Practical Reasoning for Expressive Description Logics

Ian Horrocks[1] and Ulrike Sattler[2] and Stephan Tobies[2]

[1] Department of Computer Science, University of Manchester[†]
[2] 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 qualifying number 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.

1 Motivation

Description Logics (DLs) are a well-known family of knowledge representation formalisms [DLNS96]. 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 [SS91,DLNdN91,Sat96,DL96,CDL99].

To be used in a specific application, the expressivity of the DL must be sufficient to describe relevant properties of objects in the application domain. For example, transitive roles (e.g. "ancestor") and inverse roles (e.g. "successor"/"predecessor") play an important rôle not only in the adequate representation of complex, aggregated objects [HS99], but also for reasoning with conceptual data models [CLN94]. Moreover, reasoning with respect to cyclic definitions is crucial for applying DLs to reasoning with database schemata [CDL98a].

The relevant inference problems for (extensions of) DLs that allow for transitive and inverse roles are known to be decidable [DL96], and appropriate inference algorithms have been described [DM98], but their high degree of nondeterminism 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

[†] Part of this work was carried out while being a guest at IRST, Trento.
[‡] This work was supported by the Esprit Project 22469 – DWQ and the DFG, Project No. GR 1324/3-1.

the transitive closure of roles. It has been shown [Sat96] that restricting a 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 [Hor98]. However, it remained to show that this is still true when inverse roles and qualifying number restrictions are also present.

This paper extends our understanding of these issues in several directions. Firstly, we present an algorithm that decides satisfiability of $\mathcal{ALC}$ [SS91] (which can be seen as a notational variant of the multi modal logic K_m) extended with transitive and inverse roles, role hierarchies, and qualifying number restrictions, i.e., concepts of the form ($\geqslant 3$ *hasChild Female*) that allow the description of objects by restricting the number of objects of a given type they are related to via a certain role. The algorithm can also be used for checking satisfiability and subsumption with respect to general concept inclusion axioms (and thus cyclic definitions) 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. In general, blocking is used to ensure termination of the algorithm in cases where it would otherwise be stuck in a loop. The enhanced blocking technique allows such cases to be detected earlier and should provide useful efficiency gains in implementations of this and 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 allowed in number restrictions leads to the undecidability of all inference problems.

Due to a lack of space we can only present selected proofs. For full details please refer to [HST98,HST99].

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 $\mathcal{ALC}$ [SS91] to include transitively closed primitive roles [Sat96]; we will call this logic $\mathcal{S}$ due to its relationship with the proposition (multi) modal logic $\mathbf{S4_{(m)}}$ [Sch91].[1] This basic DL is then extended in a variety of ways—see Figure 1 for an overview.

Definition 1. *Let* $\mathbf{C}$ *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*

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

*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.*

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 $R = S^-$. We also define a function* Trans *which returns* true *iff R is a transitive role. More precisely,* $\mathsf{Trans}(R) = \text{true}$ *iff $R \in \mathbf{R}_+$ or $\mathsf{Inv}(R) \in \mathbf{R}_+$.*

$\mathcal{SHI}$ 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 roles, each of which can be inverse. For a set of role inclusion axioms $\mathcal{R}$,*

$$\mathcal{R}^+ := (\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}, \ \underline{\sqsubseteq}^*)$$

is called a role hierarchy, *where $\underline{\sqsubseteq}^*$ is the transitive-reflexive closure of $\sqsubseteq$ over $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}$.*

$\mathcal{SHIQ}$ is obtained from $\mathcal{SHI}$ by allowing, additionally, for qualifying number restrictions, *i.e., for concepts of the form $(\geqslant n\ R\ C)$ and $(\leqslant n\ R\ C)$, where R is a* simple *(possibly inverse) role and n is a non-negative integer. A role is called* simple *iff it is neither transitive nor has transitive sub-roles.*

$\mathcal{SHIN}$ is the restriction of $\mathcal{SHIQ}$ where qualifying number restrictions may only be of the form $(\geqslant n\ R\ \top)$ and $(\leqslant n\ R\ \top)$. In this case, we omit the symbol $\top$ and write $(\geqslant n\ R)$ and $(\leqslant n\ R)$ instead.

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* valuation $\cdot^{\mathcal{I}}$ *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, where $\sharp M$ denotes the cardinality of a set M. An interpretation satisfies a role hierarchy $\mathcal{R}^+$ iff $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for each $R \underline{\sqsubseteq}^* S \in \mathcal{R}^+$; we denote this fact by $\mathcal{I} \models \mathcal{R}^+$ and say that $\mathcal{I}$ is a model of $\mathcal{R}^+$.*

A concept C is called satisfiable *with respect to a role hierarchy $\mathcal{R}^+$ iff there is some interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{R}^+$ and $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a* model of *C w.r.t. $\mathcal{R}^+$. A concept D* subsumes *a concept C w.r.t. $\mathcal{R}^+$ (written $C \sqsubseteq_{\mathcal{R}^+} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each model $\mathcal{I}$ of $\mathcal{R}^+$. 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 w.r.t. role hierarchies can be reduced to each other: $C \sqsubseteq_{\mathcal{R}^+} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{R}^+$, and C is unsatisfiable w.r.t. $\mathcal{R}^+$ iff $C \sqsubseteq_{\mathcal{R}^+} A \sqcap \neg A$ for some concept name A.

In [Baa91,Sch91,BBN+93], the *internalisation* of terminological axioms is introduced, a technique that reduces reasoning with respect to a (possibly cyclic) terminology to satisfiability of concepts. In [Hor98], we saw how role hierarchies can be used for this reduction. In the presence of inverse roles, this reduction must be slightly modified.

