D together with their inverses, and $sub(D)$ the subconcepts of D. Then $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a *tableau* for D iff $\mathbf{S}$ is a set of individuals, $\mathcal{L} : \mathbf{S} \to 2^{sub(D)}$ maps each individual to a set of concepts, $\mathcal{E} : \mathbf{R}_D \to 2^{\mathbf{S}\times\mathbf{S}}$ maps each role to a set of pairs of individuals, and there is some individual $s \in \mathbf{S}$ such that $D \in \mathcal{L}(s)$. Furthermore, for all $s, t \in \mathbf{S}$, $C, E \in sub(D)$, and $R, S \in \mathbf{R}_D$, it holds that:

1. if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$,
2. if $C \sqcap E \in \mathcal{L}(s)$, then $C \in \mathcal{L}(s)$ and $E \in \mathcal{L}(s)$,
3. if $C \sqcup E \in \mathcal{L}(s)$, then $C \in \mathcal{L}(s)$ or $E \in \mathcal{L}(s)$,
4. if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(R)$, then $C \in \mathcal{L}(t)$,
5. if $\exists R.C \in \mathcal{L}(s)$, then there is some $t \in \mathbf{S}$ such that $\langle s, t\rangle \in \mathcal{E}(R)$ and $C \in \mathcal{L}(t)$,
6. if $\forall S.C \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(R)$ for some $R \,\underline{\circledast}\, S$ with $\mathsf{Trans}(R)$, then $\forall R.C \in \mathcal{L}(t)$,
7. $\langle s, t\rangle \in \mathcal{E}(R)$ iff $\langle t, s\rangle \in \mathcal{E}(\mathsf{Inv}(R))$.
8. if $\langle x, y\rangle \in \mathcal{E}(R)$ and $R \,\underline{\circledast}\, S$, then $\langle x, y\rangle \in \mathcal{E}(S)$,
9. if $\leqslant 1R \in \mathcal{L}(s)$, then $\sharp\{t \mid \langle s, t'\rangle \in \mathcal{E}(R)\} \leq 1$,
10. if $\geqslant 2R \in \mathcal{L}(s)$, then $\sharp\{t \mid \langle s, t'\rangle \in \mathcal{E}(R)\} \geq 2$,

Tableaux for $\mathcal{SI}$-concepts are defined analogously and must satisfy Properties 1-7, where, due to the absence of a role hierarchy, $\underline{\circledast}$ is the identity.

Due to the close relationship between models and tableaux, the following lemma can be easily proved by induction on the structure of concepts. As a consequence, an algorithm that constructs (if possible) a tableau for an input concept is a decision procedure for satisfiability of concepts.

Lemma 6 A $\mathcal{SHIF}$-concept (resp. $\mathcal{SI}$-concept) D is satisfiable iff D has a tableau.

3.1 Reasoning in $\mathcal{SHIF}$

In the following, we give an algorithm that, given a $\mathcal{SHIF}$-concept D, decides the existence of a tableaux for D.

Definition 7 A *completion tree* for a $\mathcal{SHIF}$-concept D is a tree where each node x of the tree is labelled with a set $\mathcal{L}(x) \subseteq$ *sub*(D) and each edge $\langle x, y\rangle$ is labelled with a set $\mathcal{L}(\langle x, y\rangle)$ of (possibly inverse) roles occurring in *sub*(D).

Given a completion tree, a node y is called an *R-successor* of a node x iff y is a successor of x and $S \in \mathcal{L}(\langle x, y\rangle)$ for some S with $S \,\underline{\circledast}\, R$. A node y is called an *R-neighbour* of x iff y is an R-successor of x, or if x is an $\mathsf{Inv}(R)$-successor of y. Predecessors and ancestors are defined as usual.

A node is *blocked* iff it is directly or indirectly blocked. A node x is *directly blocked* iff none of its ancestors are blocked, and it has ancestors x', y and y' such that

1. x is a successor of x' and y is a successor of y' *and*
2. $\mathcal{L}(x) = \mathcal{L}(y)$ and $\mathcal{L}(x') = \mathcal{L}(y')$ *and*
3. $\mathcal{L}(\langle x', x\rangle) = \mathcal{L}(\langle y', y\rangle)$.

In this case we will say that y blocks x.

A node y is *indirectly blocked* iff one of its ancestors is blocked, or—in order to avoid wasted expansion after an application of the $\leqslant$-rule—it is a successor of a node x and $\mathcal{L}(\langle x, y\rangle) = \emptyset$.

