Solving Query-answering Problems with If-and-Only-If Formulas

Kiyoshi Akama[1] and Ekawit Nantajeewarawat[2]
[1]*Information Initiative Center, Hokkaido University, Hokkaido, Japan*
[2]*Computer Science, Sirindhorn International Institute of Technology, Thammasat University, Pathumthani, Thailand*

Keywords: Equivalent Transformation, Query-answering Problems, Unfolding, Skolemization, Extended Clauses.

Abstract: A query-answering problem (QA problem) is concerned with finding all ground instances of a query atomic formula that are logical consequences of a given logical formula describing the background knowledge of the problem. A method for solving QA problems on full first-order logic has been invented based on the equivalent transformation (ET) principle, where a given QA problem on first-order logic is converted into a QA problem on extended clauses and is then further transformed repeatedly and equivalently into simpler forms until its answer set can be readily obtained. In this paper, such a clause-based solution is extended by proposing a new method for effectively utilizing a universally quantified if-and-only-if statement defining a predicate, which is called an *iff-formula*. The background knowledge of a given QA problem is separated into two parts: (i) a conjunction of iff-formulas and (ii) other types of knowledge. Special ET rules for manipulating iff-formulas are introduced. The new solution method deals with both iff-knowledge in first-order logic and a set of extended clauses. Application of this solution method is illustrated.

1 INTRODUCTION

Query-answering problems (QA problems) form an important class of problems, which has attracted increasing interest recently. In contrast to proof problems, which are "yes/no" problems, a QA problem is characteristically an "all-answers finding" problem, i.e., it is concerned with finding all ground instances of a query atomic formula that follow logically from a given logical formula representing the background knowledge of the problem.

Subclasses of QA problems have been considered in the Semantic Web community (Horrocks et al., 2005; Motik et al., 2005; Motik and Rosati, 2010) and in logic programming and deductive databases (Lloyd, 1987; Minker, 1988). These subclasses are however relatively small compared to the class of QA problems considered by human beings in natural language sentences. The class of all QA problems on full first-order logic is very important for natural language understanding and human problem solving. A large number of studies have been carried out in logic programming based on specific semantics, such as the well-founded semantics and the stable model semantics. Specific semantics for sets of clauses (possibly with negation as failure), which can be useful for programming, are however not so expressive and natural for the direct translation of natural language sentences and for natural language understanding. For this reason, we take full first-order logic with the standard semantics for QA problems.

A method for solving QA problems on full first-order logic has been discussed in (Akama and Nantajeewarawat, 2013b; Akama and Nantajeewarawat, 2014), and as far as we know, it provides the only existing general approach that deals with QA problems on full first-order logic with standard semantics. This solution method is based on the equivalent transformation (ET) principle. A given QA problem is successively transformed equivalently into simpler forms until its answer set can be readily obtained.

To enable the ET-based strategy, meaning-preserving Skolemization has been developed in (Akama and Nantajeewarawat, 2011) together with a new extended space, called the $\mathrm{ECLS}_{\mathrm{F}}$ space, over the set of all first-order formulas. This extended space includes function variables, which are variables ranging over function constants. Since function constants are mappings from tuples of ground terms to ground terms, atomic formulas (atoms) with function variables are regarded as "second-order" atoms. For problem transformation on the extended space, many ET rules have been devised in (Akama and Nantajeewarawat, 2013c; Akama and Nantajeewarawat,

Akama K. and Nantajeewarawat E..
Solving Query-answering Problems with If-and-Only-If Formulas.
DOI: 10.5220/0005097903330344
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2014), pages 333-344
ISBN: 978-989-758-049-9

2013b; Akama and Nantajeewarawat, 2014), including ET rules for unfolding, for removing useless definite clauses, for resolution, for factoring, for dealing with atoms with function variables, and for erasing independent satisfiable atoms.

In this paper, we extend the ET-based procedure in (Akama and Nantajeewarawat, 2013b; Akama and Nantajeewarawat, 2014) by introducing a method for effectively utilizing if-and-only-if formulas (iff-formulas, for short) in given background knowledge. Iff-formulas are often used for defining concepts in a knowledge base. Compared to unfolding using clauses obtained from given iff-formulas, the iff-formulas themselves allow clause transformation with unrestricted applicability for simplification of QA problems. Iff-formulas are thus useful for effective and efficient computation.

To begin with, Section 2 formalizes QA problems on first-order logic, introduces the $\mathrm{ECLS_F}$ space and meaning-preserving Skolemization, and identifies the main objective of this paper. Section 3 defines iff-formulas and a quadruple form for representing a QA problem with iff-formulas, and presents the extended ET-based procedure. Section 4 gives ET rules for clause transformation using iff-formulas and for removal of useless iff-formulas. Section 5 compares transformation using iff-formulas with unfolding. Section 6 illustrates application of our method. Section 7 provides conclusions.

2 QA PROBLEMS ON AN EXTENDED SPACE

2.1 QA Problems

A *query-answering problem* (*QA problem*) on first-order logic is a pair $\langle K,q\rangle$, where K is a first-order formula, representing background knowledge, and q is a usual atomic formula (atom), representing a query. When no confusion is caused, the qualification "on first-order logic" is often dropped. The standard semantics for first-order formulas is used, in the sense that all models of a given first-order formula are considered instead of restricting models to be considered using specific semantics. Interpretations and models are sets of ground atoms, which are similar to Herbrand interpretations and Herbrand models. The answer to a QA problem $\langle K,q\rangle$, denoted by $answer(K,q)$, is the set of all ground instances of q that are logical consequences of K. As shown in (Akama and Nantajeewarawat, 2013b), $answer(K,q)$ can be equivalently defined as

$$answer(K,q) = (\bigcap Models(K)) \cap rep(q), \qquad (1)$$

where $Models(K)$ denotes the set of all models of K and $rep(q)$ the set of all ground instances of q.

The main features of the ET-based method for solving QA problems on first-order logic with standard semantics (Akama and Nantajeewarawat, 2013b; Akama and Nantajeewarawat, 2014) include: (i) the use of a new extended space, which is an extension of first-order logic by incorporation of function variables; (ii) the use of meaning-preserving Skolemization (Akama and Nantajeewarawat, 2011), in place of the conventional Skolemization (Chang and Lee, 1973), for converting a first-order formula into a clause set in the extended space; and (iii) the use of equivalent transformation on the extended space for computation of solutions. They are described below along with the primary objective of this paper.

2.2 The Extended Space $\mathrm{ECLS_F}$

A usual function symbol in first-order logic denotes an unevaluated function; it is used for constructing from existing terms a syntactically new term without evaluating the obtained term. A different class of functions, called *function constants*, is used in the extended space. A function constant is an actual mathematical function, say f, on ground terms; when it takes ground terms, say $t_1,\ldots,t_n$, as input, $f(t_1,\ldots,t_n)$ is evaluated for determining an output term. Variables of a new type, called *function variables*, are introduced; they can be instantiated into function constants or function variables, but not into usual terms.

Given any n-ary function constant or n-ary function variable f, an expression $func(f,t_1,\ldots,t_n,t_{n+1})$, where the t_i are usual terms, is considered as an atom of a new type, called a *func-atom*. When f is a function constant and the t_i are all ground, the truth value of this atom is true iff $f(t_1,\ldots,t_n) = t_{n+1}$.

In addition to usual atoms and *func*-atoms, constraint atoms may be used in a clause. While the truth value of a ground usual atom depends on an interpretation, the truth value of a ground constraint atom is determined in advance independently of any interpretation. Examples of constraint atoms are $eq(t_1,t_2)$, $neq(t_1,t_2)$, $le(t_1,t_2)$, and $ge(t_1,t_2)$, where t_1 and t_2 are terms. When t_1 and t_2 are ground terms, $eq(t_1,t_2)$ and $neq(t_1,t_2)$ are true iff $t_1 = t_2$ and $t_1 \neq t_2$, respectively. When t_1 and t_2 are numbers, $le(t_1,t_2)$ and $ge(t_1,t_2)$ are true iff $t_1 \leq t_2$ and $t_1 \geq t_2$, respectively.

