RWTH
LTCS–Report

Aachen University of Technology
Research group for
Theoretical Computer Science

Optimised Reasoning for $\mathcal{SHIQ}$

Ian Horrocks and Ulrike Sattler

LTCS-Report 01-08

RWTH Aachen
LuFg Theoretische Informatik
http://www-lti.informatik.rwth-aachen.de

Ahornstr. 55
52074 Aachen
Germany

Optimised Reasoning for $\mathcal{SHIQ}$

Ian Horrocks and Ulrike Sattler

November 6, 2001

Abstract

The tableau algorithm implemented in the FaCT knowledge representation system decides satisfiability and subsumption in $\mathcal{SHIQ}$, a very expressive description logic providing, e.g., inverse and transitive roles, number restrictions, and general axioms. Intuitively, the algorithm searches for a tree-shaped abstraction of a model. To ensure termination of this algorithm without compromising correctness, it stops expanding paths in the search tree using a so-called "double-blocking" condition.

This condition was clearly more exacting than was strictly necessary, but it was assumed that more precisely defined blocking conditions would, on the one hand, make the proof of the algorithm's correctness far more difficult and, on the other hand (and more importantly), be so costly to check as to outweigh any benefit that might be derived.

However, FaCT's failure to solve UML derived knowledge bases led us to reconsider this conjecture and to formulate more precisely defined blocking conditions. We prove that the revised algorithm is still sound and complete, and demonstrate that it greatly improves FaCT's performance—in some cases by more than two orders of magnitude.

Contents

1 Introduction

Description Logics (DLs) form a family of knowledge representation formalisms designed for the representation of and reasoning about *terminological knowledge*. They can be viewed as offsprings of semantic networks and frame-based systems, whose development was motivated by the insight that such systems need a well-defined, implementation-independent semantics. A first attempt towards this goal was made by KL-ONE [BS85], a successful and highly influential knowledge representation system.

The two main inference problems addressed by KL-ONE were *subsumption* between pairs of concepts, which was used to arrange the concepts defined in a knowledge base into a taxonomy, and *satisfiability* of single concepts, which was used to check the consistency of the knowledge base. Unfortunately, when the underlying representational formalism was studied in detail, it turned out that the above mentioned inference problems were undecidable [Sch89]. It might be argued that semi-decidability is fine for other applications, and thus could be tolerated, but since subsumption can be reduced to *un*satisfiability and satisfiability to *non*-subsumption, one of both problems would always be truly undecidable.

Following this observation, the developers of the CLASSIC system from AT&T [BBMAR89] decided that the DL underlying CLASSIC should not only be decidable, but be *realistically* decidable, i.e., they wanted the corresponding inference problems to be decidable in polynomial time. Thus they severely restricted the expressive power of their DL, and designed a (sub-Boolean) DL with tractable, sound, and complete inference algorithms.

In parallel, the computational complexity of a variety of DLs was investigated, and it turned out that the inference problems of (almost all) DLs with interesting expressive power were at least PSPACE-complete [DLNdN91], i.e., of a complexity apparently far too high to be practicable. Despite this discouraging assessment with regard to *worst case* performance, several researchers implemented satisfiability/subsumption algorithms for such DLs [BFH$^+$94; BFT95], and developed sophisticated optimisation techniques designed to improve *typical case* performance. Surprisingly, these PSPACE algorithms proved amenable to optimisation and behaved well in practise—it was found that the pathological cases that lead to the high complexity of these DLs are so artificial that they rarely occur in practice [Neb90; HKNP94; SvRvdVM95].

In the late 90's, motivated by a medical terminology application which required even more expressive power, the DL system FaCT was implemented with an underlying DL (first $\mathcal{SHIF}$, later $\mathcal{SHIQ}$) which was of an even higher complexity, namely EXPTIME-complete [Hor98]. Interestingly, after thoughtful optimisations, this system showed the same behaviour as its predecessors, i.e., it behaved very well in practice. Other systems implementing EXPTIME-complete DLs were subsequently developed [HM01; PS99], and showed a similar behaviour—a phenomenon that lead part of the DL community to believe that, with knowledge bases stemming from realistic applications, "tractable" means "in EXPTIME".

At the same time, expressive DLs were shown to have useful applications in the database domain—in particular they were shown to be useful for reasoning about conceptual models of databases expressed, e.g., in extended entity-relationship diagrams or in UML [CLN98]. Roughly speaking, such a conceptual model can be translated into a DL knowledge base, possibly with the addition of further (integrity) constraints, and the inference services of a standard DL system can then be used to detect inconsistencies and implicit is-a links between classes, entities, or relations. This approach is especially useful when integrating databases or building data warehouses, and has been implemented in the ICOM tool for intelligent conceptual modelling [FN00]. Interestingly, this translation yields knowledge bases from realistic applications that could *not* be solved by any of the available DL systems [BCDG01], even though the UML diagrams that lead to these knowledge bases are relatively small and seemingly harmless.

In this paper, we report on an optimisation of the FaCT system that was inspired by the failure of state-of-the-art DL systems to handle these knowledge bases. Roughly speaking, FaCT performs a complete search of trees whose depth can be exponential in the size of the input. It uses back-tracking search and a cycle-detection mechanism called *blocking* that limits the tree depth (which could otherwise be infinite) to ensure termination without compromising soundness and completeness.

In order to deal with inverse roles and the possibility of concepts with only infinite models, the $\mathcal{SHIQ}$ algorithm implemented in FaCT introduced a new and more sophisticated "double-blocking" technique [HST99b]. The conditions required to trigger a "block" were more complex than in earlier tableaux algorithms for less expressive DLs, but were still provably correct (i.e., maintained soundness and completeness) and relatively easy to check. Although these conditions were more exacting than was strictly necessary, relaxing them would have significantly increased their complexity, making it harder to prove that they were still correct. Moreover, it seemed that the cost of checking more complex conditions would be prohibitive, and likely to outweigh any benefit that might derive from establishing blocks at a shallower depth.

An investigation of FaCT's behaviour when failing to solve UML derived knowledge bases has, however, lead us to reconsider this conjecture, to formulate a more detailed and less strict blocking condition and, as a matter of course, to prove that the modified algorithm is still sound and complete. The effect of the optimised blocking condition on FaCT's behaviour turned out to be dramatic—in some cases it improved the system's performance by more than two orders of magnitude. Clearly, the value of improved blocking should not be underestimated, even if the overhead seems considerable.

2 $\mathcal{SHIQ}$-Syntax, Semantics, and Tableaux

In this section, we define syntax and semantics of $\mathcal{SHIQ}$-concepts and roles. We start with $\mathcal{SHIQ}$-roles, then introduce some abbreviations, and finally define $\mathcal{SHIQ}$-concepts.

Definition 1 Let $\mathbf{R}$ be a set of *role names* with both transitive and normal role names $\mathbf{R}_+ \cup \mathbf{R}_\mathsf{P} = \mathbf{R}$, where $\mathbf{R}_\mathsf{P} \cap \mathbf{R}_+ = \emptyset$. The set of $\mathcal{SHIQ}$*-roles* is $\mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$. A *role inclusion axiom* is of the form

$$R \sqsubseteq S,$$

for two $\mathcal{SHIQ}$-roles R and S. A *role hierarchy* is a set of role inclusion axioms.

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 role to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that, for $P \in \mathbf{R}$ and $R \in \mathbf{R}_+$,

$$\langle x, y\rangle \in P^\mathcal{I} \text{ iff } \langle y, x\rangle \in P^{-\mathcal{I}},$$
$$\text{if } \langle x, y\rangle \in R^\mathcal{I} \text{ and } \langle y, z\rangle \in R^\mathcal{I}, \text{ then } \langle x, z\rangle \in R^\mathcal{I}.$$

An interpretation $\mathcal{I}$ *satisfies a role hierarchy* $\mathcal{R}$ iff $R^\mathcal{I} \subseteq S^\mathcal{I}$ for each $R \sqsubseteq S \in \mathcal{R}$; we denote this fact by $\mathcal{I} \models \mathcal{R}$.

We introduce some notation to make the following considerations easier.

