A Procedure for Description Logic $\mathcal{ALCFI}$

Yu Ding and Volker Haarslev

Department of Computer Science and Software Engineering
Concordia University, Montreal, Quebec, Canada
{ding_yu|haarslev}@cse.concordia.ca

Abstract. A combination of *inverse roles* and *functional restrictions* makes the underlying description logics (DLs) lose the *finite model property.* Consequently, sophisticated *cycle detection and termination* mechanisms are employed to terminate searching of models that potentially admit only an infinite domain. In this paper, we study the logic $\mathcal{ALCFI}$ and present a tableau-based decision procedure that uses only a *static termination condition.* To achieve this nice property, a preprocessing step is to be performed to convert a source problem to a target problem. This conversion preserves equisatisfiability. As a consequence of this conversion, the tableau-based decision procedure for the concept satisfiability test w.r.t. a set of *general concept inclusions*[BCM$^+$03] (a.k.a. general axioms) is ExpTime for $\mathcal{ALCFI}$.

1 Motivation and Brief Introduction

For description logics (DLs) with inverse roles, tableau-based decision procedures are designed to accommodate *backward propagations* of constraints, and the *termination conditions* (a.k.a. cycle detection/blocking techniques) are required to be dynamic. For logics lacking the *finite model property*, a more sophisticated strategy (e.g. the pair-wise blocking technique) is introduced to guarantee soundness. A study of the model properties of DLs with inverse roles shows that various dynamic termination conditions are sufficient. Generally speaking, the bidirectional traversal of inverse relationships is a natural consequence from enforcing the dynamic termination conditions. However, there are some negative effects from bidirectional traversals, e.g., the *tableau caching* technique turns unsound[BCM$^+$03]. Further, a complaint usually heard is that a dynamic checking of termination conditions is less efficiently implementable than the *static termination condition.* In line with this thinking, it is ideal to investigate if the static termination condition could adapt to tableau-based decision procedures for concept satisfiability tests w.r.t. general axioms in $\mathcal{ALCFI}$.

The logic $\mathcal{ALCFI}$ is well studied in the literature. It extends the basic DL $\mathcal{ALC}$ with both *inverse roles* and *functional restrictions.* The letter $\mathcal{F}$ denotes functional restriction[1] [BCM$^+$03]. A functional restriction is of the form $(\leq_1 R)$ which restricts the number of R-neighbors to be no more than one. A functional restriction can be viewed as a partial function over the interpretation

[1] Please note that $\mathcal{F}$ was for *feature agreement* in earlier years, but now it is used for *functional restriction.*

domain[Lut99]. It is known that this logic has the tree model property. A simple example in $\mathcal{ALCFI}$ that admits only an infinite model is the satisfiability problem of $\neg A \sqcap \exists R^-.C$ w.r.t. $\mathcal{T} = \{\top \sqsubseteq (\leq_1 R^-), \top \sqsubseteq \exists R.A\}$. To search for a model, the dynamic *pair-wise blocking technique* is usually used [HS99]. Generally speaking, a node x is blocked if none of its *ancestors* are blocked, and it has a *witness* x' (one ancestor node) such that the labels of the two meet certain prior conditions. For example, the static *equality blocking technique* commonly used requires $\mathcal{L}(x) = \mathcal{L}(x')$ where $\mathcal{L}(x)$ is the initial label for x. The tableau procedures can ignore those blocked nodes without sacrificing soundness.

In this paper, we first introduce a technique that transforms a source problem to a target problem and preserves equisatisfiability within $\mathcal{ALCFI}$. By this conversion, the previous example can be turned into another problem which asks about the satisfiability of $\neg A \sqcap \exists R^-.C \sqcap B$ w.r.t. $\mathcal{T} = \{\top \sqsubseteq (\leq_1 R^-), \top \sqsubseteq \exists R.A, \top \sqsubseteq \forall R.((\geq_2 R^-) \sqcup \neg B) \sqcup C\}$. The search of a model for the converted problem can use the static label equality termination mechanism (a.k.a. the equality blocking technique[BCM+03]). Based on the static blocking mechanism[2], we demonstrate a tableau-based decision procedure that runs in an exponential time in the worst case for the concept satisfiability problem in $\mathcal{ALCFI}$.

2 Syntax and Semantics

We use A for *atomic concept*, use C and D for arbitrary concepts, use R (and R^-)[3] for role names. The concept formulae in $\mathcal{ALCFI}$ are formed as following:

$C, D := \top | A | \neg C | C \sqcap D | C \sqcup D | \exists R.C | \forall R.C | \leq_1 R | \geq_2 R$

An interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$ consists of a non-empty domain $\Delta^{\mathcal{I}}$, and an interpretation function $.^{\mathcal{I}}$ which maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role name to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that

$(\top)^{\mathcal{I}} = \Delta^{\mathcal{I}}$ $\quad$ $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$

$(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$ $\quad$ $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$

$(\exists R.C)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} |$ There is $b \in \Delta^{\mathcal{I}}$ with $(a, b) \in R^{\mathcal{I}}$ and $b \in C^{\mathcal{I}}\}$

$(\forall R.C)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} |$ For all $b \in \Delta^{\mathcal{I}}$, $(a, b) \in R^{\mathcal{I}}$ implies $b \in C^{\mathcal{I}}\}$

$(\leq_1 R)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} | \forall b, c \in \Delta^{\mathcal{I}}$, $(a, b) \in R^{\mathcal{I}}$ and $(a, c) \in R^{\mathcal{I}}$ implies $b = c\}$

$(\geq_2 R)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} | \exists b, c \in \Delta^{\mathcal{I}}$ s.t. $(a, b) \in R^{\mathcal{I}}$ and $(a, c) \in R^{\mathcal{I}}$ and $b \neq c\}$

A *role* is interpreted as a binary relation on a subset of elements of the interpretation domain $\Delta^{\mathcal{I}}$. The related *inverse role* is interpreted as the inverse of that binary relation. For example, let $x, y \in \Delta^{\mathcal{I}}$, $(x, y) \in R^{\mathcal{I}}$ iff $(y, x) \in (R^-)^{\mathcal{I}}$, where R^- and R are two roles in inverse relation. Simply each binary relation has a unique inverse relation; and the inverse of an inverse relation is the original

[2] Comparing with the common static blocking techniques, ours uses slightly different static blocking conditions. For details about the specific static blocking that the decision procedure uses, we leave to the subsection on soundness.

