Embedding Proof Problems into Query-answering Problems and Problem Solving by Equivalent Transformation

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: Question-answering Problems, Proof Problems, Equivalent Transformation, Solving Logical Problems.

Abstract: A proof problem is a "yes/no" problem concerning with checking whether one logical formula is a logical consequence of another logical formula, while a query-answering problem (QA problem) is an "all-answers finding" problem concerning with finding all ground instances of a query atomic formula that are logical consequences of a given logical formula. In order to establish a precise relation between these two problem classes, the concept of an embedding mapping is introduced. When one problem class can be embedded into another problem class at low computational cost, the former class can be regarded as a subclass of the latter class and, consequently, problems in the former class can be solved through a method for solving problems in the latter one. Construction of low-cost embedding mappings from proof problems to QA problems is demonstrated. By such embedding, proof problems can be solved using a procedure for solving QA problems. A procedure for solving QA problems based on the equivalent transformation principle is presented. Application of the procedure to the two problem classes is illustrated.

1 INTRODUCTION

Given a first-order formula K, representing background knowledge, and an atomic formula (atom) a, representing a query, a *query-answering problem* (*QA problem*) is to find the set of all ground instances of a that are logical consequences of K. Characteristically, it is an "all-answers finding" problem, i.e., all ground instances of the query atom satisfying the requirement must be found. A *proof problem*, by contrast, is a "yes/no" problem; it is concerned with checking whether or not one given logical formula is a logical consequence of another given logical formula.

Historically, works on logic-based automated reasoning have been centered around proof problems (Chang and Lee, 1973; Gallier, 1986; Fitting, 1996; Newborn, 2000). Methods for solving proof problems were developed, e.g., tableau-based methods (Beth, 1955) and resolution-based methods (Robinson, 1965), and they have been subsequently adapted to address other classes of logical problems, including some specific subclasses of QA problems, e.g., QA problems on definite clauses (Lloyd, 1987). As opposed to such a proof-centered approach, we present in this paper a direct approach towards solving QA problems on the basis of the *equivalent transformation* (*ET*) principle. We show that proof problems can naturally be considered as QA problems of a special form; therefore, a method for solving QA problems also lends itself to solve proof problems in a straightforward way.

In order to clearly understand the relation between proof problems and QA problems, we introduce the notion of an embedding mapping from one problem class to another problem class. Using an embedding mapping, we demonstrate that proof problems can be formulated as a subclass of QA problems. We propose a framework for solving QA problems by ET. A given input QA problem on first-order logic is converted into an equivalent QA problem on an extended clause space, called the $\mathrm{ECLS_F}$ space, through meaning-preserving Skolemization (Akama and Nantajeewarawat, 2011). The obtained QA problem is then successively transformed on the $\mathrm{ECLS_F}$ space by application of ET rules until the answer to the original problem can be readily obtained. With an embedding mapping from proof problems to QA problems, this framework can be used for solving proof problems.

To begin with, Section 2 formalizes QA problems and proof problems. Section 3 defines an embedding mapping and shows how to embed proof problems into QA problems. Section 4 introduces extended

Akama K. and Nantajeewarawat E..
Embedding Proof Problems into Query-answering Problems and Problem Solving by Equivalent Transformation.
DOI: 10.5220/0004546202530260
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2013), pages 253-260
ISBN: 978-989-8565-81-5

clauses, the extended space $\mathrm{ECLS_F}$ and QA problems on this space. Section 5 presents our ET-based procedure for solving QA problems. Section 6 defines unfolding transformation on the $\mathrm{ECLS_F}$ space and provides some other ET rules on this space. Section 7 illustrates application of our framework. Section 8 concludes the paper.

2 QA PROBLEMS AND PROOF PROBLEMS

2.1 Interpretations and Models

In this paper, an atom occurring in a first-order formula can be either a usual atom or a constraint atom. The semantics of first-order formulas based on a logical structure given in (Akama and Nantajeewarawat, 2012) is used. The set of all ground usual atoms, denoted by $\mathcal{G}$, is taken as the interpretation domain. An *interpretation* is a subset of $\mathcal{G}$. A ground usual atom g is true with respect to an interpretation I iff g belongs to I. Unlike ground usual atoms, the truth values of ground constraint atoms are predetermined independently of interpretations. A *model* of a first-order formula E is an interpretation that satisfies E. The set of all models of a first-order formula E is denoted by $\mathit{Models}(E)$. Given first-order formulas E_1 and E_2, E_2 is a *logical consequence* of E_1 iff every model of E_1 is a model of E_2.

