Prefix Resolution
A Resolution Method for Modal and Description Logics

Carlos Areces, Hans de Nivelle, and Maarten de Rijke*

ILLC, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam,
The Netherlands. E-mail: {carlos, nivelle, mdr}@wins.uva.nl

Abstract. We provide a resolution-based proof procedure for modal
and description logics that improves on previous proposals in a number
of important ways. First, it avoids translations into large undecidable logi-
ces, and works directly on modal or description logic formulas instead.
Second, by using labeled formulas it avoids the complexities of earlier
propositional resolution-based methods for modal logic. Third, it pro-
vides a method for manipulating so-called assertional information in the
description logic setting. And fourth, we believe that it combines ideas
from the method of prefixes used in tableaux and resolution in such a
way that some of the heuristics and optimizations devised in either field
are applicable.

1 Introduction

In this paper we develop a novel direct resolution method for modal logics and
description logics. Designing resolution methods that can directly (that is, with-
out having to perform translations) be applied to modal logics, received quite a
bit of attention in the late 1980s and early 1990s, cf. [12, 17, 8]. In contrast, recent
years have witnessed an increase of attention for translation-based resolution cal-
culi for modal (and modal-like) logics; here, one translates modal languages into
a large background language (typically first-order logic), and devises strategies
that guarantee termination for the fragment corresponding to the original modal
language; see [14, 16, 9].

In parallel with these developments, the description logic community has
been very active in devising tableaux-based methods. There is some work on
devising translation-based resolution methods for description logics [20, 16], but
we are not aware of any work on direct resolution-based methods for description
logics. This is surprising for at least two reasons. First, description logics are
closely related to modal logics (see [18, 10]), and, hence, tools in one field can
easily be used in the other. Secondly, and more importantly, in contrast with
modal logic, the field of description logic has a very strong focus on decision

* Hans de Nivelle and Maarten de Rijke were supported by the Spinoza Project ‘Logic
in Action’ at ILLC, The University of Amsterdam. We thank the referees for their
careful reviews and suggestions of improvements and further work. Special thanks
to Carsten Lutz for his very detailed and helpful comments.
methods and computational tools, and, surely, resolution-based methods rather than tableaux-based methods provide the basis for most of today's powerful computational logic tools, and so it seems natural to try and apply the former in the setting of description logics.

Now, translation-based approaches to resolution for modal and description logics suffer from the drawback that termination becomes a highly non-trivial task as we are now working within full first-order logic. Existing direct resolution methods for modal logics, on the other hand, lack the elegance and efficiency of the original resolution method because they need to perform 'cuts' inside modal operators to achieve completeness.

In this paper we develop a direct resolution method for modal and description logics that retains as much as possible of the lean 'one-rule' character of traditional resolution methods. The key idea introduced here is to use labels to decorate formulas with additional information. These labels encode accessibility relations (or, in description logic terms, roles) as well as worlds (or, objects). The use of labels allows us to avoid the complexities involved in previous proposals for direct resolution methods for modal logics. The intuition is that labels make information explicit as we need it, so that the basic resolution rule only needs to be used 'at the top level.'

The main achievements of this paper can be summarized as follows:

- it proposes a resolution method that does not involve skolemization beyond the use of constants;
- it presents an elegant and direct propositional resolution calculus for classical modal and description logic;
- description logics split information in two kinds: A-boxes which contain assertional information (facts about a particular domain), and T-boxes with terminological knowledge (definitions of derived notions). As far as we know, our proposal is the first one to account for assertional information with a propositional resolution approach;
- our method is hybrid and conservative in more than one sense: it allows one to adopt ideas from different fields and amalgamate them together.

The rest of the paper is organized as follows. Because of space limitations we will restrict our attention to the description logic $\mathcal{ALC}$ and its extension $\mathcal{ALCQ}$; but the similarities between $\mathcal{ALC}$ and the basic multi-modal logic $\mathbf{K}_m$ are well-known [18], and they should allow anyone to transfer our results to the modal setting without problems. In Section 2 we provide some basics on description logic, and in Section 3 we present a resolution method for the description logic $\mathcal{ALCQ}$. Then, in Section 4 we discuss various extensions of our results, covering both modal and description logics, and in Section 5 we point out links with related work. We conclude with a summary and further questions in Section 6.

2 Basic Issues in Description Logic

In this section we provide some background information on description logics, as well as some basic definitions.
Description logics are a family of specialized languages for the representation and structuring of knowledge, together with efficient methods to perform different 'reasoning tasks.' They are specialized languages related to the KL-ONE system of Brachman and Schmolze [6]. Nowadays, description logics are generally considered to be “variations” of first-order logic—either restrictions or restrictions plus some added operators. On the one hand these variations are motivated by the undecidability of the inference problem for first-order logic, and on the other by a desire to preserve the structure of the knowledge being represented. The main tools used for providing decision methods and studying complexity-theoretic aspects in the area of description logic are based on labeled tableaux.

Let us make things more precise now.

**Definition 1 (Signature).** Let \( \mathcal{L} = \{C_i\} \cup \{R_i\} \cup \{a_i\} \) be a denumerable set of symbols. We will call \( C_i \) atomic concepts, \( R_i \) atomic roles and \( a_i \) constants. \( \mathcal{L} \) is called a signature.

