International Journal of Innovative Computing, Information and Control
Volume **18**, Number **5**, October **2022**

ICIC International ©2022 ISSN 1349-4198
pp. **1547–1558**

SOLVING QUERY-ANSWERING PROBLEMS BASED ON EQUIVALENT TRANSFORMATION

Kiyoshi Akama[1] and Ekawit Nantajeewarawat[2,*]

[1]Information Initiative Center
Hokkaido University
Kita 11, Nishi 5, Kita-ku, Sapporo, Hokkaido 060-0811, Japan
akama@iic.hokudai.ac.jp

[2]School of Information, Computer and Communication Technology
Sirindhorn International Institute of Technology
Thammasat University
99 Moo 18, Km. 41 on Paholyothin Highway, Khlong Luang, Pathum Thani 12120, Thailand
*Corresponding author: ekawit@siit.tu.ac.th

Received February 2022; revised June 2022

Abstract. *A query-answering (QA) problem is an "all-answers finding" problem and has been solved by SLD resolution. Due to the completeness of SLD resolution (Linear resolution with Selection function for Definite clauses), it is believed that SLD resolution is successfully applied to all QA problems. However, we prove that a QA problem, called a pal-pal QA problem, cannot be solved by SLD resolution. We show that the pal-pal QA problem can be solved by using equivalent transformation (ET) rules. Such a transformational method is called the ET-based method. We conclude that SLD resolution is not general enough to be applied for all QA problems on definite clauses, the difficulty of which can be overcome by adopting new ET rules in the ET-based method. We also show that the ET-based method overcomes the difficulty of correctness by the concept of finite solvability and by using many ET rules.*

Keywords: Logical problem solving framework, Model-intersection problem, Equivalent transformation, Proof problem, Query-answering problem

1. **Introduction.** Proof problems historically constitute the most important problem class in logical problem solving [1, 2, 3, 4]. Resolution provides us with a refutation proof procedure for logical formulas. SL resolution was developed as a restricted form of linear resolution [5]. SLD resolution stands for SL resolution for definite clauses. Based on the resolution proof method, automated theorem proving has been extensively investigated [6, 7, 8, 9, 10].

The ET-based method is a solution method of repeated application of equivalent transformation rules (ET rules). We can apply the ET-based method to proving theorems. From the viewpoint of solving proof problems, we compared the ET-based method with the resolution-based method, and showed that the ET-based method has the superiority over the resolution-based method, i.e., it has more formalizations, more transformation rules, more computation paths, and more solvability than the resolution-based method [11].

Built-in constraint atoms play a crucial role in knowledge representation and are essential for practical applications. We consider first-order formulas that possibly include built-in constraint atoms. The set of all such formulas is denoted by $\mathrm{FOL_c}$. A *query-answering problem* (*QA problem*) on $\mathrm{FOL_c}$ is a pair $\langle K, a\rangle$, where K is a formula in $\mathrm{FOL_c}$

DOI: 10.24507/ijicic.18.05.1547

and a is a user-defined query atom [12]. The answer to a QA problem $\langle K, a\rangle$ is defined as the set of all ground instances of a that are logical consequences of K. Characteristically, a QA problem is an "all-answers finding" problem, i.e., all ground instances of a given query atom satisfying the requirement above are to be found.

Being inspired by the resolution proof method, Prolog and a theory of SLD-resolution were invented for solving QA problems on definite clauses [13, 14, 15, 16]. To solve a QA problem $\langle K, a\rangle$, we transform it into $\langle D, G\rangle$, where D is a set of definite clauses and G is a goal. SLD resolution finds a goal sequence starting from G. This sequence is obtained by repeated resolution of a goal and a definite clause in D. The correctness theory of SLD resolution, consisting of soundness and completeness theorems, states that the set of correct answers and the set of computed answers are the same [17, 18].

The objective of this paper is to show that the ET-based method has the superiority over the resolution-based method also from the viewpoint of solving QA problems. From the correctness theory of SLD resolution, it is widely recognized that SLD resolution can be successfully applied for solving all QA problems. However, we will argue that SLD resolution has not enough power to be applied to all QA problems. We also show that the ET-based method overcomes the difficulty by the concept of finite solvability and by using many ET rules.

The rest of the paper is organized as follows. Section 2 explains the ET-based approach to logic and logical computation. It also introduces a QA problem, called a pal-pal QA problem, and its formalization (called the existence-finding formalization). Section 3 shows that the pal-pal QA problem cannot be solved by unfolding solely, which is later used to conclude that the pal-pal QA problem cannot be solved by SLD resolution. Section 4 proves that the pal-pal QA problem can be solved by using many specialized ET rules. Section 5 compares SLD resolution and the ET-based method for solving QA problems. Section 6 provides conclusions.

The notation that follows holds thereafter. Given a set A, $pow(A)$ denotes the power set of A. Given two sets A and B, $Map(A, B)$ denotes the set of all mappings from A to B, $partialMap(A, B)$ denotes the set of all partial mappings from A to B, and for any partial mapping f from A to B, $dom(f)$ denotes the domain of f, i.e., $dom(f) = \{a | (a \in A)\ \&\ (f(a) \text{ is defined})\}$.