2.2 QA Problems

A *query-answering problem* (*QA problem*) is a pair $\langle K, a\rangle$, where K is a first-order formula, representing background knowledge, and a is a usual atom, representing a query. The answer to a QA problem $\langle K, a\rangle$, denoted by $\mathit{answer}_{\mathrm{qa}}(\langle K, a\rangle)$, is defined as the set of all ground instances of a that are logical consequences of K. Using $\mathit{Models}(K)$, the answer to a QA problem $\langle K, a\rangle$ can be equivalently defined as

$$\mathit{answer}_{\mathrm{qa}}(\langle K, a\rangle) = (\bigcap \mathit{Models}(K)) \cap \mathit{rep}(a),$$

where $\mathit{rep}(a)$ denotes the set of all ground instances of a. Accordingly, a QA problem can also be seen as a model-intersection problem. When no confusion is caused, $\mathit{answer}_{\mathrm{qa}}(\langle K, a\rangle)$ is often written as $\mathit{answer}_{\mathrm{qa}}(K, a)$.

2.3 Proof Problems

A *proof problem* is a pair $\langle E_1, E_2\rangle$, where E_1 and E_2 are first-order formulas, and the answer to this problem, denoted by $\mathit{answer}_{\mathrm{pr}}(\langle E_1, E_2\rangle)$, is defined by

$$\mathit{answer}_{\mathrm{pr}}(\langle E_1, E_2\rangle) = \begin{cases} \text{“yes”} & \text{if } E_2 \text{ is a logical} \\ & \text{consequence of } E_1, \\ \text{“no”} & \text{otherwise.} \end{cases}$$

It is well known that a proof problem $\langle E_1, E_2\rangle$ can be converted into the problem of determining whether $E_1 \wedge \neg E_2$ is unsatisfiable (Chang and Lee, 1973), i.e., whether $E_1 \wedge \neg E_2$ has no model. As a result, $\mathit{answer}_{\mathrm{pr}}(\langle E_1, E_2\rangle)$ can be equivalently defined by

$$\mathit{answer}_{\mathrm{pr}}(\langle E_1, E_2\rangle) = \begin{cases} \text{“yes”} & \text{if } \mathit{Models}(E_1 \wedge \neg E_2) \\ & \text{is the empty set,} \\ \text{“no”} & \text{otherwise.} \end{cases}$$

When no confusion is caused, $\mathit{answer}_{\mathrm{pr}}(\langle E_1, E_2\rangle)$ is often written as $\mathit{answer}_{\mathrm{pr}}(E_1, E_2)$.

3 EMBEDDING PROOF PROBLEMS INTO QA PROBLEMS

3.1 Embedding Mappings

The notion of a class of problems and that of an embedding mapping are formalized below.

Definition 1. A *class* $\mathbf{C}$ *of problems* is a triple $\langle \textsc{Prob}, \textsc{Ans}, \mathit{answer}\rangle$, where

1. PROB and ANS are sets,
2. *answer* is a mapping from PROB to ANS.

The sets PROB and ANS are called the *problem space* and the *answer space*, respectively, of $\mathbf{C}$. Their elements are called *problems* and (possible) *answers*, respectively, in $\mathbf{C}$. Given a problem $\mathit{prb} \in \textsc{Prob}$, $\mathit{answer}(\mathit{prb})$ is the answer to *prb* in $\mathbf{C}$. □

Definition 2. Let $\mathbf{C}_1 = \langle \textsc{Prob}_1, \textsc{Ans}_1, \mathit{answer}_1\rangle$ and $\mathbf{C}_2 = \langle \textsc{Prob}_2, \textsc{Ans}_2, \mathit{answer}_2\rangle$ be classes of problems. An *embedding mapping* from $\mathbf{C}_1$ to $\mathbf{C}_2$ is a pair $\langle \pi, \alpha\rangle$, where π is an injective mapping from $\textsc{Prob}_1$ to $\textsc{Prob}_2$ and α is a partial mapping from $\textsc{Ans}_2$ to $\textsc{Ans}_1$ such that for any $\mathit{prb} \in \textsc{Prob}_1$, $\mathit{answer}_1(\mathit{prb}) = \alpha(\mathit{answer}_2(\pi(\mathit{prb})))$. □

Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be classes of problems. Suppose that (i) there exists an embedding mapping $\langle \pi, \alpha\rangle$ from $\mathbf{C}_1$ to $\mathbf{C}_2$, (ii) there exists a procedure P for solving problems in $\mathbf{C}_2$, and (iii) there also exist a procedure P_π for realizing π and a procedure P_α for realizing α. Then a procedure for solving problems in $\mathbf{C}_1$ can be obtained by making the composition of the

procedures P_π, P and P_α. $\mathbf{C}_1$ is regarded as a *subclass* of $\mathbf{C}_2$ iff there exists an embedding mapping $\langle \pi, \alpha \rangle$ from $\mathbf{C}_1$ to $\mathbf{C}_2$ such that π and α can be realized at low computational cost.