**Definition 2 (Interpretation).** Given a signature \( \mathcal{L} = \{C_i\} \cup \{R_i\} \cup \{a_i\} \), an interpretation \( \mathcal{I} \) for \( \mathcal{L} \) is a tuple \( \mathcal{I} = (\Delta, \cdot^\mathcal{I}) \), where

- \( \Delta \) is a non empty set.
- \( \cdot^\mathcal{I} \) is a function assigning an element \( a_i^\mathcal{I} \in \Delta \) to each constant \( a_i \); a subset \( C_i^\mathcal{I} \subseteq \Delta \) to each atomic concept \( C_i \); and a relation \( R_i^\mathcal{I} \subseteq \Delta \times \Delta \) to each atomic role \( R_i \).

**Definition 3 (Concepts and Roles).** Given a signature \( \mathcal{L} \), each description logic will define a set of defined concepts and a set of defined roles (usually just called concepts and roles). Table 1 below defines the standard connectives together with their usual names and semantics.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>concept name</td>
<td>( C )</td>
<td>( C^\mathcal{L} )</td>
</tr>
<tr>
<td>top</td>
<td>( \top )</td>
<td>( \Delta^\mathcal{I} )</td>
</tr>
<tr>
<td>bottom</td>
<td>( \bot )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>conjunction</td>
<td>( C_1 \land C_2 )</td>
<td>( C_1^\mathcal{I} \land C_2^\mathcal{I} )</td>
</tr>
<tr>
<td>disjunction (( \lor ))</td>
<td>( C_1 \lor C_2 )</td>
<td>( C_1^\mathcal{I} \lor C_2^\mathcal{I} )</td>
</tr>
<tr>
<td>negation (( \neg ))</td>
<td>( \neg C )</td>
<td>( \Delta^\mathcal{I} \setminus C^\mathcal{I} )</td>
</tr>
<tr>
<td>univ. quant.</td>
<td>( \forall R.C )</td>
<td>{ ( d_1 \mid \forall d_2 \in \Delta \cdot (d_1, d_2) \in R^\mathcal{I} \to d_2 \in C^\mathcal{I} } }</td>
</tr>
<tr>
<td>exist. quant. (( \exists ))</td>
<td>( \exists R.C )</td>
<td>{ ( d_1 \mid \exists d_2 \in \Delta \cdot (d_1, d_2) \in R^\mathcal{I} \land d_2 \in C^\mathcal{I} } }</td>
</tr>
<tr>
<td>role name</td>
<td>( R )</td>
<td>( R^\mathcal{I} )</td>
</tr>
<tr>
<td>role conj. (( \cap ))</td>
<td>( R_1 \land R_2 )</td>
<td>( R_1^\mathcal{I} \land R_2^\mathcal{I} )</td>
</tr>
</tbody>
</table>

**Table 1.** Common operators of description logics
The above semantic definition of $\forall R.C$ and $\exists R.C$ matches the semantic definition of the modal operators $\Box$ and $\Diamond$; the connection was made explicit in [18].

Historically, a number of description logics received a special name; it is customary to define systems by postfixing the names of these original systems with the added operators. The logic $\mathcal{FL}^-$ [3] is defined as the description logic allowing universal quantification, conjunction and unqualified existential quantifications of the form $\exists R.T$. The logic $\mathcal{ALC}$ [19] extends $\mathcal{FL}^-$ with negation of atomic concept names. The names in parentheses in Table 1 are the usual ones for defining extensions. Hence, $\mathcal{ALC}$ is $\mathcal{AL}$ extended with full negation. In the system $\mathcal{ALC}$, all the other operators in Table 1 can be defined.

In description logics we are interested in performing inferences given certain background knowledge.

**Definition 4 (Knowledge Bases).** A knowledge base $\Sigma$ is a pair $\Sigma = \langle T, A \rangle$ such that

- $T$ is the T(erminal)-Box, a (possibly empty) set of expressions of the forms $C_1 \sqsubseteq C_2$ or $R_1 \sqsubseteq R_2$ ($C_1, C_2 \in \text{Concepts}, R_1, R_2 \in \text{Roles}$)
- $A$ is the A(SSERTIONAL)-Box, a (possibly empty) set of expressions of the forms $a : C$ or $(a, b) : R$ ($C \in \text{Concepts}, R \in \text{Roles}, a, b \in \text{Constants}$).

**Definition 5 (Models).** Let $I$ be an interpretation and $\varphi$ an expression of the kind specified below. Then $I$ models $\varphi$ (notation: $I \models \varphi$) if

- $\varphi = C_1 \sqsubseteq C_2$ and $C_1^I \subseteq C_2^I$, or
- $\varphi = R_1 \sqsubseteq R_2$ and $R_1^I \subseteq R_2^I$, or
- $\varphi = a : C$ and $a^I \in C^I$, or
- $\varphi = (a, b) : R$ and $(a^I, b^I) \in R^I$.

Let $\Sigma = \langle T, A \rangle$ be a knowledge base and $I$ an interpretation, then $I$ models $\Sigma$ (notation: $I \models \Sigma$) if for all $\varphi \in T \cup A, I \models \varphi$.

**Definition 6 (Reasoning Tasks).** The following are some of the standard reasoning tasks considered for description logic. Let $\Sigma$ be a knowledge base:

- Subsumption ($\Sigma \models C_1 \sqsubseteq C_2$): check whether for all interpretations $I$ such that $I \models \Sigma$ we have $C_1^I \subseteq C_2^I$.
- Instance Checking ($\Sigma \models a : C$): check whether for all interpretations $I$ such that $I \models \Sigma$ we have $a^I \in C^I$.
- Concept Consistency ($\Sigma \not\models C \sqsubseteq \bot$): check whether for some interpretation $I$ such that $I \models \Sigma$ we have $C^I \neq \emptyset$.
- Knowledge Base Consistency ($\Sigma \not\models \bot$): check whether there exists $I$ such that $I \models \Sigma$.