[3] For brevity, here R^- is used as the (unique) role name that has an inverse relationship to the role name R. Note that this usage is different from the convention.

relation itself. In the paper we assume each role has a unique inverse role. For a role R, for example, we consider R^- as the only *inverse role*[4] to R.

A Tbox $\mathcal{T}$ of general axioms is a set of axioms of the form $\top \sqsubseteq C$[5]. An $\mathcal{ALCFI}$ concept C is satisfiable w.r.t. a Tbox $\mathcal{T}$ iff C and $\mathcal{T}$ have a model (i.e. non-empty interpretation) in common.

3 Preprocess

We require concept formulae/expressions are put in Negation Normal Form (NNF) by pushing negation signs inward to concept names. We introduce a preprocessing operation for the input concept formula E_0 and the Tbox $\mathcal{K}_0$ of general axioms of the form $\top \sqsubseteq C$, where E_0 and C are in NNF.

Definition 1. *The operation $tag(.)$ on the expression (incl. axiom) x is:*
(1) if x is $C \sqcap D$, then $tag(x) = tag(C) \sqcap tag(D)$;
(2) if x is $C \sqcup D$, then $tag(x) = tag(C) \sqcup tag(D)$;
(3) if x is $\exists R.C$, then $tag(x) = P(x) \oplus \exists R.(tag(C))$;
(4) if x is $\forall R.C$, then $tag(x) = Q(x) \oplus \forall R.(tag(C))$;
(5) if x is $\top \sqsubseteq C$, then $tag(x) = \top \sqsubseteq tag(C)$;
(6) otherwise $tag(x) = x$.
where $P(x)$, $Q(x)$ are fresh names unique for each x; C, D are subformulae; the symbol $\oplus$ represents the conjunction operator in exactly the same way as $\sqcap$.

E_0/E_1 denote the formulae before and after tagging; $\mathcal{K}_0/\mathcal{K}_1$ denote the Tboxes before and after tagging. By the tagging operation, it is able to collect tagged *universal constraints* of the form $Q(x) \oplus \forall R.(tag(C))$ into a set $\mathcal{U}(R)$, and to collect tagged *existential constraints* of the form $P(x) \oplus \exists R.(tag(C))$ into a set $\mathcal{E}(R)$, where R is a role, and $\mathcal{U}$ and $\mathcal{E}$ are sets indexed by different roles. Let x denote any (sub)formula of E_0 and $\mathcal{K}_0$, we have:

- for $x = \exists R.C$, there is $P(x) \oplus \exists R.tag(C) \in \mathcal{E}(R)$;
- for $y = \forall R.D$, there is $Q(x) \oplus \forall R.tag(D) \in \mathcal{U}(R)$;

Definition 2. *Initialize $\mathcal{K}_a = \emptyset$. For $\alpha \in \mathcal{E}(*)$ and $\beta \in \mathcal{U}(*)$ where $*$ denote any role name, perform the following operations:*
(1) if α is $P(x) \oplus \exists R.tag(C) \in \mathcal{E}(R)$, then
$\mathcal{K}_a := \mathcal{K}_a \cup \{\top \sqsubseteq \forall R^-.(\neg P(x) \sqcup \geq_2 R) \sqcup tag(C)\}$;
(2) if β is $Q(x) \oplus \forall R.tag(D) \in \mathcal{U}(R)$, then
$\mathcal{K}_a := \mathcal{K}_a \cup \{\top \sqsubseteq \forall R^-.\neg Q(x) \sqcup tag(D)\}$.

Hereafter, we use $\mathcal{K}_2$ to denote $\mathcal{K}_a \cup \mathcal{K}_1$, and E_2 to denote E_1. The following tableaux procedure works on E_2 and $\mathcal{K}_2$, which is still in the logic $\mathcal{ALCFI}$.

[4] It takes a linear cost to identify equivalent roles that are implied by inverse relationship declarations in a namespace (of role names).
[5] Equality axioms of the form $C \equiv D$ are converted to inclusion axioms like $C \sqsubseteq D$ and $D \sqsubseteq C$. An inclusion axiom of the form $C \sqsubseteq D$ can be converted to $\top \sqsubseteq \neg C \sqcup D$.

4 A Calculus and An Algorithm for $\mathcal{ALCFI}$

We consider the converted problem. Without loss of generality, we require the Tbox of a set of general axioms is simplified and expressed as one general axiom $\top \sqsubseteq G_2$. To be precise, $G_2 = \sqcap r_i$ for all $\top \sqsubseteq r_i \in \mathcal{K}_2$. The following is a set of tableaux expansion rules for (the converted problem in) $\mathcal{ALCFI}$.

```
$\sqcap$-rule: if   1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$, $x$ is not blocked, and
                    2. not $\{C_1, C_2\} \subseteq \mathcal{L}(x)$
               then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule: if   1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$, $x$ is not blocked, and
                    2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
               then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_i\}$ for some $C_i \in \{C_1, C_2\}$
$\exists$-rule: if  1. $\exists R.C \in \mathcal{L}(x)$, $x$ is not blocked, and
                    2. $(\leq_1 R) \notin \mathcal{L}(x)$ and $x$ has no $R$-neighbor $y$ with $C \in \mathcal{L}(y)$
               then create a successor node $y$ with $\mathcal{L}(\langle x, y\rangle) := \{R\}$ and $\mathcal{L}(y) := \{C\}$;
               if   1. $\exists R.C \in \mathcal{L}(x)$, $x$ is not blocked, and
                    2. $(\leq_1 R) \in \mathcal{L}(x)$ and $x$ has no $R^-$-predecessor, and
                    3. $x$ has no $R$-successor $y$
               then create a successor node $y$ with $\mathcal{L}(\langle x, y\rangle) := \{R\}$ and $\mathcal{L}(y) := \{C\}$;
               if   1. $\exists R.C \in \mathcal{L}(x)$, $x$ is not blocked, and
                    2. $(\leq_1 R) \in \mathcal{L}(x)$ and $x$ has no $R^-$-predecessor, and
                    3. $x$ has an $R$-successor $y$
               then $\mathcal{L}(y) := \mathcal{L}(y) \cup \{C\}$
$\geq$-rule: if     1. $(\geq_2 R) \in \mathcal{L}(x)$, $x$ is not blocked, and
                    2. $x$ has no $R$-neighbor $y$
               then create a successor node $y$ with $\mathcal{L}(\langle x, y\rangle) := \{R\}$ and $\mathcal{L}(y) := \emptyset$
$\forall$-rule: if  1. $\forall R.C \in \mathcal{L}(x)$, $x$ is not blocked, and
                    2. there is an $R$-successor $y$ of $x$ with $C \notin \mathcal{L}(y)$
               then $\mathcal{L}(y) := \mathcal{L}(y) \cup \{C\}$
```

