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

Ian Horrocks
Department of Computer Science,
University of Manchester

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

Abstract

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

1 Motivation

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

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

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

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

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

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

2 Preliminaries

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

The logics we will discuss are all based on an extension of the well known DL $\mathcal{ALC}$ [Schmidt-Schauß & Smolka, 1991] to include transitively closed primitive roles [Sattler, 1996]; we will call this logic $\mathcal{S}$ due to its relationship with the proposition (multi) modal logic $\mathcal{S}_4(m)$ [Schild, 1991].

1This logic has previously been called $\mathcal{ALC}_{R^*}$, but this becomes too cumbersome when adding letters to represent additional features.
Definition 1 Let $N_C$ be a set of concept names and $R$ a set of role names with transitive role names $R_i \subseteq R$. The set of $SI$-roles is $R(\overline{R}) = \{ R \mid R \in R \}$. The set of $SI$-concepts is the smallest set such that every concept name is a concept, and, if $C$ and $D$ are concepts and $R$ is an $SI$-role, then $(C \sqcap D), (C \sqcup D), (\neg C)$, $(\forall R.C)$, and $(\exists R.C)$ are also concepts.

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

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

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

A concept $C$ is called satisfiable iff there is some interpretation $I$ such that $C^I \not= \emptyset$. Such an interpretation is called a model of $C$. A concept $D$ subsumes a concept $C$ (written $C \subseteq D$) iff $C^I \subseteq D^I$ holds for every interpretation $I$. For an interpretation $I$, an individual $x \in \Delta^I$ is called an instance of a concept $C$ iff $x \in C^I$.

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

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

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

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

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

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

Definition 2 A terminology $T$ is a finite set of general concept inclusion axioms, $\mathcal{T} = \{ C_1 \subseteq D_1, \ldots, C_n \subseteq D_n \}$, where $C_i, D_i$ are arbitrary $SHIF$-concepts. An interpretation $I$ is said to be a model of $\mathcal{T}$ iff $C_i^I \subseteq D_i^I$ holds for all $C_i \sqsubseteq D_i \in T$. $C$ is satisfiable with respect to $\mathcal{T}$ iff there is a model $I$ of $\mathcal{T}$ with $C^I \not= \emptyset$. Finally, $D$ subsumes $C$ with respect to $\mathcal{T}$ iff for each model $I$ of $\mathcal{T}$ we have $C^I \subseteq D^I$. The following lemma shows how general concept inclusion axioms can be internalised using a “universal” role $U$, a transitive super-role of all roles occurring in $\mathcal{T}$ and their respective inverses.

Lemma 3 Let $\mathcal{T}$ be terminology and $C, D$ be $SHIF$-concepts and let $C_T := \bigwedge_{C \sqsubseteq D \in T} \neg C_i \sqcup D_i$.

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

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

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

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

3 Reasoning for $SI$ Logics

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

Please note that SHITF no longer has the finest model
property: for example the following concept, where \( R \) is a
transitive super-role of \( F \), is satisfiable, but each of its mod-
els has an infinite domain.

\[-C \sqcap \exists F^-, C \sqsubseteq \exists F^- \land \forall R^- (\exists F^- . (C \sqsubseteq \exists F^-))]\]

The correctness of the algorithms can be proved by showing
that they create a tableau for a concept if it is satisfi-
ble. 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 SHITF-concept can easily be
transformed to an equivalent one in NNF by pushing nega-
tions inwards [Hollund et al., 1990].

**Definition 5** Let \( D \) be a SHITF-concept in NNF, \( R_D \),
the set of roles occurring in \( D \) together with their inverses,
and \( sub(D) \) the subconcepts of \( D \). Then \( T = (S, L, E) \) is a tab-
leau for \( D \) if \( S \) is a set of individuals, \( L : S \rightarrow 2^{\\text{NNF}(D)} \)
maps each individual to a set of concepts, \( E : R_D \rightarrow 2^{S \times S} \)
maps each role to a set of pairs of individuals, and there is some
individual \( s \in S \) such that \( D \in L(s) \). Furthermore, for all \( s, t \in S, C, E \in sub(D), \) and \( R, S \in R_D \), it holds that:
1. if \( C \in L(s) \), then \( \neg C \notin L(s) \)
2. if \( C \sqcap E \in L(s) \), then \( C \in L(s) \land E \notin L(s) \)
3. if \( C \sqcup E \in L(s) \), then \( C \notin L(s) \land E \in L(s) \)
4. if \( \forall R, C \in L(s) \) and \( (s, t) \in E(R) \), then \( C \in L(t) \).
5. if \( \exists R, C \in L(s) \), then there is some \( t \in S \) such that
   \( (s, t) \in E(R) \) and \( C \in L(t) \).
6. if \( \forall S, C \in L(s) \) and \( (s, t) \in E(R) \) for some \( R \in S \) with
   \( \text{Trans}(R) \), then \( \forall RV, C \in L(t) \).
7. if \( (s, t) \in E(R) \) iff \( (t, s) \in E(\text{Inv}(R)) \).
8. if \( (x, y) \in E(R) \) and \( R \in S \), then \( (x, y) \in E(S) \).
9. if \( \exists_1 R \in L(s) \), then \( \exists t \{ (s, t') \in E(R) \} \leq 1 \).
10. if \( \exists_2 R \in L(s) \), then \( \exists t \{ (s, t') \in E(R) \} \geq 2 \).

