Optimised Reasoning for $\mathcal{SHIQ}$

E0175

Abstract.

We present an optimised version of the tableau algorithm implemented in the FaCT knowledge representation system which decides satisfiability and subsumption in $\mathcal{SHIQ}$, a very expressive description logic providing, e.g., inverse and transitive roles, number restrictions, and general axioms. 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.

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 seen in the successful and highly influential knowledge representation system KL-ONE [4].

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 [18]. 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 the two problems would always be truly undecidable.

Following this observation, the developers of the CLASSIC system [3] decided that the underlying DL 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 [7], 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 [1, 5], 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 [16, 11, 19].

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 [15]. 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 [10, 17], 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 [6]. 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 [9]. Interestingly, this translation yields knowledge bases from realistic applications that could *not* be solved by any of the available DL systems [2], 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 [14]. 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 Preliminaries

In this section, we define the 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 $R \in \mathbf{R}_+$,

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

An interpretation $\mathcal{I}$ *satisfies a role hierarchy* $\mathcal{R}$ iff $R^\mathcal{I} \subseteq S^\mathcal{I}$ for each $R \sqsubseteq S \in \mathcal{R}$; such an interpretation is called a *model* of $\mathcal{R}$.

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:

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

2. Since set inclusion is transitive and $R^\mathcal{I} \subseteq S^\mathcal{I}$ implies $\mathsf{Inv}(R)^\mathcal{I} \subseteq \mathsf{Inv}(S)^\mathcal{I}$, for a role hierarchy $\mathcal{R}$, we introduce $\underline{\sqsubseteq}\!*$ 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 $R \underline{\sqsubseteq}\!* S$ and $S \underline{\sqsubseteq}\!* R$.
3. Obviously, a role R is transitive if and only if its inverse $\mathsf{Inv}(R)$ is transitive. However, 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 $\mathtt{true}$ 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): $\mathsf{Trans}(R) := \mathtt{true}$ if, for some S with $S \equiv R$, $S \in \mathbf{R}_+$ or $\mathsf{Inv}(S) \in \mathbf{R}_+$, and false otherwise.

Definition 2 A role R is called *simple* w.r.t. $\mathcal{R}$ iff not $\mathsf{Trans}(S)$ for each $S \underline{\sqsubseteq}\!* R$.

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 a $\mathcal{SHIQ}$-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

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

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 a model $\mathcal{I}$ of $\mathcal{R}$ with $C^\mathcal{I} \neq \emptyset$. 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 every model $\mathcal{I}$ of $\mathcal{R}$. Two concepts C, D are *equivalent* w.r.t. $\mathcal{R}$ (written $C \equiv_\mathcal{R} D$) iff they are mutually subsuming.

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

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 duality between universal and existential and at-most ($\leqslant$) and at-least ($\geqslant$) restrictions. 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 $\mathit{clos}(D)$ to be the smallest set that contains D, is closed under sub-formulae and $\sim$, and which contains, for each subconcept $\forall R.C \in \mathit{clos}(D)$ and role $R' \underline{\sqsubseteq}\!* R$, also the concept $\forall R'.C$. Note that $\#\mathit{clos}(D)$ is linear in $|D| + |\mathcal{R}|$.

A tableau algorithm tries to construct, for an input concept D, an *abstraction* of a model of D, i.e., a so-called *tableau* for D. The advantage of constructing/testing the existence of tableaux rather than models is that in tableaux, all conditions are *local*, whereas there are *global* conditions in the definition of models (e.g., transitivity of $r^\mathcal{I}$ for $r \in \mathbf{R}_+$). A definition of a $\mathcal{SHIQ}$ tableau can be found in [12].

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 $\mathcal{R}$.*

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 [13] but for the modified pairwise-blocking condition.

The algorithm presented here tries to construct, for an input concept D, a tableau whose relational structure forms a tree where nodes are labelled with concepts from $\mathit{clos}(D)$ and with D in the label of the root node. We must take special care to prevent the algorithm from generating a tree with arbitrarily long paths, i.e., from failing to terminate. 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. This three-fold condition is rather strict, e.g., the root node can never block another node, and this can lead to later blocking and longer paths than is absolutely necessary.