Fig-1. The tableaux expansion rules for $\mathcal{ALCFI}$

We use the common notion of a *completion tree* and the relevant notations. Given a completion tree, a node y is called an *R-neighbor* of a node x if y is an R-successor of x, or if x is an R^--successor of y. Predecessor nodes are deliberately ignored in some of the rules above. The correctness relies on the *back-propagation-don't-care property* to be introduced in the subsection on soundness.

To obtain a model which is potentially infinite, it is necessary to apply the *unraveling technique* on the completion tree using the notion of paths[HS99]. Instead of going to a directly blocked node, the path goes to the blocking node. Thus, if blocking occurred, an infinite model can be obtained through *unraveling* (in cases when an ancestor node and a descendant node are involved).

To generalize the notion of the *clash triggers* [BCM$^+$03] for $\mathcal{ALCFI}$, we introduce the notion of $\perp$-set (a.k.a. `Nogood`) which was originally introduced in [DM99]. To avoid generating irrelevant $\perp$-sets (which would be the case if the $\perp$-rules were considered as a parallel calculus to derive inconsistent concept sets bottom-up rather than top-down), inconsistency inference based on the $\perp$-rules are carried out on demand by tableau procedures[DM99].

The following are the inconsistency propagation rules for $\bot$-set.

$\bot$-0: $\{\neg\top\} \in \bot$-sets.
$\bot$-1: $\{C, \neg C\} \in \bot$-sets.
$\bot$-2: $\{(\leq_1 R), (\geq_2 R)\} \in \bot$-sets.
$\bot$-3: if $\alpha \cup \{G_2\} \in \bot$-sets, then $\alpha \in \bot$-sets.
$\bot$-4: if $\alpha \in \bot$-sets and $\alpha \subseteq \beta$, then $\beta \in \bot$-sets.
$\bot$-5: if $\alpha \cup \{C\} \in \bot$-sets and $\alpha \cup \{D\} \in \bot$-sets, then $\alpha \cup \{C \sqcup D\} \in \bot$-sets.
$\bot$-6: if $\alpha \in \bot$-set and $\beta = \{\exists R.E_0, \forall R.E_1, ..., \forall R.E_n\}$(where R is any role) and $\alpha \subseteq \{E_0, E_1, E_2, ..., E_n\}$, then $\beta \in \bot$-sets.
$\bot$-7: if $\alpha \in \bot$-set and $\beta = \{(\geq_2 R), \forall R.E_1, ..., \forall R.E_n\}$(where R is any role) and $\alpha \subseteq \{E_1, E_2, ..., E_n\}$, then $\beta \in \bot$-sets.

Fig-2. The inconsistency propagation rule for $\mathcal{ALCFI}$

To get a sound and complete exponential-time decision procedure, certain search strategies must be integrated with the set of expansion/propagation rules. The following outlines two procedures that cooperate and decide the concept satisfiability problem w.r.t. a set of general axioms in $\mathcal{ALCFI}$. A procedure call of $TEST(E_2, G_2)$ gives the satisfiability result of E_2 w.r.t. $\mathcal{K}_2$.

```
PROCEDURE $TEST(C, GCI)$
[01]Nogood := $\emptyset$;
[02]WHILE (true)
[03]        Witness := $\emptyset$;
[04]        $len$ := $sizeof$(Nogood);
[05]        allocate a tableau node $x_0$ and $\mathcal{L}(x_0) = \{C\}$;
[06]        $SAT(x_0, GCI, null, null)$;
[07]        IF ($C \in$ Nogood) Return unsatisfiable;
[08]        IF ($len$ == $sizeof$(Nogood)) Return satisfiable;
[09]ENDWHILE
END-PROCEDURE {TEST}
```

The procedure $TEST(.,.)$ takes a concept C and a general axiom GCI as input. It invokes the sub-procedure $SAT(.,.,.,.)$ in the WHILE-loop.

Nogood and Witness are two abstract global data structures to be considered as sets (of set of formulae). Nogood is initialized only once and memorizes everything augmented by the sub-procedure $SAT(.,.,.,.)$. Witness, used for the *static blocking* purpose, is however emptied in each loop before invoking $SAT(.,.,.,.)$.

The utility function $sizeof(.)$ gets the number of elements of a set. The variable len is used for keeping the number of current elements in Nogood.

There is a special tableau node named x_0 and $\mathcal{L}(x_0)$ is a set of labels (a singleton set of the concept that is subject to satisfiability test).

There are two cases to exit the infinite loop. The first is when the input concept C is found in the global data structure Nogood, and $TEST(.,.)$ decides the problem as unsatisfiable. The second is when Nogood is not augmented by the sub-procedure $SAT(.,.,.,)$, and $TEST(.,.)$ decides the problem as satisfiable.

PROCEDURE $SAT(x, G, parent, edge2parent)$

```
[10]  IF ($\mathcal{L}(x) \in$ Nogood) Return $false$;
[11]  IF ($\mathcal{L}(x) \in$ Witness) Return $true$;
[12]  $\mathcal{BS}$ := all the propositional branches of $\mathcal{L}(x) \cup \{G\}$;
[13]  selected := $false$;
[14]  FOR each $\mathcal{B} \in \mathcal{BS}$
[15]      IF ($\mathcal{B} \in$ Nogood) Continue;
[16]      IF ($true == hasclash(\mathcal{B})$) THEN
[17]          Nogood := Nogood $\cup\{\mathcal{B}\}$;
[18]          Terminate;
[19]      ENDIF
[20]      IF ($\mathcal{B} \in$ Witness) THEN
[21]          Witness := Witness $\cup\{\mathcal{L}(x)\}$;
[22]          Return $true$;
[23]      ENDIF
[24]      selected := $true$;
[25]     Break;
[26]  ENDFOR
[27]  IF ($false ==$ selected ) THEN
[28]      Nogood := Nogood $\cup\{\mathcal{L}(x)\}$;
[29]      Terminate;
[30]  ENDIF
[31]  $Successors(\mathcal{B}, Succ(x, .), edge2parent)$;
[32]  $AllSuccessors(\mathcal{B}, Succ(x, .))$;
[33]  Witness := Witness $\cup$ $\{\mathcal{B}\}$;
[34]  Witness := Witness $\cup$ $\{\mathcal{L}(x)\}$;
[35]  FOR each successor $s \in Succ(x, R)$     ($comment$: $R$ is any role name)
[36]      ret := $SAT(s, G, x, R^-)$;
[37]      IF ($false ==$ ret) THEN
[38]          Nogood := Nogood $\cup$ $\{\mathcal{B}\}$;
[39]          Terminate;
[40]      ENDIF
[41]  ENDFOR
[42]  Return $true$;
END-PROCEDURE {SAT}
```