1. The inverse relation on roles is symmetric, and to avoid considering roles such as R^{--}, we define a function Inv which returns the inverse of a role, more precisely

$$\mathsf{Inv}(R) := \begin{cases} R^- & \text{if } R \text{ is a role name,} \\ S & \text{if } R = S^- \text{ for a role name } S. \end{cases}$$

2. Since set inclusion is transitive and $R^\mathcal{I} \subseteq S^\mathcal{I}$ implies $\mathsf{Inv}(R)^\mathcal{I} \subseteq \mathsf{Inv}(S)^\mathcal{I}$, we introduce $\sqsubseteq^*$ as the transitive-reflexive closure of $\sqsubseteq$ on

$$\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}.$$

   We use $R \equiv S$ as an abbreviation for $R \sqsubseteq^* S$ and $S \sqsubseteq^* R$.

3. Obviously, a role R is transitive if and only if its inverse $\mathsf{Inv}(R)$ is transitive. However, either R or $\mathsf{Inv}(R)$ is a role name, and only role names can be elements of $\mathbf{R}_+$. Moreover, in cyclic cases such as $R \equiv S$, S is transitive if R or $\mathsf{Inv}(R)$ is a transitive role name. In order to avoid these case distinctions, the function Trans returns true iff R is a transitive role—regardless whether it is a role name, the inverse of a role name, or equivalent to a transitive role name (or its inverse):

$$\mathsf{Trans}(R) := \begin{cases} \text{true} & \text{if, for some } S \text{ with } S \equiv R,\ S \in \mathbf{R}_+ \text{ or } \mathsf{Inv}(S) \in \mathbf{R}_+ \\ \text{false} & \text{otherwise.} \end{cases}$$

We are now ready to define $\mathcal{SHIQ}$-concepts.

Definition 2 A role R is called *simple* with respect to $\mathcal{R}$ iff not $\mathsf{Trans}(R)$ and, for any $S \sqsubseteq^* R$, S is also a simple role.

Let N_C be a set of *concept names*. The set of $\mathcal{SHIQ}$*-concepts* is the smallest set such that

1. every concept name $C \in N_C$ is a concept,
2. if C and D are concepts and R is an $\mathcal{SHIQ}$-role, then $(C \sqcap D)$, $(C \sqcup D)$, $(\neg C)$, $(\forall R.C)$, and $(\exists R.C)$ are concepts, and
3. if C is a concept, R is a simple $\mathcal{SHIQ}$-role and $n \in \mathbb{N}$, then $(\leqslant n\ R\ C)$ and $(\geqslant n\ R\ C)$ are concepts.

The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps, additionally, every concept to a subset of $\Delta^{\mathcal{I}}$ such that

$$
\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}},\\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}},\\
\neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},\\
(\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \text{There is some } y \in \Delta^{\mathcal{I}} \text{ with } \langle x, y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\},\\
(\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \text{For all } y \in \Delta^{\mathcal{I}}, \text{ if } \langle x, y\rangle \in R^{\mathcal{I}}, \text{ then } y \in C^{\mathcal{I}}\},\\
(\leqslant n\ R\ C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \sharp R^{\mathcal{I}}(x, C) \leqslant n\},\\
(\geqslant n\ R\ C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \sharp R^{\mathcal{I}}(x, C) \geqslant n\},
\end{aligned}
$$

where for a set M we denote the cardinality of M by $\sharp M$ and $R^{\mathcal{I}}(x, C)$ is defined as $\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$.

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 with respect to $\mathcal{R}$. A concept D *subsumes* a concept C with respect to $\mathcal{R}$ (written $C \sqsubseteq_{\mathcal{R}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{R}$. Two concepts C, D are *equivalent* with respect to $\mathcal{R}$ (written $C \equiv_{\mathcal{R}} D$) iff they are mutually subsuming. For an interpretation $\mathcal{I}$, an individual $x \in \Delta^{\mathcal{I}}$ is called an *instance* of a concept C iff $x \in C^{\mathcal{I}}$.

2.1 A $\mathcal{SHIQ}$-Tableau

As usual, we define appropriate abstractions of models, *tableaux*, whose existence can be tested by a tableaux algorithm. The advantage of this abstraction is that they allow to replace the "global" condition of the interpretation of transitive roles into "local" conditions.

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 using a combination of DeMorgan's laws and the following equivalences:

$$
\begin{array}{rcl}
\neg(\exists R.C) & \equiv & (\forall R.\neg C) \\
\neg(\forall R.C) & \equiv & (\exists R.\neg C) \\
\neg(\leqslant n\ R\ C) & \equiv & (\geqslant (n+1)\ R\ C) \\
\neg(\geqslant n\ R\ C) & \equiv & \begin{cases} (\forall R.\neg C) & \text{if } n = 1 \\ (\leqslant (n-1)\ R\ C) & \text{otherwise} \end{cases}
\end{array}
$$

For a concept C we will denote the NNF of $\neg C$ by $\sim C$.

For a $\mathcal{SHIQ}$-concept D in NNF and a role hierarchy, we define $clos(D)$ to be the smallest set that contains D, is closed under sub-formulae and $\sim$, and which contains, for each subconcept $\forall R.C \in clos(D)$ and role $R' \sqsubseteq^* R$, also the concept $\forall R'.C$. Then $\#clos(D)$ is linear in $|D| + |\mathcal{R}|$.

Definition 3 If $\mathcal{R}$ is a role hierarchy, D is a $\mathcal{SHIQ}$-concept in NNF and $\mathbf{R}_D$ is the set of roles occurring in D, together with their inverses, a tableau T for D with respect to $\mathcal{R}$ is defined to be a triple $(\mathbf{S}, \mathcal{L}, \mathcal{E})$ such that: $\mathbf{S}$ is a set of individuals, $\mathcal{L} : \mathbf{S} \to 2^{clos(D)}$ maps each individual to a set of concepts which is a subset of $clos(D)$, $\mathcal{E} : \mathbf{R}_D \to 2^{\mathbf{S}\times\mathbf{S}}$ maps each role in $\mathbf{R}_D$ to a set of pairs of individuals, and there is some individual $s \in \mathbf{S}$ such that $D \in \mathcal{L}(s)$. For all $s \in \mathbf{S}$, $C, C_1, C_2 \in clos(D)$, and $R, S \in \mathbf{R}_D$, T must satisfy:

T1 if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$,

T2 if $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,

T3 if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,

T4 if $\forall S.C \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(S)$, then $C \in \mathcal{L}(t)$,

T5 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 $C \in \mathcal{L}(t)$,

T6 if $\forall S.C \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(R)$ for some $R \sqsubseteq^* S$ with $\mathsf{Trans}(R)$, then $\forall R.C \in \mathcal{L}(t)$,

T7 $\langle s, t\rangle \in \mathcal{E}(R)$ iff $\langle t, s\rangle \in \mathcal{E}(\mathsf{Inv}(R))$,

T8 if $\langle s, t\rangle \in \mathcal{E}(R)$ and $R \sqsubseteq^* S$ then $\langle s, t\rangle \in \mathcal{E}(S)$,

T9 if $(\leqslant n\ S\ C) \in \mathcal{L}(s)$, then $\sharp S^T(s, C) \leqslant n$,

T10 if $(\geqslant n\ S\ C) \in \mathcal{L}(s)$, then $\sharp S^T(s, C) \geqslant n$,

T11 if $(\leqslant n\ S\ C) \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(S)$ then $C \in \mathcal{L}(t)$ or $\sim C \in \mathcal{L}(t)$,

for

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

Lemma 1 *A $\mathcal{SHIQ}$-concept D is satisfiable with respect to a role hierarchy $\mathcal{R}$ iff there exists a tableau for D with respect to $\mathcal{R}$.*

Proof: For the *if* direction, the construction of a model of D from a tableau for D is similar to the one presented in [HST99a]. If $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a tableau for D with $D \in \mathcal{L}(s_0)$, a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of D can be defined as follows:

$$\begin{array}{lcl} \Delta^{\mathcal{I}} & = & \mathbf{S} \\ A^{\mathcal{I}} & = & \{s \mid A \in \mathcal{L}(s)\} \text{ for all concept names A in } \mathit{clos}(D) \\ R^{\mathcal{I}} & = & \mathcal{E}(R) \cup \bigcup_{P \sqsubseteq^{*} R,\ \mathsf{Trans}(R)} \mathcal{E}(P)^{+} \quad \text{for role names } R \end{array}$$

From the definition of $R^{\mathcal{I}}$, T7, and T8, it follows that, if $\langle s,t\rangle \in S^{\mathcal{I}}$, then either $\langle s,t\rangle \in \mathcal{E}(S)$ or there exists a path $\langle s,s_1\rangle, \langle s_1,s_2\rangle, \ldots, \langle s_n,t\rangle \in \mathcal{E}(R)$ for some R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^{*} S$.