A *clause* C in the extended space is a formula of the form

$$a_1,\ldots,a_m \leftarrow b_1,\ldots,b_n,\mathbf{f}_1,\ldots,\mathbf{f}_o,$$

where (i) $a_1, \ldots, a_m$ are usual atoms, (ii) each of $b_1, \ldots, b_n$ is a usual atom or a constraint atom, and (iii) $\mathbf{f}_1, \ldots, \mathbf{f}_o$ are *func*-atoms. The sets $\{a_1, \ldots, a_m\}$ and $\{b_1, \ldots, b_n, \mathbf{f}_1, \ldots, \mathbf{f}_o\}$ are called the *left-hand side* and the *right-hand side*, respectively, of the clause C, denoted by $lhs(C)$ and $rhs(C)$, respectively. When $m = 0$, C is called a *negative clause*. When $m = 1$, C is called a *definite clause*, the only atom in $lhs(C)$ is called the *head* of C, denoted by $head(C)$, and the set $rhs(C)$ is also called the *body* of C, denoted by $body(C)$. When $m > 1$, C is called a *multi-head clause*. All usual variables in a clause are universally quantified and their scope is restricted to the clause itself.

The set of all clause sets in the extended space is called the $\mathrm{ECLS_F}$ space. Function variables in a clause set in $\mathrm{ECLS_F}$ are all existentially quantified and their scope covers entirely all clauses in the set. Given a clause set Cs in $\mathrm{ECLS_F}$, let $Models(Cs)$ denote the set of all models of Cs.

2.3 Meaning-Preserving Skolemization

In the conventional proof theory, a first-order formula is usually converted into a conjunctive normal form in the usual first-order formula space. The conversion involves removal of existential quantifications by Skolemization, i.e., by replacement of an existentially quantified variable with a Skolem term determined by its relevant quantification structure. The conventional Skolemization, however, does not generally preserve the logical meaning of a formula (Chang and Lee, 1973); as a result, it causes difficulties in solving QA problems by equivalent transformation.

In order to transform a first-order formula equivalently into a set of clauses, meaning-preserving Skolemization was invented in (Akama and Nantajeewarawat, 2008; Akama and Nantajeewarawat, 2011). Let $\mathrm{MPS}(K)$ denote the result of meaning-preserving Skolemization of a given first-order formula K. $\mathrm{MPS}(K)$ is obtained from K by repeated subformula transformation and conversion to a clausal form. For subformula transformation, say T, model-preserving transformation is used. For example, $T(\neg(\neg E)) = E$ and $T(\neg(E_1 \vee E_2)) = (\neg E_1) \wedge (\neg E_2)$. Although the forms of these transformations are similar to those in the conventional Skolemization, they are totally different in the sense that the formulas E, E_1, and E_2 may contain *func*-atoms, function variables, and function constants. When $K = (\forall x_1 \forall x_2 \cdots \forall x_n \exists y : E)$, the transformation T introduces a new function variable and a new *func*-atom, i.e., $T(K)$ is the formula

$$\exists h \forall x_1 \forall x_2 \cdots \forall x_n \forall y : (E \vee \neg func(h, x_1, x_2, \ldots, x_n, y)),$$

where h is an n-ary function variable. For example,

$$T(\forall x \exists y : motherOf(y, x)) = \exists h \forall x \forall y : (motherOf(y, x) \vee \neg func(h, x, y)),$$

which is further converted into the extended clause $(motherOf(y, x) \leftarrow func(h, x, y))$. The transformation rules used in (Akama and Nantajeewarawat, 2011) for meaning-preserving Skolemization are given in the appendix.

It was shown in (Akama and Nantajeewarawat, 2008) that:

Theorem 1. $Models(K) = Models(\mathrm{MPS}(K))$ for any first-order formula K. □

2.4 A Triple Form and Equivalent Transformation (ET)

A triple form of a QA problem is introduced in (Akama and Nantajeewarawat, 2014) for flexible representation and transformation. Let $\mathcal{A}$ be the set of all usual atoms and for any atom $a \in \mathcal{A}$, let $rep(a)$ denote the set of all ground instances of a. A triple form of a QA problem on $\mathrm{ECLS_F}$ is a tuple $\langle Cs, q, \pi \rangle$, where Cs is a clause set in $\mathrm{ECLS_F}$ representing background knowledge, q is a usual atom representing a query, and π is a partial mapping from $\mathcal{A}$ to $\mathcal{A}$ such that the range of π contains all instances of q. The answer to the QA problem $\langle Cs, q, \pi \rangle$, denoted by $answer(Cs, q, \pi)$, is defined by

$$answer(Cs, q, \pi) = \pi((\bigcap Models(Cs)) \cap rep(q)). \quad (2)$$

An ET-based procedure for solving QA problems is a state-transition procedure consisting of three main phases:

1. A QA problem $\langle K, q \rangle$ on first-order logic is first converted into a QA problem $\langle Cs, q, id \rangle$ on $\mathrm{ECLS_F}$, where $Cs = \mathrm{MPS}(K)$ and id is the identity mapping. By Theorem 1, $answer(K, q) = answer(Cs, q, id)$.
2. The QA problem $\langle Cs, q, id \rangle$ is transformed by successive application of various ET rules. In general, each application of an ET rule transforms a given QA problem $\langle \hat{Cs}, \hat{q}, \hat{\pi} \rangle$ into $\langle \tilde{Cs}, \tilde{q}, \tilde{\pi} \rangle$ preserving the answer set, i.e., $answer(\hat{Cs}, \hat{q}, \hat{\pi}) = answer(\tilde{Cs}, \tilde{q}, \tilde{\pi})$.
3. From the resulting simplified QA problem, the answer set of the original QA problem is derived.

Each transition step preserves the answer set of a given input QA problem and therefore the correctness of this procedure is guaranteed.

2.5 The Primary Objective of This Paper

Given a QA problem $\langle K, q\rangle$ on first-order logic, the first-order formula K often includes a universally quantified closed formula of the form $\forall(a \leftrightarrow F)$, where a is a usual atom and F is a first-order formula. This form of knowledge is referred to herein as *if-and-only-if knowledge* (for short, *iff-knowledge*). It is very useful since it enables direct transformation of a QA problem by replacement of an instance of a with its corresponding instance of F. However, the transformation to a clausal form in the previous triple-form method (Section 2.4) does not utilize this advantage (see Section 5).

The primary purpose of the paper is to develop a new method for effectively utilizing iff-knowledge. More precisely, we divide the background knowledge of a QA problem into two parts: (i) a conjunction of iff-knowledge and (ii) other types of knowledge. We introduce special ET rules for manipulating iff-knowledge. For ease of transformation, we assume in this paper that the form of the formula F in iff-knowledge $\forall(a \leftrightarrow F)$ is a disjunction of atom conjunctions.

3 SOLVING QA PROBLEMS WITH IFF-FORMULAS

The class of iff-formulas considered in this paper is formally defined in Section 3.1 along with related notation. In order to make a clear separation between iff-formulas and knowledge of other types, a quadruple form of a QA problem is introduced in Section 3.2. An ET-based procedure for solving QA problems with iff-formulas is presented in Section 3.3.

In the rest of this paper, let $\mathcal{A}$ be the set of all usual atoms and for any atom $a \in \mathcal{A}$, let $rep(a)$ denote the set of all ground instances of a.

3.1 If-and-Only-If Formulas (Iff-Formulas)

Given an atom or a constraint atom a, let $var(a)$ denote the set of all variables occurring in a. Given a set A of atoms and/or constraint atoms, let $var(A) = \bigcup\{var(a) \mid a \in A\}$.

An *if-and-only-if formula* (for short, *iff-formula*) I on $\mathcal{A}$ is a formula of the form

$$a \leftrightarrow (conj_1 \vee \cdots \vee conj_n),$$

where $a \in \mathcal{A}$ and each of the $conj_i$ is a set of atoms in $\mathcal{A}$ and/or constraint atoms. The atom a is called the *head* of the iff-formula I, denoted by $head(I)$. When emphasis is given to its head, an iff-formula whose head is an atom a is often referred to as *iff*(a).