Construct Name	Syntax	Semantics	
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	
universal concept	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$	
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	$\geqslant nR$ $\leqslant nR$	$\{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}}\} \geqslant n\}$ $\{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}}\} \leqslant n\}$	$\mathcal{N}$
qualifying number restrictions	$\geqslant nR.C$ $\leqslant nR.C$	$\{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\} \geqslant n\}$ $\{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\} \leqslant n\}$	$\mathcal{Q}$

Fig. 1. Syntax and semantics of the $\mathcal{SI}$ family of DLs

Definition 2. *A* terminology $\mathcal{T}$ *is a finite set of* general concept inclusion axioms, $\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$, *where C_i, D_i are arbitrary $\mathcal{SHIQ}$-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 $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.*

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

Lemma 1. *Let $\mathcal{T}$ be a terminology, $\mathcal{R}$ a set of role inclusion axioms and C, D $\mathcal{SHIQ}$-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 that does not occur in $\mathcal{T}, C, D$, or $\mathcal{R}$. We set

$$\mathcal{R}_U := \mathcal{R} \cup \{R \sqsubseteq U, \mathsf{Inv}(R) \sqsubseteq U \mid R \textit{ occurs in } \mathcal{T}, C, D, \textit{ or } \mathcal{R}\}.$$

Then C is satisfiable w.r.t. $\mathcal{T}$ and $\mathcal{R}^+$ iff $C \sqcap C_{\mathcal{T}} \sqcap \forall U.C_{\mathcal{T}}$ is satisfiable w.r.t. $\mathcal{R}_U^+$. Moreover, D subsumes C with respect to $\mathcal{T}$ and $\mathcal{R}^+$ iff $C \sqcap \neg D \sqcap C_{\mathcal{T}} \sqcap \forall U.C_{\mathcal{T}}$ is unsatisfiable w.r.t. $\mathcal{R}_U^+$.

The proof of Lemma 1 is similar to the ones that can be found in [Sch91,Baa91]. Most importantly, it must be shown that, (a) if a $\mathcal{SHIQ}$-concept C is satisfiable with respect to a terminology $\mathcal{T}$ and a role hierarchy $\mathcal{R}^+$, 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 1. *Satisfiability and subsumption of $\mathcal{SHIQ}$-concepts (resp. $\mathcal{SHI}$-concepts) w.r.t. terminologies and role hierarchies are polynomially reducible to (un)satisfiability of $\mathcal{SHIQ}$-concepts (resp. $\mathcal{SHI}$-concepts) w.r.t. role hierarchies.*

3 Reasoning for $\mathcal{SI}$ Logics

In this section, we present two tableaux algorithms: the first decides satisfiability of $\mathcal{SHIQ}$-concepts, and can be used for all $\mathcal{SHIQ}$ reasoning problems (see Theorem 1); the second decides satisfiability (and hence subsumption) of $\mathcal{SI}$-concepts in PSPACE. Please note that $\mathcal{SHIN}$ (and hence $\mathcal{SHIQ}$) 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.

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

This concept requires the existence of an infinite F^--path, where the first element on the path satisfies $\neg C$ while all other elements satisfy $C \sqcap \leqslant 1 F$. This path cannot collapse into a cycle: (a) it cannot return to the first element because this element cannot satisfy both C and $\neg C$; (b) it cannot return to any subsequent element on the path because then this node would not satisfy $\leqslant 1 F$.

The correctness of the algorithms we are presenting 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 $\mathcal{SHIQ}$-concept can easily be transformed to an equivalent one in NNF by pushing negations inwards [HNS90]; with $\sim C$ we denote the NNF of $\neg C$. For a concept C in NNF we define $\mathit{clos}(C)$ as the smallest set of concepts that contains C and is closed under subconcepts and $\sim$. Please note that size of $\mathit{clos}(C)$ is linearly bounded by the size of C.

Definition 3. *Let D be a $\mathcal{SHIQ}$-concept in NNF, $\mathcal{R}^+$ a role hierarchy, and $\mathbf{R}_D$ the set of roles occurring in D and $\mathcal{R}^+$ together with their inverses. Then $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a* tableau *for D w.r.t. $\mathcal{R}^+$ iff $\mathbf{S}$ is a set of individuals, $\mathcal{L} : \mathbf{S} \to 2^{\mathit{clos}(D)}$ 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 $s, t \in \mathbf{S}$, $C, C_1, C_2 \in \mathit{clos}(D)$, 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 $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,*
3. *if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,*
4. *if $\forall S.C \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(S)$, then $C \in \mathcal{L}(t)$,*
5. *if $\exists S.C \in \mathcal{L}(s)$, then there is some $t \in \mathbf{S}$ such that $\langle s, t \rangle \in \mathcal{E}(S)$ 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* $R \mathrel{\underline{\sqsubseteq}^*} S$ *with* $\mathsf{Trans}(R)$*, then* $\forall R.C \in \mathcal{L}(t)$,
7. $\langle x,y\rangle \in \mathcal{E}(R)$ *iff* $\langle y,x\rangle \in \mathcal{E}(\mathsf{Inv}(R))$,
8. *if* $\langle s,t\rangle \in \mathcal{E}(R)$ *and* $R \mathrel{\underline{\sqsubseteq}^*} S$*, then* $\langle s,t\rangle \in \mathcal{E}(S)$,
9. *if* $(\leqslant n\ S\ C) \in \mathcal{L}(s)$*, then* $\sharp S^T(s,C) \leqslant n$,
10. *if* $(\geqslant n\ S\ C) \in \mathcal{L}(s)$*, then* $\sharp S^T(s,C) \geqslant n$,
11. *if* $(\bowtie\ n\ S\ C) \in \mathcal{L}(s)$ *and* $\langle s,t\rangle \in \mathcal{E}(S)$ *then* $C \in \mathcal{L}(t)$ *or* ${\sim}C \in \mathcal{L}(t)$,

where we use $\bowtie$ *as a placeholder for both* $\leqslant$ *and* $\geqslant$ *and we define*

$$S^T(s,C) := \{t \in \mathbf{S} \mid \langle s,t\rangle \in \mathcal{E}(S) \text{ and } C \in \mathcal{L}(t)\}.$$

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

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

Lemma 2. *A* $\mathcal{SHIQ}$*-concept (resp.* $\mathcal{SI}$*-concept)* D *is satisfiable w.r.t. a role hierarchy* $\mathcal{R}^+$ *iff* D *has a tableau w.r.t.* $\mathcal{R}^+$*.*

3.1 Reasoning in $\mathcal{SHIQ}$

In the following, we give an algorithm that, given a $\mathcal{SHIQ}$-concept D, decides the existence of a tableaux for D. We implicitly assume an arbitrary but fixed role hierarchy $\mathcal{R}^+$. The tableaux algorithm works on a finite *completion tree* (a tree some of whose nodes correspond to individuals in the tableau, each node being labelled with a set of $\mathcal{SHIQ}$-concepts), and employs a *blocking* technique [HS99] to guarantee termination: If a path contains two pairs of successive nodes that have pair-wise identical label and whose connecting edges have identical labels, then the path beyond the second pair is no longer expanded, it is said to be blocked. Blocked paths can be "unravelled" to construct an infinite tableau. The identical labels make sure that copies of the first pair and their descendants can be substituted for the second pair of nodes and their respective descendants.