The instruction `Terminate` facilitates the *restart strategy*. An execution of Line-18 or Line-19 or Line-39 will transfer the program control to Line-07.

Line-15 uses `Continue`, when executed, the program control goes to Line-14.

Line-35 to Line-41 performs a depth-first exploration of the tableau tree.

Line-17, Line-28, Line-38, and *hashclash*(.) implement the $\perp$-rules. The utility function *hashclash*(.) implements the detection of primitive clashes.

Line-11 and Line-20 to Line-23 implement the specific *static blocking* and works on a new notion of *propositional branches*. A propositional branch abstracts the exhaustive application of the $\sqcap$-rule and the $\sqcup$-rule over the topmost $\sqcap$ and $\sqcup$ operators in $\mathcal{L}(x) \cup \{G\}$. It takes a cost in a single exponential function of the number of affected propositional operators (outside *role fillers*[BCM$^+$03]) to enumerate all propositional branches. The soundness largely relies on the *back-propagation-don't-care* property introduced in a later subsection on soundness.

$Succ(x, .)$ is a data structure holding a set of successors of node x indexed by different role names.

```
PROCEDURE $Successors(\mathcal{B}, Succ(x, .), edge2parent)$
[43]  FOR each existential-restriction $e \in \mathcal{B}$   ($comment$: $e$ is of form $\exists R.C$)
[44]      CASE $((\leq_1 R) \in \mathcal{B})$ and $(edge2parent \neq R)$:
[45]          IF $(Succ(x, R) == \emptyset)$ THEN
[46]              allocate a new tableau node $s$;
[47]              $Succ(x, R) := Succ(x, R) \cup \{s\}$;
[48]              $\mathcal{L}(s) := \{C\}$;
[49]              $\mathcal{L}(\langle x, s\rangle) = \{R\}$;
[50]          ELSE
[51]              $s :=$ the single element in $Succ(x, R)$;
[52]              $\mathcal{L}(s) := \mathcal{L}(s) \cup \{C\}$;
[53]          ENDIF
[54]      CASE $((\leq_1 R) \notin \mathcal{B})$:
[55]          IF ($x$ has no $R$-neighbor $y$ s.t. $C \in \mathcal{L}(y)$) THEN
[56]              allocate a new tableau node $s$;
[57]              $Succ(x, R) := Succ(x, R) \cup \{s\}$;
[58]              $\mathcal{L}(s) := \{C\}$;
[59]              $\mathcal{L}(\langle x, s\rangle) := \{R\}$;
[60]          ENDIF
[61]      ENDCASE
[62]  ENDFOR
END-PROCEDURE {Successors}
```

```
PROCEDURE $AllSuccessors(\mathcal{B}, Succ(x, .))$
[63]  FOR each universal-restriction $v \in \mathcal{B}$   ($comment$: $v$ is of form $\forall R.C$)
[64]      IF ($x$ has no $R$-neighbor) and $((\geq_2 R) \in \mathcal{B}(x))$ THEN
[65]          allocate a new tableau node $s$;
[66]          $\mathcal{L}(\langle x, s\rangle) = \{R\}$;
[67]          $\mathcal{L}(s) = \emptyset$;
[68]          $Succ(x, R) = Succ(x, R) \cup \{s\}$;
[69]      ENDIF
[70]      FOR each $s \in Succ(x, R)$   ($comment$: $R$ stands for any role)
[71]          $\mathcal{L}(s) := \mathcal{L}(s) \cup \{C\}$;
[72]      ENDFOR
[73]  ENDFOR
END-PROCEDURE {AllSuccessors}
```

$Successors(\mathcal{B}, Succ(x, .), edge2parent)$ implements the $\exists$-rule and the $\leq$-rule. $Succ(x, .)$ is an output parameter. $\mathcal{B}$ and $edge2parent$ are input parameters.

$AllSuccessors(\mathcal{B}, Succ(x, .))$ implements a combination of the $\forall$-rule and the $\geq$-rule. $Succ(x, .)$ is an input and output parameter. $\mathcal{B}$ is an input parameter.

5 Equisatisfiability of the Conversion

Here is a quick explanation for the *tagging operation*. Let λ be the set of all modal constraints for $\mathcal{K}_1 \cup E_1$, and let $\{Q_1, Q_2, ..., Q_m\}$ be the set of unique names (Ramsey atoms[6]) introduced. Roughly speaking, the $tag(.)$ operation essentially provides two functions: (1) a multi-valued function $f : \lambda \to 2^{\{Q_1,Q_2,...,Q_m\}}$ such that if $x \neq y$ then $f(x) \cap f(y) = \emptyset$; (2) a single-valued function $g : \{Q_1, Q_2, ..., Q_m\} \to \lambda$. The space of Ramsey atoms is partitioned according to different modal constraints. This partitioning is the essential utility provided with the *tagging operation*.

A few comments are necessary: (1) $C \sqcap D$ is a *conjunction* which has two *conjuncts* C and D; (2) Formulae like $\exists R.C$ are *existential constraints*, formulae like $\forall R.C$ are *universal constraints*. Both of them are called *modal constraints* and are generally denoted as $\overset{\exists}{\forall}R.tag(C)$ hereafter; (3) The $tag(.)$ operation (recursively) maps each occurrence of a specific *modal constraint* to a conjunction having two conjuncts, one of which is a Ramsey atom; (4) In each tagged constraint we stipulate that the Ramsey atom is on the left and the modal constraint is on the right. This excludes cases like $\overset{\exists}{\forall}R.tag(C) \sqcap Q(x)$.