Let $I = (a \leftrightarrow (conj_1 \vee \cdots \vee conj_n))$ be an iff-formula. For each $i \in \{1,\ldots,n\}$, $conj_i$ corresponds to the the existentially quantified atom conjunction $\mathrm{FOL}(conj_i, a)$ given by

$$\mathrm{FOL}(conj_i, a) = \exists y_1 \cdots \exists y_k : \bigwedge\{b \mid b \in conj_i\},$$

where $\{y_1,\ldots,y_k\} = var(conj_i) - var(a)$. The iff-formula I corresponds to the universally quantified formula

$$\forall(a \leftrightarrow (\mathrm{FOL}(conj_1, a) \vee \cdots \vee \mathrm{FOL}(conj_n, a))),$$

which is denoted by $\mathrm{FOL}(I)$.

An iff-formula $(a \leftrightarrow (conj_1 \vee \cdots \vee conj_n))$ is in a *standard form* iff for any $i, j \in \{1,\ldots,n\}$, if $i \neq j$, then

$$(var(conj_i) - var(a)) \cap (var(conj_j) - var(a)) = \varnothing.$$

An iff-formula I can always be converted into a standard form, with its meaning given by $\mathrm{FOL}(I)$ being preserved, through variable renaming. It is assumed that all iff-formulas considered henceforth are in standard forms.

Assume that I is an iff-formula. The if-part and the only-if-part of $\mathrm{FOL}(I)$ are denoted by $\mathrm{FOL}_{\mathrm{If}}(I)$ and $\mathrm{FOL}_{\mathrm{onlyIf}}(I)$, respectively. Let $\mathrm{IF}(I)$, $\mathrm{ONLYIF}(I)$, and $\mathrm{CLS}(I)$ be the clause sets defined as follows:

- $\mathrm{IF}(I) = \mathrm{MPS}(\mathrm{FOL}_{\mathrm{If}}(I))$.
- $\mathrm{ONLYIF}(I) = \mathrm{MPS}(\mathrm{FOL}_{\mathrm{onlyIf}}(I))$.
- $\mathrm{CLS}(I) = \mathrm{IF}(I) \cup \mathrm{ONLYIF}(I)$.

Note that $\mathrm{CLS}(I)$ can be equivalently defined as $\mathrm{MPS}(\mathrm{FOL}(I))$, i.e., it is the clause set obtained by converting $\mathrm{FOL}(I)$ into a conjunctive normal form by meaning-preserving Skolemization.

Example 1. Suppose that I is an iff-formula

$$p(x,y) \leftrightarrow (\{q(x,y,z), r(z)\} \vee \{eq(x,w), s(x,y,w)\}),$$

where w, x, y, and z are usual variables. Then

$$\begin{aligned}\mathrm{FOL}(I) &= \forall x \forall y : p(x,y) \leftrightarrow ((\exists z : q(x,y,z) \wedge r(z)) \vee \\ &\qquad (\exists w : eq(x,w) \wedge s(x,y,w))),\end{aligned}$$

$$\begin{aligned}\mathrm{IF}(I) = \{&(p(x,y) \leftarrow q(x,y,z), r(z)),\\ &(p(x,y) \leftarrow eq(x,w), s(x,y,w))\},\end{aligned}$$

$$\begin{aligned}&\mathrm{ONLYIF}(I)\\ &= \{(q(x,y,z), eq(x,w) \leftarrow p(x,y), func(f_0,y,x,z),\\ &\qquad func(f_1,y,x,w)),\\ &\quad (q(x,y,z), s(x,y,w) \leftarrow p(x,y), func(f_0,y,x,z),\\ &\qquad func(f_1,y,x,w)),\\ &\quad (r(z), eq(x,w) \leftarrow p(x,y), func(f_0,y,x,z),\\ &\qquad func(f_2,y,x,w)),\\ &\quad (r(z), s(x,y,w) \leftarrow p(x,y), func(f_0,y,x,z),\\ &\qquad func(f_2,y,x,w))\}.\end{aligned}$$

□

Two iff-formulas I and I' on $\mathcal{A}$ are said to be *disjoint* iff $rep(head(I))$ and $rep(head(I'))$ are disjoint. Let E be a set of mutually disjoint iff-formulas on $\mathcal{A}$. E corresponds to the conjunction $\bigwedge\{\text{FOL}(I) \mid I \in E\}$, which is denoted by $\text{FOL}(E)$. Let $\text{IF}(E) = \bigcup\{\text{IF}(I) \mid I \in E\}$, $\text{ONLYIF}(E) = \bigcup\{\text{ONLYIF}(I) \mid I \in E\}$, and $\text{CLS}(E) = \text{IF}(E) \cup \text{ONLYIF}(E)$.

A *ground substitution* for an iff-formula $(a \leftrightarrow (conj_1 \vee \cdots \vee conj_n))$ is a substitution θ such that $a\theta$, $conj_1\theta, \ldots, conj_n\theta$ are all ground.

3.2 Quadruples for Transformation of QA Problems

3.2.1 A Quadruple Form

In order to clearly separate iff-formulas from clauses in background knowledge, we extend a QA problem on ECLS_F into a quadruple $\langle Cs, E, q, \pi \rangle$ on $\mathcal{A}$, where (i) Cs is a clause set in the ECLS_F space such that each usual atom appearing in Cs belongs to $\mathcal{A}$, (ii) E is a set of mutually disjoint iff-formulas on $\mathcal{A}$, (iii) $q \in \mathcal{A}$, and (iv) π is a partial mapping from $\mathcal{A}$ to $\mathcal{A}$ such that the domain of π contains all ground instances of q. The answer to the QA problem $\langle Cs, E, q, \pi \rangle$, denoted by $answer(Cs, E, q, \pi)$, is defined by

$$answer(Cs, E, q, \pi) = \pi((\bigcap Models(Cs \cup \text{CLS}(E))) \cap rep(q)). \quad (3)$$

3.2.2 Transformation into Quadruples

A QA problem $\langle K, q \rangle$ on first-order logic is transformed into a quadruple form on ECLS_F as follows:

1. From K, identify a first-order formula K' and a set E of mutually disjoint iff-formulas such that $K = K' \wedge \text{FOL}(E)$.
2. Convert K' by meaning-preserving Skolemization into a clause set Cs in the ECLS_F space, i.e., $Cs = \text{MPS}(K')$.
3. Construct $\langle Cs, E, q, id \rangle$, where id is the identity mapping, as the resulting quadruple.

Finding a nonempty set E of iff-formulas for converting K into $K' \wedge \text{FOL}(E)$ is useful for solving QA problems since iff-formulas increase the possibility of transforming QA problems with less cost (see Section 5). As the number of iff-formulas in the set E increases, such possibility is higher.

3.3 A Procedure for Solving QA Problems with Iff-Formulas

Assume that a QA problem $\langle K, q \rangle$ on first-order logic is given. To solve this problem using ET, perform the following steps:

1. Transform $\langle K, q \rangle$ into a quadruple $\langle Cs, E, q, id \rangle$ using the transformation given in Section 3.2.
2. Successively transform the quadruple $\langle Cs, E, q, id \rangle$ in the ECLS_F space using the following ET rules: Assume that $\langle \hat{Cs}, \hat{E}, \hat{q}, \pi \rangle$ is a QA problem.
   (a) If $\hat{E}$ contains an iff-formula $iff(a)$ and $\tilde{Cs}$ is obtained from $\hat{Cs}$ by replacement using $iff(a)$, then transform $\langle \hat{Cs}, \hat{E}, \hat{q}, \pi \rangle$ into $\langle \tilde{Cs}, \hat{E}, \hat{q}, \pi \rangle$,
   (b) If $\hat{E}$ contains an iff-formula $iff(a)$ and for each atom b that occurs in $\hat{Cs}$ or $\hat{E} - \{iff(a)\}$, a and b are not unifiable, then transform $\langle \hat{Cs}, \hat{E}, \hat{q}, \pi \rangle$ into $\langle \hat{Cs}, \hat{E} - \{iff(a)\}, \hat{q}, \pi \rangle$.
   (c) Transform $\langle \hat{Cs}, \hat{E}, \hat{q}, \pi \rangle$ by transformation of $\hat{Cs}$ and/or π using ET rules on ECLS_F, including the ET rules for unfolding (UNF) and definite-clause removal (RMD) in Section 5.2, and the ET rules given in (Akama and Nantajeewarawat, 2014) for
      - side-change transformation (SCH),
      - resolution (RESO),
      - elimination of isolated *func*-atoms (EIF),
      - elimination of subsumed clauses (ESUB),
      - elimination of valid clauses (EVAD),
      - erasing independent satisfiable atoms (EIS), and
      - elimination of satisfiable independent clauses (ESI).
   (d) Transform $\langle \hat{Cs}, \hat{E}, \hat{q}, \pi \rangle$ using ET rules for constraints, e.g., ET rules for equality constraints.
3. Assume that the transformation yields a quadruple $\langle Cs', E', q', \phi \rangle$. Then:
   (a) If Cs' is not satisfiable, then output $rep(\phi(q'))$ as the answer set.
   (b) If Cs' is a set of unit clauses the head of which are instances of q', then output the answer set $\bigcup_{C \in Cs'} rep(\phi(head(C)))$.
   (c) Otherwise stop with failure.