2. ET-Based Method.

We explain the ET-based method of logical problem solving [19], together with evaluation measure of a solver by using a finitely solvable area. We also introduce a pal-pal QA problem and formalize it as a model-intersection problem, which will be used, in later sections, for establishment of superiority of the ET-based method over SLD resolution.

2.1. A solution method based on equivalent transformation.

To solve QA problems, we take a transformational approach based on equivalent transformation. We formalize a QA problem by a pair of a mapping φ and a formula $K \wedge Cs$, which is called a model-intersection (MI) problem, and solve the problem by calculating

$$\varphi\left(\bigcap Models(K \wedge Cs)\right),$$

where φ is called an extraction mapping and $Models(X)$ is the set of all models of X.

To calculate $\varphi\left(\bigcap Models(K \wedge Cs)\right)$, we find a sequence starting from Cs $(= Cs_1)$:

$$\langle K, Cs_1\rangle \to \langle K, Cs_2\rangle \to \cdots \to \langle K, Cs_n\rangle.$$

This sequence is obtained by equivalent transformation (ET) rules with respect to φ and K. An answer is obtained from $K \wedge Cs_n$ by application of a partial mapping, called an answer mapping. This QA solution method is called an ET-based method.

2.2. **Pal-pal QA problem.** We take a QA problem to illustrate formalization as a model-intersection problem. This problem will be used, in later sections, for comparison of SLD resolution with the ET-based method.

Assume as background knowledge a set consisting of the following three if-and-only-if formulas, where *pal*, *rev*, and *app* stand for "palindrome", "reverse", and "append", respectively.

(1) $\forall x$: $pal(x) \leftrightarrow rev(x, x)$.
(2) $\forall x$: $rev(x, y) \leftrightarrow$
$(eq(x, [\,]) \wedge eq(y, [\,])\ \vee$
$\exists a, w, u : (eq(x, [a|w]) \wedge rev(w, u) \wedge app(u, [a], y)))$.
(3) $\forall x$: $app(x, y, z) \leftrightarrow$
$(eq(x, [\,]) \wedge eq(y, z)\ \vee$
$\exists a, w, u : (eq(x, [a|w]) \wedge eq(z, [a|u]) \wedge app(w, y, u)))$.

Consider the problem "find all ground atoms $exists(s, t)$ such that two lists $[1, s|t]$ and $[2, s|t]$ are palindromes", which is called the pal-pal QA problem.

To represent the pal-pal QA problem, we introduce a formula E_1:

$$\forall A(\forall X((pal([1, A|X]) \wedge pal([2, A|X])) \rightarrow exists(A, X))).$$

Let $\mathcal{A}_{\mathrm{u}}$ be the set of all ground user-defined atoms. Let $\mathcal{G}_{\mathrm{u}}$ be the set of all ground user-defined atoms. We consider $\bigcap Models(F_0 \wedge E_1)$, where F_0 is the conjunction of the three if-and-only-if formulas in Section 2.2. This set results in the union of two sets:

$$\bigcap Models(F_0 \wedge E_1) = \bigcap Models(F_0) \cup G_{ans},$$

where

$$G_{ans} = \{exists(s, t) | [1, s|t] \text{ and } [2, s|t] \text{ are palindromes}\}.$$

Hence the answer to the pal-pal QA problem is formalized as

$$answer_{pal\text{-}pal} = \varphi_1 \left(\bigcap Models(F_0 \wedge E_1) \right),$$

where φ_1 is a mapping defined by

Definition 2.1. *φ_1 is a mapping from $pow(\mathcal{G}_{\mathrm{u}})$ to $\mathcal{G}_{\mathrm{u}}$ such that for each $G \subseteq \mathcal{G}_{\mathrm{u}}$, $\varphi_1(G) = \{exists(s, t) | exists(s, t) \in G\}$.*

The formalization with the extraction mapping φ_1 is called an existence-finding formalization.

2.3. **The correctness of the ET-based approach.** Let a logical problem formalized by $\varphi\,(\bigcap Models(F \wedge E))$ be given. We solve it by successive transformation using rules.

A rewriting rule is a partial mapping on the set of first-order formulas.

Definition 2.2. *Let φ be an extraction mapping and K a first-order formula. A rewriting rule r is an equivalent transformation rule (ET rule) with respect to φ and K iff for any first-order formulas Cs and Cs', if $r(Cs) = Cs'$, then $\varphi\,(\bigcap Models(K \wedge Cs)) = \varphi\,(\bigcap Models\,(K \wedge Cs'))$.*

When successive transformation reaches an end point, a partial mapping produces an output as the answer of the given problem.

Definition 2.3. *A partial mapping ans is an answer mapping for φ iff*

$$ans(K \wedge Cs, \varphi) = \varphi\left(\bigcap Models(K \wedge Cs)\right)$$

if it is defined.

The correctness of the ET-based solution was proved by the following theorem [11].