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

Definition 3 Let $\mathcal{R}$ be a role hierarchy and D a $\mathcal{SHIQ}$-concept in NNF. A *completion tree* w.r.t. $\mathcal{R}$ and D is a tree $\mathbf{T}$ where each node x of the tree is labelled with a set $\mathcal{L}(x) \subseteq$ *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 *clos*(D). Additionally, we keep track of inequalities between nodes of the tree with a symmetric binary relation $\neq$ between the nodes of $\mathbf{T}$.[1]

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 *indirectly blocked* if its predecessor is blocked, and (in order to avoid wasted expansion after an application of the $\leqslant$-rule, which is explained later) a node y will also be taken to be indirectly blocked if it is a successor of a node x and $\mathcal{L}(\langle x, y\rangle) = \emptyset$. A node is *directly blocked* if it is *c-blocked* or *a-blocked*.[2]

A node w is *a-blocked* (see Figure 3 in Appendix A 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

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 1) 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 4 in Appendix A for an illustration) if none of its ancestors are blocked, it has ancestors v and w' such that w is a successor of v, it satisfies B1 and B2, and

B5 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
B6 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] The $\neq$ relation is used to prohibit identification of nodes introduced by an application of the $\geqslant$-rule, which could lead to non-termination due to infinite sequences of $\geqslant$- and $\leqslant$-rule applications.

[2] A c-block leads to a **c**ycle in the tableau to be constructed, whereas an a-block is unravelled in the standard way–"a" stands for **a**cyclic.

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.

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 $n+1$ 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 $0 \leq i < j \leq n$.

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 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.

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 ensures that w' satisfies all $\forall$ restrictions on v. B2 ensures 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. Illustrations of the two blocking conditions are given in Appendix A.

Lemma 2 *Let D be a $\mathcal{SHIQ}$-concept in NNF and $\mathcal{R}$ a role hierarchy.*
1. The application of the tableau algorithm to D and $\mathcal{R}$ terminates.
2. If the expansion rules can be applied to D such that they yield a complete and clash-free completion tree w.r.t. $\mathcal{R}$, then D has a tableau w.r.t. $\mathcal{R}$.
3. If D has a tableau w.r.t. $\mathcal{R}$, then the tableau algorithm can be applied to D such that it yields a complete and clash-free completion tree w.r.t. $\mathcal{R}$.

Sketch of the Proof: (1.) Termination is due to the fact that the tableau algorithm constructs, in a monotonic way, a tree with bounded depth and width. (2.) From a complete and clash-free completion tree, we can construct a tableau by almost standard unravelling. The only non-standard elements are (i) cyclic parts of the tableau in c-blocking situations and (ii) a slightly more complex unravelling to make sure that at-least restrictions are satisfied in situations where two successors of the same node are a-blocked by the same node. (3.) A tableau can be used to trigger the application of the non-

$\rightarrow_{\sqcap}$:	if 1.	$C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
	then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\rightarrow_{\sqcup}$:	if 1.	$C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$\rightarrow_{\exists}$:	if 1.	$\exists S.C \in \mathcal{L}(x)$, x is not blocked and 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 $\mathcal{L}(y) = \{C\}$
$\rightarrow_{\forall}$:	if 1.	$\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and there is an S-neighbour y of x with $C \notin \mathcal{L}(y)$
	then	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$
$\rightarrow_{\forall_+}$:	if 1.	$\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and there is some R with $\mathsf{Trans}(R)$ and $R \underline{\overset{*}{\sqsubseteq}} S$,
	2.	and an R-neighbour y of x with $\forall R.C \notin \mathcal{L}(y)$
	then	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall R.C\}$
$\rightarrow_{\bowtie}$:	if 1.	$(\leqslant n\ S\ C) \in \mathcal{L}(x)$, x is not indirectly blocked, and 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 $E \in \{C, {\sim}C\}$
$\rightarrow_{\geqslant}$:	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 $1 \leq i < j \leq n$.
$\rightarrow_{\leqslant}$:	if 1.	$(\leqslant n\ S\ C) \in \mathcal{L}(x)$, x is not indirectly blocked, $\sharp S^{\mathbf{T}}(x, C) > n$, and
	2.	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 1. The Expansion Rules for $\mathcal{SHIQ}$

deterministic expansion rules in such a way that the tableau algorithm yields a complete and clash-free completion tree.