Definition 3. (p-model) *Given E_1 and $\mathcal{K}_1$ the tagged formula and the tagged Tbox, $Q(x) \sqcap \overset{\exists}{\forall}R.tag(C)$ a tagged constraint in E_1 and $\mathcal{K}_1$, a p-model for E_1 and $\mathcal{K}_1$ is a model $(\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ such that for any $n \in \Delta^{\mathcal{I}}$ there are*
(1) if $n \in (Q(x))^{\mathcal{I}}$, then $n \in (\overset{\exists}{\forall}R.tag(C))^{\mathcal{I}}$; and
(2) if $n \in (\overset{\exists}{\forall}R.tag(C))^{\mathcal{I}}$ where $\overset{\exists}{\forall}R.tag(C)$ was tagged, then there is some Ramsey atom $Q(x)$ for $\overset{\exists}{\forall}R.tag(C)$ such that $n \in (Q(x))^{\mathcal{I}}$.

Lemma 1. *If E_1 and $\mathcal{K}_1$ is satisfiable, then it is satisfiable in a p-model.*

Proof. Given E_1 and $\mathcal{K}_1$ as the converted objects and $(\Delta^{\mathcal{I}_1}, .^{\mathcal{I}_1})$ be a model for them. To construct a p-model $(\Delta^{\mathcal{I}_2}, .^{\mathcal{I}_2})$, consider each $n \in \Delta^{\mathcal{I}_1}$ having:

(1) $n \in (Q_i(x))^{\mathcal{I}_1}$ but there is no $\overset{\exists}{\forall}R.tag(C)$ s.t. $n \in (\overset{\exists}{\forall}R.tag(C))^{\mathcal{I}_1}$; or

(2) $n \in (\overset{\exists}{\forall}R.tag(C))^{\mathcal{I}_1}$, but there is no $Q_i(x)$ s.t. $n \in (Q_i(x))^{\mathcal{I}_1}$.

Use the sub-model generating technique to exclude the superfluous $Q_i(x)$ or $\overset{\exists}{\forall}R.tag(C)$ in both cases when copying from n to $n' \in \Delta^{\mathcal{I}_2}$. Copying other interpretations for n unchanged to n', copying other elements in the domain, and copying role interpretations, it constructs a $(\Delta^{\mathcal{I}_2}, .^{\mathcal{I}_2})$ to meet the two requirements of being a p-model. So, we have $(\Delta^{\mathcal{I}_2}, .^{\mathcal{I}_2})$ a p-model provided that $(\Delta^{\mathcal{I}_1}, .^{\mathcal{I}_1})$ be a model. □

Lemma 2. *If E_1 and $\mathcal{K}_1$ has a model, then $\mathcal{K}_a$ is satisfiable in it.*

Proof. Suppose E_1 and $\mathcal{K}_1$ has a p-model $\mathcal{M} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ in which $\mathcal{K}_a$ is not satisfied. This means the existence of at least one axiom $s \in \mathcal{K}_a$ being violated at some element $n \in \Delta^{\mathcal{I}}$.

Case-1: By the definition of $\mathcal{K}_a$, this s corresponds to some tuple $Q(x) \oplus \forall R.tag(C) \in \mathcal{U}(R)$ for some role R. Suppose the axiom being violated at element n be $\top \sqsubseteq tag(C) \sqcup \forall R^{-}.\neg Q(x)$. This means there is some $m \in \Delta^{\mathcal{I}}$ such that

[6] In honor of Frank P. Ramsey, who in 1926 observed the C-rule (a.k.a. the Ramsey's Rule[Ram31]) which is about the equivalence of converse modalities in modal logics.

(1) $(m,n) \in R^{\mathcal{I}}$, and
(2) $m \in (Q(x))^{\mathcal{I}}$.
From the definition of p-model, we know $Q(x)^{\mathcal{I}} \subseteq (\forall R.tag(C))^{\mathcal{I}}$. So,
(3) $m \in (\forall R.tag(C))^{\mathcal{I}}$.
By (3) and (1), it follows
(4) $n \in (tag(C))^{\mathcal{I}}$.
By (4), the axiom $\top \sqsubseteq tag(C) \sqcup \forall R^{-}.\neg Q(x)$ is satisfied at n.
Case-2: Suppose this axiom s corresponds to some tuple $P(x) \oplus \exists R.tag(C) \in \mathcal{E}(R)$ for some role R. In this case, $\top \sqsubseteq tag(C) \sqcup \forall R^{-}.(\neg P(x) \sqcup \geq_2 R)$ is the axiom being violated at element n, which requires an element $m \in \Delta^{\mathcal{I}}$ such that
(1′) $(m,n) \in R^{\mathcal{I}}$, and
(2′) $m \in (P(x))^{\mathcal{I}}$, and
(3′) $m \in (\leq_1 R)^{\mathcal{I}}$.
By the definition of p-model, we know $P(x)^{\mathcal{I}} \subseteq (\exists R.tag(C))^{\mathcal{I}}$. So,
(4′) $m \in (\exists R.tag(C))^{\mathcal{I}}$.
Consider (1′) (3′) (4′), we have
(5′) $n \in (tag(C))^{\mathcal{I}}$.
By (5′), the axiom $\top \sqsubseteq tag(C) \sqcup \forall R^{-}.(\neg P(x) \sqcup \geq_2 R)$ is satisfied at n.
Since in both cases supposedly violated axioms are actually satisfied, this conculdes that a p-model for both E_1 and $\mathcal{K}_1$ is also a model for $\mathcal{K}_a$. □

Lemma 3. *E_1 is satisfiable w.r.t. $\mathcal{K}_1 \cup \mathcal{K}_a$ iff E_1 is satisfiable w.r.t. $\mathcal{K}_1$.*

Proof. (ONLY IF) Let $\mathcal{M}_1$ be a model for E_1 and $\mathcal{K}_1 \cup \mathcal{K}_a$. Observe that $\mathcal{K}_1 \cup \mathcal{K}_a$ is a superset of $\mathcal{K}_1$, E_1 is satisfiable w.r.t. $\mathcal{K}_1$ (trivially in $\mathcal{M}_1$).