Theorem 2.1. *Let φ be an extraction mapping and K a first-order formula. Let F and E be first-order formulas. If ans is an answer mapping for φ, R is a set of ET rules with respect to φ and K, and there is a sequence of pairs of first-order formulas*

$$\langle K, Cs_1\rangle \to \langle K, Cs_2\rangle \to \cdots \to \langle K, Cs_n\rangle$$

such that $\varphi\left(\bigcap Models(F \wedge E)\right) = \varphi\left(\bigcap Models(K \wedge Cs_1)\right)$, $r_i \in R$ and $r_i(Cs_i) = Cs_{i+1}$ for any $i \in \{1, 2, \ldots, n-1\}$, and $\langle K \wedge Cs_n, \varphi\rangle \in dom(ans)$, then

$$\varphi\left(\bigcap Models(F \wedge E)\right) = ans(K \wedge Cs_n, \varphi).$$

This theorem establishes a sufficient condition for the correctness of all transformational logical problem solving methods. The conventional SLD resolution methods for proof/QA problems can be considered as specific classes of transformational logical problem solving methods. The conventional SLD resolution methods for proof/QA problems often produce incorrect results by the violation of the first condition $\varphi\left(\bigcap Models(F \wedge E)\right) = \varphi\left(\bigcap Models(K \wedge Cs_1)\right)$ using the conventional Skolemization. They often fail to produce answers by failing to satisfy the last condition $\langle K \wedge Cs_n, \varphi\rangle \in dom(ans)$.

2.4. **Answer mapping for the pal-pal QA problem.** We define a partial mapping ans_1 corresponding to the extraction mapping φ_1 as follows.

Definition 2.4. *Let ans_1 be a partial mapping to assign a set of ground atoms to each problem that is defined by: if Cs is a set of unit clauses, then*

$$ans_1(K \wedge Cs, \varphi_1) = \{exists(s,t) | (exists(s,t) = a\theta \in \mathcal{G}_u)\ \&\ ((a \leftarrow) \in Cs)\ \&\ (\theta \in \mathcal{S})\},$$

where $\mathcal{S}$ is the set of all substitutions, and it is undefined otherwise.

The partial mapping ans_1 is an answer mapping for φ_1, i.e., if Cs is a set of unit clauses,

$$ans_1\left(K \wedge Cs, \varphi_1\right) = \varphi_1\left(\bigcap Models(K \wedge Cs)\right),$$

a proof of which is as follows:

$$ans_1(K \wedge Cs, \varphi_1) = \bigcap Models(Cs) = \varphi_1\left(\bigcap Models(K \wedge Cs)\right).$$

2.5. **Evaluation by a finitely solvable area.** A given problem is solved when a sequence of problems starting from the given problem reaches the domain of a given answer mapping. A control selects an ET rule that is applied to a problem at each step of computation to determine a sequence of problems. For any control *ctrl*, we define $FS(R, ctrl, A)$ as the set of all problems that are successfully solved by using ET rules in R, the control *ctrl*, and answer mappings in A, where FS stands for "finite solvability". A solver is evaluated in the ET-based method by the set of all problems that can be solved in finite steps, which is called a finitely solvable area (FSA).

Consider the set of all problems that can be solved successfully by some control, using a set R of ET rules and a set A of answer mappings, which is denoted by $FSA(R, A)$ and is called a finitely solvable area with respect to R and A, i.e.,

$$FSA(R, A) = \bigcup\{FS(R, ctrl, A) | ctrl \in CTRL\},$$

where $CTRL$ is the set of all controls. Given a problem $prob$, we try to solve $prob$, i.e., we try to find R and A such that

$$prob \in FSA(R, A).$$

In general, R may contain more than one rule and A may contain more than one answer mapping. When we have more answer mappings and more ET rules, we may have more successful computation paths and hence may obtain more problems in FSA. Maximization of $FSA(R, A)$ by increasing R and A is of central importance in the research of logical computation.

2.6. **SLD resolution as ET-based method.** In the ET-based method illustrated in Theorem 2.1, the initial transformation should satisfy

$$\varphi\left(\bigcap Models(F \wedge E)\right) = \varphi\left(\bigcap Models\,(K \wedge Cs_1)\right).$$

A typical method for this transformation will be obtained from Theorem 2.2.

A subset A of $\mathcal{A}_u$ is closed (under substitution) iff if $a \in A$ and $\theta \in \mathcal{S}$, then $a\theta \in A$. Assume that A_0 and A_1 are subsets of $\mathcal{A}_u$ that are closed under substitution. The set of all definite clauses $(H \leftarrow B)$ such that $H \in A_1$ and $B \subseteq A_0$ is denoted by $dc(A_0, A_1)$.