For a node x, $\mathcal{L}(x)$ is said to contain a *clash* iff $\{A, \neg A\} \subseteq \mathcal{L}(x)$ or $\{\geqslant 2R, \leqslant 1S\} \subseteq \mathcal{L}(x)$ for roles $R \,\underline{\circledast}\, S$. A completion tree is called *clash-free* iff none of its nodes contains a clash; it is called *complete* iff none of the expansion rules in Figure 2 is applicable.

For a $\mathcal{SHIF}$-concept D, the algorithm starts with a completion tree consisting of a single node x with $\mathcal{L}(x) = \{D\}$. It applies the expansion rules, stopping when a clash occurs, and answers "D is satisfiable" iff the completion rules can be applied in such a way that they yield a complete and clash-free completion tree.

The soundness and completeness of the tableaux algorithm is an immediate consequence of Lemmas 6 and 8.

Lemma 8 Let D be an $\mathcal{SHIF}$-concept.

1. The tableaux algorithm terminates when started with D.
2. If the expansion rules can be applied to D such that they yield a complete and clash-free completion tree, then D has a tableau.
3. If D has a tableau, then the expansion rules can be applied to D such that they yield a complete and clash-free completion tree.

Before we sketch the ideas of the proof, we will discuss the different expansion rules and their correspondence to the language constructors.

The $\sqcap$-, $\sqcup$-, $\exists$- and $\forall$-rules are the standard $\mathcal{ALC}$ tableaux rules [Schmidt-Schauß & Smolka, 1991]. The $\forall_+$-rule is used to handle transitive roles, where the $\underline{\circledast}$-clause deals with the role hierarchy. See [Horrocks & Sattler, 1999] for details.

The functional restriction rules merit closer consideration. In order to guarantee the satisfaction of a $\geqslant 2R$-constraint, the $\geqslant$-rule creates two successors and uses a fresh atomic concept A to prohibit identification of these successors by the $\leqslant$-rule. If a node x has two or more R-neighbours and contains a functional restriction $\leqslant 1R$, then the $\leqslant$-rule merges two of the

```
⊓-rule:    if   1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$, $x$ is not indirectly blocked, and
                2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
           then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
⊔-rule:    if   1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$, $x$ is not indirectly blocked, and
                2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
           then, for some $C \in \{C_1, C_2\}$, $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$
∃-rule:    if   1. $\exists S.C \in \mathcal{L}(x)$, $x$ is not blocked, and
                2. $x$ has no $S$-neighbour $y$ with $C \in \mathcal{L}(y)$
           then create a new node $y$ with
                $\mathcal{L}(\langle x, y\rangle) = \{S\}$ and $\mathcal{L}(y) = \{C\}$
∀-rule:    if   1. $\forall S.C \in \mathcal{L}(x)$, $x$ is not indirectly blocked, and
                2. there is an $S$-neighbour $y$ of $x$ with $C \notin \mathcal{L}(y)$
           then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$
$\forall'_+$-rule: if 1. $\forall S.C \in \mathcal{L}(x)$, $x$ is not indirectly blocked,
                2. there is some $R$ with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$, and
                3. $x$ has an $R$-neighbour $y$ with $\forall R.C \notin \mathcal{L}(y)$
           then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall R.C\}$
⩾-rule:    if   1. $(\geqslant 2\ R) \in \mathcal{L}(x)$, $x$ is not blocked, and
                2. there is no $R$-neighbour $y$ of $x$ with $A \in \mathcal{L}(y)$
           then create two new nodes $y_1, y_2$ with
                $\mathcal{L}(\langle x, y_1\rangle) = \mathcal{L}(\langle x, y_2\rangle) = \{R\}$,
                $\mathcal{L}(y_1) = \{A\}$ and $\mathcal{L}(y_2) = \{\neg A\}$
⩽-rule:    if   1. $(\leqslant 1\ R) \in \mathcal{L}(x)$, $x$ is not indirectly blocked,
                2. $x$ has two $R$-neighbours $y$ and $z$
                   s.t. $y$ is not an ancestor of $z$,
           then 1. $\mathcal{L}(z) \longrightarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ and
                2. if $z$ is an ancestor of $y$
                     then $\mathcal{L}(\langle z, x\rangle) \longrightarrow \mathcal{L}(\langle z, x\rangle) \cup \mathsf{Inv}(\mathcal{L}(\langle x, y\rangle))$
                     else $\mathcal{L}(\langle x, z\rangle) \longrightarrow \mathcal{L}(\langle x, z\rangle) \cup \mathcal{L}(\langle x, y\rangle)$
                3. $\mathcal{L}(\langle x, y\rangle) \longrightarrow \emptyset$
```

