Abstract.

We present an optimised version of the tableau algorithm implemented in the FaCT knowledge representation system which decides satisfiability and subsumption in $SHIQ$, 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 concepts (to arrange the concepts defined in a knowledge base into a taxonomy), and satisfiability of concepts (to check the consistency of the knowledge base). Unfortunately, when the underlying representational formalism was studied in detail, these two inference problems turned out to be undecidable [18].

Subsequently, a variety of decidable DLs and their computational complexity were studied. It turned out that the inference problems of (almost all) DLs with interesting expressive power were PSPACE-hard [7], i.e., of a complexity far beyond practicability. Despite this discouraging assessment w.r.t. 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 practice—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 $SHIF$, later $SHIQ$) which was even more complex, 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 behaving similarly were subsequently developed [10, 17].

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 correctness.

In order to deal with inverse roles and the possibility of concepts with only infinite models, the $SHIQ$ 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 tableau 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 $SHIQ$-concepts and roles. We start with $SHIQ$-roles, then introduce some
1. every concept name
2. Since set inclusion is transitive and the form
3. Obviously, a role

The next abbreviations make the following considerations easier.

\[ \text{Definition 1} \text{ Let } R \text{ be a set of role names with both transitive and normal role names } R_1 \cup R_2 = R, \text{ where } R_2 \cap R_+ = \emptyset. \text{ The set of } \mathcal{SHIQ}\text{-roles is } R \cup \{ R^\text{-} | R \in R_\mathcal{R} \}. A \text{ role inclusion axiom is of the form } R \subset S, \text{ for two } \mathcal{SHIQ}\text{-roles } R \text{ and } S. \text{ A role hierarchy is a set of role inclusion axioms.}

An interpretation \( I = (\Delta^I, \mathcal{I}) \) consists of a set \( \Delta^I \), called the domain of \( I \), and a function \( \mathcal{I} \) which maps every role to a subset of \( \Delta^I \times \Delta^I \) such that, for \( P \in R \) and \( R \in R_\mathcal{R} \), \( \{ (x, y) \in P^I \text{ iff } (y, x) \in P^I, \text{ and } (x, y) \in R^I \text{ and } (y, z) \in R^I \text{, then } (x, z) \in R^I. \} \).

The interpretation function \( \mathcal{I} \) satisfies a role hierarchy \( R \) if \( R^I \subseteq S^I \) for each \( R \subseteq S \in R \); such an interpretation is called a model of \( R \).

The next abbreviations 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 \( \text{Inv}(R) \) which returns the inverse of a role: \( \text{Inv}(R) := R^{-} \) if \( R \) is a role name, and \( \text{Inv}(R) := S \) if \( R = S^+ \) for a role name \( S \).
2. Since set inclusion is transitive and \( R^I \subseteq S^I \) implies \( \text{Inv}(R)^I \subseteq \text{Inv}(S)^I \), for a role hierarchy \( R \), we introduce \( R^I \text{ as an abbreviation for } R \subseteq S \text{ and } S \subseteq R \).
3. Obviously, a role \( R \) is transitive if and only if its inverse \( \text{Inv}(R) \) is transitive. However, in cyclic cases such as \( R = S \), \( S \) is transitive if \( R \) or \( \text{Inv}(R) \) is a transitive role name. To avoid these case distinctions, the function \( \text{Trans} \) returns true iff \( R \) is a transitive role—regardless of the reason: \( \text{Trans}(R) := \text{true if, for some } S \text{ with } S \equiv R, S \in R_+ \text{ or } \text{Inv}(S) \in R_+ \text{, and false otherwise.}

\[ \text{Definition 2} \text{ A role } R \text{ is called simple w.r.t. } R \text{ iff not } \text{Trans}(S) \text{ for each } S \equiv R \text{.}

Let \( N_C \) be a set of concept names. The set of \( \mathcal{SHIQ}\text{-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 \( \mathcal{SHIQ}\text{-role, then } (C \cap D), (C \cup D), (\neg C), (\forall R.C), \) and \( (\exists R.C) \) are concepts, and
3. if \( C \) is a concept, \( R \) is a simple \( \mathcal{SHIQ}\text{-role and } n \in N \text{, then } (\leq n R C) \text{ and } (\geq n R C) \text{ are concepts.}

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

\[ \{ (x, y) \in R^I \text{ with } (x, y) \in R \text{ and } (y, z) \in C^I \} \text{; for each } C \in N_C \text{, then } C^I \subseteq \Delta^I \}

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

A concept \( C \) is called satisfiable w.r.t. a role hierarchy \( R \) if there is a model \( I \) of \( R \) with \( C^I \neq \emptyset. \) Such an interpretation is called a model of \( C \) w.r.t. \( R. \) A concept \( D \) subsumes a concept \( C \) w.r.t. \( R \) (written \( C \subseteq R \) \( D \)) if \( C^I \subseteq D^I \) holds for every model \( I \) of \( R. \) Two concepts \( C, D \) are equivalent w.r.t. \( R \) (written \( C \equiv R \) \( D \)) if 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}\text{-concept can easily be transformed to an equivalent one in NNF by pushing negations inwards using a combination of De Morgan’s laws and the duality between universal and existential and at-most (\( \leq \)) and at-least (\( \geq \)) restrictions. For a concept \( C \) we will denote the NNF of \( \neg C \) by \( \neg C. \)

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

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 \( x^2 \) for \( x \in R_+ \)). A definition of a \( \mathcal{SHIQ}\text{-tableau can be found in [12].}

\[ \text{Lemma 1} \text{ A } \mathcal{SHIQ}\text{-concept } D \text{ is satisfiable w.r.t. a role hierarchy } R \text{ iff there exists a tableau for } D \text{ w.r.t. } R. \]

