International Journal of Innovative Computing, Information and Control
Volume **18**, Number **6**, December **2022**

 ISSN 1349-4198
pp. **1799–1813**

REPRESENTATION AND COMPUTATION OF LOGICAL PROBLEMS WITH IF-AND-ONLY-IF FORMULAS

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 May 2022; revised August 2022

Abstract. *Many logical problems, such as proof problems and query-answering problems, on first-order formulas can be mapped into model-intersection (MI) problems on an extended clause space. Increasing the power of solving MI problems is essential to solve logical problems correctly and more efficiently. MI problems constitute one of the largest classes of logical problems and are of fundamental importance. This paper extends the representation power of MI problems by an introduction of iff-formulas. It also extends the computation power of MI problems by inventing new equivalent transformation rules such as replacement using an iff-formula and removal of an iff-formula. The correctness of this new solution method is guaranteed.*

Keywords: Model-intersection problem, Extended clause, Function variable, Equivalent transformation, If-and-only-if formula

1. **Introduction.** Logical structure should be of fundamental and central importance for human intelligence. The conventional theory of logic is proof-centered and inference-based one on the first-order formula space. The current theory of logic is, however, unsatisfactory. The theory of logic should be reconstructed extensively, seeking appropriate structure to capture human intelligence and to maximize representational and computational power. We need to change the structure of logic by extending problem classes, computation methods, and the underlying formula space.

Proof problems have long been the main target for logical problem solving. A problem in this class is a "yes/no" problem concerning with checking whether one logical formula is a logical consequence of another logical formula. A query-answering (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. After the resolution principle for proof problems was invented by Robinson [1], many logical problems, such as proof problems and QA problems, on first-order formulas have been investigated extensively [2-6].

However, the class of problems solved by conventional methods is not large enough. Complete integration of proof problems and QA problems on first-order logic is not realized. Recently, the deep learning community has given increasing attention to integration

DOI: 10.24507/ijicic.18.06.1799

of logical reasoning into neural network architectures. Such integration usually realizes limited-size constraint solvers in deep learning architectures, and does not extend the solvability obtained by the theories of logical computation [7-16].

SLD resolution is incomplete, both for the class of all proof problems and for the class of all QA problems, that are defined by using first-order formulas. The conventional computation theory based on SLD resolution has such a theoretical limitation.

To change the core structure of logic, we use a model-intersection problem (MI problem) [17] and equivalent transformation (ET). The set of all MI problems constitutes a very large class of logical problems. MI problems integrate proof problems and QA problems. The integration is realized by a general conversion method to map all proof problems and all QA problems on first-order logic into MI problems. MI problems enable us to solve proof problems and QA problems by using equivalent transformation rules (ET rules) with full guarantee of correctness. Integration into MI problems is of fundamental importance to establish a general solution method to solve all deductive problems on first-order formulas, overcoming the limitation of conventional methods based on inference within the first-order formula space. Increasing the power of solving MI problems by ET rules is essential to solve logical problems correctly and more efficiently.

The space used in the classical theorem-proving theory is first-order logic, which is identified with the usual clause space based on the conversion by the conventional Skolemization. Since this space has serious limitations for representation and computation, we use first-order logic possibly with built-in constraints and the extended clause space [18].

This paper extends further the power of representation and computation of MI problems by (i) enrichment of the expressive power of MI problems with if-and-only-if formulas (*iff*-formulas) [19], and (ii) invention of ET rules such as replacement using an *iff*-formula and removal of an *iff*-formula. The resulting representative power of problem descriptions is higher than other representation schemes for proof problems and QA problems considered in our previous works, including [20, 21]. The correctness of this new solution method is guaranteed based on the ET principle.

This paper keeps MI problems as the central problem class and computation by ET rules, while it increases the underlying formula space and extends computation methods as follows:

- Extension of the space from the usual clause space to a pair space, i.e., a pair of a clause set and an *iff*-formula set is used;
- Invention of ET rules on the space, i.e., ET rules for replacement and removal regarding *iff*-formulas.

The rest of the paper is organized as follows. Section 2 gives an introductory example, where representation by definite clauses, that by first-order formulas, and that by *iff*-formulas are illustrated. Representation by a pair of extended clauses and *iff*-formulas is introduced. Section 3 introduces MI problems in triple forms, each of which consists of extended clauses and *iff*-formulas, and an extraction mapping. Section 4 provides a solution method for MI problems in triple forms by ET. Section 5 introduces ET rules related to *iff*-formulas. Section 6 illustrates a solution of a puzzle by using *iff*-formulas, and explains that the proposed computation framework is an extension of SLD resolution for proof problems and QA problems. Section 7 provides conclusions.