Figure 2: The complete tableaux expansion rules for $\mathcal{SHIF}$

neighbours *and* the edges connecting them with x. Labelling edges with sets of roles allows a single node to be both an R and S-successor of x even if R and S are not comparable by $\sqsubseteq^*$. Finally, contradicting functional restrictions are taken care of by the definition of a clash.

We now sketch the main ideas behind the proof of Lemma 8:

1. Termination: Let $m = |\mathit{sub}(D)|$ and $n = |\mathbf{R}_D|$. Termination is a consequence of the following properties of the expansion rules:

(a) The expansion rules never remove nodes from the tree or concepts from node labels. Edge labels can only be changed by the $\leqslant$-rule which either expands them or sets them to $\emptyset$; in the latter case the node below the $\emptyset$-labelled edge is blocked. (b) Successors are only generated for concepts of the form $\exists R.C$ and $\geqslant 2R$. For a node x, each of these concepts triggers the generation of at most two successors. If for one of these successors y the $\leqslant$-rule subsequently causes $\mathcal{L}(\langle x, y\rangle)$ to be changed to $\emptyset$, then x will have some R-neighbour z with $\mathcal{L}(z) \supseteq \mathcal{L}(y)$. This, together with the definition of a clash, implies that the concept that led to the generation of y will not trigger another rule application. Obviously, the out-degree of the tree is bounded by $2m$. (c) Nodes are labelled with non-empty subsets of $\mathit{sub}(D)$ and edges with subsets of R_D, so there are at most 2^{2mn} different possible labellings for a pair of nodes and an edge. Therefore, on a path of length at least 2^{2mn} there must be 2 nodes x, y such that x is directly blocked by y. Since a path on which nodes are blocked cannot become longer, paths are of length at most 2^{2mn}.

2. Soundness: A complete and clash-free tree $\mathbf{T}$ for D induces the existence of a tableaux $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for D as follows. Individuals in $\mathbf{S}$ correspond to *paths* in $\mathbf{T}$ from the root node to some node that is not blocked. Instead of going to a directly blocked node, these paths jump back to the blocking node, which yields paths of arbitrary length. Thus, if blocking occurs, this construction yields an infinite tableau. This rather complicated tableau construction is necessary due to the presence of functional restrictions; its validity is ensured by the blocking condition, which considers both the blocked node and its predecessor.

3. Completeness: A tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for D can be used to "steer" the application of the non-deterministic $\sqcup$- and $\leqslant$-rules in a way that yields a complete and clash-free tree.

The following theorem is an immediate consequence of Lemma 8, Lemma 6, and Lemma 3.

Theorem 9 The tableaux algorithm is a decision procedure for the satisfiability and subsumption of $\mathcal{SHIF}$-concepts with respect to terminologies.

3.2 A PSPACE-algorithm for $\mathcal{SI}$

To obtain a PSPACE-algorithm for $\mathcal{SI}$, the $\mathcal{SHIF}$ algorithm is modified as follows: (a) As $\mathcal{SI}$ does not allow for functional restrictions, the $\geqslant$- and the $\leqslant$-rule can be omitted; blocking no longer involves two pairs of nodes with identical labels but only two nodes with "similar" labels. (b) Due to the absence of role hierarchies, edge labels can be restricted to roles (instead of sets of roles). (c) To obtain a PSPACE algorithm, we employ a refined blocking strategy which necessitates a second label $\mathcal{B}$ for each node. In the following, we will describe and motivate this blocking technique; detailed proofs as well as a similar result for $\mathcal{SIN}$ can be found in [Horrocks *et al.*, 1998].

Please note that using naively using a cut rule does not yield a PSpace algorithm: A cut rule similar to the one presented in [De Giacomo & Massacci, 1998] (non-deterministically) guesses which constraints will be propagated "up" the completion tree by universal restrictions on inverted roles. For $\mathcal{SI}$ this technique may lead to paths of polynomial length due to equality blocking. A way to avoid these long paths would be to stop the investigation of a path at some polynomial bound. However, to prove the correctness of this approach, it would be necessary to establish a "short-path-model" property similar to Lemma 12. Furthermore, we believe that our algorithm is better suited for an implementation since it makes less use of don't-know non-determinism.