The obtained answer set is always correct since all transformation steps in the procedure are answer-preserving.

4 ET RULES IN THE PRESENCE OF IFF-FORMULAS

Next, the replacement operation using an iff-formula is defined. It is followed by ET rules for replacement using iff-formulas and for removing useless iff-formulas.

4.1 Replacement Using Iff-Formulas

Assume that (i) Cs is a set of clauses, (ii) occ is an occurrence of an atom b in a clause $C \in Cs$, (iii) $iff(a)$ is an iff-formula $(a \leftrightarrow (conj_1 \vee \cdots \vee conj_n))$, (iv) ρ is a renaming substitution for usual variables such that C and $iff(a)\rho$ have no usual variable in common, and (v) θ is the most general matcher of $a\rho$ into b (i.e., the most general substitution such that $a\rho\theta = b$). Then:

- Let $\textsc{Repl}(C, occ, iff(a), \rho, \theta)$ denote the first-order formula obtained by replacing b at occ with the disjunction $\textsc{Fol}(conj_1\rho\theta, a\rho\theta) \vee \cdots \vee \textsc{Fol}(conj_n\rho\theta, a\rho\theta)$ using ρ and θ.
- Let $\textsc{Repl}(Cs, C, occ, iff(a), \rho, \theta)$ denote the conjunction of $\textsc{Repl}(C, occ, iff(a), \rho, \theta)$ and all clauses in $Cs - \{C\}$.

Note that occ is an occurrence at any arbitrary position in C (i.e., it can be in the left-hand side or the right-hand side of C). In general, $\textsc{Repl}(C, occ, iff(a), \rho, \theta)$ is not a clause. After the replacement of occ, a new clause set, say Cs', is obtained by using meaning-preserving Skolemization, i.e., $Cs' = \mathrm{MPS}(\textsc{Repl}(Cs, C, occ, iff(a), \rho, \theta))$. The resulting clause set Cs' is often simply said to be obtained by replacement using $iff(a)$ at the occurrence occ of b in C.

4.2 ET Rules for Iff-Formulas and Their Correctness

An ET rule on $\mathrm{ECLS_F}$ for replacement using an iff-formula is given by Theorem 2 and that for removing a useless iff-formula is given by Theorem 3. Let $\langle Cs, E, q, \pi \rangle$ be a QA problem on $\mathrm{ECLS_F}$.

Theorem 2. (Replacement Using an Iff-Formula) Assume that:

1. E contains an iff-formula $iff(a)$.
2. $rep(a) \cap rep(q) = \varnothing$.
3. occ is an occurrence of an atom b in a clause $C \in Cs$.
4. ρ is a renaming substitution for usual variables such that C and $iff(a)\rho$ have no usual variable in common.
5. θ is the most general matcher of $a\rho$ into b.
6. $Cs' = \mathrm{MPS}(\textsc{Repl}(Cs, C, occ, iff(a), \rho, \theta))$.

Then $\langle Cs, E, q, \pi \rangle$ can be equivalently transformed into $\langle Cs', E, q, \pi \rangle$.

Proof. Assume that $iff(a) = (a \leftrightarrow (conj_1 \vee \cdots \vee conj_n))$ and F is the disjunction

$$\textsc{Fol}(conj_1\rho\theta, a\rho\theta) \vee \cdots \vee \textsc{Fol}(conj_n\rho\theta, a\rho\theta).$$

Then Cs' is obtained by applying meaning-preserving Skolemization to the formula resulting from replacing $a\rho\theta$ in Cs with F. Since $a\rho\theta$ is logically equivalent to F, Cs and Cs' are logically equivalent in the presence of $\textsc{Cls}(E)$. Hence $answer(Cs, E, q, \pi) = answer(Cs', E, q, \pi)$. □

Theorem 3. (Removal of an Iff-Formula) Assume that:

1. E contains an iff-formula $iff(a)$.
2. $rep(a) \cap rep(q) = \varnothing$.
3. For each atom b that occurs in $\hat{Cs}$ or $\hat{E} - \{iff(a)\}$, a and b are not unifiable.

Then $\langle Cs, E, q, \pi \rangle$ can be equivalently transformed into $\langle Cs, E - \{iff(a)\}, q, \pi \rangle$.

Proof. Let $Cs_1 = Cs \cup \textsc{Cls}(E)$ and $Cs_2 = Cs \cup (\textsc{Cls}(E) - \textsc{Cls}(iff(a)))$. Obviously, $Models(Cs_1) \subseteq Models(Cs_2)$. Based on this, we prove that

$$rep(q) \cap (\bigcap Models(Cs_1)) = rep(q) \cap (\bigcap Models(Cs_2)) \quad (4)$$

by further showing as follows that for any $G \in Models(Cs_2)$, there exists $M \in Models(Cs_1)$ such that $G \cap rep(q) = M \cap rep(q)$. Assume that $G \in Models(Cs_2)$. Let $G' = G - rep(a)$. By Assumption 3 of this theorem, G' is also a model of Cs_2. Assuming that $iff(a) = (a \leftrightarrow (conj_1 \vee \cdots \vee conj_n))$, let D_p be defined as the set of definite clauses

$$\{C \mid (i \in \{1, \ldots, n\}) \;\&\; (\theta \text{ is a ground substitution for } iff(a)) \;\&\; (G' \text{ contains all atoms in } conj_i\theta \text{ that are not instances of } a) \;\&\; (C \text{ is a definite clause with } head(C) = a\theta \text{ and } body(C) = conj_i\theta - G')\}.$$

Let $M = \mathcal{M}(D_p) \cup G'$, where $\mathcal{M}(D_p)$ is the least model of D_p. By Assumption 3 of this theorem, M is a model of Cs_1. By Assumption 2 of this theorem, $G \cap rep(q) = G' \cap rep(q) = M \cap rep(q)$.

As a result, (4) holds. It follows from Equation (3) in Section 3.2 that $answer(Cs, E, q, \pi) = answer(Cs, E - \{iff(a)\}, q, \pi)$. □

Application of Theorems 2 and 3 is illustrated below.

Example 2. Assume that ans is a query atom and K is the union of a clause set Cs and the set $\{I_B, C_1, C_2\}$, where I_B is the iff-formula

$$I_B\colon\ B \leftrightarrow (\{C, D\} \vee \{H\})$$

and C_1 and C_2 are the following clauses:

C_1: $ans \leftarrow A, B$
C_2: $B \leftarrow A, C$

By Theorem 2, the clause C_1 can be transformed by replacement using I_B into:

C_3: $ans \leftarrow A, C, D$
C_4: $ans \leftarrow A, H$

Again by Theorem 2, replacement using I_B is applicable to C_2, and the replacement transforms C_2 into:

C_5: $C, H \leftarrow A, C$
C_6: $D, H \leftarrow A, C$

Note that $\{C_5, C_6\} = \mathrm{MPS}(((C \wedge D) \vee H) \leftarrow (A \wedge C))$. By Theorem 3, if the clause set Cs contains no occurrence of B, then the iff-formula I_B can be removed. □

5 A COMPARISON BETWEEN REPLACEMENT AND UNFOLDING

After introducing the unfolding operation on $\mathrm{ECLS_F}$ in Section 5.1 and presenting ET rules for unfolding and for definite-clause removal in Section 5.2, replacement using an iff-formula is compared with unfolding in Section 5.3. Section 5.4 illustrates how iff-formulas are useful for reduction of computation cost.

