A Hybrid Reasoning System for Terminologies and First-Order Clauses in Knowledge Bases[1]

Ken KANEIWA

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

kaneiwa@nii.ac.jp

Abstract Description Logics (DLs) theoretically explore knowledge representation and reasoning in concept languages. However, since they are conceptually oriented, they are not equipped with rule-based reasoning mechanisms for assertional knowledge bases – specifically, rules and facts in Logic Programming (LP), or the interaction of rules and facts with terminological knowledge. To combine rule-based reasoning with terminological knowledge, this paper presents a hybrid reasoning system for DL knowledge bases (TBox and ABox) and first-order clause sets. The primary result of this study involves the design of a sound and complete *resolution* method for the *composed knowledge bases*, and this method possesses features of an effective deduction procedure such as Robinson's Resolution Principle.

Keywords: Resolution, Logic Programming, Description Logic, Knowledge Representation, Rule-based Reasoning.

§1 Introduction

Description logics are a theoretical foundation of knowledge representation and reasoning in concept languages. Deciding satisfiability and subsumption of concepts[4] constitutes standard reasoning in description logics. For many description logics, the standard reasoning can be completely implemented using

[*1] This is an extended version of the paper.[21]

tableau-like algorithms.[18, 2] In order to practically apply knowledge bases, we consider reasoning on rules and facts in logic programming[26] together with terminological knowledge. It is required that the reasoning algorithm deals with not only the DL knowledge bases but also the clause sets in first-order logic (enable the representation of rules and facts). To address this issue, the logic programming languages CARIN-$\mathcal{ALCNR}$[25] and $\mathcal{AL}$-log[11] were proposed by combining Horn clauses and description logics.

However, this combination with description logics is syntactically limited where concept and role names (corresponding to unary and binary predicates) cannot be used to represent the head of each Horn clause. That is, in Horn clauses with DL concepts:

$$p_1(\vec{t_1}), \ldots, p_n(\vec{t_n}) \rightarrow q(\vec{t}),$$

the predicates $p_1, \ldots, p_n$ are either concept names, role names, or ordinary predicates (n-ary predicates); however, the predicate q must be an ordinary predicate. Although this limitation avoids the inference of disjunctive conclusions or negative conclusions from the head, we cannot completely obtain the expressiveness of the combination of logic programming and description logics. For example, let us consider the following Horn clauses:

Facts:

$\rightarrow \mathit{acted}(John, Mary, e_1)$
$\rightarrow \mathit{died}(Mary, e_2)$
$\rightarrow \mathit{after}(e_2, e_1)$

Rule:

$\mathit{acted}(x, y, z_1), \mathit{died}(y, z_2), \mathit{after}(z_2, z_1), Human(x), Human(y) \rightarrow \mathit{killed}(x, y)$

where *acted*, *died*, and *after* are ordinary predicates, $Human$ is a concept name (as a unary predicate), and *killed* is a role name (as a binary predicate). This rule implies that "if a human x acted against a human y at z_1 and the human y died at z_2 after z_1, then the fact that x killed y can be concluded." Additionally, we provide the following equations of DL concepts:

$Murderer \equiv \exists \mathit{killed}.Human \sqcap Human$
$Human \equiv Male \sqcup Female$

where $Murderer$, $Human$, $Male$, and $Female$ are concept names and *killed* is a role name. These equations indicate that "murderers are humans who have killed humans" and "humans are male or female." For the two types of logical form,

if the head $killed(t_1, t_2)$ as a role assertion is derived from the rule, the concept equations imply $Human(t_2)$ and therefore $Male \sqcup Female(t_2)$ holds. However, the disjunctive assertion $Male \sqcup Female(t_2)$ exceeds the expressiveness of Horn clauses[*2]

On the other hand, the following approaches to integrating logic programming and description logics have been studied in the past. In order to interoperate rules and ontologies in the Semantic Web, Grosof et al.[15] defined an expressive class as a combination of two paradigms (called Description Logic Programming) by the intersection of the description logic $\mathcal{SHOIQ}$ and Horn logic programs. With regard to abductive logic programming, Denecker's group[30, 8] presented Open Logic Programming where predicates may be undefined (as abducible predicates). The undefined predicates and the completion semantics of logic programs suitably yield a mapping from the description logic $\mathcal{ALCN}$ to the logic programs. However, these two approaches do not provide reasoning on the composed knowledge bases that arise from the integration of rules and complex concepts such as derivation of disjunctive conclusions or negative conclusions. Heymans and Vermeir[16, 17] presented Conceptual Logic Programming as a language based on disjunctive logic programming. In this language, the syntactic structure of each rule is restricted in order to obtain the tree model property (whereas in the mapping from DL to LP, disjunctive/negative conclusions are expressible). Consequently, the restricted logic programs make the satisfiability checking decidable. Instead of the computational benefit, every rule in the logic programs must have a tree-structure; however, the rule in the above example does not have a tree-structure. This limited expressiveness appears to be fatal to rule-based reasoning since the advantage of rules is to represent implication forms including various combinations of many variables, e.g., $p_1(x, y, z), p_2(x, z), p_3(y, x) \rightarrow q(x, z)$.

In order to remedy the insufficient combination of logic programming and description logics, we need to embed *general* clauses and DL concepts into a rule-based reasoning system. Resolution proof systems[28] for clausal forms in first-order logic have been implemented well as rule-based reasoning methods for knowledge bases, e.g., logic programming. As an unusual approach, a resolution system for description logics was proposed by Areces et al,[1] based on the modal

[*2] In Section 3.3, we will illustrate the derivation of such expressions in an extended knowledge base. It should be noted that a concept name or role name can be used in the head of each rule although this syntax is forbidden in CARIN-$\mathcal{ALCNR}$[25] and $\mathcal{AL}$-log[11].

resolution system[12)*3].

In this paper, we present a hybrid resolution system for combining (i) knowledge bases consisting of the TBox and ABox in the description logic $\mathcal{ALC}$ and (ii) clause sets in first-order logic. This is the first refutation method for knowledge bases composed of DLs and LP. In order for it to easily resolve both ABox-statements and first-order clauses, we introduce a clausal form of DL concepts (called clausal concepts) and compose this form and the first-order clauses. We generalize a resolution method by unification of first-order terms in assertions on clausal concepts and n-ary predicates and by incorporating the following resolution rules:

1. Resolution principle and assertional rules for the composition of first-order clauses and ABox-statements of clausal concepts
2. Terminological rules for concept definitions of clausal concepts (as TBox-statements)

Technically, the important aspect is that each resolution rule must be designed in order to refute clauses composed of DL assertions and first-order clauses (called extended clauses); that is, its resolution step deletes an inconsistent pair $(E, \neg E)$ of literals in extended clauses.

This paper is organized as follows: Section 2 presents a concept language of the description logic $\mathcal{ALC}$ and first-order clauses with concept and role names. Subsequently, we define an extended knowledge base including both logical forms. In Section 3, we develop a hybrid resolution system for the extended knowledge base where DL concepts are simplified to their clausal form (clausal concepts). In Section 4, we prove the soundness and completeness of the hybrid resolution system. Section 5 discusses the related work and finally Section 6 provides conclusions and discusses future work.

§2 Combining DL and First-Order Clauses

We define the syntax and semantics of the description logic $\mathcal{ALC}$ and first-order clauses, and extend knowledge bases by including the two types of logical forms.

2.1 Description Logic $\mathcal{ALC}$

[*3] Moreover, a resolution system for non-classical logics was developed by Gabbay and Reyle.[14)]

A concept language in the basic description logic $\mathcal{ALC}$[29] contains the set $\mathbf{C}$ of concept names A including $\bot$ and $\top$, the set $\mathbf{R}$ of role names R, and the set $\mathbf{I}$ of individual names a, b. The bottom concept $\bot$ and the universal concept $\top$ represent the empty set and the set of individuals, respectively. The concepts in $\mathcal{ALC}$ (called $\mathcal{ALC}$-concepts) are constructed from concept names A, role names R, the connectives $\neg, \sqcap, \sqcup$, and the universal and existential quantifiers $\forall, \exists$.

Definition 2.1
The set of $\mathcal{ALC}$-concepts C, D is defined inductively as follows:

$$C, D \longrightarrow A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

As explained by Donini et al.,[10] concepts are used to represent classes as sets of individuals, and roles are used to specify their properties and attributes. Let $Male, Human$ be concept names, and let *has-child* be a role name. Then, for instance, the $\mathcal{ALC}$-concept $\neg Male$ (negation of a concept) expresses "individuals who are not male." The $\mathcal{ALC}$-concepts $Male \sqcap Human$ (intersection of concepts) and $Male \sqcup Female$ (union of concepts) represent "individuals who are male and human" and "individuals who are male or female," respectively. Moreover, $\exists$*has-child*.$Male$ (existential quantification) represents "individuals who have sons" and $\forall$*has-child*.$Male$ (universal quantification) expresses "individuals whose children are all male."