Definition 2.5. *Assume that $D \subseteq dc(A_0, A_1)$, where A_0 and A_1 are closed subsets of $\mathcal{A}_u$. A mapping m_D from $pow(A_0)$ to $pow(A_0 \cup A_1)$ is defined by: for any $G \subseteq A_0$,*

$$m_D(G) = G \cup \{H | ((H \leftarrow B) \text{ is a ground instance of } C) \;\&\; (C \in D) \;\&\; (B \subseteq G)\}.$$

Theorem 2.2. *Assume that A_0 and A_1 are mutually disjoint subsets of $\mathcal{A}_u$ that are closed under substitution. Assume that F, K, and E are first-order formulas, and D and Cs_1 are subsets of $dc(A_0, A_1)$. If*

- *F is the completion of D, i.e., $F = comp(D)$,*
- *K is equivalent to D, and*
- *E is equivalent to Cs_1,*

then

$$\varphi\left(\bigcap Models(F \wedge E)\right) = \varphi\left(\bigcap Models(K \wedge Cs_1)\right).$$

Proof: By $F = comp(D)$, $\bigcap Models(F) = \bigcap Models(D)$. It follows that $\bigcap Models(F \wedge E) = m_E(\bigcap Models(F)) = m_{Cs_1}(\bigcap Models(D)) = m_{Cs_1}(\bigcap Models(K)) = \bigcap Models(K \wedge Cs_1)$. □

For the pal-pal QA problem, let D_0 be the set consisting of the definite clauses in Figure 1. Then, the formula F_0 is represented by $F_0 = comp(D_0)$. We take $F = F_0$, $D = D_0$, $K = D_0$, $E = E_1$, and $Cs_1 = MPD(E_1)$, where $MPD(X)$ is a meaning-preserving decomposition of a formula X [20]. By Theorem 2.2, we have

$$\varphi\left(\bigcap Models(F_0 \wedge E_1)\right) = \varphi\left(\bigcap Models(D_0 \wedge Cs_1)\right).$$

The formula $D_0 \wedge Cs_1$ in the right-hand side is also written as a clause set $D_0 \cup Cs_1$, and is used for SLD resolution of Cs_1 with respect to D_0, or application of unfolding with respect

(1) $pal(X) \leftarrow rev(X, X)$
(2) $rev([\,], [\,]) \leftarrow$
(3) $rev([A|X], Y) \leftarrow rev(X, R), app(R, [A], Y)$
(4) $app([\,], X, X) \leftarrow$
(5) $app([A|X], Y, [A|Z]) \leftarrow app(X, Y, Z)$

FIGURE 1. Definite clauses defining the predicates pal, rev, and app

to D_0 to Cs_1. By this transformation, SLD resolution for QA problems is considered as a subclass of the ET-based solution for QA problems.

2.7. **ET-based method vs SLD resolution.** We will show that

- the conventional SLD resolution method has a limitation in spite of the correctness theorem, and
- the ET-based method overcomes the limitation of the conventional SLD resolution method.

Using the pal-pal QA problem, we show that

- the pal-pal QA problem cannot be solved by the resolution method (Section 3 and Section 5), and
- the pal-pal QA problem can be solved by the ET-based method (Section 4).

The ET-based method overcomes the difficulty by using many ET rules.

3. Failure with General Unfolding.

We prove that the pal-pal QA problem cannot be solved by unfolding only.

3.1. **Failure by unfolding only transformation.** A definite clause whose head is an atom "exists" is called an "exists" clause. Recall the formula E_1, which was introduced in 2.2. Let C_1 be the "exists" clause

$$exists(A, X) \leftarrow pal([1, A|X]), pal([2, A|X]).$$

Let Cs_1 be the singleton set $\{C_1\}$. Then E_1 is equivalent to Cs_1, i.e., $Models(E_1) = Models(Cs_1)$. We consider the unfolding rule with respect to D_0, and let $R = \{unfolding\}$.

In this section we show that R cannot solve the pal-pal QA problem by the existence-finding formalization and $\{ans_1\}$, i.e.,

$$\text{Pal-pal QA problem} = \langle F_0 \wedge Cs_1, \varphi_1 \rangle \notin FSA(\{unfolding\}, \{ans_1\}).$$

This means that the pal-pal QA problem cannot reach the domain of ans_1 by using only the unfolding rule with respect to D_0.

3.2. **Infinite computation.** By unfolding, C_1 is typically transformed sequentially from C_1 to C_5 in Figure 2, where the body atoms selected for unfolding are underlined. Depending on the selection of body atoms, we have many possible sequences of clause sets. However, the following proposition holds, where we use D_1, which is a subset of D_0, consisting of the following three clauses:

C_p: $(pal(X) \leftarrow rev(X, X))$,
C_r: $(rev([A|X], Y) \leftarrow rev(X, R), app(R, [A], Y))$,
C_a: $(app([A|X], Y, [A|Z]) \leftarrow app(X, Y, Z))$.

C_1: $exists(A, X) \leftarrow \underline{pal([1, A|X])}, pal([2, A|X])$.
C_2: $exists(A, X) \leftarrow rev([1, A|X], [1, A|X]), \underline{pal([2, A|X])}$.
C_3: $exists(A, X) \leftarrow \underline{rev([1, A|X], [1, A|X])}, rev([2, A|X], [2, A|X])$.
C_4: $exists(A, X) \leftarrow rev([A|X], B), app(B, [1], [1, A|X]), \underline{rev([2, A|X], [2, A|X])}$.
C_5: $exists(A, X) \leftarrow rev([A|X], B), app(B, [1], [1, A|X]), rev([A|X], C), app(C, [2], [2, A|X])$.