5.1 Unfolding Operation on $\mathrm{ECLS_F}$

Assume that Cs is a clause set in $\mathrm{ECLS_F}$, D is a definite-clause set in $\mathrm{ECLS_F}$, and occ is an occurrence of an atom b in the right-hand side of a clause C in Cs. By unfolding Cs using D at occ, Cs is transformed into

$$(Cs - \{C\}) \cup (\bigcup \{resolvent(C, C', b) \mid C' \in D\}),$$

where for each $C' \in D$, $resolvent(C, C', b)$ is defined as follows, assuming that ρ is a renaming substitution for usual variables such that C and $C'\rho$ have no usual variable in common:

1. If b and $head(C'\rho)$ are not unifiable, then $resolvent(C, C', b) = \varnothing$.
2. If they are unifiable, with θ being their most general unifier, then $resolvent(C, C', b) = \{C''\}$, where C'' is the clause obtained from C and $C'\rho$ as follows:
   (a) $lhs(C'') = lhs(C\theta)$
   (b) $rhs(C'') = (rhs(C\theta) - \{b\theta\}) \cup body(C'\rho\theta)$

The resulting clause set is denoted by $\mathrm{UNF}(Cs, C, occ, D)$.

5.2 ET by Unfolding and Definite-Clause Removal

For any predicate p, let $Atoms(p)$ denote the set of all atoms having the predicate p. ET rules on $\mathrm{ECLS_F}$ for unfolding and for definite-clause removal (Akama and Nantajeewarawat, 2013c) are given below. Assume that $\langle Cs, E, q, \pi \rangle$ is a QA problem on $\mathrm{ECLS_F}$.

5.2.1 ET by Unfolding (UNF)

Suppose that:

1. p_q is the predicate of the query atom q.
2. p is a predicate such that $p \neq p_q$.
3. D is a set of definite clauses in Cs that satisfies the following conditions:
   (a) For any definite clause $C \in D$, $head(C) \in Atoms(p)$.
   (b) For any clause $C' \in (Cs \cup \mathrm{CLS}(E)) - D$, $lhs(C') \cap Atoms(p) = \varnothing$.
4. occ is an occurrence of an atom in $Atoms(p)$ in the right-hand side of a clause C in $(Cs \cup \mathrm{IF}(E)) - D$.

Then $\langle Cs, E, q, \pi \rangle$ can be equivalently transformed into the QA problem $\langle \mathrm{UNF}(Cs, C, occ, D), E, q, \pi \rangle$.

In order to apply unfolding to $\langle Cs, E, q, \pi \rangle$, we have to find a set D of definite clauses in Cs that satisfies Condition 3. By Conditions 3a and 3b, we select a predicate p and collect all definite clauses with p-atoms in their heads. To satisfy Condition 3b, a p-atom can neither appear in the left-hand side of a multi-head clause in Cs nor appear in an iff-formula in E. These conditions often disable application of unfolding ET rules in solving problems with multi-head clauses.

5.2.2 ET by Definite-Clause Removal (RMD)

Suppose that:

1. p_q is the predicate of the query atom q.
2. p is a predicate such that $p \neq p_q$.
3. D is a set of definite clauses in Cs that satisfies the following conditions:
   (a) For any definite clause $C \in D$, $head(C) \in Atoms(p)$.
   (b) For any clause $C' \in (Cs \cup \mathrm{CLS}(E)) - D$, $lhs(C') \cap Atoms(p) = \varnothing$.
4. For any clause $C' \in (Cs \cup \mathrm{CLS}(E)) - D$, $rhs(C') \cap Atoms(p) = \varnothing$.

Then $\langle Cs, E, q, \pi \rangle$ can be equivalently transformed into the QA problem $\langle Cs - D, E, q, \pi \rangle$.

Next, an example showing application of unfolding and definite-clause removal is given.

Example 3. Consider the Oedipus problem described in (Baader et al., 2007). Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Polyneikes also had children, among them Thersandros, who is not a patricide. The problem is to find a person who has a patricide child who has a non-patricide child. The difficulty of this problem arises from the absence of information as to whether Polyneikes is a patricide or not.

Assume that "*oe*," "*io*," "*po*," and "*th*" stand, respectively, for Oedipus, Iokaste, Polyneikes, and Thersandros. This problem is represented as a QA problem with the query atom $prob(x)$ and the background knowledge consisting of the following clauses:

C_1: $prob(x), pat(y) \leftarrow isChildOf(z,x), pat(z), isChildOf(y,z)$
C_2: $isChildOf(oe, io) \leftarrow$
C_3: $isChildOf(po, io) \leftarrow$
C_4: $isChildOf(po, oe) \leftarrow$
C_5: $isChildOf(th, po) \leftarrow$
C_6: $pat(oe) \leftarrow$
C_7: $\leftarrow pat(th)$

Since C_1 is a multi-head clause containing a *pat*-atom in its left-hand side, unfolding at the *pat*-atom in the right-hand side of C_1 is disabled. By unfolding at the first *isChildOf*-atom, i.e., $isChildOf(z,x)$, in its right-hand side, the clause C_1 is transformed into the following four clauses:

C_8: $prob(io), pat(y) \leftarrow pat(oe), isChildOf(y, oe)$
C_9: $prob(io), pat(y) \leftarrow pat(po), isChildOf(y, po)$
C_{10}: $prob(oe), pat(y) \leftarrow pat(po), isChildOf(y, po)$
C_{11}: $prob(po), pat(y) \leftarrow pat(th), isChildOf(y, th)$

By further unfolding at *isChildOf*-atoms four times successively, the clauses C_8–C_{11} are transformed into:

C_{12}: $prob(io), pat(po) \leftarrow pat(oe)$
C_{13}: $prob(io), pat(th) \leftarrow pat(po)$
C_{14}: $prob(oe), pat(th) \leftarrow pat(po)$

Since the predicate *isChildOf* does not appear in the right-hand side of any of C_6, C_7 and C_{12}–C_{14}, the definite clauses C_2–C_5 are removed. The resulting clause set is $Cs = \{C_6, C_7, C_{12}, C_{13}, C_{14}\}$. At this point, *pat* is the only predicate of a possible target body atom for unfolding. However, since each of C_{12}, C_{13} and C_{14} also contains a *pat*-atom in its left-hand side, no further unfolding is applicable to *Cs*.

Using other equivalent transformation rules, *Cs* can be further transformed as follows: By forwarding transformation (Akama and Nantajeewarawat, 2012b) with respect to C_7, the clauses C_{13} and C_{14} are changed into:

C_{15}: $prob(io) \leftarrow pat(po)$
C_{16}: $prob(oe) \leftarrow pat(po)$

By erasing independent satisfiable atoms (Akama and Nantajeewarawat, 2014), C_{12} is replaced with:

C_{17}: $prob(io), pat(po) \leftarrow$

By resolution and elimination of subsumed clauses, C_{15} and C_{17} are replaced with:

C_{18}: $prob(io) \leftarrow$

The resulting clause set is $Cs' = \{C_6, C_7, C_{16}, C_{18}\}$. Since no atom in the left-hand side of any clause in Cs' can be instantiated into $pat(po)$, C_{16} is removed. The obtained answer set is thus the singleton set $\{prob(io)\}$, i.e., Iokaste is the only answer to this problem (no matter whether Polyneikes is a patricide).