The meaning of $\mathcal{ALC}$-concepts are formally given by an interpretation in the following definition:

Definition 2.2
A DL interpretation is an ordered pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of a non-empty set $\Delta^{\mathcal{I}}$ (called the universe of $\mathcal{I}$) and an interpretation function $\cdot^{\mathcal{I}}$ for $\mathbf{C} \cup \mathbf{R} \cup \mathbf{I}$ where $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ (in particular, $\bot^{\mathcal{I}} = \emptyset$ and $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$), $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The DL interpretation $\mathcal{I}$ is expanded to $\mathcal{ALC}$-concepts including connectives and

quantifiers as follows:

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} - C^{\mathcal{I}}$$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

$$(\forall R.C)^{\mathcal{I}} = \{d_1 \in \Delta^{\mathcal{I}} \mid \forall d_2[(d_1, d_2) \in R^{\mathcal{I}} \text{ implies } d_2 \in C^{\mathcal{I}}]\}$$

$$(\exists R.C)^{\mathcal{I}} = \{d_1 \in \Delta^{\mathcal{I}} \mid \exists d_2[(d_1, d_2) \in R^{\mathcal{I}} \text{ and } d_2 \in C^{\mathcal{I}}]\}$$

This interpretation will be used to define the semantics of a first-order language with concept and role names.

2.2 First-Order Clauses with Concept and Role Names

Description logics do not deal with reasoning for facts ϕ and rules $\phi_1, \ldots, \phi_n \rightarrow \phi_0$ where each ϕ and ϕ_i are atomic formulas. We employ general clausal forms (not restricted to Horn clauses) in first-order logic to be embedded in the knowledge base reasoning for description logics. For the compatibility with concept languages, concept names and role names in $\mathcal{ALC}$ are used to denote unary predicates and binary predicates, respectively, in a first-order language $\mathcal{L}$ involving ordinary n-ary predicate names. Hence, the language $\mathcal{L}$ includes the set $\mathbf{P}$ of n-ary predicate names p with $\mathbf{C} \cup \mathbf{R} \subseteq \mathbf{P}$, the set $\mathbf{F}$ of n-ary function names f ($n \geq 1$), the set $\mathbf{I}$ of individual names a, b (as constant names), and the set $\mathbf{V}$ of variables x, y.

Definition 2.3
The set of terms t (in language $\mathcal{L}$) is defined inductively as follows:

1. Every individual name and variable are terms.
2. If f is an n-ary function name and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

Definition 2.4
The set of formulas (in language $\mathcal{L}$) is defined inductively as follows:

1. If $p \in \mathbf{P}$ is an n-ary predicate name and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is an atomic formula (simply called atom).
2. If $\phi_1, \ldots, \phi_n, \phi'_1, \ldots, \phi'_m$ are atomic formulas, then $\phi_1, \ldots, \phi_n \rightarrow \phi'_1, \ldots, \phi'_m$ is a clausal form. In particular, it is called the empty clause if

$$n = m = 0.$$

$\forall F$ is the universal closure of F, i.e., $\forall x_1 \cdots \forall x_n F$ where $x_1, \ldots, x_n$ are all the free variables occurring in F. Hence, $\phi_1, \ldots, \phi_n \rightarrow \phi'_1, \ldots, \phi'_m$ expresses the universal closure $\forall(\neg\phi_1 \vee \cdots \vee \neg\phi_n \vee \phi'_1 \vee \cdots \vee \phi'_m)$ in the first-order logic. It should be noted that this clausal form corresponds to the indefinite facts and rules in disjunctive logic programming.[27] Expressive logic programming is required to deal with indefinite information of DL concepts (i.e. disjunctive concepts). Let E be an expression. $Var(E)$ denotes the set of free variables occurring in E. A substitution is a mapping θ from a finite subset of $\mathbf{V}$ into the set of terms such that $\theta(x) \neq x$. The restriction of a substitution θ to a set V of variables is defined by $\theta\!\upharpoonright\! V = \{(x,t) \in \theta \mid x \in V\}$. The substitution is expanded to terms and formulas in the usual manner of first-order logic. A substitution θ to variables occurring in E is denoted by $E\theta$ (called an instance of E). An expression E is ground if it is without variables. $ground(E)$ denotes the set of all ground instances of E. Let ES be a set of expressions. $ground(ES) = \cup_{E_i \in ES} ground(E_i)$. Let E_1, E_2 be expressions. A substitution θ is a unifier for E_1 and E_2 if $E_1\theta = E_2\theta$. A most general unifier for E_1 and E_2 is expressed by $mgu(E_1, E_2)$.

2.3 Extended Knowledge Bases

We define a knowledge base consisting of the TBox and ABox and a clause set. Let A be a concept name and C, D be $\mathcal{ALC}$-concepts. We define TBox $\mathcal{T}$ as a set of TBox-statements of the form $C \equiv D$. Note that we use TBoxes that are acyclic and sets of concept definitions of the form $A \equiv C$. Let a, b be individual names and R be a role name. An ABox $\mathcal{A}$ is a set of ABox-statements of the forms $C(a)$, $R(a,b)$. Normally, a DL knowledge base is defined as an ordered pair $(\mathcal{T}, \mathcal{A})$ of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. We extend it to an ordered tuple $KB = (\mathcal{T}, \mathcal{A}, \mathcal{P})$ by attaching a clause set $\mathcal{P}$. TBox- and ABox-statements and clausal forms are generally called knowledge base statements. A knowledge base KB is said to be ground if all the clausal forms in KB are ground.

Here, we interpret the first-order formulas including concept and role names and variables. A DL interpretation $\mathcal{I}$ is extended with $p^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$ for every n-ary predicate name $p \in \mathbf{P}$ and $f^{\mathcal{I}} : (\Delta^{\mathcal{I}})^n \rightarrow \Delta^{\mathcal{I}}$ for every n-ary function name $f \in \mathbf{F}$. A variable assignment is a mapping α from the set $\mathbf{V}$ of variables into the universe $\Delta^{\mathcal{I}}$. The variable assignment $\alpha[x/d]$ denotes $(\alpha - \{(x, \alpha(x))\}) \cup$

$\{(x, d)\}$. An interpretation of a first-order language $\mathcal{L}$ with concept and role names (called an FOL interpretation) is an ordered pair $\mathcal{I}_\alpha = (\mathcal{I}, \alpha)$, where $\mathcal{I}$ is a DL interpretation extended with $p^\mathcal{I} \subseteq (\Delta^\mathcal{I})^n$ and $f^\mathcal{I} : (\Delta^\mathcal{I})^n \to \Delta^\mathcal{I}$; and α is a variable assignment. The interpretation $t^{\mathcal{I}_\alpha}$ of terms t is defined by: (i) $x^{\mathcal{I}_\alpha} = \alpha(x)$, (ii) $a^{\mathcal{I}_\alpha} = a^\mathcal{I}$, and (iii) $(f(t_1, \ldots, t_n))^{\mathcal{I}_\alpha} = f^\mathcal{I}(t_1^{\mathcal{I}_\alpha}, \ldots, t_n^{\mathcal{I}_\alpha})$. The satisfiability relation for knowledge base statements is given by the following definition:

Definition 2.5
Let $\mathcal{I}_\alpha = (\mathcal{I}, \alpha)$ be an FOL interpretation and E be a knowledge base statement. The satisfiability relation $\mathcal{I}_\alpha \models E$ is defined as follows:

1. $\mathcal{I}_\alpha \models A \equiv C$ iff $A^\mathcal{I} = C^\mathcal{I}$
2. $\mathcal{I}_\alpha \models C(a)$ iff $a^\mathcal{I} \in C^\mathcal{I}$
3. $\mathcal{I}_\alpha \models R(a, b)$ iff $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$
4. $\mathcal{I}_\alpha \models p(t_1, \ldots, t_n)$ iff $(t_1^{\mathcal{I}_\alpha}, \ldots, t_n^{\mathcal{I}_\alpha}) \in p^\mathcal{I}$
5. $\mathcal{I}_\alpha \models \neg\phi$ iff $\mathcal{I}_\alpha \not\models \phi$
6. $\mathcal{I}_\alpha \models \phi_1, \ldots, \phi_n \to \phi'_1, \ldots, \phi'_m$ iff for any k elements $d_1, \ldots, d_k \in \Delta^\mathcal{I}$, $\{l \in \{\neg\phi_1, \ldots, \neg\phi_n, \phi'_1, \ldots, \phi'_m\} \mid \mathcal{I}_{\alpha[x_1/d_1]\cdots[x_k/d_k]} \models l\} \neq \emptyset$.

Although the clausal forms are related to the formalization of disjunctive logic programming and nonmonotonic reasoning, we adhere to the declarative semantics of first-order logic. More precisely, they are interpreted in the open-world semantics and therefore its reasoning is monotonic. This is because description logics assume that knowledge bases are incomplete. In contrast, (disjunctive) logic programming assumes that absence of information is interpreted as negative information in the closed-world semantics, in which the knowledge bases are complete and lead to nonmonotonic reasoning.

An FOL interpretation $\mathcal{I}_\alpha$ satisfies a knowledge base KB (denoted $\mathcal{I}_\alpha \models KB$) if $\mathcal{I}_\alpha$ satisfies all the elements in $KB = (\mathcal{T}, \mathcal{A}, \mathcal{P})$. A knowledge base statement E (resp. a knowledge base KB) is satisfiable if, for some $\mathcal{I}_\alpha$, $\mathcal{I}_\alpha \models E$ (resp. $\mathcal{I}_\alpha \models KB$). Otherwise, E (resp. KB) is unsatisfiable. A knowledge base statement E is a consequence of KB (denoted $KB \models E$) if every model of KB is a model of E.