(IF) Let $\mathcal{M}_2$ be a p-model for E_1 and $\mathcal{K}_1$. According to the lemma above, $\mathcal{K}_a$ is always satisfied in the p-model for both E_1 and $\mathcal{K}_1$. It follows that $\mathcal{M}_2$ is a model for E_1 and $\mathcal{K}_1 \cup \mathcal{K}_a$. □

Lemma 4. *Given a concept formula E_0 and a Tbox $\mathcal{K}_0$, let the tagged formula and Tbox be E_1 and $\mathcal{K}_1$. E_0 is satisfiable w.r.t. $\mathcal{K}_0$ iff E_1 is satisfiable w.r.t. $\mathcal{K}_1$.*

Proof. Given a model $\mathcal{M}_1$ of E_1 and $\mathcal{K}_1$, E_0 and $\mathcal{K}_0$ are trivially satisfied in $\mathcal{M}_1$. In the other direction, given a model $\mathcal{M}_0$ of E_0 and $\mathcal{K}_0$, adding Q_i accordingly as guided by $\mathcal{M}_0$ will produce a model for E_1 and $\mathcal{K}_1$. This concludes that the tagging operation (a partitioning of Ramsey atoms) preserves satisfiability. □

Theorem 1. *E_0 is satisfiable w.r.t. $\mathcal{K}_0$ iff E_1 is satisfiable w.r.t. $\mathcal{K}_1 \cup \mathcal{K}_a$.*

Proof. This immediately follows from Lemma 3 and Lemma 4. □

6 Correctness of the Procedure

6.1 Completeness

For the completeness, we need to prove the correctness for what regards concept unsatisfiability. Recall the procedure $SAT(.,.,.,.)$ and that `Nogood` is a global and permanent data structure. Taking the approach in [DM99], we start with a lemma saying that $\perp$-rules correctly propagate inconsistencies.

Lemma 5. *The $\bot$-rules generate only unsatisfiable sets.*

Proof. By induction on the application of $\bot$-rules.

Base cases. Consider rules $\bot$-0, $\bot$-1 and $\bot$-2. They are clearly unsatisfiable.

Inductive cases. Suppose the claim holds for the antecedent of each $\bot$-rule. We analyze the application of each $\bot$-rule.

- ($\bot$-3): Give C is unsatisfiable w.r.t. G_2. Consider that $\top \sqsubseteq G_2$, in every model of G_2, G_2 is equivalent to $\top$. Then it is clear that C is unsatisfiable.
- ($\bot$-4): We prove the claim by contradiction. Suppose $\alpha \subseteq \beta$, α is unsatisfiable and β is satisfiable. Let M be a model for β. Using the sub-model generating technique, there is a sub-model N of M satisfies α, and this contradicts the hypothesis that α is unsatisfiable.
- ($\bot$-5): We prove the claim by contradiction. Suppose $\alpha \sqcap C$ and $\alpha \sqcap D$ are unsatisfiable, but $\alpha \sqcap (C \sqcup D)$ is satisfiable. Let M be a model for $\alpha \sqcap (C \sqcup D)$, then either $\alpha \sqcap C$ or $\alpha \sqcap D$ is satisfied in M. This contradicts the hypothesis.
- ($\bot$-6): Give α is unsatisfiable and $\alpha \subseteq \{E_0, E_1, ..., E_n\}$. By $\bot$-4, $\{E_0, E_1, ..., E_n\}$ is unsatisfiable. Suppose β is satisfiable in a model M. This gives a sub-model for $\{E_0, E_1, ..., E_n\}$. This results in a contradiction.
- ($\bot$-7): Give α is unsatisfiable and $\alpha \subseteq \{E_1, ..., E_n\}$. By $\bot$-4, we have $\{E_1, ..., E_n\}$ is unsatisfiable. Suppose β is satisfied in a model M. This gives a sub-model for $\{E_1, ..., E_n\}$ and is in contradiction.

□

Lemma 6. *If $n \in$ `Nogood`, then n is unsatisfiable.*

Proof. Recall that `Nogood` is a data structure for storing $\bot$-sets that are obtained by application of only the $\bot$-rules in the sub-procedure $SAT(.,.,.,.)$. □

Lemma 7. (Completeness) *If E_1 is satisfiable w.r.t. G_2, then $E_1 \notin$ `Nogood`.*

6.2 Soundness

$\mathbf{T}$ denotes the tree constructed by the decision procedure. For each node $t_i \in \mathbf{T}$, $\mathcal{L}(t_i)$ denotes t_i's initial label, $\mathcal{B}(t_i)$ denotes t_i's current propositional branch.

$SAT(.,.,.,.)$ takes a depth-first traversal to expand $\mathbf{T}$ starting from its root. If t_i is explored/expanded/completed before t_j, we write $t_i \succ t_j$. The blocking relationship of nodes, written in the symbol $\tilde{\succ}$, is required to conform to the node exploration ordering[7] (in symbol $\succ$). If t_i blocks t_j, we write $t_i \tilde{\succ} t_j$.

We allow *transitive blockings* as long as the blocking node has been *propositionally completed* (i.e., it has a current propositional branch not known to be unsatisfiable and to which the $\sqcap$-rule and $\sqcup$-rule is no longer applicable.).

Only a completed node enters its label set in `Witness`. If $w \in$ `Witness` comes from a tableau node x, we write $node(w) = x$. Recall the procedure $SAT(.,.,.,.)$, if $\mathcal{L}(x)$ and $\mathcal{B}(x)$ are not in `Nogood` and have no clashes, they enter in `Witness`

[7] $\succ$ and $\tilde{\succ}$ are two binary relations on tree nodes. Since $\succ$ is acyclic, $\tilde{\succ}$ is acyclic.

right before exploring x's successors. When visiting/exploring/expanding a nascent node y, the procedure first checks if $\mathcal{L}(y)$ or $\mathcal{B}(y)$ matches any element of `Witness`. If there is $w \in$ `Witness` equals either $\mathcal{B}(y)$ or $\mathcal{L}(y)$, then y is blocked by $node(w)$. Clearly $node(w)$ is explored ahead of y, and the blocking is $node(w)\tilde{\succ}y$. Notice that if $\mathcal{B}(y) \in$ `Witness`, also $\mathcal{L}(y)$ enters in `Witness`[8]. However, `Witness` is purged (from the *restart strategy*) whenever one new `Nogood` is inferred.