To show that $\mathcal{I}$ is a model of D w.r.t. $\mathcal{R}$, we have to prove (1) $\mathcal{I} \models \mathcal{R}$ and (2) $D^{\mathcal{I}} \neq \emptyset$. The first part is obvious due to T8 and the definition of $\mathsf{Trans}(\cdot)$ and $\cdot^{\mathcal{I}}$. The second part is shown by proving $C \in \mathcal{L}(s) \Rightarrow s \in C^{\mathcal{I}}$ for any $s \in \mathbf{S}$. This implies $D^{\mathcal{I}} \neq \emptyset$ since T is a tableau for D and hence there must be some $s \in \mathbf{S}$ with $D \in \mathcal{L}(s)$.

This will be proven by induction over *norm* $\|\cdot\|$ of a concept C. The norm $\|C\|$ for concept in NNF is inductively defined as follows:

$$\begin{array}{llllll} \|A\| & := & \|\neg A\| & := & 0 & \text{for } A \in N_C \\ \|C_1 \sqcap C_2\| & := & \|C_1 \sqcup C_2\| & := & 1 + \|C_1\| + \|C_2\| & \\ \|\forall R.C\| & := & \|\exists R.C\| & := & 1 + \|C\| & \\ \|(\bowtie\ n\ S\ C)\| & & & := & 1 + \|C\|, & \end{array}$$

where we use $\bowtie$ as a placeholder for both $\leqslant$ and $\geqslant$. The two base cases of the induction are $C = A$ or $C = \neg A$. If $A \in \mathcal{L}(s)$, then, by definition, $s \in A^{\mathcal{I}}$. If $\neg A \in \mathcal{L}(s)$, then by T1, $A \notin \mathcal{L}(s)$ and hence $s \notin A^{\mathcal{I}}$. For the induction step we have to distinguish several cases:

- $C = C_1 \sqcap C_2$. T2 and $C \in \mathcal{L}(s)$ imply that $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$. Hence, by induction, we have $s \in C_1^{\mathcal{I}}$ and $s \in C_2^{\mathcal{I}}$, which yields $s \in (C_1 \sqcap C_2)^{\mathcal{I}}$.
- $C = C_1 \sqcup C_2$. Similar to the previous case.
- $C = \exists S.E$. T5 and $C \in \mathcal{L}(s)$ implies the existence of an individual $t \in \mathbf{S}$ such that $\langle s,t\rangle \in \mathcal{E}(S)$ and $E \in \mathcal{L}(t)$. By induction, we have $t \in E^{\mathcal{I}}$ and, from the definition of $S^{\mathcal{I}}$ and T7, it follows that $\langle s,t\rangle \in S^{\mathcal{I}}$ and hence $s \in C^{\mathcal{I}}$.
- $C = \forall S.E$. Let $s \in \mathbf{S}$ with $C \in \mathcal{L}(s)$, let $t \in \mathbf{S}$ be an arbitrary individual such that $\langle s,t\rangle \in S^{\mathcal{I}}$. There are two possibilities:
  - $\langle s,t\rangle \in \mathcal{E}(S)$. Then T4 implies $E \in \mathcal{L}(t)$ and, by induction, $t \in E^{\mathcal{I}}$.

– $\langle s,t\rangle \notin \mathcal{E}(S)$. Then there exists a path $\langle s,s_1\rangle, \langle s_1,s_2\rangle, \ldots, \langle s_n,t\rangle \in \mathcal{E}(R)$ for some R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$. Then $\mathsf{T6}$ implies $\forall R.E \in \mathcal{L}(s_i)$ for all $1 \leq i \leq n$ and, from $\mathsf{T4}$, $E \in \mathcal{L}(t)$ also holds. Again, by induction, this implies $t \in E^{\mathcal{I}}$.

In both cases, we have $t \in E^{\mathcal{I}}$ and, since t has been chosen arbitrarily, $s \in C^{\mathcal{I}}$ holds.

- $C = (\geqslant n\ S\ E)$. For an s with $C \in \mathcal{L}(s)$, we have $\sharp S^T(s,E) \geqslant n$. Hence there are n individuals $t_1,\ldots,t_n$ such that $t_i \neq t_j$ for $i \neq j$, $\langle s,t_i\rangle \in \mathcal{E}(S)$, and $E \in \mathcal{L}(t_i)$ for all i. By induction, we have $t_i \in E^{\mathcal{I}}$ and, since $\mathcal{E}(S) \subseteq S^{\mathcal{I}}$, also $s \in C^{\mathcal{I}}$.

- $C = (\leqslant m\ S\ E)$. For this case, we need that S is a simple role, which implies $S^{\mathcal{I}} = \mathcal{E}(S)$. Let s be an individual with $C \in \mathcal{L}(s)$. Due to $\mathsf{T11}$, we have $E \in \mathcal{L}(t)$ or $\sim E \in \mathcal{L}(t)$ for each t with $\langle s,t\rangle \in \mathcal{E}(S)$. Moreover, $\sharp S^T(s,E) \leqslant n$ holds due to $\mathsf{T9}$. We can show that $\sharp S^{\mathcal{I}}(s,E) \leqslant \sharp S^T(s,E)$: assume $\sharp S^{\mathcal{I}}(s,E) > \sharp S^T(s,E)$. This implies the existence of some t with $\langle s,t\rangle \in S^{\mathcal{I}}$ with $t \in E^{\mathcal{I}}$ but $E \notin \mathcal{L}(t)$ (because $S^{\mathcal{I}} = \mathcal{E}(S)$). By $\mathsf{T11}$ this implies $\sim E \in \mathcal{L}(t)$, which, by induction yields $t \in (\sim E)^{\mathcal{I}}$, in contradiction to $t \in E^{\mathcal{I}}$.

For the *only-if* direction, we have to show that satisfiability of D with respect to $\mathcal{R}$ implies the existence of a tableau T for D with respect to $\mathcal{R}$.

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model of D with $\mathcal{I} \models \mathcal{R}$. A tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for D can be defined as follows:

$$\begin{aligned} \mathbf{S} &= \Delta^{\mathcal{I}} \\ \mathcal{E}(R) &= R^{\mathcal{I}} \\ \mathcal{L}(s) &= \{C \in \mathit{clos}(D) \mid s \in C^{\mathcal{I}}\} \end{aligned}$$

It remains to demonstrate that T is a tableau for D:

- The Properties $\mathsf{T1}$–$\mathsf{T5}$, $\mathsf{T7}$, and $\mathsf{T9}$–$\mathsf{T11}$ are satisfied as a direct consequence of the definition of the semantics of $\mathcal{SHIQ}$-concepts.

- If $s \in (\forall S.C)^{\mathcal{I}}$ and $\langle s,t\rangle \in R^{\mathcal{I}}$ for R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$, then $t \in (\forall R.C)^{\mathcal{I}}$ unless there is some u such that $\langle t,u\rangle \in R^{\mathcal{I}}$ and $u \notin C^{\mathcal{I}}$. In this case, if $\langle s,t\rangle \in R^{\mathcal{I}}$, $\langle t,u\rangle \in R^{\mathcal{I}}$ and $\mathsf{Trans}(R)$, then $\langle s,u\rangle \in R^{\mathcal{I}}$. Hence $\langle s,u\rangle \in S^{\mathcal{I}}$ and $s \notin (\forall S.C)^{\mathcal{I}}$—in contradiction to the assumption. T therefore satisfies $\mathsf{T6}$.

- $\mathsf{T8}$ is satisfied because $\mathcal{I} \models \mathcal{R}$. ∎

3 An optimised blocking condition for $\mathcal{SHIQ}$

In this section, we present an optimised version of the tableaux algorithm for $\mathcal{SHIQ}$ from [HST99a]. The optimisation concerns the blocking condition, i.e., the mechanism that guarantees termination of the algorithm by preventing it from non-termination.