Definition 10 A *completion tree* for a $\mathcal{SI}$ concept D is a tree where each node x of the tree is labelled with two sets $\mathcal{B}(x) \subseteq \mathcal{L}(x) \subseteq \mathit{sub}(D)$ and each edge $\langle x, y\rangle$ is labelled with a (possibly inverse) role $\mathcal{L}(\langle x, y\rangle)$ occurring in $\mathit{sub}(D)$.

R-neighbours, -successors, and -predecessors are defined as in Definition 7. Due to the absence of role hierarchies, $\sqsubseteq^*$ is the identity on $\mathbf{R}$.

```
$\sqcap$-rule:  if 1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$ and
                   2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
                then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule:  if 1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$ and
                   2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
                then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$\forall$-rule:  if 1. $\forall S.C \in \mathcal{L}(x)$ and
                   2. there is an $S$-successor $y$ of $x$ with $C \notin \mathcal{B}(y)$
                then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$ and
                     $\mathcal{B}(y) \longrightarrow \mathcal{B}(y) \cup \{C\}$ or
                  2'. there is an $S$-predecessor $y$ of $x$ with $C \notin \mathcal{L}(y)$
                then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$.
$\forall_+$-rule: if 1. $\forall S.C \in \mathcal{L}(x)$ and $\mathsf{Trans}(S)$ and
                   2. there is an $S$-succ. $y$ of $x$ with $\forall S.C \notin \mathcal{B}(y)$
                then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall S.C\}$ and
                     $\mathcal{B}(y) \longrightarrow \mathcal{B}(y) \cup \{\forall S.C\}$ or
                  2'. there is an $S$-predec. $y$ of $x$ with $\forall S.C \notin \mathcal{L}(y)$
                then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall S.C\}$.
$\exists$-rule:  if 1. $\exists S.C \in \mathcal{L}(x)$, $x$ is not blocked and no other rule
                      is applicable to any of its ancestors, and
                   2. $x$ has no $S$-neighbour $y$ with $C \in \mathcal{L}(y)$
                then create a new node $y$ with
                     $\mathcal{L}(\langle x, y\rangle) = S$ and $\mathcal{L}(y) = \mathcal{B}(y) = \{C\}$
```

Figure 3: Tableaux expansion rules for $\mathcal{SI}$

A node x is *blocked* iff for an ancestor y, y is blocked or

$$\mathcal{B}(x) \subseteq \mathcal{L}(y) \quad \text{and} \quad \mathcal{L}(x)/\,\mathsf{Inv}(S) = \mathcal{L}(y)/\,\mathsf{Inv}(S),$$

where x' is the predecessor of x, $\mathcal{L}(\langle x', x\rangle) = S$, and $\mathcal{L}(x)/\,\mathsf{Inv}(S) = \{\forall\,\mathsf{Inv}(S).C \in \mathcal{L}(x)\}$.

For a node x, $\mathcal{L}(x)$ is said to contain a *clash* iff $\{A, \neg A\} \subseteq \mathcal{L}(x)$. A completion tree to which none of the expansion rules given in Figure 3 is applicable is called *complete*.

For an $\mathcal{SI}$-concept D, the algorithm starts with a completion tree consisting of a single node x with $\mathcal{B}(x) = \mathcal{L}(x) = \{D\}$. It applies the expansion rules in Figure 3, stopping when a clash occurs, and answers "D is satisfiable" iff the completion rules can be applied in such a way that they yield a complete and clash-free completion tree.

As for $\mathcal{SHIF}$, correctness of the algorithm is proved by first showing that a $\mathcal{SI}$-concept is satisfiable iff it has a tableau, and next proving the $\mathcal{SI}$-analogue of Lemma 8.

Theorem 11 The tableaux algorithm is a decision procedure for satisfiability and subsumption of $\mathcal{SI}$-concepts.

Since blocking plays a major role both in the proof of Theorem 11 and in the following complexity considerations, we will discuss it here in more detail. Blocking is necessary to guarantee the termination of the algorithm. For DLs such as $\mathcal{ALC}$, termination is mainly due to the fact that the expansion rules can only add new concepts that are strictly smaller than the concept that triggered their application.