2. Representation by Clauses and *Iff*-Formulas.

Definite clauses, first-order formulas, and *iff*-formulas are compared and representation by a pair of extended clauses and *iff*-formulas is introduced.

2.1. **Pal-pal example and definite clause formalization.** Consider a logical problem, named a "pal-pal" problem, taken from [22]. This problem is to find all lists X that satisfy the conjunction of $pal([1|X])$ and $pal([2|X])$, where pal is a predicate standing for "palindrome".

Consider a set D_0 of definite clauses as the background knowledge of the pal-pal problem:

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) $app([A|X], Y, [A|Z]) \leftarrow app(X, Y, Z)$.

The predicates rev and app stand for *reverse* and *append*, respectively. The pal-pal problem can be formalized as a QA problem $\langle D_0 \cup Q_D, ans(X)\rangle$, where Q_D is a singleton set of a definite clause

$$ans(X) \leftarrow pal([1|X]), pal([2|X]),$$

defining the predicate ans that stands for *answer*. This pal-pal problem is also called a pal-pal QA problem.

2.2. **Pal-pal puzzle with first-order logic.** Definitions of the three predicates, pal, rev, and app, in first-order formulas are shown as follows:

$$\begin{aligned}
&\forall x : pal(x) \leftrightarrow rev(x, x).\\
&\forall x, y : rev(x, y) \leftrightarrow\\
&\qquad (x = [\,] \wedge y = [\,]) \vee\\
&\qquad \exists a, w, u : (x = [a|w] \wedge rev(w, u) \wedge app(u, [a], y)).\\
&\forall x, y, z : app(x, y, z) \leftrightarrow\\
&\qquad (x = [\,] \wedge y = z) \vee\\
&\qquad \exists a, w, u : (x = [a|w] \wedge z = [a|u] \wedge app(w, y, u)).
\end{aligned}$$

The conjunction of these three formulas is denoted by F_0. The pal-pal QA problem can be formalized as an MI problem $\langle F_0 \cup Q_F, \varphi_0\rangle$, where Q_F is a singleton set of a first-order formula

$$\forall x : pal([1|x]) \wedge pal([2|x]) \rightarrow ans(x),$$

and φ_0 is an extraction mapping that gives, for any set G of ground user-defined atoms, the set of all ground terms t satisfying $ans(t) \in G$. According to the general definition of an MI problem [17], the answer to this MI problem, denoted by $ans_{\mathrm{MI}}(F_0 \cup Q_F, \varphi_0)$, is given by

$$ans_{\mathrm{MI}}(F_0 \cup Q_F, \varphi_0) \;=\; \varphi_0\left(\bigcap Models(F_0 \cup Q_F)\right),$$

where for any first-order formula K, $Models(K)$ is the set of all models of K and $\bigcap Models(K)$ is the intersection of all models of K.

2.3. **Pal-pal puzzle with *iff*-formulas.** Omitting quantifiers $\forall$ and $\exists$, we represent F_0 as a set E_0 of simpler *if-and-only-if formulas*:

$$\begin{aligned}
&pal(x) \leftrightarrow rev(x, x).\\
&rev(x, y) \leftrightarrow\\
&\qquad (x = [\,] \wedge y = [\,]) \vee\\
&\qquad (x = [a|w] \wedge rev(w, u) \wedge app(u, [a], y)).\\
&app(x, y, z) \leftrightarrow\\
&\qquad (x = [\,] \wedge y = z) \vee\\
&\qquad (x = [a|w] \wedge z = [a|u] \wedge app(w, y, u)).
\end{aligned}$$

An if-and-only-if formula is also called an *iff*-formula. The three *iff*-formulas above are referred to as *iff*$(pal(x))$, *iff*$(rev(x,y))$, and *iff*$(app(x,y,z))$, respectively. F_0 is uniquely determined by E_0, i.e., $F_0 = \text{FOL}(E_0)$ using a mapping FOL to associate a set of first-order formulas with a set of *iff*-formulas.