§3 Hybrid Resolution

In this section, we develop a hybrid resolution system for extended knowledge bases $KB = (\mathcal{T}, \mathcal{A}, \mathcal{P})$. By transforming $\mathcal{ALC}$-concepts into a clausal

form of $\mathcal{ALC}$-concepts, this resolution system can be applied to both $\mathcal{ALC}$-concepts and first-order clauses in KB.

3.1 Clausal Concepts

We simplify the form of $\mathcal{ALC}$-concepts by the following operations. Let $KB = (\mathcal{T}, \mathcal{A}, \mathcal{P})$ be a knowledge base. First, we eliminate the symbols $\neg\neg, \sqcap, \exists$ from the $\mathcal{ALC}$-concepts in TBox $\mathcal{T}$ and ABox $\mathcal{A}$. Any concept is transformed into an equivalent concept by applying the rewrite rules[*4]:

$$\neg\neg C \equiv C$$

$$C \sqcap D \equiv \neg(\neg C \sqcup \neg D)$$

$$\exists R.C \equiv \neg\forall R.\neg C$$

If a left-hand side form occurs in concepts of $\mathcal{T}$ or $\mathcal{A}$, then it is transformed to its right-hand side form. This transformation is applied to all the elements in $\mathcal{T}$, $\mathcal{A}$, and therefore $\mathcal{T}'$, $\mathcal{A}'$ without $\neg\neg, \sqcap, \exists$ are derived.

Notations. L denotes a concept name A or its negation $\neg A$, and Q represents $\forall R$ or $\neg\forall R$. $\overline{L}$ is $\neg A$ if $L = A$, or A if $L = \neg A$. $Q.L$ or L is called a *literal concept*, written as $Q^*.L$.

Next, concepts in the TBox $\mathcal{T}'$ and ABox $\mathcal{A}'$ derived above are further transformed by two operations. First, if a concept E contains a concept of the form $\neg(C \sqcup D)$ or $\forall R.(C \sqcup D)$, then $A \equiv C \sqcup D$ (where A is a new concept name) is added to $\mathcal{T}'$, and $C \sqcup D$ (in E) is replaced with A. Secondly, if a concept E includes an expression of the form $Q_1.Q_2.C$, then $A \equiv Q_2.C$ is added to $\mathcal{T}'$, and $Q_2.C$ (in E) is transformed to A. Repeating these operations results in a concept of the form:

$$Q_1^*.L_1 \sqcup \cdots \sqcup Q_n^*.L_n.$$

We call this simplified concept a *clausal concept*. For example, the $\mathcal{ALC}$-concept:

$$\neg\forall \mathit{has\text{-}child}.\neg \mathit{Female} \sqcup \forall \mathit{has\text{-}child}.\mathit{Female} \sqcup \mathit{Human}$$

is a clausal concept. Then, we obtain $\mathcal{T}'', \mathcal{A}''$ by transforming all the elements in $\mathcal{T}', \mathcal{A}'$ to equivalent clausal concepts.

[*4] In the work of Areces et al.,[1)] $\neg\neg, \sqcup, \exists$ are removed from the concepts. The transformation in this paper is adjusted to obtain compatible expressions with clausal forms.

Lemma 3.1
Let $\mathcal{I}$ be a DL interpretation. Let C' be a clausal concept transformed from a concept C and let $A_1 \equiv C_1, \ldots, A_n \equiv C_n (n \geq 0)$ be the TBox-statements added by the transformation where each A_i is a new concept name. Then, there exists a DL interpretation $\mathcal{I}'$ that is an extension of $\mathcal{I}$ to interpret $A_1, \ldots, A_n$, and $(C')^{\mathcal{I}'} = C^{\mathcal{I}}$ and $A_1^{\mathcal{I}'} = C_1^{\mathcal{I}'}, \ldots, A_n^{\mathcal{I}'} = C_n^{\mathcal{I}'}$.

By this lemma, if a knowledge base $KB = (\mathcal{T}, \mathcal{A}, \mathcal{P})$ is satisfiable, then the equivalent knowledge base $KB' = (\mathcal{T}'', \mathcal{A}'', \mathcal{P})$ obtained by transforming $\mathcal{T}, \mathcal{A}$ is satisfiable.

3.2 Hybrid Resolution System

Our hybrid resolution system contains three types of inference rules: resolution principle, terminological rules, and assertional rules. The resolution rules are applied to TBox- and ABox-statements of clausal concepts, first-order clauses, and their composed expressions.

In resolution steps, we express clausal forms as sets of literals and ABox-statements of clausal concepts as sets of assertions of literal concepts. Given a clausal form $\phi_1, \ldots, \phi_n \rightarrow \phi'_1, \ldots, \phi'_m$, the set of the literals (called a *clause*) is $\{\neg\phi_1, \ldots, \neg\phi_n, \phi'_1, \ldots, \phi'_m\}$ where a literal is an atomic formula or its negation. For an ABox-statement $Q_1^*.L_1 \sqcup \cdots \sqcup Q_n^*.L_n(a)$ of a clausal concept, we have the set $\{Q_1^*.L_1(a), \cdots, Q_n^*.L_n(a)\}$ of assertions of the literal concepts. It should be noted that applying hybrid resolution rules to sets of literals and sets of assertions of literal concepts leads to an expression composed of these sets. Let P_i^+ be a concept of the form $\forall R.A$, $\forall R.\neg A$ or A, a role name R, or an n-ary predicate p, and let $\vec{t}$ be a sequence of terms $t_1, \ldots, t_k$. The compositional expression is a set of extended literals, called an *extended clause*:

$$\{P_1^+(\vec{t_1}), \ldots, P_n^+(\vec{t_n}), \neg P_{n+1}^+(\vec{t_{n+1}}), \ldots, \neg P_m^+(\vec{t_m})\}.$$

This enables us to include an expressive assertion $Q^*.L(t)$ of a literal concept $Q^*.L$ where t is a first-order term. Since the extended clauses exceed the expressiveness of ordinary clausal forms, the definition of satisfiability has to be supplemented. Therefore, the satisfiability of extended clauses $\Phi = \{P_1^+(\vec{t_1}), \ldots, P_n^+(\vec{t_n}), \neg P_{n+1}^+(\vec{t_{n+1}}), \ldots, P_m^+(\vec{t_m})\}$ is defined as follows:

1. $\mathcal{I}_\alpha \models P_i^+(\vec{t_i})$ *iff* $(\vec{t_i})^{\mathcal{I}_\alpha} \in (P_i^+)^{\mathcal{I}}$.
2. $\mathcal{I}_\alpha \models \neg P_i^+(\vec{t_i})$ *iff* $\mathcal{I}_\alpha \not\models P_i^+(\vec{t_i})$.
3. $\mathcal{I}_\alpha \models \Phi$ with $Var(\Phi) = \{x_1, \ldots, x_k\}$ *iff* for any k elements $d_1, \ldots, d_k$

$\in \Delta^{\mathcal{I}}$, $\{E \in \Phi \mid \mathcal{I}_{\alpha[x_1/d_1]\cdots[x_k/d_k]} \models E\} \neq \emptyset$.

Next, we proceed to the definition of hybrid resolution rules. The first rule is an extension of the resolution principle to extended clauses.

Definition 3.1 (Resolution principle)
Let Φ_1, Φ_2 be extended clauses. The resolution principle for knowledge bases is given as follows:

$$\frac{\Phi_1 \cup \{\neg P^+(\vec{t})\} \quad \{P^+(\vec{t'})\} \cup \Phi_2}{(\Phi_1 \cup \Phi_2)\theta} \ (res)$$

where $\theta = mgu(\vec{t}, \vec{t'})$.

Notations. Let Ψ be a clausal concept $Q_1^*.L_1 \sqcup \cdots \sqcup Q_n^*.L_n$. $\Psi(t)$ represents the sequence of assertions $Q_1^*.L_1(t)$, ... , $Q_n^*.L_n(t)$ of the literal concepts in Ψ. We write L_A for the concept name A or its negation $\neg A$. $\delta(E)$ denotes E_0 if $E = \neg\neg E_0$, or E otherwise. The function $N(E)$ is 0 if the number of negation symbols ($\neg$) in E is even or 1 if it is odd. Moreover, we obtain the following convenient notation:

$$\{\neg R_Q(t, t')\} = \begin{cases} \{\neg R(t, t')\} & \textit{if } Q = \forall R \\ \emptyset & \textit{otherwise} \end{cases}$$

We define terminological rules with regard to TBox-statements as follows:

Definition 3.2 (Terminological rules)
Let Φ be an extended clause and Ψ be a clausal concept. The terminological rules for knowledge bases are given as follows:

$$\frac{\Phi \cup \{\neg A(t)\} \quad A \equiv Q^*.L \sqcup \Psi}{\Phi \cup \{\delta(\neg Q^*.L)(t)\}} \ (T1)$$

$$\frac{\Phi \cup \{A(t)\} \quad A \equiv Q^*.L \sqcup \Psi}{\Phi \cup \{Q^*.L(t), \Psi(t)\}} \ (T2)$$

$$\frac{\Phi \cup \{Q_1.L_A(t)\} \quad A \equiv Q_2^*.L_2 \sqcup \Psi}{\Phi \cup \{\delta(\neg Q_2^*.L_2)(t')\} \cup \{\neg R_{Q_1}(t, t')\}} \ (T3)$$

where $N(Q_1.L_A) = 1$[*5], and $t' = x$ (new variable) if $Q_1 = \forall R$ and $t' = c$ (new constant) otherwise.

$$\frac{\Phi \cup \{Q_1.L_A(t)\} \quad A \equiv Q_2^*.L_2 \sqcup \Psi}{\Phi \cup \{Q_2^*.L_2(t'), \Psi(t')\} \cup \{\neg R_{Q_1}(t,t')\}} \ (T4)$$

where $N(Q_1.L_A) = 0$, and $t' = x$ (new variable) if $Q_1 = \forall R$ and $t' = c$ (new constant) otherwise.