3.1 Constructing a $\mathcal{SHIQ}$-Tableau

From Lemma 1, an algorithm which constructs a tableau for a $\mathcal{SHIQ}$-concept D can be used as a decision procedure for the satisfiability of D with respect to a role hierarchy $\mathcal{R}$. Such an algorithm will now be described in detail. It uses the same techniques as the $\mathcal{SHIQ}$-algorithm in [HST99a] but for the modified pairwise-blocking condition.

The algorithm presented here tries to construct, for an input concept C_0, a tableau whose relational structure forms a tree with C_0 in the label of the root node. We must take special care to prevent the algorithm from generating a tree with arbitrarily long paths. In the original algorithm, we introduced a so-called *double blocking condition*. Roughly speaking, if we find two nodes on a path, a node x and its successor y such that they have two ancestor nodes, again, a node x' and its successor y' such that (1) x and x' are labelled with the same concepts, (2) y and y' are labelled with the same concepts, and (3) the relations between x and y are the same as those between x' and y', then this path is no longer modified below y, i.e., it cannot become longer. Now, this three-fold condition is a rather strict one, e.g., the root node can never block another node, and thus blocking occurs rather late, which means that paths can become rather long.

In the following, we will loosen this condition such that blocking can occur earlier. Basically, we will restrict, in the conditions (1) and (2), the concepts to the relevant ones and, in condition (3), the relations to the relevant ones.

Moreover, to guarantee the termination of the algorithm, we have to make sure that the $\geqslant$- and $\leqslant$-rules cannot be applied in a way that would yield an infinite sequence of rule applications. This is enforced by recording which nodes have been introduced by an application of the $\geqslant$-rule and by prohibiting an identification of these nodes by the $\leqslant$-rule.

Definition 4 Let $\mathcal{R}$ be a role hierarchy and D a $\mathcal{SHIQ}$-concept in NNF. A *completion tree* with respect to $\mathcal{R}$ and D is a tree $\mathbf{T}$ where each node x of the tree is labelled with a set $\mathcal{L}(x) \subseteq \mathit{clos}(D)$ and each edge $\langle x, y\rangle$ is labelled with a set of role names $\mathcal{L}(\langle x, y\rangle)$ containing (possibly inverse) roles occurring in $\mathit{clos}(D)$ or $\mathcal{R}$. Additionally, we keep track of inequalities between nodes of the tree with a symmetric binary relation $\neq$ between the nodes of $\mathbf{T}$.

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

For a role S, a concept C, and a node x in $\mathbf{T}$ we define $S^{\mathbf{T}}(x, C)$ by

$$S^{\mathbf{T}}(x, C) := \{y \mid y \text{ is } S\text{-neighbour of } x \text{ and } C \in \mathcal{L}(y)\}.$$

A node is *blocked* if it is *directly* or *indirectly* blocked. A node is *directly blocked* if it is *c-blocked* or *a-blocked*.[1] A node w is *a-blocked* (see Figure 3.1 for an illustration) if none of its ancestors are blocked, it is not c-blocked, and it has ancestors v and w' such that w is a successor of v and

B1 $\mathcal{L}(w) \subseteq \mathcal{L}(w')$,

B2 if w is an $\mathsf{Inv}(S)$-successor of v and $\forall S.C \in \mathcal{L}(w')$, then

   a. $C \in \mathcal{L}(v)$, and

   b. if there is some R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$ such that w is an $\mathsf{Inv}(R)$-successor of v, then $\forall R.C \in \mathcal{L}(v)$,

B3 if $(\leqslant n\ S\ C) \in \mathcal{L}(w')$, then

   a. w is not an $\mathsf{Inv}(S)$-successor of v or

   b. w is an $\mathsf{Inv}(S)$-successor of v and $\sim C \in \mathcal{L}(v)$ or

   c. w is an $\mathsf{Inv}(S)$-successor of v, $C \in \mathcal{L}(v)$, and w' has at most $n-1$ S-successors z with $C \in \mathcal{L}(z)$, and

B4 if $(\geqslant m\ T\ E) \in \mathcal{L}(w')$ (resp. $\exists T.E \in \mathcal{L}(w')$), then

   a. w' has at least m (resp. at least one) T-successors z with $E \in \mathcal{L}(z)$ or

   b. w is an $\mathsf{Inv}(T)$-successor of v and $E \in \mathcal{L}(v)$.

A node w is *c-blocked* (see Figure 3.1 for an illustration) if none of its ancestors are blocked, it has ancestors v and w' such that w is a successor of v, and[2]

B5 $\mathcal{L}(w) \subseteq \mathcal{L}(w')$,

B6 if w is an $\mathsf{Inv}(S)$-successor of v and $\forall S.C \in \mathcal{L}(w')$, then

   a. $C \in \mathcal{L}(v)$, and

   b. if there is some R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$ such that w is an $\mathsf{Inv}(R)$-successor of v, then $\forall R.C \in \mathcal{L}(v)$,

B7 if $(\leqslant n\ T\ E) \in \mathcal{L}(w')$, then w is not an $\mathsf{Inv}(T)$-successor of v or $\sim E \in \mathcal{L}(v)$, and

B8 if w is an U-successor of v and $(\geqslant m\ U\ F) \in \mathcal{L}(v)$, then $\sim F \in \mathcal{L}(w)$.

[1] A c-block leads to a cycle in the tableau to be constructed, whereas an a-block is unravelled in the standard way.

[2] Please note that B5 is identical to B1, and B6 to B2.

In this case, we say that w' *is a c-blocking candidate for* w. We say that a c-blocking candidate w_1' for w *c-blocks* w if there is no c-blocking candidate w_2' for w "between" w_1' and w, i.e., if all c-blocking candidates w_2' for w different from w_1' are ancestors of w_1'. The definition of a node *a-blocking* another one is analogous.

A node is *indirectly blocked* if its predecessor is blocked, and in order to avoid wasted expansion after an application of the $\leqslant$-rule, a node y will also be taken to be indirectly blocked if 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* if, for some concept name $A \in N_C$, $\{A, \neg A\} \subseteq \mathcal{L}(x)$, or if for a some concept C, some role S, and some $n \in \mathbb{N}$: $(\leqslant\ n\ S\ C) \in \mathcal{L}(x)$ and there are $n+1$ nodes $y_0, \ldots, y_n$ such that $C \in \mathcal{L}(y_i)$, y_i is an S-neighbour of x, and $y_i \neq y_j$ for all $0 \leq i < j \leq n$.

The algorithm initialises the tree $\mathbf{T}$ to contain a single node x_0, called the *root* node, with $\mathcal{L}(x_0) = \{D\}$, where D is the concept to be tested for satisfiability. The inequality relation $\neq$ is initialised with the empty relation. $\mathbf{T}$ is then expanded by repeatedly applying the rules from Figure 3.1. The order in which the rules are applied is the following: all rules are applied first to the ancestors of a node x before the $\geqslant$- or the $\exists$-rule is applied to x.

The completion tree is complete if, for some node x, $\mathcal{L}(x)$ contains a clash or if none of the rules is applicable. If, for an input concept D, the expansion rules can be applied in such a way that they yield a complete, clash-free completion tree, then the algorithm returns "D is *satisfiable*", and "D is *unsatisfiable*" otherwise.

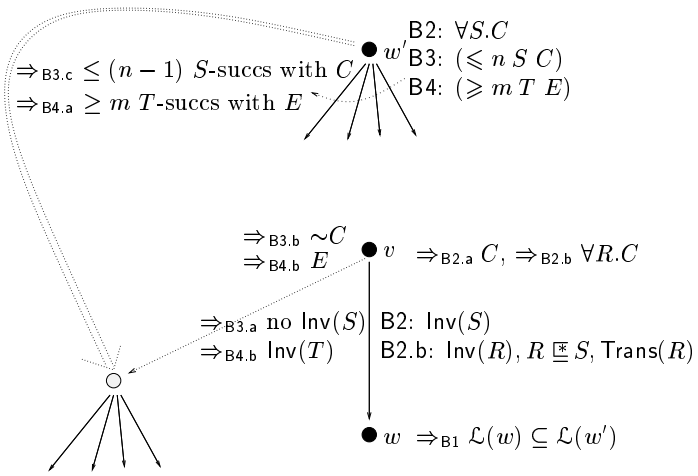

Figure 1: Illustration of an a-blocking situation. The double arrow indicates that a copy of w' and its successors is made a new successor of v when constructing a tableau.