Definition 4. *A* completion tree *for a* $\mathcal{SHIQ}$*-concept* D *is a tree where each node* x *of the tree is labelled with a set* $\mathcal{L}(x) \subseteq clos(D)$ *and each edge* $\langle x,y\rangle$ *is labelled with a set* $\mathcal{L}(\langle x,y\rangle)$ *of (possibly inverse) roles occurring in* $clos(D)$*; explicit inequalities between nodes of the tree are recorded in a binary relation* $\neq$ *that is implicitly assumed to be symmetric.*

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* $S \mathrel{\underline{\sqsubseteq}^*} R$*. 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. Since this blocking technique involves pairs of nodes, it is called* pair-wise *blocking.*

A node y is indirectly blocked *iff one of its ancestors is blocked, or it is a successor of a node x and $\mathcal{L}(\langle x, y\rangle) = \emptyset$; the latter condition avoids wasted expansions after an application of the $\leqslant$-rule.*

For a node x, $\mathcal{L}(x)$ is said to contain a clash *iff $\{A, \neg A\} \subseteq \mathcal{L}(x)$ or if, for some concept C, some role S, and some $n \in \mathbb{N}$: $(\leqslant n\ S\ C) \in \mathcal{L}(x)$ and there are $n + 1$ S-neighbours $y_0, \ldots, y_n$ of x such that $C \in \mathcal{L}(y_i)$ and $y_i \neq y_j$ for all $0 \leq i < j \leq n$. 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 $\mathcal{SHIQ}$-concept D, the algorithm starts with a completion tree consisting of a single node x with $\mathcal{L}(x) = \{D\}$ and $\neq\, = \emptyset$. It applies the expansion rules in Figure 2, 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 2 and 3.

Lemma 3. *Let D be an $\mathcal{SHIQ}$-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.*

The proof can be found in the appendix. Here, we will only discuss the intuition behind the expansion rules and their correspondence to the constructors of $\mathcal{SHIQ}$. Roughly speaking,[2] the completion tree is a partial description of a model whose individuals correspond to nodes, and whose interpretation of roles is taken from the edge labels. Since the completion tree is a tree, this would not yield a correct interpretation of transitive roles, and thus the interpretation of transitive roles is built via the transitive closure of the relations induced by the corresponding edge labels.

The $\sqcap$-, $\sqcup$-, $\exists$- and $\forall$-rules are the standard tableaux rules for $\mathcal{ALC}$ or the propositional modal logic K_m. The $\forall_+$-rule is the standard rule for $\mathcal{ALC}_{R^+}$ or the propositional modal logic $\mathsf{S4}_m$ extended to deal with role-hierarchies as follows. Assume a situation that satisfies the precondition of the $\forall_+$-rule, i.e., $\forall S.C \in$

[2] For the following considerations, we employ a simpler view of the correspondence between completion trees and models, and need not bother with the path construction mentioned above.

$\sqcap$-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 $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-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 $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$\exists$-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 $\mathcal{L}(y) = \{C\}$
$\forall$-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, and 2. there is some R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$, 3. there is an R-neighbour y of x with $\forall R.C \notin \mathcal{L}(y)$ then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall R.C\}$
choose-rule:	if 1. $(\bowtie\ n\ S\ C) \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. there is an S-neighbour y of x with $\{C, {\sim}C\} \cap \mathcal{L}(y) = \emptyset$ then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, {\sim}C\}$
$\geqslant$-rule:	if 1. $(\geqslant n\ S\ C) \in \mathcal{L}(x)$, x is not blocked, and 2. there are not n S-neighbours $y_1, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$ and $y_i \neq y_j$ for $1 \leq i < j \leq n$ then create n new nodes $y_1, \ldots, y_n$ with $\mathcal{L}(\langle x, y_i\rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$.
$\leqslant$-rule:	if 1. $(\leqslant n\ S\ C) \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\sharp S^{\mathbf{T}}(x, C) > n$ and there are two S-neighbours y, z of x with $C \in \mathcal{L}(y), C \in \mathcal{L}(z)$, y is not an ancestor of x, and not $y \neq z$ then 1. $\mathcal{L}(z) \longrightarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ and 2. if z is an ancestor of x 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. $\mathcal{L}(\langle x, y\rangle) \longrightarrow \emptyset$ 4. Set $u \neq z$ for all u with $u \neq y$

Fig. 2. The complete tableaux expansion rules for $\mathcal{SHIQ}$

$\mathcal{L}(x)$, and there is an R-neighbour y of x with $\mathsf{Trans}(R)$, $R \sqsubseteq^* S$ and $\forall R.C \notin \mathcal{L}(y)$. If y has an R-successor z, then, due to the transitivity of R, z is also an R-successor of x. Since $R \sqsubseteq^* S$, it is also an S-successor of x and hence must satisfy C. This is ensured by adding $\forall R.C$ to $\mathcal{L}(z)$

The rules dealing with qualifying number restrictions work similarly to the rules given in [BBH96]. For a concept $(\geqslant n\ R\ C) \in \mathcal{L}(x)$, the $\geqslant$-rule generates n R-successors $y_1, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$. To prevent the $\leqslant$-rule from indentifying the new nodes, it also sets $y_i \neq y_j$ for each $1 \leq i < j \leq n$. Conversely, if $(\leqslant n\ R\ C) \in \mathcal{L}(x)$ and x has more than n R-neighbours that are labelled with C, then the $\leqslant$-rule chooses two of them that are not in $\neq$ and merges them, together with the edges connecting them with x. The definition of a clash takes care of the situation where the $\neq$ relation makes it impossible to merge any two R-neighbours of x, while the *choose*-rule ensures that all R-neighbours of x are labelled with either C or $\sim C$. Without this rule, the unsatisfiability of concepts like $(\geqslant 3\ R\ A) \sqcap (\leqslant 1\ R\ B) \sqcap (\leqslant 1\ R\ \neg B)$ would go undetected. The relation $\neq$ is used to prevent infinite sequences of rule applications for contradicting number restrictions of the form $(\geqslant n\ R\ C)$ and $(\leqslant (m)\ R\ C)$, with $n > m$. 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 with respect to $\underline{*}$.

The following theorem is an immediate consequence of Lemma 2 and 3, and Theorem 1.