In the hybrid resolution system, assertions of literal concepts in extended clauses are refuted by the following assertional rules:

Definition 3.3 (Assertional rules)
Let Φ_1, Φ_2 be extended clauses. The assertional rules for knowledge bases are given as follows:

$$\frac{\Phi_1 \cup \{\forall R.L_A(t_1)\} \quad \{\overline{L_A}(t_2)\} \cup \Phi_2}{\Phi_1 \cup \Phi_2 \cup \{\neg R(t_1,t_2)\}} \ (A1)$$

$$\frac{\Phi_1 \cup \{\neg\forall R.L_A(t)\} \quad \{L_A(x)\} \cup \Phi_2}{(\Phi_1 \cup \Phi_2)\theta} \ (A2)$$

where c is a new constant and $\theta = \{x/c\}$.

$$\frac{\Phi_1 \cup \{\forall R_1.L_A(t_1)\} \quad \{Q_2.L'_A(t_2)\} \cup \Phi_2}{\Phi_1 \cup \Phi_2 \cup \{\neg R_1(t_1,t)\} \cup \{\neg R_{Q_2}(t_2,t)\}} \ (A3)$$

where $N(\forall R_1.L_A) \neq N(Q_2.L'_A)$, and $t = x$ (new variable) if $Q_2 = \forall R$ and $t = c$ (new constant) otherwise.

$$\frac{\Phi_1 \cup \{\neg\forall R.L(t_1)\} \quad \{\neg R(t_2,t)\} \cup \Phi_2}{(\Phi_1 \cup \Phi_2)\theta} \ (A4)$$

where c is a new constant or the constant introduced by an application of $(T3)$, $(T4)$, $(A2)$ or $(A3)$ with its premise $\Phi_1 \cup \{\neg\forall R.L(t_1)\}$, and θ is a mgu of (t_1, c) and (t_2, t).

In the form $\dfrac{E_1 \quad E_2}{E}$ of hybrid resolution rules, E_1, E_2 are called the premises, and E the conclusion. We assume that the premises E_1, E_2 do not have the same variables, i.e., $Var(E_1) \cap Var(E_2) = \emptyset$. Compared with the DL resolution system,[1)] our hybrid resolution system is extended to enhance the resolution for (i) assertions of clausal concepts (in $(A1), \ldots, (A4)$ and (res)), (ii) TBox- and ABox-statements (in $(T1)$, ... ,$(T4)$), and (iii) extended clauses represented by clausal concepts, n-ary predicates, and first-order terms (in (res)), and with

[*5] Recall that we use Q_i to denote $\forall R$ or $\neg\forall R$. Then, $Q_1.L_A$ is of the form $\forall R.\neg A$ or $\neg\forall R.A$ because $N(Q_1.L_A) = 1$.

unification for first-order terms. The resolution principle (res) contains resolution for positive and negative literals and can be applied to ABox-statements of clausal concepts and role names. In related work,[1)] concept definitions in the TBox are unfolded in the ABox beforehand. In our case, the terminological rules $(T1)$, ... ,$(T4)$ generate inferences from concept definitions $A \equiv C$ in the TBox and assertions $L_A(t)$ or $Q.L_A(t)$ of the defined concept names A in the extended clauses. The assertional rules $(A1), \ldots, (A3)$ eliminate an inconsistent pair $(A, \neg A)$ of a concept name A and its negation $\neg A$ in the premises. The assertional rule $(A4)$ is a resolution rule for a negative assertion $\neg R(t_1, t_2)$ of a role name R and assertions of the form $\neg \forall R.L(t_1)$.

Definition 3.4 (Resolution)
Let KB be a knowledge base and E be a knowledge base statement. The derivability relation $KB \vdash E$ is defined by the following:

1. If $E \in KB$, then $KB \vdash E$.
2. If $KB \vdash E_1$ and $KB \vdash E_2$, where E_1, E_2 are premises and E is the conclusion in a hybrid resolution rule, then $KB \vdash E$.

A derivation of E from KB is called a refutation if $E = \emptyset$. A knowledge base KB is refutable if we have a refutation $KB \vdash \emptyset$.

Definition 3.5 (Proper resolution)
A hybrid resolution rule is called proper if the following statements hold:

1. Its premises E_1 and E_2 contain an inconsistent pair of literals.
2. Its conclusion E is derived by deleting the inconsistent pair in E_1 and E_2.
3. Its unification is most general if it exists.

Proposition 3.1
The resolution principle, the terminological rules, and the assertional rules for extended knowledge bases are proper.

This proposition indicates that our hybrid resolution system possesses features of an effective deduction procedure for combining the DL $\mathcal{ALC}$ and logic programming. Namely, for refutation, it can eliminate redundant derivation steps since every proper resolution rule is applied only to clauses including an inconsistent pair of literals.

3.3 An Example of Refutation

Let us examine an example of refutation by applying the hybrid resolution system. Consider the first-order language with

$\mathbf{C} = \{\ Murderer, Human, Male, Female\ \}$
$\mathbf{R} = \{\ killed\ \}$
$\mathbf{P} = \mathbf{C} \cup \mathbf{R} \cup \{\ acted, died, after\ \}$
$\mathbf{F} = \emptyset$
$\mathbf{I} = \{\ John, Mary, e_1, e_2\ \}$

and the two knowledge bases $KB_1 = (\mathcal{T}_1, \mathcal{A}_1, \mathcal{P}_1)$ and $KB_2 = (\mathcal{T}_1, \mathcal{A}_2, \mathcal{P}_1)$ with

TBox $\mathcal{T}_1 = \{\ Murderer \equiv \exists killed.Human \sqcap Human,$
$\qquad Human \equiv Male \sqcup Female\ \}$
ABox $\mathcal{A}_1 = \{\ Male(John),\ Female(Mary)\ \}$
ABox $\mathcal{A}_2 = \{\ Human(John),\ Human(Mary)\ \}$
Clause set $\mathcal{P}_1 = \{\ \rightarrow acted(John, Mary, e_1),$
$\qquad \rightarrow died(Mary, e_2),$
$\qquad \rightarrow after(e_2, e_1),$
$acted(x, y, z_1), died(y, z_2), after(z_2, z_1), Human(x),$
$\qquad Human(y) \rightarrow killed(x, y)\ \}$

In the TBox $\mathcal{T}_1$, the concept name $Murderer$ is defined by the concept $\exists killed.$ $Human \sqcap Human$ and the concept name $Human$ is defined by the disjunctive concept $Male \sqcup Female$. The ABox $\mathcal{A}_1$ asserts that John is male and Mary is female, and the ABox $\mathcal{A}_2$ expresses the fact that John and Mary are humans. In the clause set $\mathcal{P}_1$, the rule $acted(x, y, z_1), died(y, z_2), after(z_2, z_1), Human(x),$ $Human(y) \rightarrow killed(x, y)$ and the three facts $acted(John, Mary, e_1)$, $died(Mary,$ $e_2)$, and $after(e_2, e_1)$ are described. Notice that this rule is not expressible in the logic programming languages CARIN-$\mathcal{ALCNR}$[25] and $\mathcal{AL}$-log,[11] since the role name *killed* occurs in the head of the rule.

Let us consider answering the following query in the knowledge bases KB_1, KB_2.