Similar tasks can obviously also be defined for roles whenever role definitions have a richer structure than we have considered here.

In this paper we will be concerned with knowledge base consistency, which, in sufficiently strong description logics like $\mathcal{ALC}$ and its extensions, can decide all the other reasoning tasks.
3 Decision Methods for Description Logics

Weak logics like $\mathcal{FL}^-$ have very effective decision methods. For some of the standard reasoning tasks mentioned in Definition 6 these methods are polynomial and only need to perform a structural analysis of the concepts involved (i.e., no "real deduction" is performed). It is interesting to note that at this (low) level of expressive power the different reasoning tasks cannot be mapped into each other. But, of course, once we have full boolean expressive power as in $\mathcal{ALC}$, reasoning tasks like subsumption can be translated into satisfaction queries.

However, even at the level of $\mathcal{ALC}$ there is another dimension which matters: the difference between dealing with assertional information and terminological information. More precisely, assertions increase the expressive power of description logics. The standard connection between description logics and $K_m$ is established at the terminological level. To account for the assertional information the notion of nominal or name for a world is needed. See [4] for a recent study of this topic and Section 5 for further comments.

Below we give a resolution proof procedure for the description logic $\mathcal{ALC}R$ that is able to cope with assertional information.

**Definition 7 (Weak Negation Form).** Define the following rewriting procedure WNF on concepts:

1. $\neg C \overset{\text{WNF}}{\rightarrow} \neg C$
2. $\exists R.C \overset{\text{WNF}}{\rightarrow} \exists R.\neg C$
3. $(C_1 \cup \cdots \cup C_j) \overset{\text{WNF}}{\rightarrow} (\neg C_1 \cap \cdots \cap \neg C_j)$
4. $\top \overset{\text{WNF}}{\rightarrow} (C \cap \neg C)$, for $C$ arbitrary
5. $\bot \overset{\text{WNF}}{\rightarrow} \top$

For any concept $C$, WNF converges to a unique normal form which we denote as WNF$(C)$. WNF$(C)$ is logically equivalent to $C$. WNF can trivially be extended to expressions $a : C_1$ by setting WNF$(a : C_1) = a : \text{WNF}(C_1)$. If we interpret $\cup$, $\exists R.C$, $\top$ and $\bot$ as defined operators, then WNF is slightly more than an expansion of definitions.

**Definition 8 (Clause).** Given an infinite set of labels $L$ disjoint from Constants, a clause is a set $Cl$ such that each element of $Cl$ is either

- a concept assertion of the form $t : C$ where $t$ is either a constant or a label in $L$.
- a role assertion of the form $(t_1,t_2) : R$, where $t_1,t_2$ are either constants or labels in $L$.

We will use the notation $t : C$ (with possible subindices) for concept assertions and $(t_1,t_2) : R$ (with subindices) for role assertions, and $\top : \mathbb{N}$ for both of them.

A formula in a clause is a literal if it is either a role assertion, a concept or negated concept assertion on an atomic concept, or a universal or negated universal concept assertion.
Definition 9 (Model for a Clause and a Set of Clauses). Notice that formulas in a clause are simply assertions over an expanded set of constants. Let $C$ be a clause, and $I = \langle \Delta, \tau \rangle$ a model in the expanded signature; we put $I \models C$ if $I \models \bigvee C$. A set of clauses $S$ has a model if there is model $I$ such that for all $C \in S$, $I \models C$.

Definition 10 (Set of Clauses of a Knowledge Base). Let $\Sigma = \langle T, A \rangle$ be a knowledge base (with non-cyclic definitions). It is known that $\Sigma$ can be transformed into an “unfolded” equivalent knowledge base $\Sigma' = \langle \emptyset, A' \rangle$ where all concept and role assertions use only atomic concept and role symbols [11].

The set $S_{\Sigma'}$ of clauses corresponding to $\Sigma$ is the smallest set such that

- if $a : C_1 \land \cdots \land C_n = \text{WNF}(a : C)$ ($n \geq 1$) for $a : C \in A'$ then $\{a : C_i\} \in S_{\Sigma'}$,
- for $1 \leq i \leq n$.

- if $(a, b) : R_1 \land \cdots \land R_n \in A'$ then $\{(a, b) : R_i\} \in S_{\Sigma'}$, for $1 \leq i \leq n$.

Notice that the “unfolded” assertions of $A'$ are used in this translation. Furthermore, in $S_{\Sigma'}$ we can identify a (possibly empty) subset of clauses $R A$ of the form $\{(a, b) : R\}$ which we call role assertions, and for each constant $a$ a (possibly empty) subset $C A_a$ of clauses of the form $\{a : C\}$ which we call concept assertions.

Because of the format of a knowledge base it is impossible to find in $S_{\Sigma'}$ mixed clauses containing both (in disjunction) concept and role assertions. Furthermore there are no disjunctive concept assertions on different constants, i.e., there is no clause $C$ in $S_{\Sigma'}$ such that $C = C' \| \{a : C_1\} \| \{b : C_2\}$ for $a \neq b$. These properties will be relevant in the first steps of the completeness proof.