Since terminologies (or general TBoxes) can be internalised in $\mathcal{SHIQ}$, and subsumption can be reduced to satisfiability [14], we thus have:

Theorem 1 *The tableaux algorithm decides 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 [2] or $\mathcal{SHIQ}$ translations of OIL/DAML+OIL ontologies [8]. 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 the following table.

	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

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. E.g., it includes the axioms:

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

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, e.g., in the GALEN medical terminology KB. The effect of these cyclical axioms can be seen when classifying the concept Europe. Figure 2 illustrates part of the tree built by the algorithm 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.

[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.

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 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.

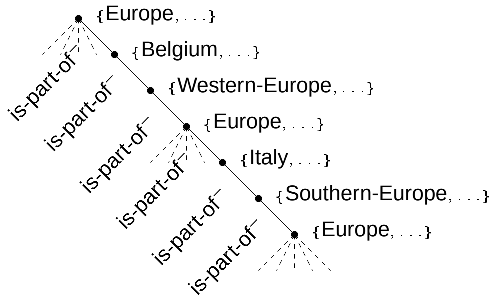


Figure 2. Tree built by unoptimised algorithm for concept Europe

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 $\rightarrow_{\bowtie}$-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.

REFERENCES

[1] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. 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).

[2] 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).

[3] R. J. Brachman, A. Borgida, D. McGuinness, and L. 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.

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

[5] P. Bresciani, E. Franconi, and S. Tessaris, 'Implementing and testing expressive description logics: Preliminary report', in *Proc. of DL'95*, pp. 131–139, (1995).

[6] D. Calvanese, M. Lenzerini, and D. Nardi, 'Description logics for conceptual data modeling', in *Logics for Databases and Information Systems*, eds., J. Chomicki and G. Saake, 229–263, Kluwer Academic Publisher, (1998).

[7] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt, 'The complexity of concept languages', in *Proc. of KR-91*, Boston, MA, USA, (1991).

[8] 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).

[9] E. Franconi and G. Ng, 'The i.com tool for intelligent conceptual modelling', in *Working Notes of the ECAI2000 Workshop KRDB2000*, (2000).

[10] V. Haarslev and R. Möller, 'RACER system description', in *Proc. of IJCAR-01*, number 2083 of LNAI, Springer-Verlag, (2001).

[11] J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich, 'An empirical analysis of terminological representation systems', *Artificial Intelligence*, **68**, 367–397, (1994).

[12] I. Horrocks and U. Sattler, 'Optimised reasoning for SHIQ', LTCS-Report LTCS-01-08, LuFG Theoretical Computer Science, RWTH Aachen, Germany, (2001). See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[13] 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.

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

[15] I. Horrocks, 'Using an expressive description logic: FaCT or fiction?', in *Proc. of KR-98*, pp. 636–647, (1998).

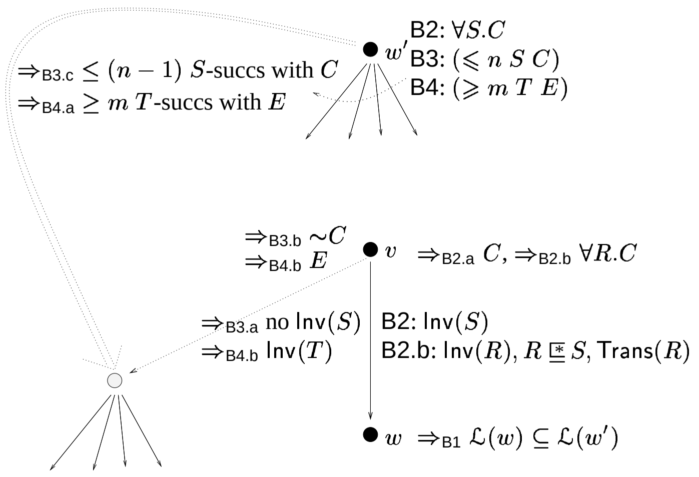
[16] B. Nebel, 'Terminological reasoning is inherently intractable', *Artificial Intelligence*, **43**, 235–249, (1990).