2.4. **Meaning-preserving decomposition (MPD).** Let FOL_c be the set of all first-order formulas possibly with built-in constraint atoms. Let K be a first-order formula in FOL_c. Meaning-preserving decomposition of K, i.e., $\text{MPD}(K)$, is the set of extended clauses that is equivalent to K. For instance, supposing that

$$K_0 = (\forall x : (man(x) \rightarrow (\exists y : (love(y,x) \wedge motherOf(y,x))))),$$

$\text{MPD}(K_0)$ is the set consisting of the following two extended clauses, where h_0 is a unary function variable:

- $love(y,x) \leftarrow man(x), func(h_0,x,y)$
- $motherOf(y,x) \leftarrow man(x), func(h_0,x,y)$

2.5. **Representation by a pair of clauses and *iff*-formulas.** We have three formalizations of the pal-pal QA problem using

- a set D_0 of definite clauses,
- a set F_0 of first-order formulas, and
- a set E_0 of *iff*-formulas.

Let *completion* denote a mapping for obtaining, from a definite clause set D, a first-order formula F such that (i) $F = \text{FOL}(E)$ for some *iff*-formula E, and (ii) the model of F is equal to the minimal model of D. Then, the relation among these three sets is shown by

$$F_0 = completion(D_0) = \text{FOL}(E_0).$$

Which is the best representation among these three forms, i.e., clauses, first-order formulas, and *iff*-formulas? Representation by first-order formulas may be the best for direct formalization from problem statements. However, first-order formulas may have smaller advantage to be transformed. For transformational purposes, a clause set Cs_0 is better used in SLD resolution for proof problems, which is determined by $Cs_0 = \text{MPD}(F_0)$. A definite clause representation is well used in SLD resolution for QA problems. Definite clauses form a subclass of clauses. *Iff*-formulas form a subclass of formulas and can be used to transform all formulas effectively.

For general representation and efficient computation for proof problems and QA problems, we propose a pair representation of clauses and *iff*-formulas. We take a pair, i.e., $\langle Cs, E\rangle$, for representation and computation of a problem, which can represent (as two special cases)

- by $\langle Cs, \emptyset\rangle$ a set Cs of clauses, and
- by $\langle \emptyset, E\rangle$ a set E of *iff*-formulas.

Given a first-order formula F, we first determine F_1, F_2 and E such that $F = F_1 \wedge F_2$ and $F_2 = \text{FOL}(E)$. We then obtain Cs by $Cs = \text{MPD}(F_1)$, and we have a pair representation $\langle Cs, E\rangle$ that satisfies

$$\text{MPD}(F) = Cs \cup \text{MPD}(\text{FOL}(E)).$$

Note that Cs and $\text{MPD}(\text{FOL}(E))$ may have common clauses. For instance, it is admitted that we have $Cs = \text{MPD}(F)$ and at the same time, E is a non-empty set of *iff*-formulas.

The importance of considering a pair representation $\langle Cs, E\rangle$ is as follows: SLD resolution for proof problems uses a set Cs of clauses, and SLD resolution for QA problems conventionally uses a set D of definite clauses. Since D is a set of clauses, we can always consider SLD resolution for proof/QA problems in a clause space. However, this view

does not give a general and unified theory. Instead of D, SLD resolution for QA problems should take a set E of *iff*-formulas. It is expected that a general and unified theory of SLD resolution for proof/QA problems is obtained by considering Cs and E on the same level, i.e., in the space of pair representations.

The pair representation will be used for defining the pair space in Section 3.

3. Extension into a Pair Space with *Iff*-Formulas.

A pair space consisting of pairs of clause sets and *iff*-formula sets is introduced as an extended space for MI problems.

3.1. User-defined atoms, constraint atoms, and *func*-atoms.

We consider an extended formula space that contains three kinds of atoms, i.e., user-defined atoms, built-in constraint atoms, and *func*-atoms. A *user-defined atom* takes the form $p(t_1, \ldots, t_n)$, where p is a user-defined predicate and the t_i are usual terms. A *built-in constraint atom*, also simply called a *constraint atom* or a *built-in atom*, takes the form $c(t_1, \ldots, t_n)$, where c is a predefined constraint predicate and the t_i are usual terms. Let $\mathcal{A}_{\rm u}$ be the set of all user-defined atoms, $\mathcal{G}_{\rm u}$ the set of all ground user-defined atoms, $\mathcal{A}_{\rm c}$ the set of all constraint atoms, and $\mathcal{G}_{\rm c}$ the set of all ground constraint atoms.