Proposition 1. Let $\Sigma$ be a knowledge base and $S_{\Sigma'}$ its corresponding set of clauses. Then $\Sigma$ is satisfiable iff $S_{\Sigma'}$ is satisfiable.

\[
\begin{align*}
(\land) & \quad C \cup \{\neg N_1 \land N_2\} & \quad (\neg \land) & \quad C \cup \{\neg (C_1 \land C_2)\} \\
\quad & \quad C \cup \{\neg N_1\} & \quad & \quad C \cup \{\neg \text{WNF}(C_1), \neg \text{WNF}(C_2)\} \\
\quad & \quad C \cup \{\neg N_2\} & \quad & \quad C \cup \{\neg \text{WNF}(C_1), \neg \text{WNF}(C_2)\}
\end{align*}
\]

\[\begin{align*}
\text{(RES)} & \quad C_{11} \cup \{\neg N\} & \quad C_{12} \cup \{\neg N\} \\
& \quad C_{11} \cup C_{12}
\end{align*}\]

\[\begin{align*}
(\forall) & \quad C_{11} \cup \{t : \forall R.C\} & \quad C_{12} \cup \{(t_1, t_2) : R\} \\
& \quad C_{11} \cup C_{12} \cup \{t_2 : C\}
\end{align*}\]

\[\begin{align*}
(\neg \forall) & \quad C \cup \{t : \forall R.C\} & \quad C \cup \{(t, n) : R\} \\
& \quad C \cup \{n : \text{WNF}(\neg C)\}, \text{where } n \text{ is new.}
\end{align*}\]

(Notice that $\land$ also covers role conjunction and that $\neg \forall$ is a mild kind of skolemization which only involves the introduction of constants.)

Table 2: The Resolution Rules
Table 2 shows the resolution rules we will consider.

**Definition 11 (Deduction).** A deduction of a clause $C_1$ from a set of clauses $S$ is a finite sequence $S_1, \ldots, S_n$ of sets of clauses such that $S = S_1$, $C_1 \in S_n$ and each $S_i$ (for $i > 1$) is obtained from $S_{i-1}$ by adding the consequent clauses of the application of one of the resolution rules in Table 2 to clauses in $S_{i-1}$. $C_1$ is a consequence of $S$ if there is a deduction of $C_1$ from $S$. A deduction of $\emptyset$ from $S$ is a refutation of $S$.

Before proving soundness, completeness and termination we present a simple example of resolution in our system.

**Example 1.** Consider the following description. Ignoring some fundamental genetic laws, suppose that children of tall people are blond (1). Furthermore, all Tom's daughters are tall (2), but he has a non-blond grandchild (3). Can we infer that Tom has a son (4)?

(0) $\text{female} \equiv \neg \text{male}$
(1) $\text{tall} \in \forall \text{CHILD} \blond$
(2) $\text{t}: \forall \text{CHILD}. \neg (\text{female} \lor \text{tall})$
(3) $\exists \text{CHILD}. \exists \text{CHILD}. \neg \blond$
(4) $\text{t}: \exists \text{CHILD}. \text{male}$

As is standard, we use a new proposition letter $\text{rest-tall}$ to complete the partial definition in (1) and we resolve with the negation of the formula we want to infer. After unfolding and applying WNF we obtain the following three clauses

1. $\{\text{t} : \forall \text{CHILD}. \neg ((\forall \text{CHILD}. \blond) \land \text{rest-tall})\}$
2. $\{\neg \forall \text{CHILD}. \forall \text{CHILD}. \blond\}$
3. $\{\forall \text{CHILD}. \neg \text{male}\}$

Now we start resolving,

4. $\{\text{t': } \neg \forall \text{CHILD}. \blond\}$ by $(\neg \forall)$ in 2.
5. $\{\text{t', t}: \text{CHILD}\}$ by $(\neg \forall)$ in 2.
6. $\{\text{t': male}\}$ by $(\forall)$ in 3.
7. $\{\text{t': } (\neg \text{male} \lor \neg ((\forall \text{CHILD}. \blond) \land \text{rest-tall}))\}$ by $(\forall)$ in 1.
8. $\{\text{t': male, t': } (\forall \text{CHILD}. \blond) \land \text{rest-tall})\}$ by $(\neg \forall)$ in 7.
9. $\{\text{t': } ((\forall \text{CHILD}. \blond) \land \text{rest-tall})\}$ by (RES) in 6 and 8.
10. $\{\text{t': } \forall \text{CHILD}. \blond\}$ by $(\forall)$ in 9.
11. $\{\text{t': rest-tall}\}$, by $(\forall)$ in 9.
12. $\emptyset$ by (RES) in 4 and 10.

**Theorem 1 (Soundness).** The resolution rules described in Table 2 are sound. That is, if $\Sigma$ is a knowledge base, then $S_\Sigma$ has a refutation only if $\Sigma$ is unsatisfiable.

**Proof.** We will prove that the resolution rules we introduced preserve satisfiability. That is, given a rule, if the premises are satisfiable, then so are the conclusions. We only discuss $(\neg \forall)$.

Let $I$ be a model of the antecedent. If $I$ is a model of $C_1$ we are done. If $I$ is a model of $t : \neg \forall R.C$, then there exists $d$ in the domain, such that $(t^I, d) \in R^I$ and
$d \in \neg C$. Let $P'$ be identical to $P$ except perhaps in the interpretation of $n$ where $n' = a$. As $n$ is a new label, also $P' \models t \rightarrow \forall R.C$. But now $P' \models CI \cup \{(t, n) : R\}$ and $P' \models CI \cup \{(n : \text{WNF}(\neg C))\}$. 

Our next aim is to prove completeness. We follow the approach used in [12]; given a set of clauses $S$ we aim to define a structure $T_S$ such that

(i) if $S$ is satisfiable, a model can be effectively constructed from $T_S$; and

(ii) if $S$ is unsatisfiable, a refutation can be effectively constructed from $T_S$.