FIGURE 2. Typical transformation sequence starting from C_1

Proposition 3.1. *Let* $C = (exists(A, X) \leftarrow pal([1, A|X]), pal([2, A|X]))$. *Let* Cs_n *be the result of* n*-times application of unfolding with respect to* D_0 *to* $\{C\}$. *Then* Cs_n *is not an empty set, and for any* $n \in \{1, 2, \ldots\}$, *all clauses in* Cs_n *are definite clauses that contain "exists" atoms in the left-hand sides, and have non-empty bodies.*

Proof: Since unfolding with respect to D_0 keeps the predicate "exists" in the head, all clauses in Cs_n are "exists" clauses. There is Cs'_n such that (a) Cs'_n is the result of n-times application of unfolding with respect to D_1 to $\{C\}$, and (b) $Cs'_n \subseteq Cs_n$. Since unification in unfolding with respect to D_1 never fails, Cs'_n is not an empty set. Since unfolding does not decrease the number of body atoms, all clauses in Cs'_n are "exists" clauses with non-empty bodies. Hence Cs_n is not an empty set. If there is a clause in Cs_n that has an empty body, there are ground terms s and t such that $[1, s|t]$ and $[2, s|t]$ are palindromes. Hence there is no clause in Cs_n that has an empty body. It follows that Cs_n is not an empty set, and all clauses in Cs_n are "exists" clauses with non-empty bodies. □

3.3. **Unreachability and unsolvability.** Unfolding is a major transformation rule for solving QA problems. However, we show that the pal-pal QA problem can never reach the domain of ans_1 by unfolding alone.

Theorem 3.1. *Let* $C = (exists(A, X) \leftarrow pal([1, A|X]), pal([2, A|X]))$. *Let* Cs_n *be the result of* n*-times application of unfolding with respect to* D_0 *to* $\{C\}$. *Then*

$$\langle comp(D_0) \wedge Cs_n, \varphi_1 \rangle \notin dom(ans_1).$$

Proof: Let Cs_n be the result of n-times application of unfolding with respect to D_0 to $\{C\}$. By Proposition 3.1, Cs_n is not an empty set, and all clauses in Cs_n are "exists" clauses with non-empty bodies. Hence $\langle comp(D_0) \wedge Cs_n, \varphi_1 \rangle \notin dom(ans_1)$. □

4. **Solution with Many ET Rules.** In ET-based problem solving, the search space is extended by inclusion of more ET rules. We give a solution with specialized ET rules to the pal-pal QA problem.

4.1. **Solution with specialized/multi-head ET rules.** The pal-pal QA problem cannot be solved by unfolding alone, i.e.,

$$\text{Pal-pal QA problem} = \langle F_0 \wedge Cs_1, \varphi_1 \rangle \notin FSA(\{unfolding\}, \{ans_1\}),$$

and cannot be solved by resolution and factoring, i.e.,

$$\text{Pal-pal QA problem} = \langle F_0 \wedge Cs_1, \varphi_1 \rangle \notin FSA(\{resolution, factoring\}, \{ans_1\}).$$

Similarly, we can say that the pal-pal QA problem cannot be solved by a combination of unfolding, resolution, and factoring:

$$\text{Pal-pal QA problem} = \langle F_0 \wedge Cs_1, \varphi_1 \rangle \notin FSA(\{unfolding, resolution, factoring\}, \{ans_1\}).$$

What rules should be used to solve the pal-pal QA problem? It is a two-head ET rule, called r_{rev_2}, which will be introduced in Section 4.2.

$$\text{Pal-pal QA problem} = \langle F_0 \wedge Cs_1, \varphi_1 \rangle \in FSA(\{unfolding, r_{rev_2}\}, \{ans_1\}).$$

Furthermore, we show that the pal-pal QA problem can be solved by using six specialized unfolding rules and the two-head rule r_{rev_2}:

$$\text{Pal-pal QA problem} = \langle F_0 \wedge Cs_1, \varphi_1 \rangle \in FSA(R_1, \{ans_1\}),$$

where $R_1 = \{r_{pal}, r_{rev_0}, r_{rev_1}, r_{app_0}, r_{app_1}, r_{app_2}, r_{rev_2}\}$.

4.2. **Solution by using a two-head ET rule.** Given a logical structure and an answer mapping, one typical improvement of finite solvability is obtained by creation and accumulation of ET rules. After the transformation from C_1 to C_5 in Figure 2, the clause

$$C_5\colon\ exists(A,X) \leftarrow \underline{rev([A|X],B)}, app(B,[1],[1,A|X]), \underline{rev([A|X],C)}, app(C,[2],[2,A|X]).$$

has two rev atoms, $rev([A|X],B)$ and $rev([A|X],C)$. Since the first arguments of the two rev atoms are the same, the second arguments are also the same, i.e., B and C can be unified. More precisely, two rev atoms, $rev(t,t_1)$ and $rev(t,t_2)$, are found, and the second rev atom can be removed after specialization by $t_1 = t_2$. This transformation is obtained by an ET rule r_{rev_2}, which is represented by

$$rev(X,Y), rev(X,Z) \Rightarrow \{=(Y,Z)\}, rev(X,Y).$$

Let us observe further transformation. By application of r_{rev_2}, we have

$$C_6\colon\ exists(A,X) \leftarrow rev([A|X],B), \underline{app(B,[1],[1,A|X])}, app(B,[2],[2,A|X]).$$