Lemma 8. (**Back-Propagation-Don't-Care**) *Let Q_1 and P_1 be the Ramsey atoms respecitvely for $\forall R^-.tag(C)$ and $\exists R^-.tag(C)$. Given a node x that is completed and one of its R-successor nodes y that is only propositionally completed: (1) if $\mathcal{B}(y) \supseteq \{Q_1, \forall R^-.tag(C)\}$, then $tag(C) \in \mathcal{B}(x)$; (2) if $\mathcal{B}(y) \supseteq \{P_1, \exists R^-.tag(C), (\leq_1 R^-)\}$, then $tag(C) \in \mathcal{B}(x)$.*

Proof. (1) Consider the axiom $\top \sqsubseteq \forall R.\neg Q_1 \sqcup tag(C)$ at node x. It is impossible for x to choose $\forall R.\neg Q_1$, for otherwise y could not be propositionally completed.

(2) Consider the axiom $\top \sqsubseteq \forall R.(\neg P_1 \sqcup (\geq_2 R^-)) \sqcup tag(C)$ at node x. It is impossible for x to choose $\forall R.(\neg P_1 \sqcup (\geq_2 R^-))$, for otherwise y could not be propositionally completed. □

Lemma 9. (Soundness) *If there is a tableau tree $\boldsymbol{T}$ for E_1 w.r.t. G_2, then there is a model T for E_1 w.r.t. G_2.*

Proof. It takes two steps. We first update the tableau (completion tree) $\mathbf{T}$ to get another one $\mathbf{T}'$, and then apply the unraveling technique on $\mathbf{T}'$ to get a model.

(1) For each node $x \in \mathbf{T}$ that is not blocked, if $(\geq_2 R) \in \mathcal{B}(x)$ and x has only one R-neighbor y, then make a copy y' of y, put y' as x's R-successor, and establish the blocking $y\tilde{\succ}y'$[9]. This results in the tableau $\mathbf{T}'$.

(2) To admit an infinite model, we consider pathes in $\mathbf{T}'$. We use the mapping $\mathbf{Tail}(p)$ to return the last element in a path p. Give a path $p = [x_0, ..., x_n]$, where x_i are nodes in $\mathbf{T}'$, $\mathbf{Tail}(p)= x_n$. Paths in $\mathbf{T}'$ are defined inductively as follows:

- for the root node x_0 in $\mathbf{T}'$, $[x_0]$ is a path in $\mathbf{T}'$.
- for a path p and a node x_i in $\mathbf{T}'$, $[p, x_i]$ is a path in $\mathbf{T}'$ iff
  - x_i is not blocked, and
    - x_i is a successor of $\mathbf{Tail}(p)$, or
    - y is a successor of $\mathbf{Tail}(p)$ and x_i (transitively) blocks y.

The model $T = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ can be defined with:
$\Delta = \{x_p |\ p$ is a path in $\mathbf{T}'\ \}$
$x_p \in (\mathcal{L}(\mathbf{Tail}(p)) \sqcap \mathcal{B}(\mathbf{Tail}(p)))^{\mathcal{I}}$
$\{\langle x_p, x_q\rangle \mid \langle x_p, x_q\rangle \in (R)^{\mathcal{I}}\} = \{\langle x_p, x_q\rangle \in \Delta \times \Delta |\ q = [p,\ \mathbf{Tail}\ (q)]$ and
1. $\mathbf{Tail}(q)$ is an R-successor of $\mathbf{Tail}(p)$, or
2. $\exists y \in \mathbf{T}'$, y is an R-successor of $\mathbf{Tail}(p)$ and $\mathbf{Tail}(q)\tilde{\succ}y\ \}$

$\bigcup\quad \{\langle x_p, x_q\rangle \in \Delta \times \Delta |\ p = [q,\ \mathbf{Tail}\ (p)]$ and
1. $\mathbf{Tail}(p)$ is an R^--successor of $\mathbf{Tail}(q)$, or
2. $\exists y \in \mathbf{T}'$, y is an R^--successor of $\mathbf{Tail}(q)$ and $\mathbf{Tail}(p)\tilde{\succ}y\ \}$

T is a model for E_1 w.r.t. G_2. □

[8] See line-33 and line-34 of the procedure.

[9] The case $(\geq_2 R) \in \mathcal{B}(x)$ and x has no R-neighbor does not exist, see the $\geq$-rule and also line-64 of the procedure.

6.3 Complexity

Lemma 10. *Each execution of the procedure $SAT(.,.,.,.)$ takes a cost of $2^{O(n)}$, where n is the problem size.*

Proof. Notice that the procedure $SAT(.,.,.,.)$ terminates if a new `Nogood` is found or it stops if the tableau (completion tree) is saturated.

(1) For any tableau node x, considering $\mathcal{L}(x) \cup \{G_2\}$, the number of (topmost[10]) $\sqcap$ and $\sqcup$ operators in it is bounded by n. This allows that the propositional branch selection takes at most c^n for some constant $c > 1$.

(2) The number of expanded[11] nodes is bounded by 2^n due to the blocking strategy (using `Witness` to avoid repetitive propositional branch or initial label) in $SAT(.,.,.,.)$. The number of nodes of the final tableau structure is bounded by $2^n + 2^n * n$.

(3) The cost per node could not surpass d^n for some $d > 1$.

The total cost for one execution of $SAT(.,.,.,.)$ is thus bounded by $(c^n + d^n) * (2^n + 2^n * n)$ which is of $2^{O(n)}$. □

Lemma 11. *The procedure $TEST(.,.)$ stops in $2^{O(n)}$ loops, where n is the problem size.*

Proof. (1) The size of `Nogood` can be bounded by $h(n) = 2^{O(n)}$. This is justifiable by considering a space of (sufficient) sub-formulae for the problem in question.

(2) Each run of $TEST(.,.)$ will report at least one new `Nogood` if that run does not produce a saturated tableau. After at most $h(n)$ runs of $TEST(.,.)$, the size of `Nogood` will not grow (for the whole sub-formulae space is exhausted). At this point, $TEST(.,.)$ is able to decide the satisfiability of the problem in question and terminates. □

Theorem 2. *The concept satisfiability test w.r.t. general axioms in $\mathcal{ALCFI}$ can be decided in deterministic exponential time by tableau decision procedures.*

7 Summary and Related Work

The work in the paper is inspired by the ExpTime tableaux procedure given in [DM99]. We borrowed the *unraveling technique* for infinite models from [HS99] and used the *tree model property* result for the description logic $\mathcal{ALCFI}$.