But in our case this proves to be more difficult than in [12] because we have to deal with A-Box information, that is, with named objects or worlds (concept assertions) and fixed constraints on relations (role assertions). We will proceed in stages. To begin, we will obtain a first structure to account for named worlds and their fixed relation constraints. After that we can use a simple generalization of results in [12]. We base our construction on trees which will help in guiding the construction of the corresponding refutation proof.

Let $\Sigma$ be a knowledge base and $S_\Sigma$ its corresponding set of clauses. Let $a$ be a constant and $CA_a$ the subset of $CA$ of concept assertions concerning the constant $a$. Define the following operation to be performed on $CA_a$.

We construct for each $CA_a$ a binary tree $T_a$ inductively. Let the original tree $u$ consist of the single node $CA_a$ and repeat the following operations in an alternating fashion.

---

**Operation A1.** Repeat the following steps as long as possible:
- choose a leaf $w$. Replace any clause of the form \{a : \neg(C_1 \land C_2)\} by \{a : \text{WNF}(\neg C_1), a : \text{WNF}(\neg C_2)\}; and any clause of the form \{a : C_1 \land C_2\} by \{a : C_1\} and \{a : C_2\}.

**Operation A2.** Repeat the following steps as long as possible:
- choose a leaf $w$ of $u$ and a clause $CL$ in $w$ of the form $CL = \{a : C_1, a : C_2\} \cup CL'$;
- add two children $w_1$ and $w_2$ to $w$, where $w_1 = w \setminus \{CL\} \cup \{a : C_1\}$ and $w_2 = w \setminus \{CL\} \cup \{a : C_2\}$. 

The leaves of $T_a$ give us the possibilities for “named worlds” in our model (remember that concept prefixes act as names for worlds/objects). We can view each leaf as a set $S_a^w$, representing a possible configuration for world $a$.

**Proposition 2.** Operation $A$ (the combination of A1 and A2) terminates, and upon termination

1. all the leaves $S_a^{w_1}$ to $S_a^{w_n}$ of the tree are sets of unit literal clauses;
2. if all $S_a^{w_1}, \ldots, S_a^{w_n}$ are refutable, then $CA_a$ is refutable;
3. if one $S_a^{w_i}$ is satisfiable, then $CA_a$ is satisfiable.

**Proof.** Termination is trivial. Item 1 holds by virtue of the construction, and item 2 is proved by induction on the depth of the tree. We need only realize that by simple propositional resolution if the two children of a node $w$ are refutable, then so is $w$. Item 3 is also easy. Informally, Operation A “splits” disjunctions and “carries along” conjunctions. Hence if some $S_a^{w_i}$ has a model we have a model satisfying all conjuncts in $CA_a$ and at least one of each disjuncts. \qed
We should now consider the set \( RA \) of role assertions. Let \( \text{NAMES} \) be the set of constants which appear in \( \Sigma \). If \( a \) is in \( \text{NAMES} \) but \( CA_a \) is empty in \( S_\Sigma \), define \( S_a^R = \{ a : C, a : \neg C \} \) for some concept \( C \). We will construct a set of sets of nodes \( N = \{ N_i \mid N_i \text{ contains exactly one leaf of each } T_a \} \). Each \( N_i \) is a possible set of constraints for the named worlds in a model of \( S_\Sigma \).

**Proposition 3.** If for all \( i \), \( \bigcup N_i \cup RA \) is refutable, then so is \( S_\Sigma \).

**Proof.** If for all \( i \), \( \bigcup N_i \cup RA \) is refutable, then for some constant \( a \) we have that for all \( S_a^R \) obtained from \( CA_a \), \( S_a^R \cup RA \) is refutable. Hence by Proposition 2, \( CA_a \cup RA \) is refutable, and so is \( S_\Sigma \). \( \square \)

For all \( i \), we will now extend each set in \( N_i \) with further constraints. For each \( S_a \in N_i \), start with a node \( w_a \) labeled by \( S_a \).

\[\]

- **Operation B1.** Equal to **Operation A1.**
- **Operation B2.** Repeat the following steps as long as possible:
  - choose nodes \( w_{a1}, w_b \) such that \( \{ (a,b) : R \} \) in RA, \( \{ a : \forall R_a C_i \} \in w_a \), \( \{ b : C_i \} \notin w_b \), where \( w_b \) is without children;
  - add a child to \( w_a, w_b ' = w_b \cup \{ b : C_i \} \).

Call \( N_i^* \) the set of all leaves obtained from the forest constructed in B.

**Proposition 4.** Operation B terminates, and upon termination

1. all nodes created are derivable from \( \bigcup N_i \cup RA \), and hence if a leaf is refutable so is \( \bigcup N_i \cup RA \);
2. if some \( \bigcup N_i^* \) is satisfiable, then \( S_\Sigma \) is satisfiable.

**Proof.** To prove termination, notice that in each cycle the quantifier depth of the formulas considered decreases. Furthermore, it is not possible to apply twice the operation to a node named by \( a \) and \( b \) and a formula \( a : \forall R_a C_i \).

As to item 1, each node is created by an application of the \( (\forall) \) rule to members of \( N_i \cup RA \) or clauses previously derived by such applications. To prove item 2, let \( \mathcal{I} \) be a model of \( N_i^* \). Define a new model \( \mathcal{I}' = (\mathcal{A}', \mathcal{R}') \) as follows.