3.2 Embedding Proof Problems into QA Problems

Next, we show how to embed proof problems into QA problems. Assume that:

- $\mathbf{C}_{\text{qa}} = \langle \text{PROB}_{\text{qa}}, \text{ANS}_{\text{qa}}, \mathit{answer}_{\text{qa}} \rangle$ is the class of QA problems defined by Section 2.2, i.e., PROB_{qa} is the set of all QA problems, ANS_{qa} is the power set of $\mathcal{G}$, and $\mathit{answer}_{\text{qa}} : \text{PROB}_{\text{qa}} \to \text{ANS}_{\text{qa}}$ is given by Section 2.2.
- $\mathbf{C}_{\text{pr}} = \langle \text{PROB}_{\text{pr}}, \text{ANS}_{\text{pr}}, \mathit{answer}_{\text{pr}} \rangle$ is the class of proof problems defined by Section 2.3, i.e., PROB_{pr} is the set of all proof problems, $\text{ANS}_{\text{pr}} = \{\text{“yes”}, \text{“no”}\}$, and $\mathit{answer}_{\text{pr}} : \text{PROB}_{\text{pr}} \to \text{ANS}_{\text{pr}}$ is given by Section 2.3.

In order to construct an embedding mapping from $\mathbf{C}_{\text{pr}}$ to $\mathbf{C}_{\text{qa}}$, we want to construct from any arbitrary given proof problem $\langle E_1, E_2 \rangle$ a QA problem $\langle K, \mathit{yes} \rangle$ such that $\mathit{answer}_{\text{pr}}(E_1, E_2) = \text{“yes”}$ iff $\mathit{answer}_{\text{qa}}(K, \mathit{yes}) = \{\mathit{yes}\}$, where *yes* is a 0-ary predicate symbol and the atom *yes* occurs in neither E_1 nor E_2. The following approaches can be taken for constructing such a formula K:

- Construct K such that every model of K contains *yes* iff $\mathit{answer}_{\text{pr}}(E_1, E_2) = \text{“yes”}$.
- Construct K such that K has no model iff $\mathit{answer}_{\text{pr}}(E_1, E_2) = \text{“yes”}$.

We refer to the first approach as *positive* construction, and the second one as *negative* construction. They are given below.

3.2.1 Embedding using Positive Construction

Positive construction of an embedding mapping from $\mathbf{C}_{\text{pr}}$ to $\mathbf{C}_{\text{qa}}$ can be obtained by Proposition 1.

Proposition 1. Let E_1 and E_2 be first-order formulas. Assume that:

1. *yes* is a 0-ary predicate symbol and *yes* occurs in neither E_1 nor E_2.
2. prb_1 is the proof problem $\langle E_1, E_2 \rangle$.
3. prb_2 is the QA problem $\langle \mathit{yes} \leftrightarrow (E_1 \to E_2), \mathit{yes} \rangle$.

Then $\mathit{answer}_{\text{pr}}(\mathit{prb}_1) = \text{“yes”}$ iff $\mathit{answer}_{\text{qa}}(\mathit{prb}_2) = \{\mathit{yes}\}$. □

Proposition 1 determines an embedding mapping $\langle \pi_{\text{a}}, \alpha_{\text{a}} \rangle$ from $\mathbf{C}_{\text{pr}}$ to $\mathbf{C}_{\text{qa}}$ as follows:

- For any proof problem $\langle E_1, E_2 \rangle$,
$$\pi_{\text{a}}(\langle E_1, E_2 \rangle) = \langle \mathit{yes} \leftrightarrow (E_1 \to E_2), \mathit{yes} \rangle.$$
- $\alpha_{\text{a}}(\{\mathit{yes}\}) = \text{“yes”}$ and $\alpha_{\text{a}}(\emptyset) = \text{“no”}$.

3.2.2 Embedding using Negative Construction

The next proposition illuminates negative construction of an embedding mapping from $\mathbf{C}_{\text{pr}}$ to $\mathbf{C}_{\text{qa}}$.

Proposition 2. Let E_1 and E_2 be first-order formulas. Assume that:

1. *yes* is a 0-ary predicate symbol and *yes* occurs in neither E_1 nor E_2.
2. prb_1 is the proof problem $\langle E_1, E_2 \rangle$.
3. prb_2 is the QA problem $\langle E_1 \wedge \neg E_2, \mathit{yes} \rangle$.

Then $\mathit{answer}_{\text{pr}}(\mathit{prb}_1) = \text{“yes”}$ iff $\mathit{answer}_{\text{qa}}(\mathit{prb}_2) = \{\mathit{yes}\}$. □

Proposition 2 determines an embedding mapping $\langle \pi_{\text{b}}, \alpha_{\text{b}} \rangle$ from $\mathbf{C}_{\text{pr}}$ to $\mathbf{C}_{\text{qa}}$ as follows:

- For any proof problem $\langle E_1, E_2 \rangle$,
$$\pi_{\text{b}}(\langle E_1, E_2 \rangle) = \langle E_1 \wedge \neg E_2, \mathit{yes} \rangle.$$
- $\alpha_{\text{b}}(\{\mathit{yes}\}) = \text{“yes”}$ and $\alpha_{\text{b}}(\emptyset) = \text{“no”}$.