Theorem 2. *The tableaux algorithm is a decision procedure for the satisfiability and subsumption of $\mathcal{SHIQ}$-concepts with respect to terminologies.*

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

To obtain a (worst-case) optimal algorithm for $\mathcal{SI}$, the $\mathcal{SHIQ}$ algorithm is modified as follows. (a) Since $\mathcal{SI}$ does not allow for qualifying number restrictions the $\geqslant$-, $\leqslant$-, and *choose*-rule can be omitted. In the absence of the *choose*-rule we may assume all concepts appearing in labels to be in NNF from the (smaller) set of all subconcepts of D denoted by $sub(D)$, and in the absence of role hierarchies, edge labels can be restricted to roles (instead of sets of roles). Due to the absence of number restrictions the logic still has the finite model property, and blocking no longer need involve two pairs of nodes with identical labels, but only two nodes with (originally) identical labels. (b) To obtain a PSPACE algorithm, we employ a refined blocking strategy which further loosens this "identity" condition to a "similarity" condition. This is achieved by using a second label $\mathcal{B}$ for each node. In the following, we will describe and motivate this blocking technique; detailed proofs as well as an extension of this result to $\mathcal{SIN}$ can be found in [HST98].

Establishing a PSPACE-result for $\mathcal{SI}$ is not as straightforward as it might seem at a first glance. One problem is the presence of inverse roles which might lead to constraints propagating upwards in the tree. This is not compatible with the standard trace technique [SS91] that keeps only a single path in memory at the same time, because constraints propagating upwards in the tree may have an influence on paths that have already been visited and have been discarded from memory. There are at least two possibilities to overcome this problem: (1) by guessing which constraints might propagate upwards beforehand; (2) by a *reset-restart* extension of the trace technique described later in this section. Unfortunately, this is not the only problem. To apply either of these two techniques, it is also necessary to establish a polynomial bound on the length of paths in the completion tree. This is easily established for logics such as $\mathcal{ALC}$ that do not allow for transitive roles. For $\mathcal{ALC}$ with transitive roles (i.e., $\mathcal{S}$), this bound is due to the fact that, for a node x to block a node y, it is sufficient that $\mathcal{L}(y) \subseteq \mathcal{L}(x)$. In the presence of inverse roles, we use a more sophisticated blocking technique to establish the polynomial bound.

$\sqcap$-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-predecessor 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 $\mathcal{L}(y) = \mathcal{B}(y) = \{C\}$

Fig. 3. Tableaux expansion rules for $\mathcal{SI}$

Definition 5. *A* completion tree *for an $\mathcal{SI}$ concept D is a tree where each node x of the tree is labelled with two sets $\mathcal{B}(x) \subseteq \mathcal{L}(x) \subseteq sub(D)$, and each edge $\langle x, y\rangle$ is labelled with a (possibly inverse) role $\mathcal{L}(\langle x, y\rangle)$ occurring in $sub(D)$.*

R-neighbours, -successors, and -predecessors are defined as in Definition 4 where, in the absence of role hierarchies, $\underline{\overset{*}{\sqsubseteq}}$ is the identity on $\mathbf{R}$.

A node x is blocked *iff x has a blocked ancestor y, or x has an ancestor y and a predecessor x' with $\mathcal{L}(\langle x', x\rangle) = S$, and*

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

where $\mathcal{L}(x)/\,\mathsf{Inv}(S) = \{\forall\,\mathsf{Inv}(S).C \in \mathcal{L}(x)\}$.

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 $\mathcal{SHIQ}$, correctness of the algorithm can be 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 3, see [HST98].

Theorem 3. *The tableaux algorithm is a decision procedure for satisfiability and subsumption of $\mathcal{SI}$-concepts.*

Since blocking plays a major rôle both in the proof of Theorem 3 and especially in the following complexity considerations, we will discuss it here in more detail. Blocking guarantees 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 introduces 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 nodes are never touched again) and subset-blocking is sufficient (i.e., for a node x to be blocked by an ancestor y, it is sufficient that $\mathcal{L}(x) \subseteq \mathcal{L}(y)$), it is possible to give a polynomial bound on the length of paths.

For $\mathcal{SI}$, *dynamic blocking* was introduced in [HS99], i.e., blocks are not established on a once-and-for-all basis, but established and broken dynamically. Moreover, blocks must be established on the basis of label *equality*, since value restrictions can now constrain predecessors as well as successors. 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 $\mathcal{L}(y)$. 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 $x_0, x_1, \ldots, x_n$ where all edges are labelled R with $\mathsf{Trans}(R)$, the only kind of path along which the length of the longest concept in the labels might not decrease. If no rules can be applied, then we have, for $1 \leq i < n$,

$$\mathcal{L}(x_{i+1})/\,\mathsf{Inv}(R) \subseteq \mathcal{L}(x_i)/\,\mathsf{Inv}(R) \quad \text{and}$$
$$\mathcal{B}(x_i) \subseteq \mathcal{B}(x_{i+1}) \cup \{C_i\}$$

(where $\exists R.C_i \in \mathcal{L}(x_i)$ triggered the generation of x_{i+1}). This limits the number of different labels and guarantees blocking after a polynomial number of steps.

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

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 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. A *reset-restart* mechanism solves 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 (*reset*). 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 [HST98] as it formerly relied on the fact that we never removed any nodes from the completion tree. Summing up we get:

Theorem 4. *The modified algorithm is a* PSPACE *decision procedure for satisfiability and subsumption of* $\mathcal{SI}$*-concepts.*

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

Like earlier DLs that combine a hierarchy of (transitive and non-transitive) roles with some form of number restrictions [HS99,HST98], $\mathcal{SHIN}$ only allows *simple* roles in restrictions, i.e. roles that are neither transitive nor have transitive subroles. The justification for this limitation has been partly on the grounds of a doubtful semantics (of transitive functional roles) and partly to simplify decision procedures. In this section, we will show that allowing arbitrary roles in $\mathcal{SHIN}$ number restrictions leads to undecidability. For convenience, we denote $\mathcal{SHIN}$ with arbitrary roles in number restrictions by $\mathcal{SHIN}^+$.

The undecidability proof uses a reduction of the domino problem [Ber66] adapted from [BS96]. This problem asks whether, 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 exactly one of the domino types, and adjacent dominoes are "compatible" with respect to some predefined criteria.

Definition 6. *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 each point is associated with exactly one domino type and that a point and its neighbours satisfy the compatibility conditions induced by H and

V is simple for most logics (via the introduction of concepts C_{D_i} for domino types D_i, and the use of 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.

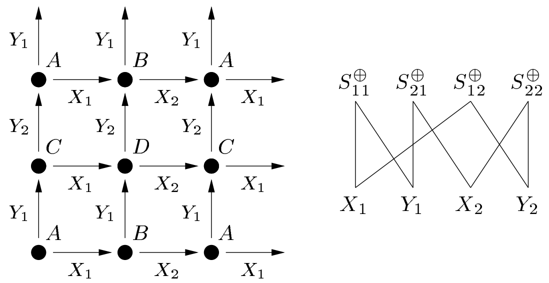

Fig. 4. Visualisation of the grid and role hierarchy.