- \( \mathcal{A}' = \Delta_i \)
- \( \mathcal{A}' = \mathcal{A} \) for all constants \( a \);
- \( C_i \mathcal{I}' = C_i \mathcal{I} \) for all atomic concepts \( C_i \); and
- \( \mathcal{R}' = \mathcal{R} \cup \{ \{ a,b \} : R \} \in RA \).

Observe that \( \mathcal{I}' \) differs from \( \mathcal{I} \) only in an extended interpretation of role symbols. By definition, \( \mathcal{I}' \models RA \). It remains to prove that \( \mathcal{I}' \models CA \). By Proposition 2, we are done if we prove that \( \mathcal{I}' \models \bigcup N_i^* \). Now, since we only expanded the interpretation of relations, \( \mathcal{I} \) and \( \mathcal{I}' \) can only disagree on universal concepts of the form \( a : \forall R_a C_i \). By induction on the quantifier depth we prove this to be false.

Assume that \( \mathcal{I} \) and \( \mathcal{I}' \) agree on all formulas of quantifier depth less than \( n \), and let \( a : \forall R_a C_i \) be of quantifier depth \( n \), for \( \{ a : \forall R_a C_i \} \in S_a^R \). Suppose
$I' \not\models \forall RC$. This holds iff there exists $b$ such that $(a', b') \in R$ and $I' \models b : C$.

By the inductive hypothesis, $I \not\models b : C$. Now, if $(a', b') \in R$ we are done. Otherwise, by definition $\{(a, b) : R\} \in RA$. But then $(b : C) \in S^*_b$ by construction and as $I \models S^*_b$, we also have $I \models b : C$ — a contradiction. \hfill \square

As we said above, each $N^*_b$ represents the “named core” of a model of $S$. The final step is to define the non-named part of the model. The following operations are performed to each set in each of the $N^*_b$ obtaining in such a way a forest $F_i$.

Fix $N^*_b$ and $a$. We construct a tree “hanging” from the corresponding $S^*_b \in N^*_b$. The condition that each node of the tree is named by either a constant of a new label (that is, all the formulas have the same prefix) will be preserved as an invariant during the construction. Set the original tree $u$ to $S^*_b$ and repeat the following operations C1, C2 and C3 in succession until the end-condition holds.

---

**Operation C1.** Equal to Operation A1.

**Operation C2.** Equal to Operation A2.

**Operation C3.** For each leaf $u$ of $u$,

- if for some concept we have $\{C\}, \{-C\} \in u$, do nothing;
- otherwise, since $u$ is a set of unit clauses, we can write $u = \{\{t : C_t\}, \ldots, \{t : C_m\}, \{t : \forall R, A_t\}, \ldots, \{t : \forall R, A_m\}, \{t : \forall R, P_t\}, \ldots, \{t : \forall R, P_m\}\}$.

Form the sets $w_t = \{\{WNF(t' : -P_t)\} \cup S^*_b$ where $t'$ is a new label and $S^*_b = \{\{t' : A_t\} \cup \{t : \forall R, A_t\} \in w\}$, and append each of them to $w$ as children marking the edges as $R_t$ links. The nodes $w_t$ are called the projections of $w$.

**End-condition.** Operation C3 is inapplicable.

---

**Proposition 5.** Operation C cannot be applied indefinitely.

**Definition 12.** We call nodes to which Operation C1 or C2 has been applied of type 1, and those to which Operation C3 has been applied of type 2. The set of closed nodes is recursively defined as follows,

- if for some concept $\{t : C\}, \{t : \neg C\}$ are in $w$ then $w$ is closed.
- if $w$ is of type 1 and all its children are closed, $w$ is closed.
- if $w$ is of type 2 and some of its children is closed, $w$ is closed.

Let $F_i$ be a forest that is obtained by applying Operations C1, C2, and C3 to $N^*_b$ as often as possible. Then $F_i$ is closed if any of its roots is closed.

**Lemma 1.** If one of the forest $F_i$ obtained from $S^*_b$ is non-closed, then $S^*_b$ has a model.

**Proof.** Let $F_i$ be a non-closed forest. By a simple generalization of the results in [12, Lemma 2.7] we can obtain a model $I = \{A_i \models \}$ of all roots $S^*_b$ in $F_i$, from the trees “hanging” from them, i.e., a model of $\bigcup N^*_b$. By Proposition 4, $S^*_b$ has a model.

Lemma 1 establishes the property $(\dagger)$ we wanted in our structure $T_S$. To establish $(\dagger)$ we need a further auxiliary result.
Proposition 6. Let \( w \) be a node of type 2. If one of its projections \( w_i \) is refutable, then so is \( w \).

Proof. Let \( w \) be a set of unit clauses \( w = \{ \{ t : C_i \}, \ldots, \{ t : C_m \}, \{ t : \neg R_{k_1}, A_1 \}, \ldots, \{ t : \neg R_{k_n}, A_n \}, \{ t : \neg R_{k_i}, P_i \}, \ldots, \{ t : \neg R_{k_q}, P_q \} \} \). And let \( w_i \) be its refutable projection: \( w_i = \{ \{ \text{WNE}(t' : \neg P_i) \} \cup S_i, \text{where } t' \text{ is a new label, and } S_i = \{ \{ t' : A_{k_i} \} \mid \{ t : \neg R_{k_i}, A_{k_i} \} \in w \} \}. \) We use resolution on \( w \) to arrive to the clauses in \( w_i \) from which the refutation can be carried out: Apply \((\neg \forall)\) to \( \{ t : \neg R_{k_i}, P_i \} \) in \( w \) to obtain \( \{ t' : \text{WNE}(t' : \neg P_i) \} \) and \( \{ (t, t') : R_i \} \). Now apply \((\forall)\) to all the clauses \( \{ t : \neg R_{k_i}, A_{k_i} \} \) in \( w \) to obtain \( \{ t' : A_{k_i} \} \). \( \square \)

Lemma 2. In a forest \( F_i \), every closed node is refutable.