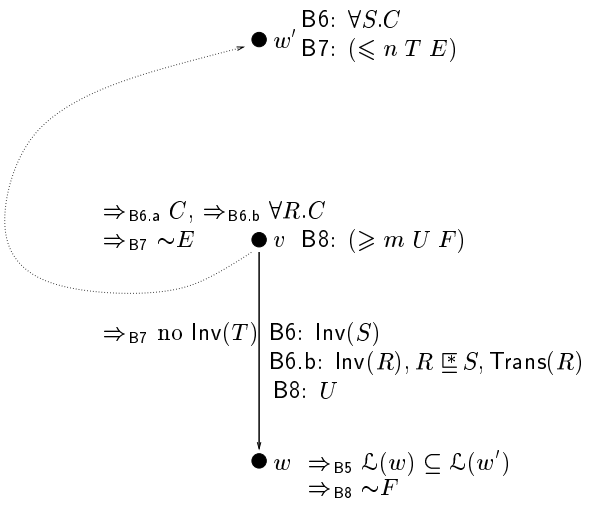

Figure 2: Illustration of a c-blocking situation. The arrow going up to w' indicates that w' is made a new successor of v when constructing a tableau.

Remark: (a) Please note that some of the rules are non-deterministic—hence the somewhat strange return behaviour of the algorithm.
(b) The intuition for the blocking conditions are as follows: when building a tableau from a completion tree, an a-block is unravelled in the standard way (i.e., a copy of w' and its successors is made a successor of v), while a c-block leads to a cylic tableau since the "original" w' is made a successor of v. B1 and B5 ensure that w' satisfies all $\forall$ restrictions on v. B2 and B5 ensure that v satisfies all "backward" $\forall$ restrictions on w'. In the a-blocking case, B3 and B4 ensure that, when a copy of w' has v as a predecessor (instead of its former predecessor), this copy still satisfies its at-most and at-least restrictions. In the c-blocking case, B5 ensures that at-most restrictions on w' are still satisfied with the new neighbour v, and B6 ensures that at-least restrictions on v are still satisfied even if several of its successors are c-blocked by the same node.
(c) A-blocking alone would have been enough to ensure correctness and termination—however, c-blocks may occur earlier, and may thus lead to a better performance.
(d) To make the following proofs easier, the blocking conditions are such that a node cannot be both a-blocked *and* c-blocked. If a node is c-blocked, it cannot be a-blocked.

3.2 Soundness and Completeness

We will show that the algorithm is terminating, sound, and complete.

$\sqcap$-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.	$(\leqslant n\ S\ C) \in \mathcal{L}(x)$, 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 no n nodes $y_1, \ldots, y_n$ such that $C \in \mathcal{L}(y_i)$, y_i is an S-neighbour of x, 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,
	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 a successor 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 a successor of x then $\mathcal{L}(\langle x, z\rangle) \longrightarrow \mathcal{L}(\langle x, z\rangle) \cup \mathcal{L}(\langle x, y\rangle)$ else (z is a predecessor of x) $\mathcal{L}(\langle z, x\rangle) \longrightarrow \mathcal{L}(\langle z, x\rangle) \cup \{\mathsf{Inv}(R) \mid R \in \mathcal{L}(\langle x, y\rangle)\}$ 3. $\mathcal{L}(\langle x, y\rangle) \longrightarrow \emptyset$ 4. Set $u \neq z$ for all u with $u \neq y$

Figure 3: The Expansion Rules for $\mathcal{SHIQ}$

Lemma 2 *For each $\mathcal{SHIQ}$-concept D and role hierarchy $\mathcal{R}$, the tableaux algorithm terminates.*

Proof: Let $m = |clos(D)|$, k the number of roles occurring in 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 fact that, in principle, the expansion rules build a completion tree monotonically with bounded depth and breadth:

1. 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 and will remain blocked forever.

2. Successors of a node x are the result of an application of the $\exists$- or the $\geqslant$-rule to concepts of the form $\exists R.C$ (which yields one successor) and $(\geqslant \ n \ S \ C)$ (which yields n successors) in $\mathcal{L}(x)$. 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 $m \cdot n_{max}$.

3. Suppose a node y has ancestors x, y', and x' with

   - y is a successor of x, y' is a successor of x',
   - $\mathcal{L}(y) = \mathcal{L}(y')$,
   - $\mathcal{L}(x) = \mathcal{L}(x')$, and
   - $\mathcal{L}(\langle x, y\rangle) = \mathcal{L}(\langle x', y'\rangle)$,

   and the $\geqslant$- or the $\exists$-rule can be applied to y. Hence no rules can be applied to any ancestors of y. In this case, y is a-blocked according to Definition 4.

   Nodes are labelled with non-empty subsets of $clos(D)$ and edges with subsets of R_D, so there are at most 2^{2m+k} different possible labellings for a pair of nodes and an edge. Therefore, if a path p is of length at least 2^{2m+k} then, from the a-blocking conditions defined in Definition 4 and the fact that rules are applied first to ancestors of a node before new successors of this node are generated, there must be two nodes y, y' on p such that y is directly a-blocked by y'. Since a path on which nodes are blocked cannot become longer, paths are of length at most 2^{2m+k}. ∎

Lemma 3 (Soundness) *If the expansion rules can be applied to a $\mathcal{SHIQ}$-concept D such that they yield a complete and clash-free completion tree with respect to $\mathcal{R}$, then D has a tableau with respect to $\mathcal{R}$.*

Proof: We build the tableau by (almost) standard unravelling similar to the one in [HST99a]. The only non-standard elements are due to (1) number restrictions and (2) the optimised blocking conditions: for (1), we must distinguish different successors of a node that are a-blocked by the same node—in standard unravelling, they would yield the same path, and thus at-least number restrictions on their predecessor might be violated. For c-blocking, B8 implies that the blocked node may not be a "witness" for an at-least restriction on its predecessor, and thus we do not need to distinguish different successors of a node that are blocked by the same node. For (2), if a node is c-blocked by another one, then we can build a cyclic model, i.e., make the blocking node a successor of the blocked node's predecessor.

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 $[\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}]$. Let $p = [\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}]$ be 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 a-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 $p = [p'|\frac{x}{x'}]$, x is not blocked, x' is neither c-blocked nor indirectly blocked, and x' is a-blocked iff $x \neq x'$. Furthermore, $\mathcal{L}(x') \subseteq \mathcal{L}(x)$ holds.

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

$$
\begin{array}{rcl}
\mathbf{S} & = & \mathsf{Paths}(\mathbf{T}) \\
\mathcal{L}(p) & = & \mathcal{L}(\mathsf{Tail}(p)) \\
\mathcal{E}(R) & = & \{\langle p, [p|\frac{x}{x'}]\rangle \in \mathbf{S} \times \mathbf{S} \mid x' \text{ is an } R\text{-successor of } \mathsf{Tail}(p)\} \cup \\
 & & \{\langle [q|\frac{x}{x'}], q\rangle \in \mathbf{S} \times \mathbf{S} \mid x' \text{ is an } \mathsf{Inv}(R)\text{-successor of } \mathsf{Tail}(q)\} \cup \\
 & & \{\langle p, [q|\frac{y}{y}]\rangle \mid p = [q|\frac{y}{y}|q'] \text{ and } y \text{ c-blocks an } R\text{-successor of } \mathsf{Tail}(p)\} \cup \\
 & & \{\langle [q|\frac{y}{y}], p\rangle \mid p = [q|\frac{y}{y}|q'] \text{ and } y \text{ c-blocks an } \mathsf{Inv}(R)\text{-successor of } \mathsf{Tail}(p)\}
\end{array}
$$

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

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

- $D \in \mathcal{L}([\frac{x_0}{x_0}])$ since $D \in \mathcal{L}(x_0)$.

- T1 holds because $\mathbf{T}$ is clash-free; T2 and T3 hold because $\mathsf{Tail}(p)$ is not indirectly blocked and $\mathbf{T}$ is complete.

- T4: Let $\forall S.C \in \mathcal{L}(p)$ 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 completeness implies $C \in \mathcal{L}(x')$. Since $\mathcal{L}(x') \subseteq \mathcal{L}(x) = \mathcal{L}(q)$, we have $C \in \mathcal{L}(q)$.