For $\mathcal{S}$ this is no longer true: the $\forall_+$-rule can introduce new concepts that are the same size as the triggering concept. To ensure termination, nodes labelled with a subset of the label of an ancestor are *blocked*. Since rules can be applied "top-down" (successors are only generated if no other rules are applicable, and the labels of inner leaves are never touched again) and subset-blocking is sufficient, it is possible to give a polynomial bound on the length of paths.

For $\mathcal{SI}$, dynamic blocking was introduced in [Horrocks & Sattler, 1999]. Here blocks must be established on the basis of label *equality* since value restrictions can now constrain successors as well as predecessors. Unfortunately, this may lead to completion trees with exponentially long paths because there are exponentially many possibilities to label sets on such a path. Due to the non-deterministic $\sqcup$-rule, these exponentially many sets may actually occur.

This non-determinism is not problematical for $\mathcal{S}$ because disjunctions need not be completely decomposed to yield a subset-blocking situation. For an optimal $\mathcal{SI}$ algorithm, the additional label $\mathcal{B}$ was introduced to enable a sort of subset-blocking which is independent of the $\sqcup$-non-determinism. Intuitively, $\mathcal{B}(x)$ is the restriction of $\mathcal{L}(x)$ to those non-decomposed concepts that x must satisfy, whereas $\mathcal{L}(x)$ contains boolean decompositions of these concepts as well as those that are imposed by value restrictions in descendants. If x is blocked by y, then all concepts in $\mathcal{B}(x)$ are eventually decomposed in $\mathcal{L}(y)$ (if no clash occurs). However, in order to substitute x by y, x's constraints on predecessors must be at least as strong as y's; this is taken care of by the second blocking condition.

Let us consider a path $x_1, \ldots, x_n$ where all edges are labelled R with $\mathsf{Trans}(R)$, the only kind of paths along which the length of the longest concept in the labels might not decrease. If no rules can be applied, we have $\mathcal{L}(x_{i+1})/\,\mathsf{Inv}(R) \subseteq \mathcal{L}(x_i)/\,\mathsf{Inv}(R)$ and $\mathcal{B}(x_i) \subseteq \mathcal{B}(x_{i+1}) \cup \{C_i\}$ (where $\exists R.C_i$ triggered the generation of x_{i+1}). This limits the number of labels and guarantees blocking after a polynomial number of steps.

Lemma 12 The paths of a completion tree for a concept D have a length of at most m^4 where $m = |\mathit{sub}(D)|$.

Finally, a slight modification of the expansion rules given in Figure 3 yields a PSPACE algorithm. This modification is necessary because the original algorithm must keep the whole completion tree in its memory—which needs exponential space even though the length of its paths is polynomially bounded. The original algorithm may not forget about branches because restrictions which are pushed *upwards* in the tree might make it necessary to revisit paths which have been considered before. We solve this problem as follows:

Whenever the $\forall$- or the $\forall_+$-rule is applied to a node x and its *predecessor* y (Case 2' of these rules), we delete all successors of y from the completion tree. While this makes it necessary to restart the generation of successors for y, it makes it possible to implement the algorithm in a depth-first manner which facilitates the re-use of space.

This modification does not affect the proof of soundness and completeness for the algorithm, but of course we have to re-prove termination [Horrocks *et al.*, 1998] as it formerly relied on the fact that we never removed any nodes from the completion tree. Summing up we get:

Theorem 13 The modified algorithm is a PSPACE decision procedure for satisfiability and subsumption of $\mathcal{SI}$-concepts.

4 The Undecidability of Unrestricted $\mathcal{SHIN}$

We are currently working on an algorithm for $\mathcal{SHIN}$ based on the $\mathcal{SHIF}$-algorithm already presented. Like earlier DLs that combine a hierarchy of (transitive and non-transitive) roles with some form of number restrictions [Horrocks & Sattler, 1999; Horrocks *et al.*, 1998], $\mathcal{SHIN}$ will only allow simple roles in restrictions. The justification for this limitation has been partly on the grounds of a doubtful semantics (of functional roles) and partly to simplify decision procedures. In this section we will show that, at least with respect to $\mathcal{SHIN}$, allowing arbitrary roles in $\mathcal{SHIN}$ number restrictions leads to undecidability. For convenience, we will refer to $\mathcal{SHIN}$ with arbitrary roles in number restrictions as $\mathcal{SHIN}^+$.