Proof. For \( w \) a node in \( F_i \), let \( d(w) \) be the longest distance from \( w \) to a leaf.

If \( d(w) = 0 \), then \( w \) is a leaf, thus for some concept \( C_i \), \( \{ t : C_i \} \) and \( \{ t : \neg C_i \} \) are in \( w \). Using (RES) we immediately derive \( \{ \} \).

For the induction step, suppose the proposition holds for all \( w' \) such that \( d(w') < n \) and that \( d(w) = n \). If \( w \) is of type 1, let \( w_1 = w \setminus \{ C_i \} \cup \{ C_i \} \) and \( w_2 = w \setminus \{ C_i \} \cup \{ C_i \} \) be its children. By the inductive hypothesis there is a refutation for \( w_1 \) and \( w_2 \). By propositional resolution there is a refutation of \( w \): repeat the refutation proof for \( w_2 \) but starting with \( w \); instead of the empty clause we should obtain a derivation of \( C_i \), now use the refutation of \( w_2 \).

Suppose \( w \) is of type 2. Because \( w \) is closed, one of its projections is closed. Hence, by the inductive hypothesis it has a refutation. By Proposition 6, \( w \) itself has a refutation. \( \square \)

Theorem 2 (Completeness). The resolution method described above is complete: if \( \Sigma \) is a knowledge base, then \( S_\Sigma \) is refutable whenever \( \Sigma \) is unsatisfiable.

Proof. It is only necessary to put together the previous pieces. If \( \Sigma \) does not have a model then, by Proposition 1, there is no model for \( S_\Sigma \). Hence by Lemma 1 all the forests \( F_i \) obtained from \( S_\Sigma \) are closed, and by Lemma 2, for each \( N_i \), one of the sets \( S_{\partial_i} \) is refutable. By Proposition 4, for all \( i \), \( N_i \cup R_i \) is refutable. By Proposition 3, \( S_\Sigma \) is refutable. \( \square \)

Because we have shown how to effectively obtain a refutation from an inconsistent set of clauses we have also established termination. Notice that during the completeness proof we have used a specific strategy in the application of the resolution rules (for example, the \((\neg \forall)\) rule is never applied twice to the same formula).

Theorem 3 (Termination). Given a knowledge base \( \Sigma \), the resolution method (with the strategy described above) terminates with answer YES if \( \Sigma \) is inconsistent and with answer NO otherwise.

As a corollary of the results above, we obtain soundness, completeness and termination of our resolution method for \( K_m \). Notice that this is really a weaker result than the ones proved above, since we don’t have to bother about assertional A-Box information when dealing with \( K_m \). When using our resolution method for \( K_m \) the prefix labels are really metalogical entities and not part of the logic. We will discuss this matter further in Section 3.
4 Extensions and Variations

In addition to the basic results in Section 3, we will now discuss some extensions and variations. Because of space constraints we provide few details.

Modal Extensions. The natural step, from a classical modal point of view, is to consider systems above \( \mathbf{K}_m \). We choose systems \( \mathbf{T} \), \( \mathbf{D} \), and \( 4 \) as examples. Each system is defined as an extension of the basic system \( \mathbf{K} \) by the addition of an axiom scheme which characterizes certain properties of the accessibility relation:

<table>
<thead>
<tr>
<th>Name</th>
<th>Axiom scheme</th>
<th>Accessibility Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{T} )</td>
<td>( p \rightarrow \lozenge p )</td>
<td>reflexivity: ( \forall x. xR_x )</td>
</tr>
<tr>
<td>( \mathbf{D} )</td>
<td>( \Box p \rightarrow \lozenge p )</td>
<td>seriality: ( \forall x\exists y. xR_y )</td>
</tr>
<tr>
<td>4</td>
<td>( \lozenge \Box p \rightarrow \lozenge p )</td>
<td>transitivity: ( \forall xyz. (xR_y \land yR_z \rightarrow xR_z) )</td>
</tr>
</tbody>
</table>

Corresponding to each of the axioms we add a new resolution rule.

\[
(T_i) \quad \frac{\mathcal{C} \cup \{t; \forall R_x C\}}{\mathcal{C} \cup \{t; C\}}
\]

\[
(D_i) \quad \frac{\mathcal{C} \cup \{t; \forall R_x C\}}{\mathcal{C} \cup \{t; \neg \Box R_y \text{ WNF}(\neg C)\}}
\]

\[
(4_i) \quad \frac{\mathcal{C}_1 \cup \{t_1; \forall R_x C\} \quad \mathcal{C}_2 \cup \{(t_1, t_2); R_i\}}{\mathcal{C}_1 \cup \mathcal{C}_2 \cup \{t_2; \forall R_i, C\}}
\]

Of course, because we are in a multi-modal formalism, these rules can be specified for any particular relation \( R_i \). From the description logic point of view these extensions can be understood as forcing certain properties on a specific relation. There exist description logics which permit the definition of the reflexive-transitive closure of a relation \( (R^*) \). Seriality is related to functionality of roles, another feature common in description logic formalisms.

Soundness for these systems is immediate:

**Theorem 4.** The resolution methods obtained by adding the rules \((T), (D)\) and \((4)\) for a particular relation \( R_i \), are sound with respect to the class of knowledge bases where the relation \( R_i \) is always interpreted as reflexive, serial and transitive, respectively.