  If $p = [q|\frac{x}{x'}]$, then x' is an $\mathsf{Inv}(S)$-successor of $\mathsf{Tail}(q)$. If $x = x'$, then $\forall S.C \in \mathcal{L}(x')$ and thus completeness implies that $C \in \mathcal{L}(\mathsf{Tail}(q))$. If $x \neq x'$, then $\forall S.C \in \mathcal{L}(x)$ together with B2.a implies that $C \in \mathcal{L}(\mathsf{Tail}(q))$, and thus $C \in \mathcal{L}(q)$.

  If $q = [q_1|\frac{y}{y}]$ and $p = [q_1|\frac{y}{y}|q']$, then y c-blocks an S-successor z of $\mathsf{Tail}(p)$. Since $\mathbf{T}$ is complete, $C \in \mathcal{L}(z)$, and B5 implies that $C \in \mathcal{L}(y)$. Hence $C \in \mathcal{L}(q)$.

  If $p = [p_1|\frac{y}{y}]$ and $q = [p_1|\frac{y}{y}|p']$, then y c-blocks an $\mathsf{Inv}(S)$-successor z of $\mathsf{Tail}(q)$ and $\forall S.C \in \mathcal{L}(y)$. In this case, B6.a ensures that $C \in \mathcal{L}(\mathsf{Tail}(q))$, and thus $C \in \mathcal{L}(q)$.

- T6 is quite similar to T4: Let $\forall S.C \in \mathcal{L}(p)$ and $\langle p, q\rangle \in \mathcal{E}(R)$ 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 completeness of $\mathbf{T}$ implies $\forall R.C \in \mathcal{L}(x')$. If $x \neq x'$, then B1 implies $\mathcal{L}(x') \subseteq \mathcal{L}(x)$. Thus $\forall R.C \in \mathcal{L}(q)$.

  If $p = [q|\frac{x}{x'}]$, then x' is an $\mathsf{Inv}(R)$-successor of $\mathsf{Tail}(q)$ and hence $\mathsf{Tail}(q)$ is an R-neighbour of x'. If $x' = x$, then $\forall S.C \in \mathcal{L}(x)$ and completeness implies $\forall R.C \in \mathcal{L}(q)$. If $x' \neq x$, then x a-blocks x' and $\forall S.C \in \mathcal{L}(x)$. Due to B2.b, $\forall R.C \in \mathcal{L}(\mathsf{Tail}(q))$, and thus $\forall R.C \in \mathcal{L}(q)$.

  If $q = [q_1|\frac{y}{y}]$ and $p = [q_1|\frac{y}{y}|q']$, then y c-blocks an R-successor z of $\mathsf{Tail}(p)$. Since $\mathbf{T}$ is complete, $\forall R.C \in \mathcal{L}(z)$, and B5 implies that $\forall R.C \in \mathcal{L}(y)$. Hence $\forall R.C \in \mathcal{L}(q)$.

  If $p = [p_1|\frac{y}{y}]$ and $q = [p_1|\frac{y}{y}|p']$, then y c-blocks an $\mathsf{Inv}(R)$-successor z of $\mathsf{Tail}(q)$ and $\forall S.C \in \mathcal{L}(y)$. In this case, B6.b ensures that $\forall R.C \in \mathcal{L}(\mathsf{Tail}(q))$, and thus $\forall R.C \in \mathcal{L}(q)$.

- T5: Let $\exists S.C \in \mathcal{L}(p)$ and $x = \mathsf{Tail}(p)$. Since $\mathbf{T}$ is complete, there are two possibilities:

  - x has an S-successor y in $\mathbf{T}$ with $C \in \mathcal{L}(y)$.
    - If y is not blocked, then $q = [p|\frac{y}{y}] \in \mathbf{S}$, $\langle p, q\rangle \in \mathcal{E}(S)$, and $C \in \mathcal{L}(q)$ because $\mathcal{L}(y) = \mathcal{L}(q)$.
    - If y is a-blocked by some node z in $\mathbf{T}$, then $q = [p|\frac{z}{y}] \in \mathbf{S}$, $\langle p, q\rangle \in \mathcal{E}(S)$, and $C \in \mathcal{L}(q)$ because $C \in \mathcal{L}(y) \subseteq \mathcal{L}(z) = \mathcal{L}(q)$.
    - If y is c-blocked by some node z in $\mathbf{T}$, then $p = [p_1|\frac{z}{z}|p']$, $\langle p, [p_1|\frac{z}{z}]\rangle \in \mathcal{E}(S)$, and B5 implies that $C \in \mathcal{L}(z) = \mathcal{L}([p_1|\frac{z}{z}])$.

– otherwise, completeness implies that x is an $\mathsf{Inv}(S)$-successor of some y in $\mathbf{T}$ with $C \in \mathcal{L}(y)$. Thus p is of the form $p = [q|\frac{x}{x'}]$ and there are only two possibilities:

* If $\mathsf{Tail}(q) = y$, then $\langle p, q\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}(q)$.
* Let $\mathsf{Tail}(q) = u \neq y$. Now x only has one predecessor in $\mathbf{T}$, hence u is not the predecessor of x. This implies $x \neq x'$, x a-blocks x' in $\mathbf{T}$, and u is the predecessor of x' due to the construction of Paths. Since x has no S-successor z with $C \in \mathcal{L}(z)$, $\mathsf{B4.b}$ implies that x' is an $\mathsf{Inv}(S)$-successor of u and $C \in \mathcal{L}(u) = \mathcal{L}(q)$. Hence $\langle p, q\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}(q)$.

- $\mathsf{T7}$ is satisfied due to the symmetric definition of $\mathcal{E}$. $\mathsf{T8}$ is satisfied due to the definition of R-successor that takes into account $\sqsubseteq^*$.

- For $\mathsf{T9}$, let $p \in \mathbf{S}$ with $(\leqslant n\ S\ C) \in \mathcal{L}(p)$. Let $x = \mathsf{Tail}(p)$, $x' = \mathsf{Tail}'(p)$, and
$$P := \{q \in \mathbf{S} \mid \langle p, q\rangle \in \mathcal{E}(S) \text{ and } C \in \mathcal{L}(q)\}.$$
By definition of $\mathcal{E}$, P contains at most one q that is of the form $p = [q|\frac{x}{x'}]$. Due to $\mathsf{B7}$ which disallows c-blocking in the case where $(\leqslant n\ S\ C)$ is in the label of the blocking node and C is in the label of the blocked node's ancestor in case the blocked node is an $\mathsf{Inv}(S)$-successor, all other elements q of P are either of the form

  – $q = [p|\frac{y}{y'}]$ for y' an S-successor of x or
  – $q = [p_1|\frac{u}{u}]$ for $p = [p_1|\frac{u}{u}|p']$ and u c-blocks an S-successor of x.

  These elements of P are called "forward" elements in the following.

  Since (1) $\mathbf{T}$ is clash-free and complete and $(\leqslant n\ S\ C) \in \mathcal{L}(x)$, (2) each y' for each $[p|\frac{y}{y'}] \in P$ is an S-successor of x, (3) each u c-blocks an S-successor of x for each $q = [p_1|\frac{u}{u}]$ with $p = [p_1|\frac{u}{u}|p']$, and (4) each blocked node is blocked by exactly one ancestor, there are at most n forward elements in P.

  It remains to show that, if there is some q with $p = [q|\frac{x}{x'}]$ in P, then there are at most $n-1$ forward elements in P (and thus at most n elements in P).

  So, let $q \in P$ with $p = [q|\frac{x}{x'}]$ and $\mathsf{Tail}(q) = z$.

  – If $x = x'$, then z is a predecessor of x and observations (1) to (4) above yield that $\sharp P \leq n$.
  – If $x \neq x'$, then x a-blocks x' and x' is an $\mathsf{Inv}(S)$-successor of z. Moreover, all y' with $[p|\frac{y}{y'}] \in P$ are S-successors of x, and all u with $q = [p_1|\frac{u}{u}]$ and $p = [p_1|\frac{u}{u}|p']$ c-block an S-successor of x. In this case, $\mathsf{B3.a}$ and $\mathsf{B3.b}$ are not possible, and $\mathsf{B3.c}$ implies that P contains at most $(n-1)$ forward elements. Thus P contains at most n elements, and $\mathsf{T9}$ is satisfied.