The alternation of X and Y roles in the grid means that one of the transitive super-roles S_{ij} connects each point (m, n) to the points $(m+1, n)$, $(m, n+1)$ and $(m+1, n+1)$, and to no other points. A number restriction of the form $\leqslant 3S_{ij}$ 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:

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

It only remains to add axioms which encode the local compatibility conditions (as described in [BS96]) and to assert that A, B, C, and D are subsumed by the disjunction of all domino types to enforce the placement of a tile on each point of the grid. The concept A is now satisfiable w.r.t. the various axioms (which can be internalised as described in Lemma 1) iff there is a compatible tiling of the grid.

5 Discussion

A new DL system is being implemented based on the $\mathcal{SHIQ}$ algorithm described in Section 3.1. Pending the completion of this project, the existing FaCT system [Hor98] has been modified to deal with inverse roles using the $\mathcal{SHIQ}$ blocking strategy, giving a DL which is equivalent to $\mathcal{SHI}$ extended with functional roles [HS99]; we will refer to this DL as $\mathcal{SHIF}$ and 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 [CDL+98] (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 [CDR98] 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 [DL95], 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* [DM98], the application of which introduces considerable additional non-determinism. It seems inevitable that this would lead to a further degradation in empirical tractability.

As far as complexity is concerned, we have already been successful in extending the PSPACE-result for $\mathcal{SI}$ to $\mathcal{SIN}$ [HST98]. Currently we are working on an extension of this result to $\mathcal{SIQ}$ combining the techniques from this paper with those presented in [Tob99].

References

[Baa91] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of IJCAI-91*, 1991.

[BBH96] F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[BBN+93] 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. *J. of Logic, Language and Information*, 2:1–18, 1993.

[Ber66] R. Berger. The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, 66, 1966.

[BS96] F. Baader and U. Sattler. Number restrictions on complex roles in description logics. In *Proc. of KR-96*, pages 328–339, 1996.

[CDL98a] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT*

SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98), pages 149–158, 1998.

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

[CDL99] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of the 16th Int. Joint. Conf. on Artificial Intelligence (IJCAI'99)*, 1999.

[CDR98] D. Calvanese, G. De Giacomo, and R. Rosati. A note on encoding inverse roles and functional restrictions in $\mathcal{ALC}$ knowledge bases. In *Proc. of DL'98*, 1998.

[CLN94] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. A unified framework for class based representation formalisms. *Proc. of KR-94*, pages 109–120. M. Kaufmann, Los Altos.

[DL95] 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.

[DL96] G. De Giacomo and M. Lenzerini. Tbox and Abox reasoning in expressive description logics. In *Proc. of KR-96*, pages 316–327. M. Kaufmann, Los Altos, 1996.

[DLNdN91] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of KR-91*, Boston, MA, USA, 1991.

[DLNS96] 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.

[DM98] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. *Information and Computation*, 1998. To appear.

[HNS90] B. Hollunder, W. Nutt, and M. Schmidt-Schauss. Subsumption algorithms for concept description languages. In *ECAI-90*, Pitman Publishing, London, 1990.

[Hor98] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR-98*, pages 636–647, 1998.

[HS99] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 1999. To appear.

[HST98] 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/Forschung/Papers.html.

[HST99] I. Horrocks, U. Sattler, and S. Tobies. A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions. LTCS-Report 99-08, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1999.

[Sat96] U. Sattler. A concept language extended with different kinds of transitive roles. In *20. Deutsche Jahrestagung für KI*, LNAI 1137. Springer-Verlag, 1996.

[Sch91] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, Sydney, 1991.

[SS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Tob99] S. Tobies. A PSpace algorithm for graded modal logic. In *Proc. of CADE-16*, LNCS. Springer, 1999.

Appendix

In this appendix we present the proof of Lemma 3, which is repeated here for easier reference.

Lemma. *Let D be an $\mathcal{SHIQ}$-concept.*

1. *(Termination) The tableaux algorithm terminates when started with D.*
2. *(Soundness) 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. *(Completeness) 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.*

Termination Let $m = |clos(D)|$, $k = |\mathbf{R}_D|$, and n_{max} the maximum n that occurs in a concept of the form $(\bowtie\ n\ S\ C) \in clos(D)$. Termination is a consequence of the following properties of the expansion rules:

- 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.
- Successors are only generated for concepts of the form $\exists R.C$ (which yields one successor) and $(\geqslant n\ S\ C)$ (which yields n successors). For a node x, each of these concepts can trigger the generation of successors at most once. For the $\exists$-rule, if a successor y of x was generated for a concept $\exists S.C \in \mathcal{L}(x)$ and later $\mathcal{L}(\langle x, y\rangle)$ is set to $\emptyset$ by an application of the $\leqslant$-rule, then there will be some S-neighbour z of x such that $C \in \mathcal{L}(z)$. For the $\geqslant$-rule, if $y_1, \ldots, y_n$ were generated by an application of the $\geqslant$-rule for a concept $(\geqslant n\ S\ C)$, then $y_i \neq y_j$ holds for all $1 \leq i < j \leq n$. This implies that there will always be n S-neighbours $y'_1, \ldots, y'_n$ of x with $C \in \mathcal{L}(y'_i)$ and $y'_i \neq y'_j$ for all $1 \leq i < j \leq n$, since the $\leqslant$-rule can never merge two nodes y'_i, y'_j (because $y'_i \neq y'_j$), and, whenever an application of the $\leqslant$-rule sets $\mathcal{L}(\langle x, y'_i\rangle)$ to $\emptyset$, then there will be some S-neighbour z of x with $C \in \mathcal{L}(z)$ and z "inherits" all inequalities from y'_i.
  Since $clos(D)$ contains a total of at most m $\exists R.C$ and $(\geqslant n\ S\ C)$ concepts, the out-degree of the tree is bounded by $m \cdot n_{max}$.
- Nodes are labelled with non-empty subsets of $clos(D)$ and edges with subsets of R_D, so there are at most 2^{2mk} different possible labellings for a pair of nodes and an edge. Therefore, if a path p is of length at least 2^{2mk}, then from the pair-wise blocking condition there must be two nodes x, y on p 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}. □

Soundness Let $\mathbf{T}$ be a complete and clash-free completion tree. A path is a sequence of pairs of nodes of $\mathbf{T}$ of the form $p = [\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}]$. For such a path we define $\mathsf{Tail}(p) := x_n$ and $\mathsf{Tail}'(p) := x'_n$. With $[p|\frac{x_{n+1}}{x'_{n+1}}]$ we denote the path $[\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}, \frac{x_{n+1}}{x'_{n+1}}]$. The set $\mathsf{Paths}(\mathbf{T})$ is defined inductively as follows:

- For the root node x_0 of $\mathbf{T}$, $[\frac{x_0}{x_0}] \in \mathsf{Paths}(\mathbf{T})$, and
- For a path $p \in \mathsf{Paths}(\mathbf{T})$ and a node z in $\mathbf{T}$:
  - if z is a successor of $\mathsf{Tail}(p)$ and z is not blocked, then $[p|\frac{z}{z}] \in \mathsf{Paths}(\mathbf{T})$, or
  - if, for some node y in $\mathbf{T}$, y is a successor of $\mathsf{Tail}(p)$ and z blocks y, then $[p|\frac{z}{y}] \in \mathsf{Paths}(\mathbf{T})$.