Alternatively, after the original background knowledge is simplified by unfolding and definite-clause removal into $Cs = \{C_6, C_7, C_{12}, C_{13}, C_{14}\}$, the simplified QA problem $\langle Cs, prob(x) \rangle$ can also be solved by using bottom-up computation (Akama and Nantajeewarawat, 2012a) or by using a SAT solver (Akama and Nantajeewarawat, 2013a). □

5.3 Replacement Using Iff-Formulas vs. Unfolding

Assume that *Cs* is a clause set in the ECLS_F space, *C* is a clause in *Cs*, *occ* is an occurrence of an atom *b* in the right-hand side of *C*, and $iff(a)$ is an iff-formula. Consider two QA problems Prb_1 and Prb_2, given by:

- $Prb_1 = \langle Cs, E \cup \{iff(a)\}, q, id \rangle$,
- $Prb_2 = \langle Cs \cup \text{CLS}(iff(a)), E, q, id \rangle$.

By the definitional equation (3), Prb_1 is equivalent to Prb_2.

Whenever replacement is applicable using a renaming substitution ρ and the most general matcher θ of $a\rho$ into *b*, Prb_1 is transformed into an equivalent QA problem $Prb_1' = \langle Cs', E \cup \{iff(a)\}, q, id \rangle$, where $Cs' = \text{MPS}(\text{REPL}(Cs, C, occ, iff(a), \rho, \theta))$. Referring to Section 5.2, if the required conditions for ET by unfolding $Cs \cup \text{CLS}(iff(a))$ using $\text{IF}(iff(a))$ at *occ* are satisfied, then Prb_2 can be equivalently transformed by unfolding into a QA problem $Prb_2' = \langle Cs'', E, q, id \rangle$, where $Cs'' = \text{UNF}(Cs \cup \text{CLS}(iff(a)), C, occ, \text{IF}(iff(a)))$. The changes are made

by the above two transformation steps only in the clause-set parts, i.e., at the first arguments of the quadruples representing Prb_1 and Prb_2.

It can be shown that the changes made by them are exactly the same (i.e., $Cs' \cup \textsc{Cls}(\mathit{iff}(a)) = Cs''$) as follows: Assume that $\mathit{iff}(a) = (a \leftrightarrow (\mathit{conj}_1 \vee \cdots \vee \mathit{conj}_n))$, ρ is a renaming substitution for usual variables such that C and $\mathit{iff}(a)\rho$ have no usual variable in common, and θ is the most general matcher of $a\rho$ into b. The substitution θ used above is also the most general unifier of $a\rho$ and b. It is thus also used for unifying $a\rho$ and b in the unfolding step.

- By replacement using $\mathit{iff}(a)$ followed by conversion using meaning-preserving Skolemization, n copies of C are produced and for each $i \in \{1,\ldots,n\}$, the atom b at occ in the ith copy is replaced with $\mathit{conj}_i\rho\theta$. All other atoms in each copy of C are unchanged.
- By the unfolding operation with respect to $\textsc{If}(\mathit{iff}(a))$, which is the set $\{(a \leftarrow \mathit{conj}_i) \mid i \in \{1,\ldots,n\}\}$, C is transformed into n clauses, say, $C_1,\ldots,C_n$, and for each $i \in \{1,\ldots,n\}$, the atom b at the occurrence occ is replaced with $\mathit{conj}_i\rho\theta$ in the construction of C_i. Although θ is also applied to all other atoms in C, it makes no change to those atoms since C and $a\rho$ have no usual variable in common and θ only instantiates variables occurring in $a\rho$.

Hence the two transformation steps make the same change to the target clause C. As a result, $Cs' \cup \textsc{Cls}(\mathit{iff}(a)) = Cs''$.

However, the required conditions for applicability of the two transformations are totally different. While replacement is always applicable, the required conditions for unfolding (cf. Section 5.2) are easily violated when Cs contains a multi-head clause with an instance of a in its left-hand side. Such violation disables unfolding. As a consequence, replacement by iff-formulas in the quadruple form gives higher possibility of transformation compared to unfolding in the quadruple form (and, thus, also unfolding in the triple form).

Example 4. Assume that the query atom is $prob(x)$ and the background knowledge K includes the conjunction of the following first-order formulas, where "*Co*," "*nt*," "*te*," "*AC*," and "*BC*" are abbreviations for "course," "non-teaching professor," "teach," "advanced course," and "basic course," respectively:

F_1: $\forall x : ((\exists y : (Co(y) \wedge te(x,y))) \rightarrow prob(x))$
F_2: $\forall x : (nt(x) \leftrightarrow \neg(\exists y : te(x,y) \wedge Co(y)))$
F_3: $\forall x : (Co(x) \leftrightarrow (AC(x) \vee BC(x)))$

By meaning-preserving Skolemization (Akama and Nantajeewarawat, 2011), the conjunction $F_1 \wedge F_2 \wedge F_3$ is converted into the following extended clauses, where f is a unary function variable:

C_1: $prob(x) \leftarrow Co(y), te(x,y)$
C_2: $\leftarrow nt(x), te(x,y), Co(y)$
C_3: $te(x,y), nt(x) \leftarrow func(f,x,y)$
C_4: $Co(x), nt(y) \leftarrow func(f,y,x)$
C_5: $Co(x) \leftarrow AC(x)$
C_6: $Co(x) \leftarrow BC(x)$
C_7: $AC(x), BC(x) \leftarrow Co(x)$

These clauses are used in the triple form. Using them, unfolding at the *Co*-atom in the body of C_1 is blocked due to the presence of a *Co*-atom in the left-hand side of the multi-head clause C_4. If such an unfolding step is not blocked, then it transforms C_1 into:

C_a: $prob(x) \leftarrow AC(y), te(x,y)$
C_b: $prob(x) \leftarrow BC(y), te(x,y)$

By contrast, if the quadruple form is used, then the clauses C_5–C_7 are replaced with a single iff-formula

I_{Co}: $Co(x) \leftrightarrow (\{AC(x)\} \vee \{BC(x)\})$

and replacement using I_{Co} is applicable at the *Co*-atom in C_1. This replacement transforms C_1 into the following two clauses:

C_8: $prob(x) \leftarrow AC(y), te(x,y)$
C_9: $prob(x) \leftarrow BC(y), te(x,y)$

The resulting clauses C_8 and C_9 are equal to C_a and C_b, respectively. □

5.4 Overcoming Computation Difficulty by Using Iff-Formulas

Assume that K is the first-order formula

$$\begin{aligned}\forall x\forall y\forall z : (&app(x,y,z)\\ &\leftrightarrow ((eq(x,[]) \wedge eq(y,z)) \vee\\ &\quad (\exists A\exists X\exists Z : (eq(x,[A|X]) \wedge eq(z,[A|Z])\\ &\qquad \wedge app(X,y,Z))))),\end{aligned}$$

where "*app*" stands for "append." Consider the QA problem $\langle K,q\rangle$, where $q = app([1,2,3],[4,5],x)$. This problem is solved by successively transforming the clause

A_1: $ans(z) \leftarrow app([1,2,3],[4,5],z)$

into unit clauses.

We first show that a solution with the triple form, using only unfolding and resolution, results in computation difficulty. By meaning-preserving Skolemization, the first-order formula K is converted into the set Cs consisting of the following clauses, where f_0–f_5 are function variables:

C_1: $app(x,y,z) \leftarrow eq(x,[]), eq(y,z)$
C_2: $app(x,y,z) \leftarrow eq(x,[A|X]), eq(z,[A|Z]), app(X,y,Z)$
C_3: $eq(x,[]), eq(x,[A|X]) \leftarrow app(x,y,z), func(f_0,z,y,x,A), func(f_1,z,y,x,X), func(f_2,z,y,x,Z)$
C_4: $eq(x,[]), eq(z,[A|Z]) \leftarrow app(x,y,z), func(f_0,z,y,x,A), func(f_1,z,y,x,X), func(f_2,z,y,x,Z)$
C_5: $eq(x,[]), app(X,y,Z) \leftarrow app(x,y,z), func(f_0,z,y,x,A), func(f_1,z,y,x,X), func(f_2,z,y,x,Z)$
C_6: $eq(y,z), eq(x,[A|X]) \leftarrow app(x,y,z), func(f_3,z,y,x,A), func(f_4,z,y,x,X), func(f_5,z,y,x,Z)$
C_7: $eq(y,z), eq(z,[A|Z]) \leftarrow app(x,y,z), func(f_3,z,y,x,A), func(f_4,z,y,x,X), func(f_5,z,y,x,Z)$
C_8: $eq(y,z), app(X,y,Z) \leftarrow app(x,y,z), func(f_3,z,y,x,A), func(f_4,z,y,x,X), func(f_5,z,y,x,Z)$

Among C_1–C_8, there are four clauses whose left-hand sides contain *app*-atoms, i.e., C_1, C_2, C_5, and C_8. Since C_5 and C_8 are multi-head clauses, unfolding at the *app*-atom in the body of the clause A_1 is blocked.