Unfolding by selecting $app(B,[1],[1,A|X])$ gives

$$C_7\colon\ exists(A,X) \leftarrow rev([A|X],[1|Y]), app(Y,[1],[A|X]), \underline{app([1|Y],[2],[2,A|X])}.$$

Unfolding by selecting $app([1|Y],[2],[2,A|X])$ gives no clause. So the resulting clause set contains no "exists" clause, and thus, the resulting MI problem enters the domain of ans_1. Hence, the pal-pal QA problem can be solved by using two ET rules, unfolding and r_{rev_2}:

$$\text{Pal-pal QA problem} = \langle F_0 \wedge Cs_1, \varphi_1\rangle \in FSA(\{unfolding, r_{rev_2}\}, \{ans_1\}).$$

4.3. **Computing with specialized ET rules.** A specialized unfolding rule with respect to an atom B and a set D of definite clauses, denoted by $Unfold(B,D)$, is an unfolding rule with respect to D that is only applicable to all body atoms that are instances of B.

$$\begin{array}{ll}
r_{pal}\colon & pal(X) \Rightarrow rev(X,X).\\
r_{rev_0}\colon & rev([\,],X) \Rightarrow \{=(X,[\,])\}.\\
r_{rev_1}\colon & rev([A|X],Y) \Rightarrow rev(X,V), app(V,[A],Y).\\
r_{app_0}\colon & app([\,],Y,Z) \Rightarrow \{=(Y,Z)\}.\\
r_{app_1}\colon & app([A|X],Y,Z) \Rightarrow \{=(Z,[A|W])\}, app(W,Y,Z).\\
r_{app_2}\colon & app(X,Y,[A|Z]) \Rightarrow \{=(X,[\,]),=(Y,[A|Z])\};\\
 & \qquad\qquad\ \Rightarrow \{=(X,[A|V])\}, app(V,Y,Z).\\
r_{rev_2}\colon & rev(X,Y), rev(X,Z) \Rightarrow \{=(Y,Z)\}, rev(X,Y).
\end{array}$$

FIGURE 3. Examples of rewriting rules

Refer to D_0 in Figure 1. In Figure 3, we have one two-head rule r_{rev_2}, and six specialized ET rules, which are specialized unfolding rules as follows:

- $r_{pal} = Unfold(pal(X), D_0)$,
- $r_{rev_0} = Unfold(rev([\,],X), D_0)$,
- $r_{rev_1} = Unfold(rev([A|X],Y), D_0)$,
- $r_{app_0} = Unfold(app([\,],Y,Z), D_0)$,
- $r_{app_1} = Unfold(app([A|X],Y,Z), D_0)$,
- $r_{app_2} = Unfold(app(X,Y,[A|Z]), D_0)$.

These six rules can be used in place of unfolding in Section 3.3. By using these six rules together with r_{rev_2}, we have a sequence of sets of clauses:

$$\{C_1\} \to \{C_2\} \to \{C_3\} \to \{C_4\} \to \{C_5\} \xrightarrow{r_{rev_2}} \{C_6\} \to \{C_7\} \to \{\},$$

where the clauses from C_1 to C_7 are shown in Section 3.2 and in Section 4.2. Hence, the pal-pal QA problem is solved by using the six specialized unfolding rules and the two-head rule r_{rev_2}:

$$\text{Pal-pal QA problem} = \langle F_0 \wedge Cs_1, \varphi_1 \rangle \in FSA(R_1, \{ans_1\}).$$

5. **Comparison of SLD Resolution and the ET-Based Method.** We compare SLD resolution and the ET-based method for solving QA problems.

5.1. **Failure with SLD resolution.** Consider the following two formalizations for the pal-pal QA problem:

- $\langle D', G' \rangle$ (a conventional SLD formalization), and
- $\varphi\left(\bigcap Models(D_0 \wedge Cs_1)\right)$ (an ET-based MI-problem formalization).

The first formalization $\langle D', G' \rangle$ is based on the theory of SLD resolution, where

- $D' = D_0 \cup \{C_1\}$,
- D_0 is the set consisting of the definite clauses in Figure 1,
- $C_1 = (exists(A, X) \leftarrow pal([1, A|X]), pal([2, A|X]))$,
- $q' = exists(A, X)$, and
- $G' = (\leftarrow q')$.

This formalization has a computation path: $G' = G_1, G_2, G_3, \ldots$, each goal of which is a negative clause. The second formalization $\varphi(\bigcap Models(D_0 \wedge Cs_1))$ is based on the theory of the ET-based solution (Section 2.6), which has a computation path: $D_0 \wedge Cs_1$, $D_0 \wedge Cs_2$, $D_0 \wedge Cs_3$, ..., each clause set of which is a set of definite clauses. Since these two computation paths are identified by a one-to-one correspondence, and they produce the same computed answers, we consider that the ET-based method can produce (a new form of) SLD resolution.