?-$Murderer(x)$.

which implies "is there a murderer x?" Before applying hybrid resolution rules, we remove the connectives $\sqcap, \exists, \neg\neg$ from TBox $\mathcal{T}_1$. Consequently, the following TBox is obtained.

$$\text{TBox } \mathcal{T}_1' = \{\ Human \equiv Male \sqcup Female,\\ Murderer \equiv \neg(\forall killed.\neg Human \sqcup \neg Human)\ \}$$

Moreover, the concepts in TBox $\mathcal{T}_1'$ are transformed to equivalent clausal concepts as follows:

$$\text{TBox } \mathcal{T}_1'' = \{\ Murderer \equiv \neg A_1,\ A_1 \equiv A_2 \sqcup A_3,\\ A_2 \equiv \forall killed.\neg Human,\ A_3 \equiv \neg Human,\\ Human \equiv Male \sqcup Female\ \}$$

The ABox-statements in $\mathcal{A}_1$,$\mathcal{A}_2$ and the clausal forms in $\mathcal{P}_1$ are transformed into sets of assertions of literal concepts and sets of literals as follows:

$$\text{ABox } \mathcal{A}_1' = \{\ \{Male(John)\},\ \{Female(Mary)\}\ \}$$
$$\text{ABox } \mathcal{A}_2' = \{\ \{Human(John)\},\ \{Human(Mary)\}\ \}$$
$$\text{Clause set } \mathcal{P}_1' = \{\ \{acted(John, Mary, e_1)\},\\ \{died(Mary, e_2)\},\ \{after(e_2, e_1)\},\\ \{\neg acted(x, y, z_1), \neg died(y, z_2), \neg after(z_2, z_1),\\ \neg Human(x), \neg Human(y), killed(x, y)\}\ \}$$

The answer to the query ?-$Murderer(x)$ in the clause set $\mathcal{P}_1'$ is decided by refutation for $\mathcal{P}_1' \cup \{\mathcal{G}\}$ with $\mathcal{G} = \{\neg Murderer(x)\}$. In Figure 1, (3) and (4) demonstrate the derivation processes for $KB_1' \cup \{\mathcal{G}\} = (\mathcal{T}_1'', \mathcal{A}_1', \mathcal{P}_1' \cup \{\mathcal{G}\})$ and $KB_2' \cup \{\mathcal{G}\} = (\mathcal{T}_1'', \mathcal{A}_2', \mathcal{P}_1' \cup \{\mathcal{G}\})$, respectively. These determine whether there exists a term t such that $Murderer(t)$ is valid in the knowledge bases $KB_1' = (\mathcal{T}_1'', \mathcal{A}_1', \mathcal{P}_1')$ and $KB_2' = (\mathcal{T}_1'', \mathcal{A}_2', \mathcal{P}_1')$. In both cases, we can conclude that $Murderer(John)$ is true since the empty clause is derived with the substitution $\theta = \{x/John\}$, i.e., $KB_1' \cup \{\mathcal{G}\}$ and $KB_2' \cup \{\mathcal{G}\}$ are refutable.

§4 Soundness and Completeness of Resolution

In this section, we prove the soundness and completeness of the hybrid resolution system. A hybrid resolution rule is called propositional if its mgu is the identity substitution, i.e., θ is empty.

Definition 4.1 (Unrestricted resolution)
Let KB be a knowledge base and E be a knowledge base statement. An unrestricted derivability relation $KB \vdash_u E$ is defined by the following:

1. If $E \in KB$, then $KB \vdash_u E$.

2. If $KB \vdash_u E_1$ and $KB \vdash_u E_2$, where $E_1\theta_1, E_2\theta_2$ are premises and E is the conclusion in a propositional hybrid resolution rule, then $KB \vdash_u E$.

An unrestricted derivation of E from KB is called an unrestricted refutation if $E = \emptyset$. KB is unrestrictedly refutable if we have an unrestricted refutation $KB \vdash_u \emptyset$.

Lemma 4.1
Let E_1, E_2 be the premises in a (propositional) hybrid resolution rule, and E be its conclusion. If $\{E_1, E_2\}$ is satisfiable, then E is satisfiable.

Theorem 4.1 (Soundness of (unrestricted) resolution)
Let KB be a knowledge base. If KB is refutable ($KB \vdash \emptyset$) or unrestrictedly refutable ($KB \vdash_u \emptyset$), then KB is unsatisfiable.

Proof. By Lemma 4.1, it is trivial. ∎

In order to show the completeness of the hybrid resolution system, we construct a canonical model obtained by a derivation tree that is based on the method given in modal resolution,[12] and then prove the completeness of unrestricted resolution and ground resolution and the lifting lemma.[9] An extended clause Φ is called a unit clause if it is a singleton, i.e., $\Phi = \{\phi_i\}$ or $\{\neg\phi_i\}$. Let KB be a ground knowledge base. We denote the set of ground instances of terms occurring in KB as $gterm(KB)$.

Definition 4.2 (Derivation trees)
Let $KB = (\mathcal{T}, \mathcal{A}, \mathcal{P})$ be a ground knowledge base and $C_{new} = \{c_1, \ldots, c_n\}$ be a set of new constants such that $|C_{new}|$ is more than the number of extended literals of the form $\neg\forall R.L(t)$ in KB. A derivation tree for KB is a tree T such that:

1. The root of T is $\mathcal{A} \cup \mathcal{P}$.
2. Every node is a set of extended clauses.
3. Every leaf is a set of unit clauses.
4. (Type 1) If w is a node and $w = w' \cup \{\Phi_1 \cup \Phi_2\}$ with $\Phi_1 \neq \Phi_2 \neq \emptyset$, then $w_1 = w' \cup \{\Phi_1\}$ and $w_2 = w' \cup \{\Phi_2\}$ are its children.
5. (Type 2-a) If w is a node and $w = w' \cup \{\{\forall R.L(t)\} \cup \Phi\}$, then $w'' = w' \cup \{\{\neg R(t, t_1), L(t_1)\} \cup \Phi, \ldots, \{\neg R(t, t_n), L(t_n)\} \cup \Phi\}$ is its child where $gterm(KB) \cup C_{new} = \{t_1, \ldots, t_n\}$.

6. (Type 2-b) If w is a node and $w = w' \cup \{\{\neg\forall R.L(t)\} \cup \Phi\}$, then $w'' = w' \cup \{\{R(t,c)\} \cup \Phi, \{\overline{L}(c)\} \cup \Phi\}$ is its child where c is a new constant in C_{new}.
7. (Type 3-a) If w is a node, $w = w' \cup \{\{A(t)\} \cup \Phi\}$ and $A \equiv Q_1^*.L_1 \sqcup \cdots \sqcup Q_n^*.L_n \in \mathcal{T}$, then $w'' = w' \cup \{\{Q_1^*.L_1(t), \ldots, Q_n^*.L_n(t)\} \cup \Phi\}$ is its child.
8. (Type 3-b) If w is a node, $w = w' \cup \{\{\neg A(t)\} \cup \Phi\}$ and $A \equiv Q_1^*.L_1 \sqcup \cdots \sqcup Q_n^*.L_n \in \mathcal{T}$, then $w'' = w' \cup \{\{\neg Q_1^*.L_1(t)\} \cup \Phi, \ldots, \{\neg Q_n^*.L_n(t)\} \cup \Phi\}$ is its child.

A node w_i in T is called a node of (Type X) if it is a child in the rule of (Type X).

A derivation tree T is said to be *complete* if there is no tree T' such that T is a proper subtree of T'.

Definition 4.3
Let T be a derivation tree for a ground knowledge base KB and w be a node in T. The set of closed nodes is defined as follows:

1. If $\{\phi_i\}, \{\neg\phi_i\} \in w$, then w is closed.
2. If all the children of w are closed, then w is closed.

A derivation tree is closed if its root is closed. A derivation tree T for KB is used to prove the completeness of resolution. In a closed derivation tree, the type of each node corresponds to resolution rules. If a node is of (Type 1), then its children are refutable by applying the resolution principle. If a node is of (Type 2-a) or (Type 2-b), then its children are refutable by applying the assertional rules (A1) – (A4) and the resolution principle. In addition, if a node is of (Type 3-a) or (Type 3-b), its children are refutable by applying the terminological rules (T1) – (T4) and the resolution principle. This property will be shown in Lemma 4.3.

Proposition 4.1
Let T be a complete derivation tree for a ground knowledge base KB. If T is not closed, then (i) there is a subtree T' of T such that every node of T' is not closed and every non-leaf node has exactly one child, and (ii) there is an FOL interpretation $\mathcal{I}_\alpha$ where:

1. For every node w_i of T' and every extended clause Φ in w_i, $\mathcal{I}_\alpha \models \Phi$,

2. For every E in the TBox $\mathcal{T}$, $\mathcal{I}_\alpha \models E$.

Proof. Let T be not closed. A subtree T' of T is constructed by a sequence $w_0, \ldots, w_n$ of nodes of T where w_0 is the root and, for $1 \leq i \leq n$, w_i is a non-closed child of w_{i-1}. For the subtree T', let us define an FOL interpretation $\mathcal{I}_\alpha$ such that (i) $\mathcal{I}_\alpha \models \phi_k$ for every $\{\phi_k\} \in w_n$ and (ii) $A^{\mathcal{I}} = (Q_1^*.L_1 \sqcup \cdots \sqcup Q_n^*.L_n)^{\mathcal{I}}$ for every concept name A with $\{L_A(t)\} \notin w_n$ and $A \equiv Q_1^*.L_1 \sqcup \cdots \sqcup Q_n^*.L_n \in \mathcal{T}$. Let E be any $A \equiv Q_1^*.L_1 \sqcup \cdots \sqcup Q_n^*.L_n \in \mathcal{T}$. Since T is complete, A does not occur in the leaf w_n by the definition of (Type 3-a) and (Type 3-b). Hence, $\mathcal{I}_\alpha \models E$. Moreover, we want to prove $\mathcal{I}_\alpha \models \Phi$ for every Φ in any node w_i of T' by induction on the height m of the node w_i. If $m = 0$, then the node w_i is the leaf w_n and every Φ in w_i is a unit clause $\{\phi_k\}$ or $\{\neg\phi_k\}$. So, $\mathcal{I}_\alpha \models \Phi$. If $m > 0$, then the node w_i has a child node of (Type 1), (Type 2-a), (Type 2-b), (Type 3-a), or (Type 3-b). If w_i has a child node w_{i+1} of (Type 1), then $w_i = w' \cup \{\Phi_1 \cup \Phi_2\}$. So, w_{i+1} is $w' \cup \{\Phi_1\}$ or $w' \cup \{\Phi_2\}$. By the induction hypothesis, $\mathcal{I}_\alpha \models \Phi_1$ or $\mathcal{I}_\alpha \models \Phi_2$. Therefore, $\mathcal{I}_\alpha \models \Phi_1 \cup \Phi_2$. If w_i has a child node w_{i+1} of (Type 2-a), then $w_i = w' \cup \{\{\forall R.L(t)\} \cup \Phi'\}$. So, $w_{i+1} = w' \cup \{\{\neg R(t, t_1), L(t_1)\} \cup \Phi', \ldots, \{\neg R(t, t_n), L(t_n)\} \cup \Phi'\}$. By the induction hypothesis, $\mathcal{I}_\alpha \models \{\neg R(t, t_i), L(t_i)\} \cup \Phi'$ for all $i \in \{1, \ldots, n\}$. Hence, $\mathcal{I}_\alpha \models \{\forall R.L(t)\} \cup \Phi'$. If w_i has a child node w_{i+1} of (Type 3-a), then $w_i = w' \cup \{\{A(t)\} \cup \Phi'\}$. So, $w_{i+1} = w' \cup \{\{Q_1^*.L_1(t), \ldots, Q_n^*.L_n(t)\} \cup \Phi'\}$. By the induction hypothesis, $\mathcal{I}_\alpha \models \{Q_1^*.L_1(t), \ldots, Q_n^*.L_n(t)\} \cup \Phi'$. By $\mathcal{I}_\alpha \models A \equiv Q_1^*.L_1 \sqcup \cdots \sqcup Q_n^*.L_n$, we have $\mathcal{I}_\alpha \models \{A(t)\} \cup \Phi'$. Similarly, we can prove $\mathcal{I}_\alpha \models \Phi$ in the case that w_i has a child node of (Type 2-b) or (Type 3-b). ∎

Lemma 4.2
Let T be a complete derivation tree for a ground knowledge base $KB = (\mathcal{T}, \mathcal{A}, \mathcal{P})$. If T is not closed, then KB has a model.

Proof. Proposition 4.1 entails that there exists an FOL interpretation satisfying the root $\mathcal{A} \cup \mathcal{P}$ of T and the TBox $\mathcal{T}$. ∎

Lemma 4.3
Let T be a complete derivation tree for a ground knowledge base KB. Every closed node in T is refutable.

Proof. This lemma is proved by induction on the height n of any closed node w. If $n = 0$, then $\{\phi_i\}, \{\neg\phi_i\} \in w$. By the resolution rule (res), $\emptyset$ is derived.

If $n > 0$, then we have to consider the nodes w of (Type 1), (Type 2-a), (Type 2-b), (Type 3-a), and (Type 3-b). (Type 1): if $w = w' \cup \{\Phi_1 \cup \Phi_2\}$, then its children $w_1 = w' \cup \{\Phi_1\}$ and $w_2 = w' \cup \{\Phi_2\}$ are refutable by the induction hypothesis. By the refutation of $w' \cup \{\Phi_1\}$, there is a derivation of Φ_2 from $w' \cup \{\Phi_1 \cup \Phi_2\}$. This and the refutation of w_2 lead to a refutation of w. (Type 2-a): if $w = w' \cup \{\{\forall R.A(t)\} \cup \Phi\}$, then, by the induction hypothesis, its child $w'' = w' \cup \{\{\neg R(t, t_1), A(t_1)\} \cup \Phi, \ldots, \{\neg R(t, t_n), A(t_n)\} \cup \Phi\}$ is refutable by hybrid resolution rules. Hence, we have a refutation of w. Moreover, in the cases of (Type 2-b), (Type 3-a), and (Type 3-b), we can show that w is refutable. ∎

Theorem 4.2 (Completeness of ground resolution)
Let KB be a ground knowledge base. If KB is unsatisfiable, then KB is refutable ($KB \vdash \emptyset$).

Proof. Suppose that KB has no model. By Lemma 4.2, a complete derivation tree of KB is closed. Therefore, KB is refutable by Lemma 4.3. ∎

Let $KB = (\mathcal{T}, \mathcal{A}, \mathcal{P})$ be a knowledge base. We denote $ground(KB) = (\mathcal{T}, \mathcal{A}, ground(\mathcal{P}))$.

Lemma 4.4
Let KB be a knowledge base. KB is unsatisfiable if and only if $ground(KB)$ is unsatisfiable.

Proof. ($\Leftarrow$) Trivial. ($\Rightarrow$) We assume that $ground(KB)$ has a model. Then, there exists a Herbrand model $\mathcal{I}_H$ such that $\mathcal{I}_H \models ground(KB)$. Therefore, $\mathcal{I}_H \models KB$ follows. ∎

Theorem 4.3 (Completeness of unrestricted resolution)
Let KB be a knowledge base. If KB is unsatisfiable, then KB is unrestrictedly refutable ($KB \vdash_u \emptyset$).

Proof. Suppose $KB = (\mathcal{T}, \mathcal{A}, \mathcal{P})$ is unsatisfiable. By Lemma 4.4, $ground(KB)$ is unsatisfiable. By Theorem 4.2, there is a refutation proof tree of $ground(KB)$ by applying hybrid resolution rules. Every leaf in the refutation proof tree is an element of $ground(KB)$. So, it is possible to construct a general tree by replacing every leaf with its corresponding general expression in KB. Notice that every element in $ground(KB)$ is an instance of the general expression in KB. This general tree yields an unrestricted refutation of KB. ∎

Lemma 4.5 (Lifting)
Let KB be a knowledge base, E, E' be extended clauses, and γ be a substitution. If $KB \vdash_u E$, then $KB \vdash E'$ with $E = E'\gamma$.

Proof. This lemma is shown by induction on the length n of $KB \vdash_u E$. If $n = 0$, then $E \in KB$. Hence, $KB \vdash E$. If $n > 0$, then there is an unrestricted resolvent E of E_1 and E_2. Let β_i denote a substitution. By the induction hypothesis, if $KB \vdash_u E_1$ and $KB \vdash_u E_2$, then $KB \vdash E_1'$ with $E_1 = E_1'\beta_1$ and $KB \vdash E_2'$ with $E_2 = E_2'\beta_2$. If one of the rules $(T1), \ldots, (T4), (A1), \ldots, (A3)$ is applied as propositional, then the conclusion E' such that $E = E'\beta_1{\uparrow}Var(E_1') \circ \beta_2{\uparrow}Var(E_2')$ can be derived from E_1' and E_2'. If one of the rules (res) and $(A4)$ is applied as propositional, then there are its premises $E_1\theta_1$ and $E_2\theta_2$ deriving $E = \Phi_1\theta_1 \cup \Phi_2\theta_2$. Hence, by $E_1'\beta_1\theta_1 = E_1\theta_1$ and $E_2'\beta_2\theta_2 = E_2\theta_2$, there is a most general unifier θ for terms in E_1' and E_2'. Therefore, by (res) or $(A4)$, $KB \vdash E'$ such that E is an instance of E'. ∎

By the completeness of unrestricted resolution and the lifting lemma, we can prove the completeness of resolution as follows:

Theorem 4.4 (Completeness of resolution)
Let KB be a knowledge base. If KB is unsatisfiable, then KB is refutable ($KB \vdash \emptyset$).

Proof. Suppose that KB has no model. By Theorem 4.3, KB is unrestrictedly refutable. Therefore, by Lemma 4.5, KB is refutable. ∎

§5 Related Work

Related to our work, two types of approaches to combining logic programming and description logics have been provided. The first one extends logic programming by separately incorporating DL knowledge bases. The second one formally provides a mapping from description logics to logic programs. Table 1 lists logic programming languages that have different expressiveness of the combination of LP and DLs and its computation. The former three languages (including our work concerning the hybrid resolution) and the latter three languages in Table 1 belong to the first and second types of approaches, respectively.

In $\mathcal{AL}$-log,[11] the deductive database language Datalog was combined with the description logic $\mathcal{ALC}$. The extended language $\mathcal{AL}$-log allows for rep-

Table 1 Approaches to combining LP and DLs

	expressiveness of the combination			reasoning	
	rule structure	disjunctive conclusions	concept inclusions	computation	algorithms
$\mathcal{AL}$-log	non-recursive	concepts in bodies	general	decidable	tableaux + SLD
CARIN-$\mathcal{ALCNR}$	non-recursive	concepts and roles in bodies	general	decidable	tableaux + bottom-up evaluation
	recursive			undecidable	
Hybrid resolution for $\mathcal{ALC}$ + clauses	disjunctive clauses	derivable	terminology		resolution
Description Logic Programs			limited	decidable	
Open Logic Programming	definite clauses		terminology		SLDNFA
Conceptual Logic Programming	trees	expressible	general	decidable	answer set operation

resenting constraints $y_i : C_i$ of $\mathcal{ALC}$-concepts C_i in the bodies of Datalog clauses as follows:

$$p_1(\vec{t_1}), \ldots, p_n(\vec{t_n}) \ \& \ y_1 : C_1, \ldots, y_m : C_m \rightarrow q(\vec{t}).$$

This constrained LP preserves decidability of the query answering problem. As a further extension to $\mathcal{AL}$-log, Levy et al.[25] developed the logic programming language $CARIN$ with description logic $\mathcal{ALCNR}$ where concept and role names can appear as unary and binary predicates, respectively, in the bodies of Horn rules. The reasoning problem for $CARIN$ knowledge bases is still decidable if Horn rules are non-recursive, and undecidable if they are recursive. In both extended logic programs, any head of Horn rules is not expressed by a concept name or role name. This limitation prevents disjunctive conclusions (and negative conclusions) derivable from unrestrictedly combining Horn rules and DL knowledge bases. For example, $p_1(x), p_2(y) \rightarrow r(x, y)$ is forbidden if r is a role name. The rule and the concept $\forall r.C_1 \sqcup C_2(c_1)$ gives rise to the disjunctive conclusion $C_1(c_2) \vee C_2(c_2)$ if $p_1(c_1)$ and $p_2(c_2)$ are true.