- T10: Assume $(\geqslant n\ S\ C) \in \mathcal{L}(p)$. This 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 $\sharp S^T(p, C) \geqslant n$. Due to B8, which prevents an S-successor y of $\mathsf{Tail}(p)$ with $C \in \mathcal{L}(y)$ to be c-blocked, there are three possibilities for each y_i,

  - 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 a-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 ensure that $[p|\frac{z}{y_i}] \neq [p|\frac{z}{y_j}]$ for $y_i \neq y_j$.
  - $\mathsf{Tail}(p)$ 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 $[q|\frac{\mathsf{Tail}(p)}{\mathsf{Tail}'(p)}]$ with $\mathsf{Tail}(q) = y_i$. Again, q has the desired properties and, obviously, q is distinct from all other paths q_j.

- T11: Let $(\leqslant n\ S\ C) \in \mathcal{L}(p)$ 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 completeness implies $\{C, \sim C\} \cap \mathcal{L}(x') \neq \emptyset$. Since $\mathcal{L}(x') \subseteq \mathcal{L}(x) = \mathcal{L}(q)$, 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 $(\leqslant n\ S\ C) \in \mathcal{L}(x)$. If $x = x'$, then completeness implies $\{C, \sim C\} \cap \mathcal{L}(\mathsf{Tail}(q)) \neq \emptyset$. If $x \neq x'$, then x blocks x'. The construction of $\mathcal{E}$ and $\langle p, q \rangle \in \mathcal{E}(S)$ imply that B3.a is not possible, and B3.b together with B3.c imply that $\{C, \sim C\} \cap \mathcal{L}(\mathsf{Tail}(q)) \neq \emptyset$.

  If $q = [p_1|\frac{u}{u}]$ for $p = [p_1|\frac{u}{u}|p']$, then u c-blocks an S-successor of $\mathsf{Tail}(p)$, and completeness together with B5 implies that $\{C, \sim C\} \cap \mathcal{L}(u) = \{C, \sim C\} \cap \mathcal{L}(\mathsf{Tail}(q)) \neq \emptyset$

  If $p = [q_1|\frac{u}{u}]$ for $q = [q_1|\frac{u}{u}|q']$, then u c-blocks an $\mathsf{Inv}(S)$-successor of $\mathsf{Tail}(q)$. Since $(\leqslant n\ S\ C) \in \mathcal{L}(u)$, B7 implies that $\sim C \in \mathcal{L}(\mathsf{Tail}(q))$, and thus $\{C, \sim C\} \cap \mathcal{L}(q) \neq \emptyset$. ∎

Lemma 4 (Completeness) *If a $\mathcal{SHIQ}$-concept D has a tableau with respect to $\mathcal{R}$, then the expansion rules can be applied to D such that they yield a complete and clash-free completion tree with respect to $\mathcal{R}$.*

Proof: 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 $\mathbf{T}$:

$$\left.\begin{array}{l} \mathcal{L}(x) \subseteq \mathcal{L}(\pi(x)) \\ \text{if } y \text{ is an } S\text{-neighbour of } x \text{ then } \langle \pi(x), \pi(y) \rangle \in \mathcal{E}(S) \\ x \not\equiv y \text{ implies } \pi(x) \neq \pi(y) \end{array}\right\} (*)$$

CLAIM: Let $\mathbf{T}$ be a completion tree and π a function satisfying $(*)$. If a rule is applicable to $\mathbf{T}$, then the rule can be applied to $\mathbf{T}$ such that it yields a completion tree $\mathbf{T}'$ for which the function π can be extended to π' satisfying $(*)$.

Let $\mathbf{T}$ be a completion tree and π be a function that satisfies $(*)$. We verify the claim for each of the expansion rules.

- The $\sqcap$-rule: If $C_1 \sqcap C_2 \in \mathcal{L}(x)$, then $C_1 \sqcap C_2 \in \mathcal{L}(\pi(x))$. $\mathsf{T2}$ implies $C_1, C_2 \in \mathcal{L}(\pi(x))$ and hence the rule yields a $\mathbf{T}'$ for which $\pi' = \pi$ satifies $(*)$.

- The $\sqcup$-rule: If $C_1 \sqcup C_2 \in \mathcal{L}(x)$, then $C_1 \sqcup C_2 \in \mathcal{L}(\pi(x))$. $\mathsf{T3}$ implies $\{C_1, C_2\} \cap \mathcal{L}(\pi(x)) \neq \emptyset$. Hence the $\sqcup$-rule can add an appropriate C_i and $\pi' = \pi$ satifies $(*)$.

- The $\exists$-rule: If $\exists S.C \in \mathcal{L}(x)$, then $\exists S.C \in \mathcal{L}(\pi(x))$ and $\mathsf{T5}$ implies the existence of 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 define π' to be the extension of π with $\pi'(y) = t$, and thus, due to $\mathsf{T8}$, the result of applying the $\exists$-rule $\mathbf{T}'$ satisfies $(*)$ with π'.

- 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, $\mathsf{T4}$ 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 $\forall S.C \in \mathcal{L}(\pi(x))$. 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 $(*)$ and $\mathsf{T8}$. Then $\mathsf{T6}$ implies $\forall R.C \in \mathcal{L}(\pi(y))$, and hence the $\forall_+$-rule can be applied without violating $(*)$.

- The *choose*-rule: If $(\leqslant n\ S\ C) \in \mathcal{L}(x)$, then $(\leqslant n\ S\ C) \in \mathcal{L}(\pi(x))$ and, if there is an S-neighbour y of x, then $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}(S)$ due to $(*)$ and $\mathsf{T8}$. Then $\mathsf{T11}$ implies $\{C, \sim C\} \cap \mathcal{L}(\pi(y) \neq \emptyset$, and thus the *choose*-rule can add an appropriate concept $E \in \{C, \sim C\}$ to $\mathcal{L}(x)$ without violating $(*)$.

- The $\geqslant$-rule: If $(\geqslant n\ S\ C) \in \mathcal{L}(x)$, then $(\geqslant n\ S\ C) \in \mathcal{L}(\pi(x))$ and $\mathsf{T10}$ 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 extending π to $\pi'(y_i) = t_i$

for each $1 \leq i \leq n$, one obtains a function π' that satisfies $(*)$ for the tree resulting from the application of the $\geqslant$-rule.

- The $\leqslant$-rule: If $(\leqslant\ n\ S\ C) \in \mathcal{L}(x)$, then $(\leqslant\ n\ S\ C) \in \mathcal{L}(\pi(x))$ and T9 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 $\pi(y) = \pi(z)$. Then $\pi(y) = \pi(z)$ implies that $y \neq z$ cannot hold because of $(*)$, and y, z can be chosen such that y is a successor of x. Hence the $\leqslant$-rule can be applied without violating $(*)$.

This claim implies 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 $\not\approx = \emptyset$, 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 $(*)$. Lemma 2 implies that any sequence of rule applications must terminate. Due to $(*)$, any tree generated by these rule-applications must be clash-free. This can be seen by investigating the two possibilities for a clash:

- $\mathbf{T}$ cannot contain a node x with $\{C, \neg C\} \in \mathcal{L}(x)$ because $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$ and $\pi(x)$ satisfies T1.
- $\mathbf{T}$ 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 $y_i \not\approx y_j$ for $0 \leq i < j \leq n$: since $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$, we have $(\leqslant\ n\ S\ C) \in \mathcal{L}(\pi(x))$ and, since $y_i \not\approx y_j$ implies $\pi(y_i) \neq \pi(y_j)$, this would imply that $\sharp S^T(\pi(x), C) > n$, in contradiction to T9. ∎

Since terminologies can be internalised in $\mathcal{SHIQ}$ [HST99b], we have the following theorem:

Theorem 1 *The tableaux algorithm is a decision procedure for the satisfiability and subsumption of $\mathcal{SHIQ}$-concepts with respect to role hierarchies and terminologies.*

4 Empirical evaluation

The modified algorithm has been implemented in the FaCT system and tested with knowledge bases (KBs) derived from realistic applications: either $\mathcal{SHIQ}$ encodings of UML diagrams [BCDG01] or $\mathcal{SHIQ}$ translations of OIL/DAML+OIL ontologies [FvHH+01]. In each case, we have measured the time taken to classify the KB both with and without the optimised blocking condition, and also measured the maximum size and depth of trees constructed by the algorithm during the classification procedure. The results of these tests are shown in Figure 4.