A *func-atom* [23] is an expression of the form $func(h, t_1, \ldots, t_n, t_{n+1})$, where h is either an n-ary function constant or an n-ary function variable, and the t_i are usual terms. It is a *ground func*-atom if h is a function constant and the t_i are ground usual terms.

Let $FVar$ be the set of all function variables and $FCon$ the set of all function constants. Let $\mathcal{G}_{\rm t}$ denote the set of all ground usual terms. Each n-ary function constant is associated with a mapping from $\mathcal{G}_{\rm t}^n$ to $\mathcal{G}_{\rm t}$. There are two types of variables: usual variables and function variables.

3.2. Extended clauses.

An *extended clause* C on $\mathcal{A}_{\rm u} \cup \mathcal{A}_{\rm c}$ is a formula of the form

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

where each of $a_1, \ldots, a_m, b_1, \ldots, b_n$ is a user-defined atom in $\mathcal{A}_{\rm u}$ or a constraint atom in $\mathcal{A}_{\rm c}$, and each of $\mathbf{f}_1, \ldots, \mathbf{f}_p$ is a *func*-atom. All usual variables occurring in C are implicitly universally quantified and their scope is restricted to the extended clause C itself. The sets $\{a_1, \ldots, a_m\}$ and $\{b_1, \ldots, b_n, \mathbf{f}_1, \ldots, \mathbf{f}_p\}$ are called the *left-hand side* and the *right-hand side*, respectively, of the extended clause C, and are denoted by $lhs(C)$ and $rhs(C)$, respectively. Let $userLhs(C)$ denote the number of user-defined atoms in the left-hand side of C. When $userLhs(C) = 0$, C is called a *negative extended clause*. When $userLhs(C) = 1$, C is called an *extended definite clause*. When $userLhs(C) > 1$, C is called a *multi-head extended clause*. When no confusion is caused, an extended clause, a negative extended clause, an extended definite clause, and a multi-head extended clause are also called a *clause*, a *negative clause*, a *definite clause*, and a *multi-head clause*, respectively.

3.3. 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 $\mathcal{A}_{\rm u} \cup \mathcal{A}_{\rm c}$ is a formula of the form

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

where $a \in \mathcal{A}_{\rm u}$ and each of the $conj_i$ is a finite subset of $\mathcal{A}_{\rm u} \cup \mathcal{A}_{\rm c}$. The atom a is called the *head* of the *iff*-formula I. It is a *ground iff*-formula iff a and $conj_1, \ldots, conj_n$ are all ground. 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 existentially quantified atom conjunction $\textsc{Fol}(conj_i, a)$ defined by

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

where $\{y_1, \ldots, y_k\} = var(conj_i) - var(a)$. Note that $\textsc{Fol}(conj_i, a)$ may contain variables in $var(a)$ as free variables. The *iff*-formula I corresponds to the universally quantified formula

$$\forall(a \leftrightarrow (\textsc{Fol}(conj_1, a) \vee \cdots \vee \textsc{Fol}(conj_n, a))),$$

which is denoted by $\textsc{Fol}(I)$. All variables in $var(a)$ are quantified universally at the top of the *iff*-formula.

3.4. **A pair space with extended clauses and *iff*-formulas.** Let ECLS_F denote the set of all extended clauses on $\mathcal{A}_\text{u} \cup \mathcal{A}_\text{c}$, and IFF the set of all *iff*-formulas on $\mathcal{A}_\text{u} \cup \mathcal{A}_\text{c}$. Let ECLS_pair denote the set of all $\langle Cs, E\rangle$ such that $Cs \subseteq \text{ECLS}_\text{F}$ and $E \subseteq \text{IFF}$. ECLS_pair is called a pair space.

Assuming that both Cs and E may possibly include function variables, we can best explain the role of function variables in $\langle Cs, E\rangle$ as follows: Implicit existential quantifications of function variables that appear in $Cs \cup E$ and implicit conjunction of elements in $Cs \cup E$ are assumed. Function variables that appear in $Cs \cup E$ are all existentially quantified and their scope covers all clauses and *iff*-formulas in $Cs \cup E$. With occurrences of function variables, clauses in Cs and *iff*-formulas in E are connected through shared function variables. After instantiating all function variables that appear in $Cs \cup E$ into function constants, clauses and *iff*-formulas in the instantiated set are totally separated.