For completeness and termination we should modify the construction we defined previously (in particular \((4_i)\) needs a mechanism of cycle detection); this can be done again using methods from [12].

**Theorem 5.** The resolution methods obtained by adding the rules \((T), (D)\) and \((4)\) for a particular relation \( R_i \), are complete and terminate with respect to the class of knowledge bases where the relation \( R_i \) is always interpreted as reflexive, serial and transitive, respectively.
DL Extensions. In the description logic community one considers a kind of extensions of the language that is different from the ones we already introduced. For instance, recently in [7] some attention has been given to \( n \)-ary relations in description logics (in modal logic terms, \( n \)-dimensional modal operators). Our approach seems to generalize without further problems to account for this.

Finally, another direction for extensions is to consider additional structure on roles. We have limited ourselves to conjunction, but disjunction, negation, composition, etc. can be considered. Description logics allowing these operations are known as very expressive description logics, and their worst case complexity is high, even though they perform well in some limited cases; their modal logic counterparts are related to dynamic logics based on PDL. For a translation based resolution treatment of these, see [16].

5 Related Work

The Connection with Resolution for Modal Logics. Resolution methods for modal logics (without translation) have been investigated before [13, 17, 12, 8]. The innovation introduced in this paper is in the use of labels. We think this is the key to simplify the complexities involved in previous proposals. Previously, resolution had to be performed “inside” modalities (in a similar way as how new tableaux had to be started in non-prefixed tableaux systems). Labels allow us to make information explicit and resolution can then always be performed at the “top level.” Because we have labels available, we can also deal with properties on relations—like reflexivity, seriality or transitivity—in a tableaux-like fashion, and a single new rule is all that is necessary to account for them.

Comparison with the Tableaux Method. Once labels are introduced the resolution method is very close to the tableaux approach, but we are still doing resolution. The rules (\( \top \)), (\( \bot \)) and (\( \neg \)), prepare formulas to be “fed” into the resolution rules (RES) and (\( \lor \)). The aim is still to derive the empty clause instead of finding a model by exhausting a branch.

But, is this method any better than tableaux? We don’t think this is the correct question to ask. We believe that we learn different things from studying different methods. For example, [15] studies a number of interesting optimizations of the tableaux implementation which were tested on the tableaux based theorem prover DLP. Some of their ideas were already incorporated in our resolution method (lexical normalization and early detection of clashes), and others might perhaps be used in implementations of our method. But what is perhaps more interesting to the description logic community, is that new optimizations, specific to the resolution approach, can now be exploited.

Strategies for Modal Resolution. In implementations of the resolution algorithm, strategies for selecting the resolving pairs are critical. Heuristics for the case of modal logics have been investigated in [1]. Some of their results extend to our framework, and in certain cases proofs are simpler because of our explicit use of resolution via labels. We cannot give a full description of this issue here.

\(^1\) (\( \lor \)) is added to account for the “hidden” negation in the guard of the quantifier.
Assertional Information and Hybrid Logics. There is a final topic on which we would like to comment: the relation between nominals and assertional information. The similarity between \(\mathcal{ALC}\) and the basic multi-modal logic \(\mathbf{K}_m\) is well-known. But this connection concerns the terminological part of \(\mathcal{ALC}\). Recent work on nominals and hybrid languages [4] explains how assertions enter the picture. This paper investigates \(\mathcal{ALCN} \cup \mathcal{A}\), \(\mathcal{ALC}\) plus counting, plus the set formation operator in terms of individuals: \(O(a_1, \ldots, a_n) = \{a_1^*, \ldots, a_n^*\}\), embedding this logic in a very expressive hybrid formalism \(\mathcal{H}(\forall)\). To account for \(\mathcal{ALC}\) (including assertions) a subset of a weaker system called \(\mathcal{H}(\forall)\) is enough. For this language, labeled tableaux appear as a very natural choice [3]; however, by using our labeled calculus, resolution has become just as natural a choice.

6 Conclusions and Further Work

In this paper we have provided a propositional resolution method for deciding knowledge base consistency for \(\mathcal{ALC}\). This result is further extended to account for reflexive, serial and transitive relations. Because of the connection between \(\mathcal{ALC}\) and \(\mathbf{K}_m\), our methods can also be used as resolution methods for deciding theorems of modal logics. Due to space limitations, only the basics of related issues such as more expressive description logics, and optimized strategies where discussed. These issues are being dealt with in the full version of the paper.

There is a number of important questions which are still open at this stage of our research. First, up to now we have no implementation, but this issue is high on our agenda. We believe that the ideas behind our resolution method should be simple enough so that even adapting already available provers should not prove to be a very difficult task.

Further, a very attractive idea which matches nicely with the resolution approach is to incorporate a limited kind of subsumption on universal prefixes to account for “on the fly” unfolding of terminological definitions. The use of such “universal labels” should make it unnecessary to perform a complete unfolding of the knowledge base as a pre-processing step. The leitmotiv would be “perform expansion by definitions only when needed in deduction.” On the fly unfolding has already been implemented in tableaux based systems like KRS [2].

As to the complexity of resolution: we have not attempted to formally establish the complexity of our resolution method so far. We conjecture that a PSPACE heuristic for prefixed resolution exists, even though in this first account the naïve heuristic we have introduced requires exponential space.

Finally, our completeness proof is constructive: when a refutation cannot be found we can actually define a model for the knowledge base. Hence, our method can also be used for model extraction. How does this method perform in comparison with traditional model extraction from tableaux systems?

References