Description Logic Programming (DLP)[15] attempted to present an ex-

pressive class of FOL by fusing the description logic $\mathcal{SHOIQ}$ and Horn logic programs. In other words, DLP accounts for a small combination of LP and DLs, which is expected to clarify a tractable class of dealing with rules and ontologies in the Semantic Web. In DLP, the rules and ontologies can be interoperated by mapping from DL to LP and from LP to DL. The following table describes a correspondence between concept inclusions and Horn rules.

concept inclusions in DL	Horn rules in LP
$C_1 \sqcap C_2 \sqsubseteq D$	$C_1(x), C_2(x) \rightarrow D(x)$
$C \sqsubseteq D_1 \sqcap D_2$	$C(x) \rightarrow D_1(x)$ and $C(x) \rightarrow D_2(x)$
$C_1 \sqcup C_2 \sqsubseteq D$	$C_1(x) \rightarrow D(x)$ and $C_2(x) \rightarrow D(x)$
$C \sqsubseteq \forall P.D$	$C(x), P(x, y) \rightarrow D(x)$
$\exists P.C \sqsubseteq D$	$P(x, y), C(y) \rightarrow D(x)$

This cannot provide a mapping from concept inclusions of the forms $D \sqsubseteq C_1 \sqcup C_2$, $\forall P.D \sqsubseteq C$, and $D \sqsubseteq \exists P.C$ to Horn rules.

Open Logic Programming (OLP) proposed by Denecker's group[30, 8] is an extension of logic programming with undefined predicates. In OLP, primitive concept names and role names in concept definitions can be mapped to undefined predicates, and the open-world assumption is supported in the distinction between defined and undefined predicates. By the completion of programs:

$$body_1 \vee \cdots \vee body_n \leftrightarrow A(x),$$

one concept definition $A \equiv C$ corresponds to the Horn rules $body_1 \rightarrow A(x)$, ... , $body_n \rightarrow A(x)$ with the defined unary predicate A where $body_1 \vee \cdots \vee body_n$ is equivalent to $C(x)$.

Heymans and Vermeir[16, 17] presented Conceptual Logic Programming (CLP) to obtain a mapping from the description logic $\mathcal{SHIQ}^*$ to logic programs. Although CLP is a restricted disjunctive logic programming, it can simulate $\mathcal{SHIQ}^*$-concepts, including the disjunctive concept $D \sqsubseteq C_1 \sqcup C_2$ and other complex concepts. However, in order to make the satisfiability checking decidable, the syntactic structure of clausal rules are limited to and classified into several types of rules: free rules, tree rules, and tree constraints. Free rules are of the form $p(\vec{x}) \vee \neg p(\vec{x})$ that do not reflect satisfiability, and tree rules and tree constraints must have a tree-structure. Due to the syntactic restriction, the following Horn rule (shown in the example of Section 1) cannot be expressed in CLP:

$$acted(x, y, z_1), died(y, z_2), after(z_2, z_1), Human(x), Human(y) \rightarrow killed(x, y)$$

Furthermore, order-sorted logic[13, 7] is a well-known approach to combining rule-based reasoning with sort-hierarchy (as terminological knowledge). Beierle's group[3] and Kaneiwa[20, 23, 22] developed order-sorted systems with sort predicates (denoting their corresponding unary predicates), in which sort-hierarchy reflects clausal reasoning in terms of sort predicates. Bürckert[5, 6] presented a resolution principle for clauses with constraints in which variables are constrained by a restriction theory. He adopted a refutation proof method as its reasoning algorithm; however, the testing of satisfiability for constraints was not realized by the proof method.

On the basis of the related work, we have to emphasize the distinguishing points of this paper. Instead of the decidability result of several approaches, our work provides a fully expressive combination of the DL $\mathcal{ALC}$ and LP, such as interoperation of the rule and complex concepts discussed in the example of Section 1 and its effective reasoning method for refutation, i.e., being able to refute clauses in the composed knowledge bases. In addition, for theoretically illustrating the issue of decidability (or computation), our system encounters the following difficulties:

- Expressiveness of useful rules exceeds tree-structures, while the other approaches and description logics have the tree model property to be decidable.
- A combination of expressive rules and existential and universal roles in DL concepts gives rise to increasing complexity of reasoning.

Unlike decidable approaches, our example is represented by expressive rules (i.e. non tree-structures), and Horrocks and Patel-Schneider[19] indicated the expressive power of rules (e.g., combining composed roles in rules and DL concepts leads to undecidability) and its usefulness for applications in the Semantics Web. In order to obtain a decidable combination of rules and DL concepts, we have the following two strategies: (i) syntactically restricting roles occurring in rules since rules are rather expressive (which can easily express transitive roles, inverse roles, role value maps, etc.) and (ii) combining expressive rules with limited DL constructors that can preserve the expressive power of rules. Based on the strategies, further research is required to embody a useful and decidable combination of expressive rules and DL concepts.

§6 Conclusion

We have presented a rule-based reasoning system as an extension of the DL resolution system[1] where DL knowledge bases and first-order clause sets are combined. From the viewpoint of knowledge representation, the extended knowledge bases can treat practical and general data as follows:

- Facts and ABox-statements represent assertional knowledge in a context (or a concrete situation).
- Rules are taken as general knowledge in a particular application domain.
- TBox-statements express terminological knowledge commonly employed in several application domains.

We have designed a *proper* resolution method for capturing the two types of logical forms in the DL $\mathcal{ALC}$ and logic programming. A resolution rule is called proper if its two premises are composed into one conclusion by deleting an inconsistent pair $(\phi_i, \neg\phi_i)$ of literals. Within this criterion, we have obtained proper hybrid resolution rules, whereas some of the inference rules in the DL resolution system[1] were not proper. Although we have not discussed the computation of our reasoning system, the proper resolution method can be expected to provide an effective deduction procedure. For example, consider the inconsistency of the assertions: $\neg\forall R_1.\neg A_1(t_1)$, $\forall R_1.\neg A_1(t_1)$, $\forall R_2.A_4(t_2)$, $\neg\forall R_2.\neg A_2(t_2) \vee \neg\forall R_2.\neg A_3(t_2)$. As shown in Figure 2, the inconsistency can be derived in two resolution steps because proper resolution rules are applied only to clauses including an inconsistent pair of literals. However, tableau-like reasoning (as the standard DL reasoning method) requires nine steps in the worst case since redundant steps are derived. Alternatively, Kaneiwa and Tojo[24] proposed a resolution system with complex sort expressions; however, its language was not able to represent and reason with respect to DL concepts.

This study leads to a further extension of rule-based reasoning with terminological knowledge, by means of other approaches in logic programming and automated reasoning (e.g., typed logic programming, nonmonotonic reasoning, etc.) Furthermore, we have not addressed the combinations of first-order clauses and other description logics, such as sublanguages of $\mathcal{ALC}$ (e.g., $\mathcal{AL}$, $\mathcal{ALU}$) and superlanguages of $\mathcal{ALC}$ (e.g., $\mathcal{ALCN}$,[18] $\mathcal{ALCQI}$). These combinations are interesting for considering the expressiveness of logic programming and description logics.

Acknowledgment

This research has been partially supported by the Kayamori Foundation

of Informational Science Advancement and the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (14780311). The author would like to thank the anonymous referees for their comments and suggestions to improve this paper.

References

1) C. Areces, H. de Nivelle, and M. de Rijke. Resolution in modal, description and hybrid logics. *Journal of Logic and Computation*, 11(5):717–736, 2001.

2) F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.

3) C. Beierle, U. Hedtsück, U. Pletat, P.H. Schmitt, and J. Siekmann. An order-sorted logic for knowledge representation systems. *Artificial Intelligence*, 55:149–191, 1992.

4) R. J. Brachman, R. E. Fikes, and H. J. Levesque. KRYPTON: A functional approach to knowledge representation. *Computer*, 16(10):67–73, 1983.

5) H-J. Bürckert. *A resolution principle for a logic with restricted quantifiers*, volume 568 of *Lecture Notes in Artificial Intelligence and Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1991.

6) H-J. Bürckert. A resolution principle for constraint logics. *Artificial Intelligence*, 66:235–271, 1994.

7) A. G. Cohn. Taxonomic reasoning with many sorted logics. *Artificial Intelligence Review*, 3:89–128, 1989.