3.5. **Interpretations and models.** An *interpretation* is a subset of $\mathcal{G}_\text{u}$. A ground user-defined atom $g \in \mathcal{G}_\text{u}$ is true under an interpretation I iff g belongs to I. Unlike ground user-defined atoms, the truth values of ground constraint atoms are predetermined independently of interpretations. Let $\textsc{TCon}$ denote the set of all true ground constraint atoms, i.e., a ground constraint atom $g \in \mathcal{G}_\text{c}$ is true iff $g \in \textsc{TCon}$. A ground *func*-atom $func(h, t_1, \ldots, t_n, t_{n+1})$ is true iff $h(t_1, \ldots, t_n) = t_{n+1}$.

A ground clause $C = (a_1, \ldots, a_m \leftarrow b_1, \ldots, b_n, \mathbf{f}_1, \ldots, \mathbf{f}_p) \in \text{ECLS}_\text{F}$, where $\{a_1, \ldots, a_m, b_1, \ldots, b_n\} \subseteq \mathcal{G}_\text{u} \cup \mathcal{G}_\text{c}$ and $\mathbf{f}_1, \ldots, \mathbf{f}_p$ are ground *func*-atoms, is true under an interpretation I (in other words, I *satisfies* C) iff at least one of the following conditions is satisfied.

1) There exists $i \in \{1, \ldots, m\}$ such that $a_i \in I \cup \textsc{TCon}$.
2) There exists $i \in \{1, \ldots, n\}$ such that $b_i \notin I \cup \textsc{TCon}$.
3) There exists $i \in \{1, \ldots, p\}$ such that $\mathbf{f}_i$ is false.

A ground *iff*-formula $iff(a) = (a \leftrightarrow (conj_1 \vee \cdots \vee conj_n)) \in \text{IFF}$ is true under an interpretation I (in other word, I satisfies $iff(a)$) iff the following two conditions are equivalent.

1) $a \in I$.
2) There exists $i \in \{1, \ldots, n\}$ such that $conj_i \subseteq I \cup \textsc{TCon}$.

Let $Cs \subseteq \text{ECLS}_\text{F}$ and $E \subseteq \text{IFF}$. An interpretation I is a *model* of $\langle Cs, E\rangle$ iff there exists a substitution σ for function variables that satisfies the following conditions.

1) All function variables occurring in Cs are instantiated by σ into function constants.
2) For any clause $C \in Cs$ and any substitution θ for usual variables, if $C\sigma\theta$ is a ground clause, then $C\sigma\theta$ is true under I.
3) For any *iff*-formula $iff(a) \in E$ and any substitution θ for usual variables, if $iff(a)\sigma\theta$ is a ground *iff*-formula, then $iff(a)\sigma\theta$ is true under I.

For any clause set $Cs \subseteq \text{ECLS}_\text{F}$, $Models(Cs)$ denotes the set of all models of Cs. For any $Cs \subseteq \text{ECLS}_\text{F}$ and any $E \subseteq \text{IFF}$, the set of all models of $\langle Cs, E\rangle$ is denoted by $Models(\langle Cs, E\rangle)$, and is defined as $Models(Cs \cup \textsc{Cls}(E))$, where $\textsc{Cls}(E)$ is the extended clause set that is equivalent to E.

4. **MI Problems with *Iff*-Formulas and Their Solutions.** A general solution framework for MI problems on the pair space that may contain *iff*-formulas is discussed.

4.1. **Model-intersection (MI) problems on $\mathbf{FOL_c}$.** An *MI problem* on $\mathrm{FOL_c}$ is a pair $\langle K, \varphi\rangle$, where K is a first-order formula in $\mathrm{FOL_c}$ and φ is a mapping from the power set of $\mathcal{G}_\mathrm{u}$ to some set W. The answer to this problem, denoted by $ans_\mathrm{MI}(K, \varphi)$, is defined by

$$ans_\mathrm{MI}(K, \varphi) \;=\; \varphi\left(\bigcap Models(K)\right),$$

where $\bigcap Models(K)$ is the intersection of all models of K.

4.2. **MI problems in triple forms on $\mathbf{ECLS_{pair}}$.** We define an MI problem on $\mathrm{ECLS_{pair}}$ as a triple $\langle Cs, E, \varphi\rangle$, where (i) $Cs \subseteq \mathrm{ECLS_F}$, (ii) $E \subseteq \mathrm{IFF}$, and (iii) φ is a partial mapping from the power set of $\mathcal{G}_\mathrm{u}$ to some set W. The answer to the MI problem $\langle Cs, E, \varphi\rangle$, denoted by $ans_\mathrm{MI}(Cs, E, \varphi)$, is defined by

$$ans_\mathrm{MI}(Cs, E, \varphi) = \varphi\left(\bigcap Models(Cs \cup \textsc{Cls}(E))\right),$$

where $\textsc{Cls}(E)$ is the set of extended clauses that is equivalent to E.