Instead, resolution is applicable to the clause A_1 and it produces the following resolvent clauses:

A_1: $ans(z) \leftarrow app([1,2,3],[4,5],z)$
(by applying resolution to A_1 and C_2)
A_2: $ans([1|z_1]) \leftarrow app([2,3],[4,5],z_1)$
(by applying resolution to A_2 and C_2)
A_3: $ans([1,2|z_2]) \leftarrow app([3],[4,5],z_2)$
(by applying resolution to A_3 and C_2)
A_4: $ans([1,2,3|z_3]) \leftarrow app([],[4,5],z_3)$
(by applying resolution to A_4 and C_1)
A_5: $ans([1,2,3,4,5]) \leftarrow$

The clause A_5 indicates that $[1,2,3,4,5]$ is one result of concatenating $[1,2,3]$ and $[4,5]$. However, we cannot conclude this is the only result. The reason is that resolution adds resolvent clauses to the original clause set and, therefore, the above resolution steps transform $Cs \cup \{A_1\}$ as follows:

$Cs \cup \{A_1\}$
$\Rightarrow Cs \cup \{A_1, A_2\}$
$\Rightarrow Cs \cup \{A_1, A_2, A_3\}$
$\Rightarrow Cs \cup \{A_1, A_2, A_3, A_4\}$
$\Rightarrow Cs \cup \{A_1, A_2, A_3, A_4, A_5\}$

Some other result of concatenating $[1,2,3]$ and $[4,5]$ may be obtained by other clauses, i.e., C_3–C_8 and A_1–A_4. To ensure that no such additional result exists, removal of A_1–A_4 is required. Neither unfolding nor resolution removes them. Removing A_1–A_4 by other transformation rules, if possible, tends to take non-negligible computation cost.

Next, we show that the above difficulty can be resolved by using an iff-formula. The original formula K can be transformed equivalently into the iff-formula

I_{app}: $app(x,y,z) \leftrightarrow (\{eq(x,[]), eq(y,z)\} \vee \{eq(x,[A|X]), eq(z,[A|Z]), app(X,y,Z)\})$.

We then consider the quadruple

$$\langle \{A_1\}, \{I_{app}\}, ans(x), \pi \rangle,$$

where π is a mapping such that for any term t, $\pi(ans(t)) = app([1,2,3],[4,5],t)$.

By repeated replacement using I_{app} and equivalent transformation with respect to *eq*-atoms, the clause set $\{A_1\}$ is transformed into the singleton unit-clause set $\{(ans([1,2,3,4,5]) \leftarrow)\}$ as follows:

$\{(ans(z) \leftarrow app([1,2,3],[4,5],z))\}$
$\Rightarrow$ (by replacement using I_{app})
$\{(ans(z) \leftarrow eq([1,2,3],[]), eq([4,5],z)), (ans(z) \leftarrow eq([1,2,3],[A|X]), eq(z,[A|Z]), app(X,[4,5],Z))\}$
$\Rightarrow$ (by equality solving)
$\{(ans([1|z_1]) \leftarrow app([2,3],[4,5],z_1))\}$
$\Rightarrow$ (by replacement using I_{app})
$\{(ans([1|z_1]) \leftarrow eq([2,3],[]), eq([4,5],z_1)), (ans([1|z_1]) \leftarrow eq([2,3],[A|X]), eq(z_1,[A|Z]), app(X,[4,5],Z))\}$
$\Rightarrow$ (by equality solving)
$\{(ans([1,2|z_2]) \leftarrow app([3],[4,5],z_2))\}$
$\Rightarrow$ (by replacement using I_{app})
$\{(ans([1,2|z_2]) \leftarrow eq([3],[]), eq([4,5],z_2)), (ans([1,2|z_2]) \leftarrow eq([3],[A|X]), eq(z_2,[A|Z]), app(X,[4,5],Z))\}$
$\Rightarrow$ (by equality solving)
$\{(ans([1,2,3|z_3]) \leftarrow app([],[4,5],z_3))\}$
$\Rightarrow$ (by replacement using I_{app})
$\{(ans([1,2,3|z_3]) \leftarrow eq([],[]), eq([4,5],z_3)), (ans([1,2,3|z_3]) \leftarrow eq([],[A|X]), eq(z_3,[A|Z]), app(X,[4,5],Z))\}$
$\Rightarrow$ (by equality solving)
$\{(ans([1,2,3,4,5]) \leftarrow)\}$

By the removal transformation for an iff-formula, I_{app} is removed. Then

$answer(\{A_1\}, \{I_{app}\}, ans(x), \pi)$
$= answer(\{(ans([1,2,3,4,5]) \leftarrow)\}, \varnothing, ans(x), \pi)$
$= \pi(\{ans([1,2,3,4,5])\} \cap rep(ans(x)))$
$= \pi(\{ans([1,2,3,4,5])\})$
$= \{app([1,2,3],[4,5],[1,2,3,4,5])\}$,

which means $[1,2,3,4,5]$ is the only result of concatenating $[1,2,3]$ and $[4,5]$. The aforementioned difficulty is thus overcome with small computation cost.

Structural	Relational
$NFP \equiv FP \sqcap NT$	$curr(x,z) \leftarrow exam(x,y), subject(y,z)$
$NT \equiv \neg\exists teach.Co$	$\&\ x\colon St, y\colon Co, z\colon Tp$
$FP \sqsubseteq FM$	$mayDoTh(x,y) \leftarrow curr(x,z), expert(y,z)$
$AC \sqcup BC \equiv Co$	$\&\ x\colon St, z\colon Tp, y\colon FP \sqcap \exists teach.AC$
$AC \sqcap BC \sqsubseteq \bot$	$mayDoTh(x,y) \leftarrow \&\ x\colon St, y\colon NFP$
$john\colon FP,\ teach(john, ai)$	$exam(paul, ai)$
$mary\colon FP \sqcap \forall teach.AC$	$subject(ai, kr),\ subject(ai, lp)$
$paul\colon St,\ ai\colon AC,\ kr\colon Tp,\ lp\colon Tp$	$expert(john, kr),\ expert(mary, lp)$

Figure 1: A knowledge base.

C_1: $\leftarrow NT(x), teach(x,y), Co(y)$ C_2: $teach(x,y), NT(x) \leftarrow func(f,x,y)$ C_3: $Co(x), NT(y) \leftarrow func(f,y,x)$
C_4: $FM(x) \leftarrow FP(x)$ C_5: $\leftarrow AC(x), BC(x)$ C_6: $FP(john) \leftarrow$
C_7: $teach(john, ai) \leftarrow$ C_8: $FP(mary) \leftarrow$ C_9: $AC(x) \leftarrow teach(mary, x)$
C_{10}: $St(paul) \leftarrow$ C_{11}: $AC(ai) \leftarrow$ C_{12}: $Tp(kr) \leftarrow$
C_{13}: $Tp(lp) \leftarrow$ C_{14}: $exam(paul, ai) \leftarrow$ C_{15}: $subject(ai, kr) \leftarrow$
C_{16}: $subject(ai, lp) \leftarrow$ C_{17}: $expert(john, kr) \leftarrow$ C_{18}: $expert(mary, lp) \leftarrow$

C_{19}: $curr(x,z) \leftarrow exam(x,y), subject(y,z), St(x), Co(y), Tp(z)$
C_{20}: $mayDoTh(x,y) \leftarrow curr(x,z), expert(y,z), St(x), Tp(z), FP(y), AC(w), teach(y,w)$
C_{21}: $mayDoTh(x,y) \leftarrow St(x), NFP(y)$

I_1: $NFP(x) \leftrightarrow \{FP(x), NT(x)\}$ I_2: $Co(x) \leftrightarrow (\{AC(x)\} \vee \{BC(x)\})$

Figure 2: Extended clauses and iff-formulas in the $ECLS_F$ space representing the knowledge base in Fig. 1.