Tableaux for SI-concepts are defined analogous and
must satisfy Properties 1-7, where, due to the absence of a
role hierarchy, \( \Xi \) is the identity.

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

**Lemma 6** A SHITF-concept (resp. SI-concept) \( D \) is satis-
ifiable iff \( D \) has a tableau.

### 3.1 Reasoning in SHITF

In the following, we give an algorithm that, given a SHITF-
concept \( D \), decides the existence of a tableau for \( D \).

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

Given a completion tree, a node \( x \) is called an \( R \)-successor
of a node \( y \) if \( y \) is a successor of \( x \) and \( x \in L((x, y)) \)
for some \( S \in \mathbb{R} \). A node \( y \) is called an \( R \)-neighbour of \( x \) if \( x \) is an \( \text{Inv}(R) \)-successor of \( y \). Predecessors and ancestors are defined as usual.

A node is blocked iff it is directly or indirectly blocked. A
node \( x \) is directly blocked iff none of its ancestors are blocked,
and it has ancestors \( x', y \) and \( y' \) such that
1. \( x \) is a successor of \( x' \) and \( y \) is a successor of \( y' \) and
2. \( L(x) = L(y) \land L(x') = L(y') \) and
3. \( L((x', y)) = L((y', y)) \).

In this case we will say that \( y \) blocks \( x \).

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

For a node \( x \), \( L(x) \) is said to contain a clash iff \( \{ A, \neg A \} \subseteq L(x) \) or \( \{ \exists_2 R, \exists_1 S \} \subseteq L(x) \) for roles \( R \in S \). A completion
tree is called clash-free iff none of its nodes contains a clash;
it is called complete iff none of the expansion rules in Figure 2
is applicable.

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

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

**Lemma 8** Let \( D \) be an SHITF-concept.

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

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

The \( \forall \), \( \exists \), \( \land \), \( \lor \), and \( \forall \)-rules are the standard ALC tableaux
rules [Schmidt-Schaub & Smolka, 1991]. The \( \forall \)-rule is used
to handle transitive roles, where the \( \Xi \)-clause deals with the
role hierarchy. See [Horrocks & Sattler, 1999] for details.

The functional restriction rules merit closer consideration.
In order to guarantee the satisfaction of a \( \exists_2 R \)-constraint, the
\( \exists \)-rule creates two successors and uses a fresh atomic concept
\( A \) to prohibit identification of these successors by the \( \leq \)-rule.
If a node \( x \) has two or more \( R \)-neighbours and contains a
functional restriction \( \leq_1 R \), then the \( \leq \)-rule merges two of the
\[ \]
Theorem 11 The tableau algorithm is a decision procedure for satisfiability and subsumption of $\mathcal{SI}$-concepts.

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

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

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

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

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

Lemma 12 The paths of a completion tree for a concept $D$ have a length of at most $m^4$ where $m = \lceil \log |D| \rceil$.

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

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

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

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

We are currently working on an algorithm for $SHIN$ based on the $SHFL$-algorithm already presented. Like earlier DLs that combine a hierarchy of (transitive and non-transitive) roles with some form of number restrictions [Horrocks & Sattler, 1999; Horrocks et al., 1998], $SHIN$ will only allow simple rules in restrictions. The justification for this limitation has been partly on the grounds of a doubtful semantics (of functional roles) and partly to simplify decision procedures. In this section we will show that, at least with respect to $SHIN$, allowing arbitrary rules in $SHIN$ number restrictions leads to undecidability. For convenience, we will refer to $SHIN$ with arbitrary rules in number restrictions as $SHIN^+$. The undecidability proof uses a reduction of the domino problem [Berger, 1966] adapted from [Baader & Sattler, 1996]. This problem asks if, for a set of domino types, there exists a tiling of an $N^2$ grid such that each point of the grid is covered with one of the domino types, and adjacent dominoes are “compatible” with respect to some predefined criteria.

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

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

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

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

- $A \subseteq \neg B \land \neg C \land \neg D \land \exists X_1. B \land \exists Y_1. C \land \leq 3S_{11}$
- $B \subseteq \neg A \land \neg C \land \neg D \land \exists X_2. A \land \exists Y_1. D \land \leq 3S_{21}$
- $C \subseteq \neg A \land \neg B \land \neg D \land \exists X_1. D \land \exists Y_2. A \land \leq 3S_{12}$
- $D \subseteq \neg A \land \neg B \land \neg C \land \exists X_2. C \land \exists Y_2. B \land \leq 3S_{22}$

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

5 Discussion

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

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

An extension of the embedding technique can be used to eliminate number restrictions [De Giacomo & Lenzerini, 1995], but requires a target logic which supports the transitive closure of roles, i.e., converse-PDL. The even larger number of axioms which this embedding would introduce makes it unlikely that tractable reasoning could be performed on the resulting terminology. Moreover, we are not aware of any algorithm for converse-PDL which does not employ a so-called cut rule [De Giacomo & Massacci, 1998], the application of which introduces considerable additional non-determinism. It seems inevitable that this would lead to a further degradation in empirical tractability.
In order to fully capture the above mentioned encoding of database schemata, it would be necessary not only to extend our results to $SHIN$ but to $SHIQ$ by adding qualifying number restrictions [Hollunder & Baader, 1991]. The extension of the $SHIF$ algorithm, and tests of its behaviour in applications, will also be part of future work.

References