For example, the pal-pal QA problem in Section 2.1 is formalized as $ans_\mathrm{MI}(Cs_1, E_0, \varphi_1)$, where

1) Cs_1 is a singleton set of
$$ans(X) \leftarrow pal([1|X]), pal([2|X]).$$
2) E_0 is a set of *iff*-formulas
$$\{iff(pal(x)), iff(rev(x, y)), iff(app(x, y, z))\}.$$
3) φ_1 is a mapping that associates with any $G \subseteq \mathcal{G}_\mathrm{u}$ the set of all ground terms t such that $ans(t) \in G$.

4.3. **Transformation into triples.** An MI problem $\langle K, \varphi\rangle$ on $\mathrm{FOL_c}$ is transformed into a triple form on $\mathrm{ECLS_{pair}}$ as follows.

1) From K, identify a first-order formula K' and a set E of *iff*-formulas such that $K = K' \wedge \textsc{Fol}(E)$, where $\textsc{Fol}(E)$ is the first-order formula that is equivalent to E.
2) Convert K' by meaning-preserving decomposition into a clause set $Cs \subseteq \mathrm{ECLS_F}$, i.e., $Cs = \mathrm{MPD}(K')$.
3) Construct $\langle Cs, E, \varphi\rangle$ as the resulting triple.

Finding a nonempty set E of *iff*-formulas for converting K into K' and $\textsc{Fol}(E)$ is useful for solving MI problems since *iff*-formulas increase the possibility of transforming MI problems with less cost. As the number of *iff*-formulas in the set E increases, the possibility of problem transformation with small cost is higher.

4.4. **A procedure for solving MI problems with *iff*-formulas.** Assume that an MI problem $\langle K, \varphi\rangle$ on $\mathrm{FOL_c}$ is given. To solve this problem using ET, perform the following steps:

1) Transform $\langle K, \varphi\rangle$ into a triple $\langle Cs, E, \varphi\rangle$ on $\mathrm{ECLS_{pair}}$ using the transformation given in Section 4.3.
2) Successively transform the triple $\langle Cs, E, \varphi\rangle$ using the ET rules described by Items (a)-(d) below nondeterministically. Assume that $\left\langle \hat{Cs}, \hat{E}, \varphi \right\rangle$ is an MI 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 $\left\langle \hat{Cs}, \hat{E}, \varphi \right\rangle$ into $\left\langle \tilde{Cs}, \hat{E}, \varphi \right\rangle$ through the replacement rule using an *iff*-formula in Section 5.

(b) If $\hat{E}$ contains an *iff*-formula $\mathit{iff}(a)$ and for each atom b that occurs in $\hat{Cs}$ or $\hat{E} - \{\mathit{iff}(a)\}$, a and b are not unifiable, then transform $\left\langle \hat{Cs}, \hat{E}, \varphi \right\rangle$ into $\left\langle \hat{Cs}, \hat{E} - \{\mathit{iff}(a)\}, \varphi \right\rangle$ using the removal rule for an *iff*-formula in Section 5.

(c) Transform $\left\langle \hat{Cs}, \hat{E}, \varphi \right\rangle$ using ET rules on $\mathrm{ECLS}_{\mathrm{pair}}$ such as resolution, unfolding, and other ET rules in [24, 25].

(d) Transform $\left\langle \hat{Cs}, \hat{E}, \varphi \right\rangle$ using ET rules for constraints, e.g., ET rules for equality constraints and inequality constraints.

3) Assume that the transformation yields a triple $\langle Cs', E', \varphi \rangle$. Then output $\varphi(\bigcap \mathit{Models}(Cs' \cup \textsc{Cls}(E')))$, where $\textsc{Cls}(E')$ is the set of extended clauses that is equivalent to E'.

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

5. **ET Rules in the Presence of *Iff*-Formulas.** Based on the replacement operation using an *iff*-formula, we define ET rules for replacement using *iff*-formulas and for removing useless *iff*-formulas. Correctness of these ET rules is shown.