4 QA PROBLEMS ON AN EXTENDED SPACE

To solve a QA problem $\langle K, a \rangle$ on first-order logic, the first-order formula K is usually converted into a conjunctive normal form. The conversion involves removal of existential quantifications by Skolemization, i.e., by replacement of an existentially quantified variable with a Skolem term determined by a relevant part of a formula prenex. The classical Skolemization, however, does not preserve the logical meaning of a formula—the formula resulting from Skolemization is not necessarily equivalent to the original one (Chang and Lee, 1973). In (Akama and Nantajeewarawat, 2011), a theory for extending the space of first-order formulas was developed and how meaning-preserving Skolemization can be achieved in the obtained extended space was shown. A procedure for converting first-order formulas into extended conjunctive normal forms in an extended clause space, called the ECLS_{F} space, was also presented.

The basic idea of meaning-preserving Skolemization is to use existentially quantified function variables instead of usual Skolem functions. Function variables, extended clauses, extended conjunctive normal forms and QA problems on ECLS_{F} are introduced below.

4.1 Function Constants, Function Variables and *func*-Atoms

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

In order to clearly separate function constants and function variables from usual function symbols and usual terms, a new built-in predicate symbol *func* is introduced. Given any n-ary function constant or n-ary function variable $\bar{f}$, an expression $func(\bar{f},t_1,\ldots,t_n,t_{n+1})$, where the t_i are usual terms, is considered as an atom of a new type, called a *func-atom*. When $\bar{f}$ is a function constant and the t_i are all ground, the truth value of this atom is evaluated as follows: it is true iff $\bar{f}(t_1,\ldots,t_n)=t_{n+1}$.

4.2 Extended Clauses

An *extended clause* C is a closed formula of the form

$$\forall v_1,\ldots,\forall v_m : (a_1\vee\cdots\vee a_n\vee\neg b_1\vee\cdots\vee\neg b_p \vee\neg\mathbf{f}_1\vee\cdots\vee\neg\mathbf{f}_q),$$

where $v_1,\ldots,v_m$ are usual variables, each of $a_1,\ldots,a_n,b_1,\ldots,b_p$ is a usual atom or a constraint atom, and $\mathbf{f}_1,\ldots,\mathbf{f}_q$ are *func*-atoms. It is often written simply as $(a_1,\ldots,a_n\leftarrow b_1,\ldots,b_p,\mathbf{f}_1,\ldots,\mathbf{f}_q)$. The sets $\{a_1,\ldots,a_n\}$ and $\{b_1,\ldots,b_p,\mathbf{f}_1,\ldots,\mathbf{f}_q\}$ are called the *left-hand side* and the *right-hand side*, respectively, of the extended clause C, denoted by $lhs(C)$ and $rhs(C)$, respectively. When $n=0$, C is called a *negative extended clause*. When $n=1$, C is called an *extended 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 $n>1$, C is called a *multi-head extended clause*. All usual variables in an extended clause are universally quantified and their scope is restricted to the clause itself. When no confusion is caused, an extended clause, a negative extended clause, an extended definite clause and a multi-head extended clause will also be called a *clause*, a *negative clause*, a *definite clause* and a *multi-head clause*, respectively.

An extended normal form called *existentially quantified conjunctive normal form* (ECNF) is a formula of the form $\exists v_{h1},\ldots,\exists v_{hm} : (C_1\wedge\cdots\wedge C_n)$, where $v_{h1},\ldots,v_{hm}$ are function variables and $C_1,\ldots,C_n$ are extended clauses. It is often identified with the set $\{C_1,\ldots,C_n\}$, with implicit existential quantifications of function variables and implicit clause conjunction. Function variables in such a clause set are all existentially quantified and their scope covers entirely all clauses in the set.

4.3 QA Problems on $\mathrm{ECLS_F}$

The set of all ECNFs is referred to as the *extended clause space* ($\mathrm{ECLS_F}$). By the above identification of an ECNF with a clause set, we often regard an element of $\mathrm{ECLS_F}$ as a set of (extended) clauses. With occurrences of function variables, clauses contained in a clause set in the $\mathrm{ECLS_F}$ space are connected through shared function variables. By instantiating all function variables in such a clause set into function constants, clauses in the obtained set are totally separated.

A QA problem $\langle Cs,a\rangle$ such that Cs is a clause set in $\mathrm{ECLS_F}$ and a is a usual atom is called a *QA problem on* $\mathrm{ECLS_F}$. Given a QA problem $\langle K,a\rangle$ on first-order logic, the first-order formula K can be converted equivalently by meaning-preserving Skolemization, using the conversion procedure given in (Akama and Nantajeewarawat, 2011), into a clause set Cs in the $\mathrm{ECLS_F}$ space. The obtained clause set Cs may be further transformed equivalently in this space for problem simplification, by using unfolding and other transformation rules.

5 SOLVING QA PROBLEMS

Using the notation introduced in Sections 5.1 and 5.2, our ET-based procedure is presented in Section 5.3.