The undecidability proof uses a reduction of the domino problem [Berger, 1966] adapted from [Baader & Sattler, 1996]. This problem asks if, for a set of domino types, there exists a *tiling* of an $\mathbb{N}^2$ grid such that each point of the grid is covered with one of the domino types, and adjacent dominoes are "compatible" with respect to some predefined criteria.

Definition 14 A domino system $\mathcal{D} = (D, H, V)$ consists of a non-empty set of domino types $D = \{D_1, \ldots, D_n\}$, and of sets of horizontally and vertically matching pairs $H \subseteq D \times D$ and $V \subseteq D \times D$. The problem is to determine if, for a given $\mathcal{D}$, there exists a *tiling* of an $\mathbb{N} \times \mathbb{N}$ grid such that each point of the grid is covered with a domino type in D and all horizontally and vertically adjacent pairs of domino types are in H and V respectively, i.e., a mapping $t : \mathbb{N} \times \mathbb{N} \to D$ such that for all $m, n \in \mathbb{N}$, $\langle t(m,n), t(m+1,n)\rangle \in H$ and $\langle t(m,n), t(m,n+1)\rangle \in V$.

This problem can be reduced to the satisfiability of $\mathcal{SHIN}^+$-concepts, and the undecidability of the domino problem implies undecidability of satisfiability of $\mathcal{SHIN}^+$-concepts.

Ensuring that a given point satisfies the compatibility conditions is simple for most logics (using value restrictions and boolean connectives), and applying such conditions throughout the grid is also simple in a logic such as $\mathcal{SHIN}^+$ which can deal with arbitrary axioms. The crucial difficulty is representing the $\mathbb{N}\times\mathbb{N}$ grid using "horizontal" and "vertical" roles X and Y, and in particular forcing the coincidence of $X \circ Y$ and $Y \circ X$ successors. This can be accomplished in $\mathcal{SHIN}^+$ using an alternating pattern of two horizontal roles X_1 and X_2, and two vertical roles Y_1 and Y_2, with disjoint primitive concepts A, B, C, and D being used to identify points in the grid with different combinations of successors. The coincidence of $X \circ Y$ and $Y \circ X$ successors can then be enforced using number restrictions on transitive super-roles of each of the four possible combinations of X and Y roles. A visualisation of the resulting grid and a suitable role hierarchy is shown in Figure 4, where $S_{ij}^{\oplus}$ are transitive roles.

The alternation of X and Y roles in the grid means that one of the transitive super-roles S_{ij} connects each point (x, y) to the points $(x+1, y)$, $(x, y+1)$ and $(x+1, y+1)$, and to no other points. A number restriction of the form $\leqslant 3 S_{ij}$ can thus be used to enforce the necessary coincidence of $X \circ Y$ and $Y \circ X$ successors. A complete specification of the grid is given by the following axioms:

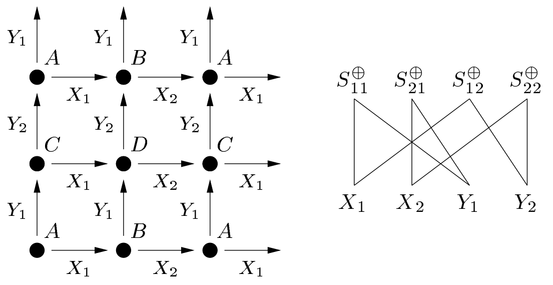

Figure 4: Visualisation of the grid and role hierarchy.

$$\begin{aligned}
A &\sqsubseteq \neg B \sqcap \neg C \sqcap \neg D \sqcap \exists X_1.B \sqcap \exists Y_1.C \sqcap \leqslant 3 S_{11},\\
B &\sqsubseteq \neg A \sqcap \neg C \sqcap \neg D \sqcap \exists X_2.A \sqcap \exists Y_1.D \sqcap \leqslant 3 S_{21},\\
C &\sqsubseteq \neg A \sqcap \neg B \sqcap \neg D \sqcap \exists X_1.D \sqcap \exists Y_2.A \sqcap \leqslant 3 S_{12},\\
D &\sqsubseteq \neg A \sqcap \neg B \sqcap \neg C \sqcap \exists X_2.C \sqcap \exists Y_2.B \sqcap \leqslant 3 S_{22}.
\end{aligned}$$

It only remains to add axioms which encode the local compatibility conditions (as described in [Baader & Sattler, 1996]) and to assert that A is subsumed by the disjunction of all domino types. The concept A is now satisfiable w.r.t. the various axioms (which can be internalised as described in Lemma 3) iff there is a compatible tiling of the grid.