6 EXAMPLE

Consider the knowledge base in Fig. 1, slightly modified from (Donini et al., 1998), where (i) the two columns refer to the structural component and the relational component and (ii) the two rows refer to the intensional level and the extensional level. The structural component is described using the description logic $\mathcal{ALC}$ (Baader et al., 2007). The intensional part of the relational component is described using an extension of Horn clauses, where class membership constraints are specified after the symbol '&'. This intensional part provides the conditions for a student to do his/her thesis with a professor. The query to be considered is to find every pair of a student s and a professor p such that s may do his/her thesis with p.

Using standard translation from $\mathcal{ALC}$ to first-order logic (Baader et al., 2007), the knowledge base in Fig. 1 can be converted into a conjunction of first-order formulas. With reference to Fig. 2, the obtained formula conjunction can further be converted into $Cs \cup E$, where Cs is the clause set consisting of C_1–C_{21} and $E = \{I_1, I_2\}$. The problem can then be formalized as the quadruple form $\langle Cs, E, mayDoTh(x,y), id\rangle$, where id is the identity mapping.

At Step 2 of the procedure in Section 3.3, the above quadruple is successively transformed. Replacement using I_1 is applied to C_{21} (Theorem 2), and I_1 is then removed (Theorem 3). Replacement using I_2 is applied to C_{19} (Theorem 2). The transformation rules mentioned at Step 2c of the procedure are next applied in the following order: UNF (25 times), RMD (8 times), SCH, UNF (6 times), RMD, EIS, ESUB (4 times), RESO (3 times), EIS, EIF, ESUB (5 times), SCH, and ESI. The final clause set consists only of the two unit clauses $(mayDoTh(paul, john) \leftarrow)$ and $(mayDoTh(paul, mary) \leftarrow)$, from which the answer set $\{mayDoTh(paul, john), mayDoTh(paul, mary)\}$ is derived.

7 CONCLUSIONS

The ET principle provides a basis for solving a very large class of QA problems. Our proposed ET-based procedure for solving QA problems is a state-transition procedure in which a state is a QA problem and application of an ET rule results in state transition. Using ET, a given QA problem is transformed equivalently into simpler forms until its answer set can be readily obtained. The design of an appropriate representation of the state space, i.e., an appropriate form for representing QA problems, is essen-

tial for ET-based problem solving. A triple form of a QA problem was previously used, where the first component is a set of extended clauses with function variables, representing the background knowledge of the problem, the second component is a query atom, and the third one is a mapping for converting ground atoms into elements of an answer set.

The background knowledge of a QA problem often includes iff-formulas, which are useful for problem transformation. By introducing a set of iff-formulas as a new component, this paper proposes a quadruple form for representing a QA problem. Iff-formulas in the quadruple form provide higher chance of transformation with less cost compared to the triple form. ET rules for using iff-formulas are invented, i.e., an ET rule for replacement using an iff-formula and that for removal of a useless iff-formula. Each transition step by an ET rule preserves the answer set of a given input problem and, consequently, the correctness of the proposed procedure with any combination of ET rules is guaranteed.

REFERENCES

Akama, K. and Nantajeewarawat, E. (2008). Meaning-Preserving Skolemization on Logical Structures. In *Proceedings of the 9th International Conference on Intelligent Technologies*, pages 123–132, Samui, Thailand.

Akama, K. and Nantajeewarawat, E. (2011). Meaning-Preserving Skolemization. In *Proceedings of the 3rd International Conference on Knowledge Engineering and Ontology Development*, pages 322–327, Paris, France.

Akama, K. and Nantajeewarawat, E. (2012a). A Delayed Splitting Bottom-Up Procedure for Model Generation. In *Proceedings of 25th Australasian Joint Conference on Artificial Intelligence*, LNCS 7691, pages 481–492, Sydney, Australia.

Akama, K. and Nantajeewarawat, E. (2012b). Conjunction-Based Clauses for Equivalent Transformation of Query-Answering Problems. *International Journal of Future Computer and Communication*, 1:5–8.

Akama, K. and Nantajeewarawat, E. (2013a). Correctness of Solving Query-Answering Problems Using Satisfiability Solvers. In *Proceedings of the 5th Asian Conference on Intelligent Information and Database Systems*, LNAI 7802, pages 404–413, Kuala Lumpur, Malaysia.

Akama, K. and Nantajeewarawat, E. (2013b). Embedding Proof Problems into Query-Answering Problems and Problem Solving by Equivalent Transformation. In *Proceedings of the 5th International Conference on Knowledge Engineering and Ontology Development*, pages 253–260, Vilamoura, Portugal.

Akama, K. and Nantajeewarawat, E. (2013c). Unfolding-Based Simplification of Query-Answering Problems in an Extended Clause Space. *International Journal of Innovative Computing, Information and Control*, 9:3515–3526.

Akama, K. and Nantajeewarawat, E. (2014). Equivalent Transformation in an Extended Space for Solving Query-Answering Problems. In *Proceedings of the 6th Asian Conference on Intelligent Information and Database Systems*, LNAI 8397, pages 232–241, Bangkok, Thailand.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2007). *The Description Logic Handbook*. Cambridge University Press, second edition.

Chang, C.-L. and Lee, R. C.-T. (1973). *Symbolic Logic and Mechanical Theorem Proving*. Academic Press.

Donini, F. M., Lenzerini, M., Nardi, D., and Schaerf, A. (1998). $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 16:227–252.

Horrocks, I., Patel-schneider, P. F., Bechhofer, S., and Tsarkov, D. (2005). OWL Rules: A Proposal and Prototype Implementation. *Journal of Web Semantics*, 3(1):23–40.

Lloyd, J. W. (1987). *Foundations of Logic Programming*. Springer-Verlag, second, extended edition.

Minker, J., editor (1988). *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Motik, B. and Rosati, R. (2010). Reconciling Description Logics and Rules. *Journal of the ACM*, 57(30):1–62.

Motik, B., Sattler, U., and Studer, R. (2005). Query Answering for OWL-DL with Rules. *Journal of Web Semantics*, 3(1):41–60.

APPENDIX

Transformation rules used in the meaning-preserving Skolemization procedure proposed in (Akama and Nantajeewarawat, 2011) for converting a first-order formula into an equivalent set of extended clauses are given below, where α, β, γ are first-order formulas, $x, x_1, \ldots, x_n, y$ are usual variables, and h is a function variable.

$$
\begin{array}{rcl}
\neg(\neg\beta) & \equiv & \beta \\
\neg(\beta \wedge \gamma) & \equiv & \neg\beta \vee \neg\gamma \\
\neg(\beta \vee \gamma) & \equiv & \neg\beta \wedge \neg\gamma \\
\beta \rightarrow \gamma & \equiv & \neg\beta \vee \gamma \\
\beta \leftrightarrow \gamma & \equiv & (\neg\beta \vee \gamma) \wedge (\neg\gamma \vee \beta) \\
(\alpha \wedge \beta) \vee \gamma & \equiv & (\alpha \vee \gamma) \wedge (\beta \vee \gamma) \\
\neg\forall x : \alpha & \equiv & \exists x : \neg\alpha \\
\neg\exists x : \alpha & \equiv & \forall x : \neg\alpha \\
(\exists x : \beta) \vee \gamma & \equiv & \exists x : (\beta \vee \gamma) \\
(\forall x : \beta) \vee \gamma & \equiv & \forall x : (\beta \vee \gamma) \\
\forall x : (\beta \wedge \gamma) & \equiv & (\forall x : \beta) \wedge (\forall x : \gamma) \\
(\exists h : \beta) \wedge \gamma & \equiv & \exists h : (\beta \wedge \gamma) \\
\forall x_1 \cdots \forall x_n \exists y : \beta & \equiv & \exists h \forall x_1 \cdots \forall x_n \forall y : \\
 & & (\beta \vee \neg \mathit{func}(h, x_1, \ldots, x_n, y))
\end{array}
$$