5.1 Inclusion of Query Information

The following notation is used. A set A of usual atoms is said to be *closed* iff for any $a\in A$ and any substitution θ for usual variables, $a\theta$ belongs to A. Assume that (i) $\mathcal{A}$ is the set of all usual atoms, (ii) $\mathcal{A}_1$ and $\mathcal{A}_2$ are disjoint closed subsets of $\mathcal{A}$, and (iii) ϕ is a bijection from $\mathcal{A}_1$ to $\mathcal{A}_2$ such that for any $a\in\mathcal{A}_1$ and any substitution θ for usual variables, $\phi(a\theta)=\phi(a)\theta$. For any $i,j\in\{1,2\}$, an extended clause C is said to be from $\mathcal{A}_i$ to $\mathcal{A}_j$ iff all usual atoms in $rhs(C)$ belong to $\mathcal{A}_i$ and all those in $lhs(C)$ belong to $\mathcal{A}_j$.

Let $\langle K, a\rangle$ be a QA problem such that K is a first-order formula in which all usual atoms belong to $\mathcal{A}_1$ and $a \in \mathcal{A}_1$. As will be detailed in Section 5.3, to solve this problem using ET, K is transformed by meaning-preserving transformation into a set Cs of extended clauses from $\mathcal{A}_1$ to $\mathcal{A}_1$ and a singleton set Q consisting only of the clause $(\phi(a) \leftarrow a)$ from $\mathcal{A}_1$ to $\mathcal{A}_2$ is constructed from the query atom a. The resulting QA problem $\langle Cs \cup Q, \phi(a)\rangle$ is then successively transformed using ET rules.

5.2 Triples for Transformation

In order to make a clear separation between a set of extended clauses from $\mathcal{A}_1$ to $\mathcal{A}_1$ and a set of those from $\mathcal{A}_1$ to $\mathcal{A}_2$ in a transformation process of QA problems, the following notation is introduced: Given a set Cs of extended clauses from $\mathcal{A}_1$ to $\mathcal{A}_1$, a set Q of extended clauses from $\mathcal{A}_1$ to $\mathcal{A}_2$ and an atom b in $\mathcal{A}_2$, let the triple $\langle Cs, Q, b\rangle$ denote the QA problem $\langle Cs \cup Q, b\rangle$. A QA problem $\langle Cs, Q, b\rangle$ can be transformed by changing Cs, by changing Q, or by changing both Cs and Q.

Definition 3. A transformation of a QA problem $\langle Cs, Q, b\rangle$ into a QA problem $\langle Cs', Q', b\rangle$ is *equivalent transformation* (*ET*) iff $answer_{\mathrm{qa}}(Cs \cup Q, b) = answer_{\mathrm{qa}}(Cs' \cup Q', b)$. □

5.3 A Procedure for Solving QA Problems by ET

Let $\mathcal{A}_1$ be a closed set of usual atoms. Assume that a QA problem $\langle K, a\rangle$ is given, where K is a first-order formula in which all usual atoms belong to $\mathcal{A}_1$ and $a \in \mathcal{A}_1$. To solve the QA problem $\langle K, a\rangle$ using ET, perform the following steps:

1. Transform K by meaning-preserving Skolemization into a clause set Cs in the $\mathrm{ECLS_F}$ space.
2. Determine (i) a closed set $\mathcal{A}_2$ of usual atoms such that $\mathcal{A}_1$ and $\mathcal{A}_2$ are disjoint and (ii) a bijection ϕ from $\mathcal{A}_1$ to $\mathcal{A}_2$ such that for any $a \in \mathcal{A}_1$ and any substitution θ for usual variables, $\phi(a\theta) = \phi(a)\theta$.
3. Successively transform the QA problem $\langle Cs, \{(\phi(a) \leftarrow a)\}, \phi(a)\rangle$ in the $\mathrm{ECLS_F}$ space using unfolding and other ET rules (see Section 6).
4. Assume that the transformation yields a QA problem $\langle Cs', Q, \phi(a)\rangle$. Then:
   (a) If $Models(Cs') = \emptyset$, then output $rep(a)$ as the answer.
   (b) If $Models(Cs') \neq \emptyset$ and Q is a set of unit clauses such that the head of each clause in Q is an instance of $\phi(a)$, then output as the answer the set
   $$\phi^{-1}(\bigcup_{C \in Q} rep(head(C))).$$
   (c) Otherwise stop with failure.

It is shown in (Akama and Nantajeewarawat, 2013) that the obtained answer is always correct.

The set $\mathcal{A}_2$ and the bijection ϕ satisfying the requirement of Step 2 can be determined as follows: First, introduce a new predicate symbol for each predicate symbol occurring in $\mathcal{A}_1$. Next, let $\mathcal{A}_2$ be the atom set obtained from $\mathcal{A}_1$ by replacing the predicate of each atom in $\mathcal{A}_1$ with the new predicate introduced for it. Finally, for each atom $a \in \mathcal{A}_1$, let $\phi(a)$ be the atom obtained from a by such predicate replacement.

6 ET RULES ON $\mathrm{ECLS_F}$

Next, ET rules for unfolding and definite-clause removal are presented, along with some other ET rules.