5 Discussion

A new DL system is being implemented based on the $\mathcal{SHIN}$ algorithm we are currently developing from the $\mathcal{SHIF}$-algorithm described in Section 3.1. Pending the completion of this project, the existing FaCT system [Horrocks, 1998] has been modified to deal with inverse roles using the $\mathcal{SHIF}$ blocking strategy, we will refer to the modified FaCT system as I-FaCT.

I-FaCT has been used to conduct some initial experiments with a terminology representing (fragments of) database schemata and inter schema assertions from a data warehousing application [Calvanese *et al.*, 1998a] (a slightly simplified version of the proposed encoding was used to generate $\mathcal{SHIF}$ terminologies). I-FaCT is able to classify this terminology, which contains 19 concepts and 42 axioms, in less than 0.1s of (266MHz Pentium) CPU time. In contrast, eliminating inverse roles using an embedding technique [Calvanese *et al.*, 1998b] gives an equisatisfiable FaCT terminology with an additional 84 axioms, but one which FaCT is unable to classify in 12 hours of CPU time.

An extension of the embedding technique can be used to eliminate number restrictions [De Giacomo & Lenzerini, 1995], but requires a target logic which supports the transitive *closure* of roles, i.e., *converse*-PDL. The even larger number of axioms which this embedding would introduce makes it unlikely that tractable reasoning could be performed on the resulting terminology. Moreover, we are not aware of any algorithm for *converse*-PDL which does not employ a so-called *cut rule* [De Giacomo & Massacci, 1998], the application of which introduces considerable additional non-determinism. It seems inevitable that this would lead to a further degradation in empirical tractability.

In order to fully capture the above mentioned encoding of database schemata, it would be necessary not only to extend our results to $\mathcal{SHIN}$ but to $\mathcal{SHIQ}$ by adding *qualifying* number restrictions [Hollunder & Baader, 1991]. The extension of the $\mathcal{SHIF}$ algorithm, and tests of its behaviour in applications, will also be part of future work.

References

[Baader *et al.*, 1993] F. Baader, H.-J. Bürckert, B. Nebel, W. Nutt, and G. Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *Journal of Logic, Language and Information*, 2, 1993.

[Baader, 1990] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. Technical Report RR-90-13, DFKI, Germany, 1990.

[Baader & Sattler, 1996] F. Baader and U. Sattler. Number restrictions on complex roles in description logics. In *Proc. of KR-96*, pages 328–339, 1996.

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

[Calvanese *et al.*, 1994] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In *Proc. of KR-94*, 1994. M. Kaufmann, Los Altos.

[Calvanese *et al.*, 1998a] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati. Source integration in data warehousing. In *Proc. of DEXA-98*. IEEE Computer Society Press, 1998.

[Calvanese *et al.*, 1998b] D. Calvanese, G. De Giacomo, and R. Rosati. A note on encoding inverse roles and functional restrictions in ALC knowledge bases. In *Proc. of DL'98*, 1998.

[De Giacomo & Lenzerini, 1995] G. De Giacomo and M. Lenzerini. What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of IJCAI-95*, 1995.

[De Giacomo & Lenzerini, 1996] G. De Giacomo and M. Lenzerini. Tbox and Abox reasoning in expressive description logics. In *Proc. of KR-96*. M. Kaufmann, Los Altos, 1996.

[De Giacomo & Massacci, 1998] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for converse-PDL. *Information and Computation*, 1998. To appear.

[Donini *et al.*, 1996] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Foundation of Knowledge Representation*. CSLI Publication, Cambridge University Press, 1996.

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

[Hollunder & Baader, 1991] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of KR-91*, pages 335–346, Boston, MA, USA, 1991.

[Horrocks *et al.*, 1998] I. Horrocks, U. Sattler, and S. Tobies. A PSPACE-algorithm for deciding $\mathcal{ALCI}_{R^+}$-satisfiability. Technical Report 98-08, LuFg Theoretical Computer Science, RWTH Aachen, 1998. See http://www-lti.informatik.rwth-aachen.de/

[Horrocks, 1998] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR-98*, 1998.

[Horrocks & Sattler, 1999] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 1999. To appear.

[Sattler, 1996] U. Sattler. A concept language extended with different kinds of transitive roles. In *Proc. of KI'96*, vol. 1137 of *LNAI*. Springer-Verlag, 1996.

[Schild, 1991] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, 1991.

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