The two computation paths never end and we cannot obtain computed answers, which can be proved similarly to the case of the ET-based solution with unfolding in Section 3. By the result of Section 3,

$$\text{Pal-pal QA problem} = \langle F_0 \wedge Cs_1, \varphi_1 \rangle \notin FSA(\{unfolding\}, \{ans_1\}).$$

This means that by using only the unfolding rule with respect to D', the pal-pal QA problem cannot reach the domain of ans_1. By the procedural similarity between unfolding and $resolution(D')$, we have

$$FSA(\{unfolding\}, \{ans_1\}) = FSA(\{resolution(D')\}, \{ans_1\}).$$

Hence we conclude that

$$\text{Pal-pal QA problem} = \langle F_0 \wedge Cs_1, \varphi_1 \rangle \notin FSA(\{resolution(D')\}, \{ans_1\}),$$

which means that the pal-pal QA problem cannot reach the domain of ans_1 by using only the resolution rule with D'. More simply, the pal-pal QA problem cannot be solved by SLD resolution.

5.2. **Improvement of solvability.** Solvability can be improved by increasing ET rules. For any sets of ET rules R and R', for any sets of answer mappings A and A', if $R \subseteq R'$ and $A \subseteq A'$, then

$$FSA(R, A) \subseteq FSA(R', A').$$

For the pal-pal QA problem, we know that

$$FSA(\{unfolding\}, \{ans_1\}) \subset FSA(\{unfolding, r_{rev_2}\}, \{ans_1\}).$$

Since $FSA(\{unfolding\}, \{ans_1\}) = FSA(\{resolution(D')\}, \{ans_1\})$, we have

$$FSA(\{resolution(D')\}, \{ans_1\}) \subset FSA(\{unfolding, r_{rev_2}\}, \{ans_1\}).$$

6. **Concluding Remarks.** We proposed an ET-based method for solving QA problems. We formalize a QA problem as an MI problem, using an extraction mapping and a formula. We try to solve the class of MI problems by using ET rules, which is a sharp contrast to the conventional resolution-centered logical problem solving by using inference rules. General inference rules, such as resolution, are usually used in the conventional methods. On the other hand, an unlimited number of ET rules, including specialized unfolding-based rules, are used.

Each time a rule set is determined, the ET-based method can produce many solution methods that are guaranteed to be correct. The ET-based method admits many formalizations. We show that the pal-pal QA problem cannot be solved only by unfolding. Any computation path that is generated by SLD resolution can be regarded as one that is generated by the ET-based method. It was shown that the ET-based method has larger finite solvability of logical problems than the conventional resolution method. For each resolution-based solution, there is a solution by the ET-based method. It was proved that the pal-pal QA problem cannot be solved by the resolution method, but can be solved by the ET-based method.

The ET-based method is useful to overcome the limitation of SLD resolution. In the ET-based method, the correctness of the result of computation is guaranteed by equivalent transformation at each step together with unlimited ET rules. By inventing more ET rules, more logical problems will be solved practically, compared to the logical computation by a limited number of inference rules in the conventional methods [12, 21, 22, 23, 24].

Theorem 2.1 establishes a sufficient condition for the correctness of all transformational logical problem solving methods. The conventional SLD resolution for proof/QA problems often produces incorrect results by the violation of the conditions in Theorem 2.1. We need to be aware of the limitations of first-order logic and the conventional theory of logical computation. Based on Theorem 2.1, we can extend the theory of logic and logical computation. Theoretical and/or practical innovation will be attained by a new logic based on a general axiomatic logic and ET-based logical problem solving [25, 26, 27].

Acknowledgments. This work was partially supported by (i) JSPS KAKENHI Grant Numbers 25280078 and 26540110, (ii) the Center of Excellence in Intelligent Informatics, Speech and Language Technology and Service Innovation (CILS), Thammasat University, and (iii) the Intelligent Informatics and Service Innovation (IISI) Center, Sirindhorn International Institute of Technology (SIIT), Thammasat University. The authors also gratefully acknowledge the helpful comments and suggestions from the reviewers.

REFERENCES

[1] J. A. Robinson, A machine-oriented logic based on the resolution principle, *Journal of the ACM*, vol.12, pp.23-41, 1965.
[2] C. L. Chang and R. C. T. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973.
[3] K. Doets, *From Logic to Logic Programming*, The MIT Press, 1994.
[4] M. Fitting, *First-Order Logic and Automated Theorem Proving*, 2nd Edition, Springer-Verlag, 1996.
[5] R. A. Kowalski and D. Kuehner, Linear resolution with selection function, *Artificial Intelligence*, vol.2, nos.3-4, pp.227-260, 1971.
[6] C. Walther, A mechanical solution of Schubert's Steamroller by many-sorted resolution, *Artificial Intelligence*, vol.26, no.2, pp.217-224, 1985.
[7] F. J. Pelletier, Seventy-five problems for testing automatic theorem provers, *Journal of Automated Reasoning*, vol.2, no.2, pp.191-216, 1986.
[8] M. Stickel, Schubert's Steamroller problem: Formulations and solution, *Journal of Automated Reasoning*, vol.2., no.2, pp.89-104, 1986.