We blurred the distinction between the *cycle detection and termination* techniques and the *global (sub)tableaux caching* techniques. The reason is that the primary purpose of both is for the termination of tableau-based decision procedures. However, the caching technique is not so scalable as the *blocking technique* in respect to DLs with inverse roles. The soundness of the conventional (sub)tableaux caching technique is a chronic problem for DLs with inverse roles. Although it is known[DM99] that the global caching of both satisfiable and

[10] We mean the $\sqcap$ and $\sqcup$ operators that are subject to the $\sqcap$-rule and the $\sqcup$-rule.

[11] We mean those nodes having successors due to the application of $\exists$-rule.

unsatisfiable label sets is sufficient to get an ExpTime tableau-based decision procedure, no example has been set up for DLs with inverse roles so far.

In [DH05][DH06], from an optimization point of view, we proposed a *profiling and compilation* approach to the soundness problem of caching. That approach is characterized by carrying out a *reachability analysis* of *potential back-propagations*. This paper is a follow-up work. Here the potential back-propagations are expressed as new axioms in the language of the logic itself.

An example is as follows. Suppose $\{\forall R.C\}$ is tested and cached as a witness. To test the satisfiability of $\neg C \sqcap \exists R^{-}.(\forall R.C)$, this *cached witness* can not be reused. An improper use of the cached witness in this scenario (and others alike) would lead to an unsound conclusion that $\neg C \sqcap \exists R^{-}.(\forall R.C)$ is satisfiable, which is actually unsatisfiable. The culprit is unanticipated propagations of constraints from the cached witness. For example, $\{\forall R.C\}$ propagates along the R^{-} edge a constraint C. This phenomenon was investigated and addressed in [DH06]. By a conversion as proposed in this paper, the source problem is converted to a target problem asking a test of $\neg C \sqcap \exists R^{-}.(A \sqcap \forall R.C)$ w.r.t. a general axiom $\top \sqsubseteq C \sqcup \forall R^{-}.\neg A$. Working on the converted problem can prevent inadvertent uses of cached witnesses. The soundness of the global (sub)tableaux caching here relies on those explicit extra axioms (introduced after the tagging operation). Obviously we have taken a different approach to addressing this problem.

The converted problem and the source problem remain in the same logic and are *equi-satisfiable*. The introduced new concept names (Ramsey atoms) is superfluous; the extra axioms actually are also redundant. Nonetheless, a combination of both the Ramsey atoms and the extra axioms leads to the *back-propagation-don't-care property*. Interestingly, the converted problem might pose a challenge to current DLs reasoners. A tableau-based DL reasoner unaware of the *back-propagation-don't-care property* would very likely suffer a dramatic performance degrading for failing to switch on its global caching functionality. Also, a reasoner with a poor caching subsystem is not expected to survive either.

In the paper, we demonstrated a decision procedure using the *restart strategy*. Whenever one new unsatisfiable set (of labels) is obtained and it is not the given problem itself, the procedure restarts. In other cases, the procedure terminates and is able to decide the satisfiability of the problem. Though the use of the *restart strategy* here is for an easy analysis of the complexity, it is also of potential advantage to run-time performance for realistic problems. The SAT community has been witnessing the performance gain from using *restart strategies* for years.

An ExpTime decision procedure for a DL with both inverse roles and functional roles would not be ready but for a polynomial-time conversion technique. This is because the proposed procedure is designed only for $\mathcal{ALCFI}$ problems with special properties. By the satisfiability-preserving conversion proposed in the paper, nonetheless this decision procedure works for $\mathcal{ALCFI}$ in general. The converted problem is of size $O(n^2)$ where n is the size of the source problem. However, a linear size expansion is possible and the idea can be found in [Lut99].

Recently, we have implemented a conversion algorithm and carried out comparison tests. The empirical analysis is reported in [DHW07]. It is observed that, as expected, the introducing of fresh concept names as well as new GCIs has a modest performance penalty[12] (of around 3 to 4 times). But for hard problems, this penalty is offset well by the global caching enabled and magnitudes of performance gain have been achieved for 50% of the test cases. It is quite promising that the proposed technique, combined with other available techniques, can be used to solve some very hard realistic problems in DLs with inverse roles.

In summary, we have presented (1) a satisfiability-preserving conversion that transforms general problems in $\mathcal{ALCFI}$ to target problems in the same logic, and (2) a tableau-based decision procedure (composed of two cooperative subprocedures) that decides the satisfiability of the target problems in deterministic exponential time, and (3) a way to use the global (sub)tableaux caching technique as freely as in the case of $\mathcal{ALC}$[DM99] for a logic with inverse roles.

References

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.

[DH05] Yu Ding and Volker Haarslev. Towards efficient reasoning for description logics with inverse roles. *DL-Workshop'05*, 2005.

[DH06] Yu Ding and Volker Haarslev. Tableau caching for description logics with inverse and transitive roles. *DL-Workshop'06*, 2006.

[DHW07] Yu Ding, Volker Haarslev, and Jiewen Wu. A new mapping from $\mathcal{ALCI}$ to $\mathcal{ALC}$. *Submitted to DL-Workshop'07*, 2007.

[DM99] Francesco M. Donini and Fabio Massacci. Exptime tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124:87–138, 1999.

[HS99] Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.

[HT00] Ian Horrocks and Stephan Tobies. Reasoning with axioms: Theory and practice. *KR-2000*, pages 285–296, 2000.

[HW06] Alexander K. Hudek and Grant Weddell. Binary absorption in tableaux-based reasoning for description logics. *DL-Workshop'06*, 2006.

[Lut99] Carsten Lutz. Complexity of terminological reasoning revisited. *LPAR-1999*, 99:181–200, 1999.

[Ram31] F. Plank Ramsey. *Truth and probability.* In The Foundations of Probability and Other Logical, Essays, R.B. Braithwaite, ed., New York: Harcourt Brace, 1931.

[12] The reason might be that the new GCIs introduced can be absorbed by current techniques (e.g., [HT00] and [HW06]) so that the penalty is not much.