Please note that, due to the construction of Paths, for $p \in \mathsf{Paths}(\mathbf{T})$ with $p = [p'|\frac{x}{x'}]$, we have that x is not blocked, x' is blocked iff $x \neq x'$, and x' is never indirectly blocked. Furthermore, $\mathcal{L}(x) = \mathcal{L}(x')$ holds.

Now we can define a tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ with:

$$\begin{aligned}
\mathbf{S} &= \mathsf{Paths}(\mathbf{T}) \\
\mathcal{L}(p) &= \mathcal{L}(\mathsf{Tail}(p)) \\
\mathcal{E}(R) &= \{\langle p, q\rangle \in \mathbf{S} \times \mathbf{S} \mid \begin{array}[t]{l} \text{Either } q = [p|\frac{x}{x'}] \text{ and} \\ \quad x' \text{ is an } R\text{-successor of } \mathsf{Tail}(p) \\ \text{or } p = [q|\frac{x}{x'}] \text{ and} \\ \quad x' \text{ is an } \mathsf{Inv}(R)\text{-successor of } \mathsf{Tail}(q)\}. \end{array}
\end{aligned}$$

Claim: T is a tableau for D with respect to $\mathcal{R}^+$.

We show that T satisfies all the properties from Definition 3.

- $D \in \mathcal{L}([\frac{x_0}{x_0}])$ since $D \in \mathcal{L}(x_0)$.
- **Property 1** holds because $\mathbf{T}$ is clash-free; **Properties 2,3** hold because $\mathsf{Tail}(p)$ is not blocked and $\mathbf{T}$ is complete.
- **Property 4**: Assume $\forall S.C \in \mathcal{L}(p)$ and $\langle p, q\rangle \in \mathcal{E}(S)$. If $q = [p|\frac{x}{x'}]$, then x' is an S-successor of $\mathsf{Tail}(p)$ and thus $C \in \mathcal{L}(x')$ (because the $\forall$-rule is not applicable). Since $\mathcal{L}(q) = \mathcal{L}(x) = \mathcal{L}(x')$, we have $C \in \mathcal{L}(q)$. If $p = [q|\frac{x}{x'}]$, then x' is an $\mathsf{Inv}(S)$-successor of $\mathsf{Tail}(q)$ and thus $C \in \mathcal{L}(\mathsf{Tail}(q))$ (because x' is not indirectly blocked and the $\forall$-rule is not applicable), hence $C \in \mathcal{L}(q)$.
- **Property 5**: Assume $\exists S.C \in \mathcal{L}(p)$. Define $x := \mathsf{Tail}(p)$. In $\mathbf{T}$ there is an S-neighbour y of x with $C \in \mathcal{L}(y)$, because the $\exists$-rule is not applicable. There are two possibilities:
  - y is a successor of x in $\mathbf{T}$. If y is not blocked, then $q := [p|\frac{y}{y}] \in \mathbf{S}$ and $\langle p, q\rangle \in \mathcal{E}(S)$ as well as $C \in \mathcal{L}(q)$. If y is blocked by some node z in $\mathbf{T}$, then $q := [p|\frac{z}{y}] \in \mathbf{S}$.
  - y is a predecessor of x. Again, there are two possibilities:
    - p is of the form $p = [q|\frac{x}{x'}]$ with $\mathsf{Tail}(q) = y$.
    - p is of the form $p = [q|\frac{x}{x'}]$ with $\mathsf{Tail}(q) = u \neq y$. x only has one predecessor in $\mathbf{T}$, hence u is not the predecessor of x. This implies $x \neq x'$, x blocks x' in $\mathbf{T}$, and u is the predecessor of x' due to the construction of Paths. Together with the definition of the blocking condition, this implies $\mathcal{L}(\langle u, x'\rangle) = \mathcal{L}(\langle y, x\rangle)$ as well as $\mathcal{L}(u) = \mathcal{L}(y)$ due to the pair-wise blocking condition.

  In all three cases, $\langle p, q\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}(q)$.