[9] T. C. Wang and W. W. Bledsoe, Hierarchical deduction, *Journal of Automated Deduction*, vol.3, no.1, pp.35-77, 1987.
[10] R. Manthey and F. Bry, SATCHMO: A theorem prover implemented in Prolog, *Proc. of the 9th International Conference on Automated Deduction*, Argonne, IL, pp.415-434, 1988.
[11] K. Akama and E. Nantajeewarawat, Solving proof problems with equivalent transformation rules, *International Journal of Innovative Computing, Information and Control*, vol.18, no.1, pp.331-344, 2022.
[12] K. Akama and E. Nantajeewarawat, Equivalent transformation in an extended space for solving query-answering problems, *Proc. of the 6th Asian Conference on Intelligent Information and Database Systems*, Bangkok, Thailand, pp.232-241, 2014.
[13] J. W. Lloyd, *Foundations of Logic Programming*, 2nd Edition, Springer-Verlag, 1987.
[14] R. A. Kowalski, Predicate logic as a programming language, *Information Processing 74 (Proc. of the 6 IFIP Congress, Stockholm)*, North-Holland, Amsterdam, pp.569-574, 1974.
[15] R. A. Kowalski, Algorithm = Logic + Control, *Communications of the ACM*, vol.22, pp.424-435, 1979.
[16] J. Minker, *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann Publishers Inc., 1988.
[17] M. Gelfond and V. Lifschitz, The stable model semantics for logic programming, *Proc. of International Logic Programming Conference and Symposium*, pp.1070-1080, 1988.
[18] M. Gelfond and V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New Generation Computing*, vol.9, pp.365-386, 1991.
[19] K. Akama and E. Nantajeewarawat, Formalization of logical problems as model-intersection problems on an extended clause space, *International Journal of Innovative Computing, Information and Control*, vol.17, no.4, pp.1103-1117, 2021.
[20] K. Akama and E. Nantajeewarawat, Skolemization that preserves logical meanings, *International Journal of Innovative Computing, Information and Control*, vol.17, no.1, pp.1-13, 2021.
[21] K. Akama and E. Nantajeewarawat, Solving query-answering problems with constraints for function variables, *Proc. of the 10th Asian Conference on Intelligent Information and Database Systems*, Dong Hoi City, Vietnam, pp.36-47, 2018.
[22] K. Akama and E. Nantajeewarawat, Model-intersection problems with existentially quantified function variables: Formalization and a solution schema, *Proc. of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, Porto, Portugal, vol.2, pp.52-63, 2016.
[23] K. Akama and E. Nantajeewarawat, Unfolding existentially quantified sets of extended clauses, *Proc. of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, Porto, Portugal, vol.2, pp.96-103, 2016.
[24] K. Akama, E. Nantajeewarawat and T. Akama, Computation control by prioritized ET rules, *Proc. of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, Seville, Spain, vol.2, pp.84-95, 2018.
[25] K. Akama, E. Nantajeewarawat and T. Akama, Logical problem solving framework, *Proc. of the 11th Asian Conference on Intelligent Information and Database Systems*, Yogyakarta, Indonesia, pp.28-40, 2019.
[26] K. Akama and E. Nantajeewarawat, Knowledge-representation-logic: An extension of first-order logic, *International Journal of Innovative Computing, Information and Control*, vol.18, no.4, pp.1055-1069, 2022.
[27] K. Akama and E. Nantajeewarawat, A foundation of logical problem solving, *International Journal of Innovative Computing, Information and Control*, vol.18, no.5, pp.1559-1570, 2022.

Author Biography

Kiyoshi Akama received the B.Eng. and M.Eng. degrees in control engineering from Tokyo Institute Technology, Japan, in 1973 and 1975, respectively; and the D.Eng. degree in control engineering from Tokyo Institute Technology, Japan, in 1989. He was an assistant professor at Faculty of Engineering, Tokyo Institute Technology, Japan, 1979-1981; a lecturer at Faculty of Letters, Hokkaido University, Japan, 1981-1989; an associate professor at Faculty of Engineering, Hokkaido University, Japan, 1989-1999; a professor at Center for multimedia Studies, Hokkaido University, Japan, 1999-2003; a professor at Information Initiative Center, Hokkaido University, Japan, 2003-2013; a specially-appointed professor at Information Initiative Center, Hokkaido University, 2013-2015; a professor at Graduate School of Information Science and Technology, Hokkaido University, Japan, 1999-2015.

Prof. Akama is currently an emeritus professor of Hokkaido University, Japan. His research interests include artificial intelligence, computer science, logic and computation, program generation and computation based on the equivalent transformation model, programming paradigms, and knowledge representation.

Ekawit Nantajeewarawat received the B.Eng. degree in computer engineering from Chulalongkorn University, Thailand, in 1987; and the M.Eng. and D.Eng. degrees in computer science from the Asian Institute of Technology, Thailand, in 1991 and 1997, respectively.

Dr. Nantajeewarawat is currently an associate professor of computer science at Sirindhorn International Institute of Technology, Thammasat University, Thailand. His research interests include knowledge representation, automated reasoning, rule-based equivalent transformation, program synthesis, formal ontologies, and object-oriented modelling.