[17] P. Patel-Schneider, 'DLP', in *Proc. of DL'99*, pp. 9–13. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-22/, (1999).

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

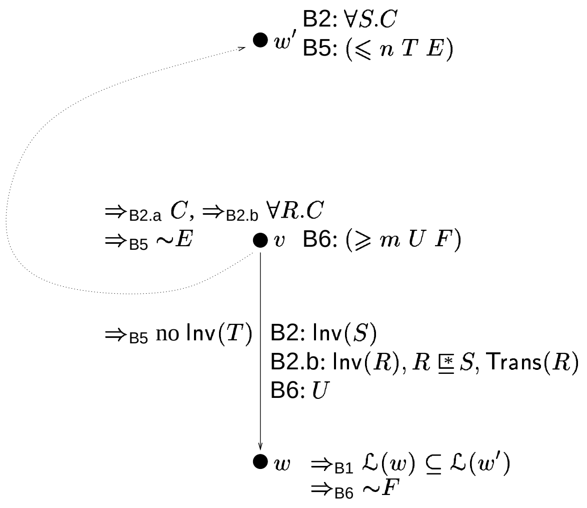
[19] P.-H. Speel, F. van Raalte, P. E. van der Vet, and N. J. I. Mars, 'Runtime and memory usage performance of description logics', in *Knowledge Retrieval, Use and Storage for Efficiency: Proc. of the 1st Int. KRUSE Symposium*, eds., G. Ellis, R. A. Levinson, A. Fall, and V. Dahl, pp. 13–27, (1995).

A Illustrations



w'
B2: $\forall S.C$
B3: $(\leqslant n\ S\ C)$
B4: $(\geqslant m\ T\ E)$
$\Rightarrow_{\mathsf{B3.c}} \leq (n-1)$ S-succs with C
$\Rightarrow_{\mathsf{B4.a}} \geq m$ T-succs with E
$\Rightarrow_{\mathsf{B3.b}} \sim C$
$\Rightarrow_{\mathsf{B4.b}} E$
v
$\Rightarrow_{\mathsf{B2.a}} C$, $\Rightarrow_{\mathsf{B2.b}} \forall R.C$
$\Rightarrow_{\mathsf{B3.a}}$ no $\mathsf{Inv}(S)$
$\Rightarrow_{\mathsf{B4.b}}$ $\mathsf{Inv}(T)$
B2: $\mathsf{Inv}(S)$
B2.b: $\mathsf{Inv}(R), R \sqsubseteq^* S, \mathsf{Trans}(R)$
w
$\Rightarrow_{\mathsf{B1}} \mathcal{L}(w) \subseteq \mathcal{L}(w')$

Figure 3. 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.



w'
B2: $\forall S.C$
B5: $(\leqslant n\ T\ E)$
$\Rightarrow_{\mathsf{B2.a}} C$, $\Rightarrow_{\mathsf{B2.b}} \forall R.C$
$\Rightarrow_{\mathsf{B5}} \sim E$
v
B6: $(\geqslant m\ U\ F)$
$\Rightarrow_{\mathsf{B5}}$ no $\mathsf{Inv}(T)$
B2: $\mathsf{Inv}(S)$
B2.b: $\mathsf{Inv}(R), R \sqsubseteq^* S, \mathsf{Trans}(R)$
B6: U
w
$\Rightarrow_{\mathsf{B1}} \mathcal{L}(w) \subseteq \mathcal{L}(w')$
$\Rightarrow_{\mathsf{B6}} \sim F$

Figure 4. 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.