5.1. **Replacement operation using an *iff*-formula.** The replacement operation using an *iff*-formula is defined below. Assume that (i) $Cs \subseteq \mathrm{ECLS}_{\mathrm{F}}$, (ii) *occ* is an occurrence of an atom b in a clause $C \in Cs$, (iii) $\mathit{iff}(a)$ is an *iff*-formula $(a \leftrightarrow (\mathit{conj}_1 \vee \cdots \vee \mathit{conj}_n))$, (iv) ρ is a renaming substitution for usual variables such that C and $\mathit{iff}(a)\rho$ 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 $\textsc{Repl}(C, \mathit{occ}, \mathit{iff}(a), \rho, \theta)$ denote the first-order formula obtained by replacing b at *occ* with the disjunction
$$\textsc{Fol}(\mathit{conj}_1\rho\theta, a\rho\theta) \vee \cdots \vee \textsc{Fol}(\mathit{conj}_n\rho\theta, a\rho\theta).$$
- Let $\textsc{Repl}(Cs, C, \mathit{occ}, \mathit{iff}(a), \rho, \theta)$ denote the conjunction of $\textsc{Repl}(C, \mathit{occ}, \mathit{iff}(a), \rho, \theta)$ 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, $\textsc{Repl}(C, \mathit{occ}, \mathit{iff}(a), \rho, \theta)$ is not a clause. After the replacement at *occ*, a new clause set, say Cs', is obtained by using meaning-preserving decomposition, i.e.,
$$Cs' = \mathrm{MPD}(\textsc{Repl}(Cs, C, \mathit{occ}, \mathit{iff}(a), \rho, \theta)).$$
The resulting clause set Cs' is often simply said to be obtained by replacement using $\mathit{iff}(a)$ at the occurrence *occ* of b in C.

5.2. **Replacement rule for an *Iff*-formula and its correctness.** Given a set P of predicates, let $\mathit{GAtoms}(P)$ denote the set of all ground atoms in $\mathcal{G}_{\mathrm{u}}$ the predicates of which belong to P. An extraction mapping φ is said to be *independent* of a set P of predicates iff for any $G_1, G_2 \subseteq \mathcal{G}_{\mathrm{u}}$, if
$$(G_1 - G_2) \cup (G_2 - G_1) \subseteq \mathit{GAtoms}(P),$$
then $\varphi(G_1) = \varphi(G_2)$. An extraction mapping φ is independent of an atom a if φ is independent of a sigleton set of the predicate of a.

An ET rule on $\mathrm{ECLS}_{\mathrm{pair}}$ for replacement using an *iff*-formula is given by Theorem 5.1.

Theorem 5.1. *(Replacement using an Iff-formula) Assume that*

1) $\langle Cs, E, \varphi \rangle$ *is an MI problem on* $\mathit{ECLS}_{\mathrm{pair}}$.

2) E contains an iff-formula iff(a).
3) φ is independent of a.
4) occ is an occurrence of an atom b in a clause $C \in Cs$.
5) ρ is a renaming substitution for usual variables such that C and iff$(a)\rho$ have no usual variable in common.
6) θ is the most general matcher of $a\rho$ into b.
7) $Cs' = MPD(\textsc{Repl}(Cs, C, occ, \mathit{iff}(a), \rho, \theta))$.
Then $\langle Cs, E, \varphi\rangle$ can be equivalently transformed into $\langle Cs', E, \varphi\rangle$.

Proof: This theorem directly follows from the fact that replacement using an *iff*-formula preserves the logical meaning of the original formula. □

A replacement rule using an *iff*-formula with a predicate p in the head is denoted by repl-iff(p). In the pal-pal QA problem, we have three *iff*-formulas, the predicates of which are *pal*, *rev*, and *app*. We can use three ET rules; repl-iff(*pal*), repl-iff(*rev*), and repl-iff(*app*) corresponding to them.

5.3. **Removal rule for an *iff*-formula and its correctness.** An ET rule on $\text{ECLS}_{\text{pair}}$ for removing a useless *iff*-formula is given by Theorem 5.2.

Theorem 5.2. *(Removal of a useless iff-formula) Assume that*
1) $\langle Cs, E, \varphi\rangle$ is an MI problem on $ECLS_{\text{pair}}$.
2) E contains an iff-formula iff(a).
3) φ is independent of a.
4) For each atom b that occurs in Cs or $E - \{\mathit{iff}(a)\}$, a and b are not unifiable.
Then $\langle Cs, E, \varphi\rangle$ can be equivalently transformed into $\langle Cs, E - \{\mathit{iff}(a)\}, \varphi\rangle$.