6.1 Unfolding Operation on $\mathrm{ECLS_F}$

Assume that (i) Cs is a set of extended clauses, (ii) D is a set of extended definite clauses, and (iii) 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:

- If b and $head(C'\rho)$ are not unifiable, then $resolvent(C, C', b) = \emptyset$.
- If they are unifiable, then $resolvent(C, C', b) = \{C''\}$, where C'' is the clause obtained from C and $C'\rho$ as follows, assuming that θ is the most general unifier of b and $head(C'\rho)$:
  - $lhs(C'') = lhs(C\theta)$
  - $rhs(C'') = (rhs(C\theta) - \{b\theta\}) \cup body(C'\rho\theta)$

The resulting clause set is denoted by $\textsc{Unfold}(Cs, D, occ)$.

6.2 ET by Unfolding and Definite-clause Removal

Let $Atoms(p)$ denote the set of all atoms having a predicate p. ET rules on $\mathrm{ECLS_F}$ for unfolding and for definite-clause removal are described below.

6.2.1 ET by Unfolding

Let $\langle Cs, a\rangle$ be a QA problem on $\mathrm{ECLS_F}$. Assume that:

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

Then $\langle Cs, a\rangle$ can be equivalently transformed into the QA problem $\langle \textsc{Unfold}(Cs, D, occ), a\rangle$.

6.2.2 ET by Definite-clause Removal

Let $\langle Cs, a\rangle$ be a QA problem on $\mathrm{ECLS_F}$. Assume that:

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

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

6.3 Some other ET Rules on $\mathrm{ECLS_F}$

Next, ET rules for merging *func*-atoms having the same call pattern, for removing isolated *func*-atoms, and for removing subsumed clauses are presented. They are used in examples in Section 7.

6.3.1 Merging *func*-Atoms with the Same Invocation Pattern

Let $\langle Cs, a\rangle$ be a QA problem on $\mathrm{ECLS_F}$. Suppose that $C \in Cs$ and $rhs(C)$ contains *func*-atoms $\mathbf{f}_1$ and $\mathbf{f}_2$ that differ only in their last arguments. Then:

1. If the last arguments of $\mathbf{f}_1$ and $\mathbf{f}_2$ are unifiable, with their most general unifier being θ, and C' is an extended clause such that
   - $lhs(C') = lhs(C\theta)$, and
   - $rhs(C') = (rhs(C) - \{\mathbf{f}_2\})\theta$,

   then $\langle Cs, a\rangle$ can be equivalently transformed into the QA problem $\langle (Cs - \{C\}) \cup \{C'\}, a\rangle$.
2. If their last arguments are not unifiable, then $\langle Cs, a\rangle$ can be equivalently transformed into the QA problem $\langle Cs - \{C\}, a\rangle$.

6.3.2 Elimination of Isolated *func*-Atoms

A *func*-atom $func(h, t_1, \ldots, t_n, v)$, where v is a usual variable, is said to be *isolated* in an extended clause C iff there is only one occurrence of v in C.

Now let $\langle Cs, a\rangle$ be a QA problem on $\mathrm{ECLS_F}$. Suppose that:

1. $C \in Cs$ such that C contains a *func*-atom that is isolated in C.
2. C' is the extended clause obtained from C by removing all *func*-atoms that are isolated in C.

Then $\langle Cs, a\rangle$ can be equivalently transformed into the QA problem $\langle (Cs - \{C\}) \cup \{C'\}, a\rangle$.

6.3.3 Elimination of Subsumed Clauses

An extended clause C_1 is said to *subsume* an extended clause C_2 iff there exists a substitution θ for usual variables such that $lhs(C_1)\theta \subseteq lhs(C_2)$ and $rhs(C_1)\theta \subseteq rhs(C_2)$.

A subsumed clause can be removed as follows: Let $\langle Cs, a\rangle$ be a QA problem on $\mathrm{ECLS_F}$. If Cs contains extended clauses C_1 and C_2 such that C_1 subsumes C_2, then $\langle Cs, a\rangle$ can be equivalently transformed into the QA problem $\langle Cs - \{C_2\}, a\rangle$.

7 EXAMPLES

Example 1 demonstrates how the procedure in Section 5.3 solves a QA problem using the ET rules in Section 6. Example 2 shows how to apply the procedure to solve a proof problem based on the embedding mapping in Section 3.2.2.

Example 1. Consider the "Tax-cut" problem discussed in (Motik et al., 2005). This problem is to find all persons who can have discounted tax, with the knowledge that (i) any person who has two children or more can get discounted tax, (ii) men and women are not the same, (iii) a person's mother is always a woman, (iv) Peter has a child named Paul, (v) Paul is a man, and (vi) Peter has a child, who is someone's mother. This background knowledge is represented

in first-order logic as the formulas F_1–F_6 below, assuming that *hc*, *ns*, *tc*, *mn*, *wm* and *mo* stand, respectively, for *hasChild*, *notSame*, *TaxCut*, *Man*, *Woman* and *motherOf*:

F_1: $\forall x\colon ((\exists y_1 \exists y_2\colon (hc(x,y_1) \wedge hc(x,y_2) \wedge ns(y_1,y_2))) \rightarrow tc(x))$
F_2: $\forall x \forall y\colon ((mn(x) \wedge wm(y)) \rightarrow ns(x,y))$
F_3: $\forall x\colon ((\exists y\colon mo(x,y)) \rightarrow wm(x))$
F_4: $hc(Peter, Paul)$
F_5: $mn(Paul)$
F_6: $\exists x\colon (hc(Peter,x) \wedge (\exists y\colon mo(x,y)))$

Accordingly, the "Tax-cut" problem is formulated as the QA problem $\langle K, tc(x)\rangle$, where K is the conjunction of F_1–F_6. Using the meaning-preserving Skolemization procedure given in (Akama and Nantajeewarawat, 2011), the first-order formula K is transformed into a clause set Cs consisting of the following extended clauses:

C_1: $tc(x) \leftarrow hc(x,y_1), hc(x,y_2), ns(y_1,y_2)$
C_2: $ns(x,y) \leftarrow mn(x), wm(y)$
C_3: $wm(x) \leftarrow mo(x,y)$
C_4: $hc(Peter, Paul) \leftarrow$
C_5: $mn(Paul) \leftarrow$
C_6: $hc(Peter,x) \leftarrow func(h_1,x)$
C_7: $mo(x,y) \leftarrow func(h_1,x), func(h_2,y)$

The clauses C_6 and C_7 together represent the first-order formula F_6, where h_1 and h_2 are 0-ary function variables.

Assume that all usual atoms occurring in Cs belong to $\mathcal{A}_1$, *ans* is a newly introduced unary predicate symbol, all *ans*-atoms belong to $\mathcal{A}_2$, and for any term t, $\phi(tc(t)) = ans(t)$. Let

$$C_0 = (ans(x) \leftarrow tc(x)).$$

To solve the QA problem $\langle K, tc(x)\rangle$, the QA problem $\langle Cs, \{C_0\}, ans(x)\rangle$ is successively transformed by applying the ET rules in Section 6 as follows:

1. By unfolding C_0 at $tc(x)$ using $\{C_1\}$, C_0 is replaced with:
   C_8: $ans(x) \leftarrow hc(x,y_1), hc(x,y_2), ns(y_1,y_2)$
2. By unfolding C_8 at the last body atom using $\{C_2\}$, C_8 is replaced with:
   C_9: $ans(x) \leftarrow hc(x,y_1), hc(x,y_2), mn(y_1), wm(y_2)$
3. By unfolding C_9 at the third body atom using $\{C_5\}$, C_9 is replaced with:
   C_{10}: $ans(x) \leftarrow hc(x,Paul), hc(x,y_2), wm(y_2)$
4. By unfolding C_{10} at the last body atom using $\{C_3\}$, C_{10} is replaced with:
   C_{11}: $ans(x) \leftarrow hc(x,Paul), hc(x,y_2), mo(y_2,z)$
5. By unfolding C_{11} at the last body atom using $\{C_7\}$, C_{11} is replaced with:
   C_{12}: $ans(x) \leftarrow hc(x,Paul), hc(x,y_2), func(h_1,y_2), func(h_2,z)$
6. By removing an isolated *func*-atom, C_{12} is replaced with:
   C_{13}: $ans(x) \leftarrow hc(x,Paul), hc(x,y_2), func(h_1,y_2)$
7. By unfolding C_{13} at the first body atom using $\{C_4, C_6\}$, C_{13} is replaced with:
   C_{14}: $ans(Peter) \leftarrow hc(Peter,y_2), func(h_1,y_2)$
   C_{15}: $ans(Peter) \leftarrow func(h_1,Paul), hc(Peter,y_2), func(h_1,y_2)$
8. By merging *func*-atoms with the same invocation pattern, C_{15} is replaced with:
   C_{16}: $ans(Peter) \leftarrow func(h_1,Paul), hc(Peter,Paul)$
9. Since C_{16} is subsumed by C_{14}, C_{16} is removed.
10. By unfolding C_{14} at the first body atom using $\{C_4, C_6\}$, C_{14} is replaced with:
    C_{17}: $ans(Peter) \leftarrow func(h_1,Paul)$
    C_{18}: $ans(Peter) \leftarrow func(h_1,y_2), func(h_1,y_2)$
11. By definite-clause removal, C_1–C_7 are removed.
12. By merging *func*-atoms with the same invocation pattern, C_{18} is replaced with:
    C_{19}: $ans(Peter) \leftarrow func(h_1,y_2)$
13. By removing an isolated *func*-atom, C_{19} is replaced with:
    C_{20}: $ans(Peter) \leftarrow$
14. Since C_{17} is subsumed by C_{20}, C_{17} is removed.