8) M. Denecker. A terminological interpretation of (abductive) logic programming. In *Logic Programming and Nonmonotonic Reasoning, Proceedings LPNMR'95*, volume 928 of *Lecture notes in Artificial Intelligence*, pages 15–29. Springer, 1995.

9) K. Doets. *From Logic to Logic Programming*. The MIT Press, 1994.

10) F. D. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logic. In G. Brewka, editor, *Principles of Knowledge Representation*. CSLI Publications, FoLLI, 1996.

11) F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-LOG: Integrating datalog and description logic. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

12) P. Enjalbert and L. Farinas del Cerro. Modal resolution in clausal form. *Theoretical Computer Science*, 65(1):1–33, 1989.

13) A. M. Frisch. A general framework for sorted deduction: Fundamental results on hybrid reasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 1989.

14) D. Gabbay and U. Reyle. Labelled resolution for classical and non-classical logics. *Studia Logica*, 59(2):179–216, 1997.

15) B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logics. In *Proc. of WWW-2003*, Budapest, Hungary, 05 2003.

16) S. Heymans and D. Vermeir. Integrating ontology languages and answer set programming. In *Proceedings of the 14th International Workshop on Expert System Applications (DEXA2003)*, pages 584–588. IEEE Computer Society, 2003.

17) S. Heymans and D. Vermeir. Integrating semantic web reasoning and answer set programming. In *Answer Set Programming: Advances in Theory and Implementation (ASP03)*, pages 194–208. CEUR Proceedings Vol. 78, 2003.

18) B. Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proceedings of GWAI–90, Fourteenth German Workshop on Artificial Intelligence*, pages 38–47, 1990.

19) I. Horrocks and P. F. Patel-Schneider. A proposal for an owl rules language. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 723–731. ACM, 2004.

20) K. Kaneiwa. The completeness of logic programming with sort predicates. *Systems and Computers in Japan*, 35(1):37–46, 2004.

21) K. Kaneiwa. A hybrid reasoning system for terminologies and clause sets. In *Proceedings of the Twenty-Second IASTED International Multi-Conference on Applied Informatics*, 2004.

22) K. Kaneiwa. Order-sorted logic programming with predicate hierarchy. *Artificial Intelligence*, 158(2):155–188, 2004.

23) K. Kaneiwa and R. Mizoguchi. Ontological knowledge base reasoning with sort-hierarchy and rigidity. In *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning (KR2004)*, pages 278–288, 2004.

24) K. Kaneiwa and S. Tojo. An order-sorted resolution with implicitly negative sorts. In *Proceedings of the 2001 Int. Conf. on Logic Programming*, pages 300–314. Springer-Verlag, 2001. LNCS 2237.

25) A. A. Levy and M-C. Rousset. CARIN: A representation language combining Horn rules and description logics. *Artificial Intelligence*, 104(1–2):165–209, 1998.

26) J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987.

27) J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. The MIT Press, 1992.

28) J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, 12(1):23–41, 1965.

29) M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

30) K. Van Belleghem, M. Denecker, and D. De Schreye. A strong correspondence between description logics and open logic programming. In *Logic Programming, Proceedings of the Fourteenth International Conference on Logic Programming*, pages 346–360. MIT Press, 1997.

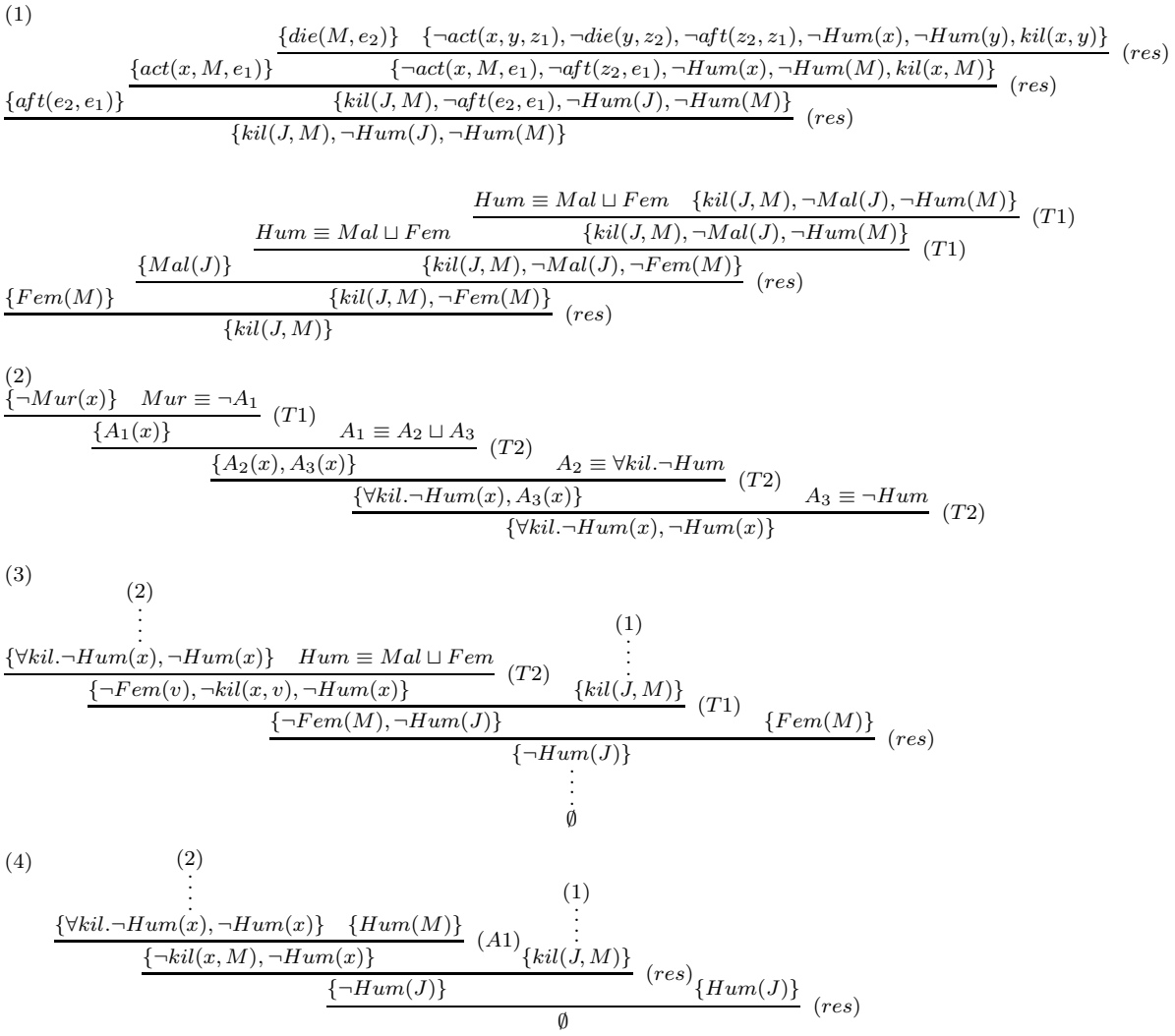

Fig. 1 Refutation from a knowledge base

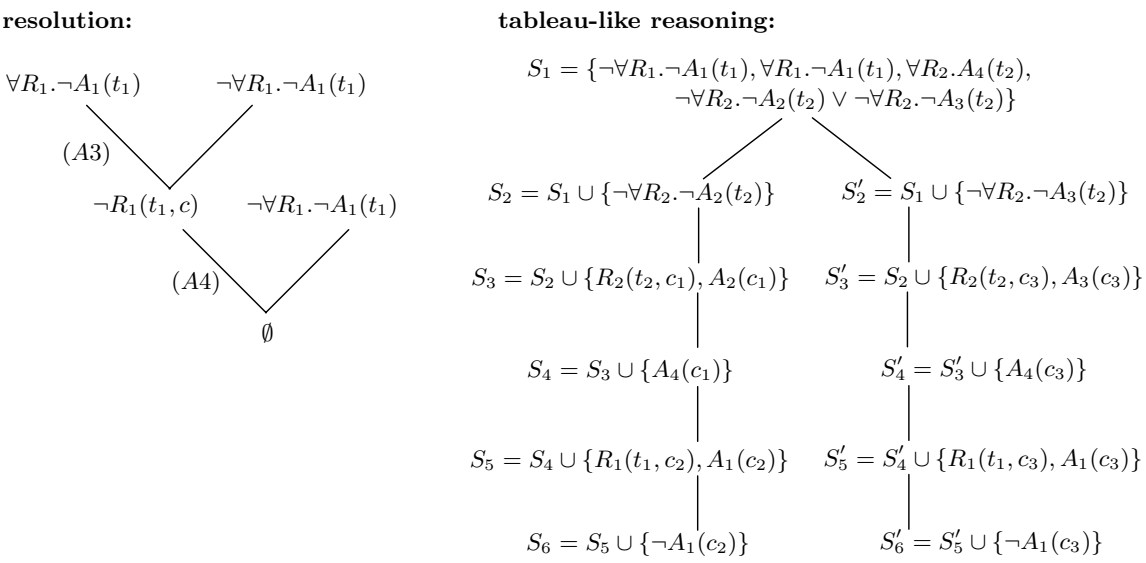

Fig. 2 Comparison between resolution and tableau-like reasoning