From Lemma 1, an algorithm which constructs a tableau for a \( \mathcal{SHIQ}\text{-concept } D \) can be used as a decision procedure for the satisfiability of \( D \) w.r.t. a role hierarchy \( R. \) Such an algorithm will now be described in detail. It uses the same techniques as the \( \mathcal{SHIQ}\text{-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 close(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 relevant ones, and in condition (3), we will restrict the relations to the relevant ones.

\[ \text{Definition 3} \text{ Let } R \text{ be a role hierarchy and } D \text{ a } \mathcal{SHIQ}\text{-concept in NNF. A completion tree w.r.t. } R \text{ and } D \text{ is a tree } T \text{ where each node } x \text{ of the tree is labelled with a set } L(x) \subseteq \text{close}(D) \text{ and each edge } (x, y) \text{ is labelled with a set of role names } L((x, y)) \text{ containing (possibly inverse) roles occurring in close}(D). \text{ Additionally, we keep track of inequalities between nodes of the tree with a symmetric binary relation } \neq \text{ between the nodes of } T. \]

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 \( S \in L((x, y)) \text{ for some } S \text{ with } S \subseteq R; \) \( y \) is

\[ 3 \text{ The } \neq \text{ relation is used to prohibit identification of nodes introduced by an application of the } \geq \text{-rule, which could lead to non-termination due to infinite sequences of } \geq \text{-rule applications.} \]
called an \( R\)-neighbour of \( x \) if \( y \) is an \( R\)-successor of \( x \), or if \( x \) is an \( \text{Inv}(R)\)-successor of \( y \).

For a role \( S \), a concept \( C \), and a node \( x \) in \( T \), we define \( S^T(x, C) \) by 
\[
S^T(x, C) := \{ y \mid y \text{ is } S\text{-neighbour of } x \text{ and } C \in \mathcal{L}(y) \}.
\]

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 \( \preceq \)-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 \( \mathcal{L}(x, y) = \emptyset \). A node is directly blocked if it is c-blocked or a-blocked.

A node \( w \) is a-blocked (see Figure 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

\[
\text{B1 } \mathcal{L}(w) \subseteq \mathcal{L}(w').
\]

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

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

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

\[
\text{B3 if } (\preceq n S.C) \in \mathcal{L}(w'), \text{ then}
\]

(a) \( w \) is not an Inv\( (S)\)-successor of \( v \) or

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

(c) \( w \) is an 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

\[
\text{B4 if } (\geq m T.E) \in \mathcal{L}(w') \text{ (resp. } \exists T.E \in \mathcal{L}(w')) \text{, then}
\]

(a) \( w' \) has at most \( m \) (resp. at least \( 1 \)) \( T\)-successors \( z \) with \( E \in \mathcal{L}(z) \) or

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

Figure 1. An a-blocking situation. When constructing a tableau, a copy of \( w' \) and its successors is made a successor of \( v \).

A node \( w \) is c-blocked (see Figure 2 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

\[
\text{B5 if } (\preceq n T.E) \in \mathcal{L}(w'), \text{ then } w \text{ is not an Inv}(T)\text{-successor of } v \text{ or } \neg E \in \mathcal{L}(v), \text{ and}
\]

\[
\text{B6 if } w \text{ is an } \mathcal{U}\text{-successor of } v \text{ and } (\geq m U.F) \in \mathcal{L}(v), \text{ then } \neg F' \in \mathcal{L}(w').
\]

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.

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 cyclic 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.

Lemma 2 Let \( D \) be a \( S\text{HIQ}\)-concept 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. \( \mathcal{R} \).
The modified algorithm has been implemented in the FaCT system, and tested with knowledge bases (KBs) derived from realistic applications: either SHIQ encodings of UML diagrams [2] or 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.

<table>
<thead>
<tr>
<th>KB</th>
<th>Optimised Blocking</th>
<th>Standard Blocking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time(s)</td>
<td>depth</td>
</tr>
<tr>
<td>hospital</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>library</td>
<td>0.25</td>
<td>9</td>
</tr>
<tr>
<td>restaurant</td>
<td>8</td>
<td>26</td>
</tr>
<tr>
<td>soccer</td>
<td>36</td>
<td>27</td>
</tr>
<tr>
<td>geography</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>

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). 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. As the name suggests, this KB describes the geography of European countries. E.g., it includes the axioms:

- Republic-of-Ireland ⊑ Irish-part-of-Ireland
- Ireland ⊑ Irish-part-of-British-Isles
- British-Isles ⊑ Irish-part-of-Western-Europe
- Western-Europe ⊑ Irish-part-of-Europe

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.:
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 4 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. 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 4. Tree built by unoptimised algorithm for concept Europe

The hospital, library, restaurant, and soccer KBs were all derived from the encoding in \( S^HTQ \) 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 \( \leq nR.C \) where \( C \) is non-atomic). The expansion of these concepts is highly non-deterministic (due to the \( \rightarrow \leq \) and the \( \rightarrow_{\geq} \)-rule), and it is critical to minimise the number of node labels in which they occur. E.g., the degree of non-determinism in the larger tree generated without the optimised blocking condition for the hospital KB is so great that, in attempting to search it, FaCT exhausts the system’s memory.

5 Discussion

To deal with inverse roles and number restrictions in a logic lacking the finite model property, the \( S^HTQ \) 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 \( SHIQ \) 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