The resulting QA problem is $\langle \emptyset, \{C_{20}\}, ans(x)\rangle$. Since $Models(\emptyset) \neq \emptyset$ and C_{20} is a unit clause whose head is an instance of $\phi(tc(x))$, the answer to the "Tax-cut" problem $\langle K, tc(x)\rangle$ is determined by

$$\begin{aligned}\phi^{-1}(\bigcap\{rep(head(C_{20}))\}) &= \phi^{-1}(\{ans(Peter)\}) \\ &= \{tc(Peter)\},\end{aligned}$$

i.e., Peter is the only one who gets discounted tax. □

Example 2. Refer to the description of the "Tax-cut" problem, the first-order formulas F_1–F_6, the clauses C_0–C_{20} and the clause set $Cs = \{C_1, \ldots, C_7\}$ in Example 1. From the background knowledge of the "Tax-cut" problem, suppose that we want to prove the existence of someone who gets discounted tax. This problem is formulated as the proof problem $\langle E_1, E_2\rangle$, where E_1 is the conjunction of F_1–F_6 and E_2 is the first-order formula $\exists x : tc(x)$.

Using Proposition 2, this proof problem is converted into the QA problem $\langle E_1 \wedge \neg E_2, yes\rangle$. Using the procedure in Section 5.3, this QA problem is solved as follows:

- Convert $E_1 \wedge \neg E_2$ by meaning-preserving Skolemization, resulting in the clause set $Cs \cup \{C'_0\}$, where C'_0 is the negative clause $(\leftarrow tc(x))$.
- Transform the QA problem

$$\langle Cs \cup \{C'_0\}, \{(\phi(yes) \leftarrow yes)\}, \phi(yes)\rangle$$

using ET rules. By following the transformation Steps 1–14 in Example 1 except that the initial target clause is C'_0 instead of C_0, the clauses C'_8–C'_{20} are successively produced, where for each $i \in \{8,\ldots,20\}$,

– $lhs(C'_i) = \emptyset$, and
– $rhs(C'_i) = rhs(C_i)$,

and C_1–C_7 are removed. As a result, $Cs \cup \{C'_0\}$ is transformed into $\{C'_{20}\}$, where $C'_{20} = (\leftarrow)$, and the QA problem

$$\langle \{C'_{20}\}, \{(\phi(yes) \leftarrow yes)\}, \phi(yes)\rangle$$

is obtained.

- Since C'_{20} is the empty clause, the clause set $\{C'_{20}\}$ has no model, i.e., $Models(\{C'_{20}\}) = \emptyset$. So the procedure outputs $rep(yes) = \{yes\}$ as the answer to the QA problem $\langle E_1 \wedge \neg E_2, yes\rangle$.

It follows from Proposition 2 that the answer to the proof problem $\langle E_1, E_2\rangle$ is "yes", i.e., there exists someone who gets discounted tax. □

8 CONCLUSIONS

Previous approaches to solving QA problems are proof-centered. They were developed for specific subclasses of QA problems; for example, answering queries in logic programming and deductive databases can be regarded as solving QA problems on definite clauses and those on a restricted form of definite clauses, respectively. There has been no general solution method for QA problems on full first-order formulas.

QA problems on full first-order logic are considered in this paper. We introduced the concept of embedding and proposed how to embed proof problems into QA problems. This embedding leads to a unified approach to dealing with proof problems and QA problems, allowing one to use a method for solving QA problems to solve proof problems. It enables a QA-problem-centered approach to solving logical problems.

Equivalent transformation (ET) is one of the most fundamental principles of computation, and it provides a simple and general basis for verification of computation correctness. We proposed a framework for solving QA problems by ET. All computation steps in this framework are ET steps, including transformation of a first-order formula into an equivalent formula in the extended clause space ECLS_F and transformation of extended clauses on ECLS_F. To the best of our knowledge, this is the only framework for dealing with the full class of QA problems on first-order formulas.

Since many kinds of ET rules can be employed, the proposed ET-based framework opens up a wide range of possibilities for computation paths to be taken. As a result, the framework enables development of a large variety of methods for solving logical problems. The range of possible computation methods can also be further extended by using computation spaces other than ECLS_F. Proof by resolution can be seen as one specific example of these possible methods. As demonstrated in (Akama and Nantajeewarawat, 2012), it can be realized by using two kinds of ET rules, i.e., resolution and factoring ET rules, on a computation space that differs slightly from ECLS_F.

REFERENCES

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

Akama, K. and Nantajeewarawat, E. (2012). Proving Theorems Based on Equivalent Transformation Using Resolution and Factoring. In *Proceedings of the Second World Congress on Information and Communication Technologies*, WICT 2012, pages 7–12.

Akama, K. and Nantajeewarawat, E. (2013). Embedding Proof Problems into Query-Answering Problems and Problem Solving by Equivalent Transformation. Technical report, Hokkaido University, Sapporo, Japan.

Beth, E. W. (1955). *Semantic Entailment and Formal Derivability*. Amsterdam : Noord-Hollandsche Uitg. Mij.

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

Fitting, M. (1996). *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, second edition.

Gallier, J. H. (1986). *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Wiley.

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

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

Newborn, M. (2000). *Automated Theorem Proving: Theory and Practice*. Springer-Verlag.

Robinson, J. A. (1965). A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12:23–41.