	Optimised Blocking			Standard Blocking		
KB	time(s)	depth	size	time(s)	depth	size
hospital	2	16	775	–	45	6874
library	0.25	9	147	1.25	11	153
restaurant	8	26	1280	672	36	5824
soccer	36	27	3840	918	32	7087
geography	9	8	70	4506	18	5983

Figure 4: Comparison of KB classification times and data structures.

It can be seen that the optimised blocking condition uniformly improves performance and that, in some cases, the improvement is quite dramatic (more than two orders of magnitude in the case of the geography knowledge base).[3] The reason for this is the reduction in the depth and size of the trees built by the optimised algorithm. Apart from the inherent cost of building larger trees, the size of the search space due to non-deterministic expansion may increase exponentially with the number of nodes in the model.

It may be interesting to consider the geography KB in more detail in order to see why the performance improvement is so dramatic.[4] As the name suggests, this KB describes the geography of European countries. For example, it includes the axioms:

$$\begin{aligned} \mathsf{Republic\text{-}of\text{-}Ireland} &\sqsubseteq \exists \mathsf{is\text{-}part\text{-}of}.\mathsf{Ireland} \\ \mathsf{Ireland} &\sqsubseteq \exists \mathsf{is\text{-}part\text{-}of}.\mathsf{British\text{-}Isles} \\ \mathsf{British\text{-}Isles} &\sqsubseteq \exists \mathsf{is\text{-}part\text{-}of}.\mathsf{Western\text{-}Europe} \\ \mathsf{Western\text{-}Europe} &\sqsubseteq \exists \mathsf{is\text{-}part\text{-}of}.\mathsf{Europe} \end{aligned}$$

If these "part-of" relationships were uni-directional, the KB would be relatively trivial to classify. However, the KB also contains axioms specifying the parts that make up various composites, e.g.:

$$\mathsf{British\text{-}Isles} \sqsubseteq \exists \mathsf{is\text{-}part\text{-}of}^{-}.\mathsf{Ireland} \sqcap \exists \mathsf{is\text{-}part\text{-}of}^{-}.\mathsf{Great\text{-}Britain}$$

This kind of cyclical construction is quite common in KBs that describe physically connected structures, and can also be seen for example in the GALEN medical terminology KB. The effect of these cyclical axioms can be seen when classifying the concept Europe. Figure 5 illustrates part of the tree built by the using the standard double blocking. It can be seen that un-blocked nodes whose label includes Europe occur several times in a single branch of the tree. The fourth node in the branch is not blocked because the first occurrence of Europe is in the label of the root node, which has no predecessor and thus cannot be a

[3] Without optimised blocking, FaCT was unable to classify the hospital KB—system resources (memory) were exhausted after 86s of processing.

[4] Please note that the authors do not make any claims for the "quality" or "correctness" of this ontology.

blocking node. The seventh node in the branch is not blocked because the label of its predecessor contains Southern-Europe, whereas the label of the predecessor of the fourth node contains Western-Europe. Note that each un-blocked node with Europe in its label will lead to the generation of a large sub-tree due to an axiom that lists all the countries that make up Europe. In contrast, the optimised blocking condition allows the root node to c-block the fourth node, greatly reducing the total size of the tree.

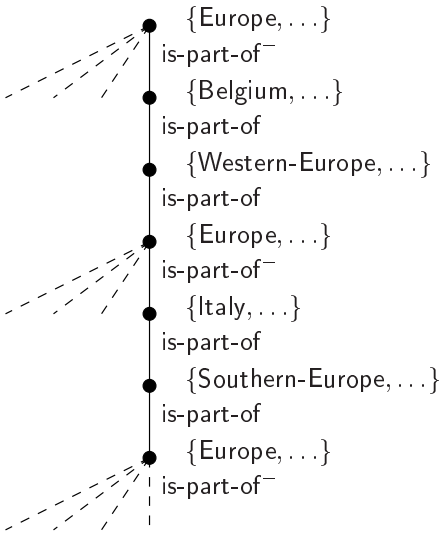

Figure 5: Tree built by unoptimised algorithm for concept Europe

The hospital, library, restaurant, and soccer KBs were all derived from the encoding in $\mathcal{SHIQ}$ of UML diagrams. The nature of the encoding means that the resulting KBs tend to be highly cyclical. Moreover, if the UML diagrams include maximum cardinality constraints on relations (e.g., single valued relations), then the encoded KB will include qualified at most restrictions, possibly with complex qualifying concepts (i.e., concepts of the form $(\leqslant nR.C)$ where C is non-atomic). The expansion of these concepts is highly non-deterministic (due to the $\rightarrow_{\leq}$- and the $\rightarrow_{choose}$-rule), and it is critical to minimise the number of node labels in which they occur. In the case of the hospital KB, for example, the degree of non-determinism in the larger tree generated without the optimised blocking condition is so great that, in attempting to search it, FaCT exhausts the system's memory.

5 Discussion

In order to deal with inverse roles and number restrictions in a logic lacking the finite model property, the $\mathcal{SHIQ}$ algorithm implemented in the FaCT system

introduced a new and more sophisticated "double-blocking" technique. The conditions under which a block could be established were clearly more exacting than was strictly necessary, but it was assumed that, apart from the difficulty of proving soundness and completeness, the increased cost of checking more precisely defined conditions would outweigh any benefit that might be derived.

The failure of the FaCT system to solve UML derived knowledge bases lead us to reconsider this conjecture, and we have presented an optimised algorithm that checks for two different kinds of block, with more precisely defined conditions under which each can be established. In spite of this increased complexity, we have been able to prove that the optimised algorithm is still sound and complete, and have shown that in some cases it can improve FaCT's performance by more than two orders of magnitude.

Clearly, the adverse effects of the stricter standard blocking condition should not have been underestimated. Inefficient blocking can lead to an increase in the size of the tree constructed by the algorithm, and given a logic with the complexity of $\mathcal{SHIQ}$ this can lead to a catastrophic blow up in the size of the search space (the number of different trees that must be explored). As we have shown, this effect can be observed in realistic knowledge bases derived both from the encoding of UML diagrams and from OIL/DAML+OIL ontologies.

Acknowledgements

The authors would like to thank Carsten Lutz for valuable comments and suggestions.

References

[BBMAR89] Ronald J. Brachman, Alexander Borgida, Deborah L. McGuinness, and Lori Alperin Resnick. The CLASSIC knowledge representation system, or, KL-ONE: the next generation. Preprints of the Workshop on Formal Aspects of Semantic Networks, Two Harbors, Cal., 1989.

[BCDG01] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML Class Diagrams using Description Logic Based Systems. In *Proc. of the KI'2001 Workshop on Applications of Description Logics*. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-44/, 2001.

[BFH+94] Franz Baader, Enrico Franconi, Bernhard Hollunder, Bernhard Nebel, and Hans-Jürgen Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.

[BFT95] P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: Preliminary report. In *Proc. of DL'95*, pages 131–139, 1995.

[BS85] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[CLN98] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In Jan Chomicki and Gnter Saake, editors, *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.

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

[FN00] E. Franconi and G. Ng. The i.com tool for intelligent conceptual modelling. In *Working Notes of the ECAI2000 Workshop KRDB2000*, 2000.

[FvHH+01] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[HKNP94] Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, and Hans-Jürgen Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 68:367–397, 1994.

[HM01] Volker Haarslev and Ralf Möller. RACER system description. In *IJCAR-01*, 2001.

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

[HST99a] I. Horrocks, U. Sattler, and S. Tobies. A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions. LTCS-Report LTCS-99-08, LuFG Theoretical Computer Science, RWTH Aachen, 1999. Revised version. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[HST99b] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proc. of LPAR'99*, number 1705 in LNAI, pages 161–180. Springer-Verlag, 1999.

[Neb90] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[PS99] Peter F. Patel-Schneider. DLP. In *Proc. of DL'99*, pages 9–13. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-22/, 1999.

[Sch89] M. Schmidt-Schauss. Subsumption in KL-ONE is undecidable. In *Proc. of KR-89*, pages 421–431, Boston (USA), 1989.

[SvRvdVM95] P.-H. Speel, F. van Raalte, P. E. van der Vet, and N. J. I. Mars. Runtime and memory usage performance of description logics. In G. Ellis, R. A. Levinson, A. Fall, and V. Dahl, editors, *Knowledge Retrieval, Use and Storage for Efficiency: Proc. of the 1st Int. KRUSE Symposium*, pages 13–27, 1995.