- **Property 6**: Assume $\forall S.C \in \mathcal{L}(p)$, $\langle p,q\rangle \in \mathcal{E}(R)$ for some $R \sqsubseteq^{*} S$ with $\mathsf{Trans}(R)$. If $q = [p|\frac{x}{x'}]$, then x' is an R-successor of $\mathsf{Tail}(p)$ and thus $\forall R.C \in \mathcal{L}(x')$ (because otherwise the $\forall_{+}$-rule would be applicable). From $\mathcal{L}(q) = \mathcal{L}(x) = \mathcal{L}(x')$ it follows that $\forall R.C \in \mathcal{L}(q)$. If $p = [q|\frac{x}{x'}]$, then x' is an $\mathsf{Inv}(S)$-successor of $\mathsf{Tail}(q)$ and hence $\mathsf{Tail}(q)$ is an R-neighbour of x'. Because x' is not indirectly blocked, this implies $\forall R.C \in \mathcal{L}(\mathsf{Tail}(q))$ and hence $\forall R.C \in \mathcal{L}(q)$.
- **Property 11**: Assume $(\bowtie\ n\ S\ C) \in \mathcal{L}(p)$, $\langle p,q\rangle \in \mathcal{E}(S)$. If $q = [p|\frac{x}{x'}]$, then x' is an S-successor of $\mathsf{Tail}(p)$ and thus $\{C, {\sim}C\} \cap \mathcal{L}(x') \neq \emptyset$ (since the *choose*-rule is not applicable). Since $\mathcal{L}(q) = \mathcal{L}(x) = \mathcal{L}(x')$, we have $\{C, {\sim}C\} \cap \mathcal{L}(q) \neq \emptyset$. If $p = [q|\frac{x}{x'}]$, then x' is an $\mathsf{Inv}(S)$-successor of $\mathsf{Tail}(q)$ and thus $\{C, {\sim}C\} \cap \mathcal{L}(\mathsf{Tail}(q)) \neq \emptyset$ (since x' is not indirectly blocked and the *choose*-rule is not applicable), hence $\{C, {\sim}C\} \cap \mathcal{L}(q) \neq \emptyset$.
- Assume **Property 9** is violated. Hence there is some $p \in \mathbf{S}$ with $(\leqslant n\ S\ C) \in \mathcal{L}(p)$ and $\sharp S^{T}(p, C) > n$. We show that this implies $\sharp S^{\mathbf{T}}(\mathsf{Tail}(p), C) > n$, in contradiction of either the clash-freeness or completeness of $\mathbf{T}$. Define $x := \mathsf{Tail}(p)$ and $P := S^{T}(p, C)$. Due to the assumption, we have $\sharp P > n$. We distinguish two cases:
  - P contains only paths of the form $q = [p|\frac{y}{y'}]$. We claim that the function Tail' is injective on P. Assume that there are two paths $q_1, q_1 \in P$ with $q_1 \neq q_2$ and $\mathsf{Tail}'(q_1) = \mathsf{Tail}'(q_2) = y'$. Then q_1 is of the form $q_1 = [p|(y_1, y')]$ and q_2 is of the form $q_2 = [p|\frac{y_2}{y'}]$ with $y_1 \neq y_2$. If y' is not blocked in $\mathbf{T}$, then $y_1 = y' = y_2$, contradicting $y_1 \neq y_2$. If y' is blocked in $\mathbf{T}$, then both y_1 and y_2 block y', which implies $y_1 = y_2$, again a contradiction.
    Since Tail' is injective on P, it holds that $\sharp P = \sharp\, \mathsf{Tail}'(P)$. Also for each $y' \in \mathsf{Tail}'(P)$, y' is an S-successor of x and $C \in \mathcal{L}(y')$. This implies $\sharp S^{\mathbf{T}}(x, C) > n$.
  - P contains a path q where p is of the form $p = [q|\frac{x}{x'}]$. Obviously, P may only contain one such path. As in the previous case, Tail' is an injective function on the set $P' := P \setminus \{q\}$, each $y' \in \mathsf{Tail}'(P')$ is an S-successor of x and $C \in \mathcal{L}(y')$ for each $y' \in \mathsf{Tail}'(P')$. To show that indeed $\sharp S^{\mathbf{T}}(x, C) > n$ holds, we have to prove the existence of a further S-neighbour u of x with $C \in \mathcal{L}(u)$ and $u \notin \mathsf{Tail}'(P')$. This will be "supplied" by $z := \mathsf{Tail}(q)$. We distinguish two cases:
    - $x = x'$. Hence x is not blocked. This implies that x is an $\mathsf{Inv}(S)$-successor of z in $\mathbf{T}$. Since $\mathsf{Tail}'(P')$ contains only successors of x, we have that $z \notin \mathsf{Tail}'(P')$ and, by construction, z is an S-neighbour of x with $C \in \mathcal{L}(z)$.
    - $x \neq x'$. This implies that x' is blocked in $\mathbf{T}$ by x and that x' is an $\mathsf{Inv}(S)$-successor of z in $\mathbf{T}$. The definition of pairwise-blocking implies that x is an $\mathsf{Inv}(S)$-successor of some node u in $\mathbf{T}$ with $\mathcal{L}(u) = \mathcal{L}(z)$. Again, since $\mathsf{Tail}'(P')$ contains only successors of x we have that $u \notin \mathsf{Tail}'(P')$ and, by construction, u is an S-neighbour of x and $C \in \mathcal{L}(u)$.

- **Property 10**: Assume $(\geqslant n\ S\ C) \in \mathcal{L}(p)$. Completeness of $\mathbf{T}$ implies that there exist n individuals $y_1, \ldots, y_n$ in $\mathbf{T}$ such that each y_i is an S-neighbour of $\mathsf{Tail}(p)$ and $C \in \mathcal{L}(y_i)$. We claim that, for each of these individuals, there is a path q_i such that $\langle p, q_i \rangle \in \mathcal{E}(S)$, $C \in \mathcal{L}(q_i)$, and $q_i \neq q_j$ for all $1 \leq i < j \leq n$. Obviously, this implies $\sharp S^T(p, C) \geqslant n$. For each y_i there are three possibilities:
  - y_i is an S-successor of x and y_i is not blocked in $\mathbf{T}$. Then $q_i = [p|\frac{y_i}{y_i}]$ is a path with the desired properties.
  - y_i is an S-successor of x and y_i is blocked in $\mathbf{T}$ by some node z. Then $q_i = [p|\frac{z}{y_i}]$ is the path with the desired properties. Since the same z may block several of the y_js, it is indeed necessary to include y_i explicitly into the path to make them distinct.
  - x is an $\mathsf{Inv}(S)$-successor of y_i. There may be at most one such y_i. This implies that p is of the form $p = [q|\frac{x}{x'}]$ with $\mathsf{Tail}(q) = y_i$. Again, q has the desired properties and, obviously, q is distinct from all other paths q_j.
- **Property 7** is satisfied due to the symmetric definition of $\mathcal{E}$. **Property 8** is satisfied due to the definition of R-successor that takes into account the role hierarchy $\underline{\circledast}$. $\square$

Completeness Let $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ be a tableau for D w.r.t. $\mathcal{R}^+$. We use this tableau to guide the application of the non-deterministic rules. To do this, we will inductively define a function π, mapping the individuals of the tree $\mathbf{T}$ to $\mathbf{S}$ such that, for each x, y in $\mathbf{T}$:

$$\left.\begin{array}{l}\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x)) \\ \text{if } y \text{ is an } S\text{-neighbour of } x, \text{ then } \langle \pi(x), \pi(y) \rangle \in \mathcal{E}(S) \\ x \neq y \text{ implies } \pi(x) \neq \pi(y)\end{array}\right\} (*)$$

Claim: Let $\mathbf{T}$ be a completion-tree and π a function that satisfies $(*)$. If a rule is applicable to $\mathbf{T}$ then the rule is applicable to $\mathbf{T}$ in a way that yields a completion-tree $\mathbf{T}'$ and an extension of π that satisfy $(*)$.

Let $\mathbf{T}$ be a completion-tree and π be a function that satisfies $(*)$. We have to consider the various rules.

- **The $\sqcap$-rule:** If $C_1 \sqcap C_2 \in \mathcal{L}(x)$, then $C_1 \sqcap C_2 \in \mathcal{L}(\pi(x))$. This implies $C_1, C_2 \in \mathcal{L}(\pi(x))$ due to Property 2 from Definition 3, and hence the rule can be applied without violating $(*)$.
- **The $\sqcup$-rule:** If $C_1 \sqcup C_2 \in \mathcal{L}(x)$, then $C_1 \sqcup C_2 \in \mathcal{L}(\pi(x))$. Since T is a tableau, Property 3 from Definition 3 implies $\{C_1, C_2\} \cap \mathcal{L}(\pi(x)) \neq \emptyset$. Hence the $\sqcup$-rule can add a concept $E \in \{C_1, C_2\}$ to $\mathcal{L}(x)$ such that $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$ holds.
- **The $\exists$-rule:** If $\exists S.C \in \mathcal{L}(x)$, then $\exists S.C \in \mathcal{L}(\pi(x))$ and, since T is a tableau, Property 5 of Definition 3 implies that there is an element $t \in \mathbf{S}$ such that $\langle \pi(x), t \rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}(t)$. The application of the $\exists$-rule generates a new variable y with $\mathcal{L}(\langle x, y \rangle = \{S\}$ and $\mathcal{L}(y) = \{C\}$. Hence we set $\pi := \pi[y \mapsto t]$ which yields a function that satisfies $(*)$ for the modified tree.

– **The $\forall$-rule:** If $\forall S.C \in \mathcal{L}(x)$, then $\forall S.C \in \mathcal{L}(\pi(x))$, and if y is an S-neighbour of x, then also $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}(S)$ due to $(*)$. Since T is a tableau, Property 4 of Definition 3 implies $C \in \mathcal{L}(\pi(y))$ and hence the $\forall$-rule can be applied without violating $(*)$.
– **The $\forall_+$-rule:** If $\forall S.C \in \mathcal{L}(x)$, then $\forall S.C \in \mathcal{L}(\pi(x))$, and if there is some $R \sqsubseteq\!\!\!*\, S$ with $\mathsf{Trans}(R)$ and y is an R-neighbour of x, then also $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}(R)$ due to $(*)$. Since T is a tableau, Property 6 of Definition 3 implies $\forall R.C \in \mathcal{L}(\pi(y))$ and hence the $\forall_+$-rule can be applied without violating $(*)$.
– **The *choose*-rule:** If $(\bowtie\ n\ S\ C) \in \mathcal{L}(x)$, then $(\bowtie\ n\ S\ C) \in \mathcal{L}(\pi(x))$, and, if there is an S-neighbour y of x, then $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}(S)$ due to $(*)$. Since T is a tableau, Property 11 of Definition 3 implies $\{C, \sim C\} \cap \mathcal{L}(\pi(y) \neq \emptyset$. Hence the *choose*-rule can add an appropriate concept $E \in \{C, \sim C\}$ to $\mathcal{L}(x)$ such that $\mathcal{L}(y) \subseteq \mathcal{L}(\pi(y))$ holds.
– **The $\geqslant$-rule:** If $(\geqslant n\ S\ C) \in \mathcal{L}(x)$, then $(\geqslant n\ S\ C) \in \mathcal{L}(\pi(x))$. Since T is a tableau, Property 10 of Definition 3 implies $\sharp S^T(\pi(x), C) \geqslant n$. Hence there are individuals $t_1, \ldots, t_n \in \mathbf{S}$ such that $\langle \pi(x), t_i \rangle \in \mathcal{E}(S)$, $C \in \mathcal{L}(t_i)$, and $t_i \neq t_j$ for $1 \leq i < j \leq n$. The $\geqslant$-rule generates n new nodes $y_1, \ldots, y_n$. By setting $\pi := \pi[y_1 \mapsto t_1, \cdots y_n \mapsto t_n]$, one obtains a function π that satisfies $(*)$ for the modified tree.
– **The $\leqslant$-rule:** If $(\leqslant n\ S\ C) \in \mathcal{L}(x)$, then $(\leqslant n\ S\ C) \in \mathcal{L}(\pi(x))$. Since T is a tableau, Property 9 of Definition 3 implies $\sharp S^T(\pi(x), C) \leqslant n$. If the $\leqslant$-rule is applicable, we have $\sharp S^{\mathbf{T}}(x, C) > n$, which implies that there are at least $n+1$ S-neighbours $y_0, \ldots, y_n$ of x such that $C \in \mathcal{L}(y_i)$. Thus, there must be two nodes $y, z \in \{y_0, \ldots, y_n\}$ such that $\pi(y) = \pi(z)$ (because otherwise $\sharp S^T(\pi(x), C) > n$ would hold). From $\pi(y) = \pi(z)$ we have that $y \not\neq z$ cannot hold because of $(*)$, and y, z can be chosen such that y is not an ancestor of z. Hence the $\leqslant$-rule can be applied without violating $(*)$.

Why does this claim yield the completeness of the tableaux algorithm? For the initial completion-tree consisting of a single node x_0 with $\mathcal{L}(x_0) = \{D\}$ and $\not\neq = \emptyset$ we can give a function π that satisfies $(*)$ by setting $\pi(x_0) := s_0$ for some $s_0 \in \mathbf{S}$ with $D \in \mathcal{L}(s_0)$ (such an s_0 exists since T is a tableau for D). Whenever a rule is applicable to $\mathbf{T}$, it can be applied in a way that maintains $(*)$, and, since the algorithm terminates, we have that any sequence of rule applications must terminate. Properties $(*)$ imply that any tree $\mathbf{T}$ generated by these rule-applications must be clash-free as there are only two possibilities for a clash, and it is easy to see that neither of these can hold in $\mathbf{T}$:

– $\mathbf{T}$ cannot contain a node x such that $\{C, \neg C\} \in \mathcal{L}(x)$ because $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$ and hence Property 1 of Definition 3 would be violated for $\pi(x)$.
– $\mathbf{T}$ cannot contain a node x with $(\leqslant n\ S\ C) \in \mathcal{L}(x)$ and $n+1$ S-neighbours $y_0, \ldots y_n$ of x with $C \in \mathcal{L}(y_i)$ and $y_i \not\neq y_j$ for $0 \leq i < j \leq n$ because $(\leqslant n\ S\ C) \in \mathcal{L}(\pi(x))$, and, since $y_i \not\neq y_j$ implies $\pi(y_i) \neq \pi(y_j)$, $\sharp S^T(\pi(x), C) > n$, in contradiction to Property 9 of Definition 3. $\square$