Proof: The *iff*-formula iff(a) is the only constraint for the atom a and is satisfiable by assigning appropriate truth values to ground instances of a, which does not affect the answer to the MI problem $\langle Cs, E, \varphi\rangle$ due to Condition 3) of the assumption part of the theorem. □

By removing useless *iff*-formulas, time-and-space efficiency may be improved.

6. **Increasing Computation Power with *Iff*-Rules.** By examining solutions of the pal-pal (QA/proof) problems, the pair computation framework is compared with the conventional SLD resolution method.

6.1. **A two-head rule.** As shown in Section 4.2, the pal-pal QA problem is formalized as $\langle Cs_1, E_0, \varphi_1\rangle$. We want to find $ans_{\text{MI}}(Cs_1, E_0, \varphi_1)$, i.e., we want to calculate

$$\varphi\left(\bigcap \mathit{Models}(Cs_1 \cup \textsc{Cls}(E_0))\right).$$

Together with *eq* rules for equality constraint solving, we can use the ET rules repl-iff(*pal*), repl-iff(*rev*), and repl-iff(*app*).

However, it is known that these three *iff*-replacement rules are not sufficient for solving the pal-pal QA problem completely. In addition, we use a two-head rule r_{rev_2}, which is represented by

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

By the two-head rule r_{rev_2}, a clause C is transformed into a clause C'' as follows:

- $rev(X, Y)$ and $rev(X, Z)$ are matched with two atoms $rev(t, s_1)$ and $rev(t, s_2)$ in the body of C by a substitution $\{X/t, Y/s_1, Z/s_2\}$.
- C is specialized into C' by the unification of s_1 and s_2.
- The body atom $rev(t, s_2)$ that is matched by $rev(X, Z)$ above is removed from C' to produce a new clause C''.

6.2. **Towards the integration of proof problems and QA problems.** We discuss *iff*-rule-based solutions to the pal-pal (QA/proof) problems for consideration in the later sections on the integrated solutions to proof problems and QA problems in the triple form.

6.2.1. *The "pal-pal" QA problem.* The "pal-pal" QA problem is to find all lists X such that $[1|X]$ and $[2|X]$ are palindromes, which is formalized as the following MI problem $\langle Cs_1, E_0, \varphi_1\rangle$.

1) Cs_1 is a singleton set of

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

2) φ_1 is a mapping to give the set of all ground terms t such that $ans(t) \in G$ for any $G \subseteq \mathcal{G}_{\mathrm{u}}$.

By the ET computation in Table 1, we have

$$ans_{\mathrm{MI}}(Cs_1, E_0, \varphi_1) = ans_{\mathrm{MI}}(\{ans([\,]) \leftarrow\}, E_0, \varphi_1) = \{[]\},$$

which shows that the answer to the pal-pal QA problem is $\{[]\}$.

TABLE 1. Transformation for the pal-pal QA problem

	Problem representation	Rule applied
p_1	$\{ans(X) \leftarrow \underline{pal([1\|X])}, pal([2\|X])\}$	repl-iff(pal)
p_2	$\{ans(X) \leftarrow rv([1\|X],[1\|X]), \underline{pal([2\|X])}\}$	repl-iff(pal)
p_3	$\{ans(X) \leftarrow \underline{rv([1\|X],[1\|X])}, rv([2\|X],[2\|X])\}$	repl-iff(rev), eq
p_4	$\{ans(X) \leftarrow rv(X,A1), ap(A1,[1],[1\|X]), \underline{rv([2\|X],[2\|X])}\}$	repl-iff(rev), eq
p_5	$\{ans(X) \leftarrow \underline{rv(X,A1)}, ap(A1,[1],[1\|X]), \underline{rv(X,A2)}, ap(A2,[2],[2\|X])\}$	r_{rev_2}
p_6	$\{ans(X) \leftarrow rv(X,A1), \underline{ap(A1,[1],[1\|X])}, ap(A1,[2],[2\|X])\}$	repl-iff(app), eq
p_7	$\{ans([\,]) \leftarrow rv([\,],[\,]), ap([\,],[2],[2]),$	
	$ans(X) \leftarrow \underline{rv(X,[1\|A3])}, ap(A3,[1],X), ap([1\|A3],[2],[2\|X])\}$	repl-iff(rev), eq
p_8	$\{ans([\,]) \leftarrow rv([\,],[\,]), ap([\,],[2],[2]),$	
	$ans([A4\|A5]) \leftarrow rv(A5,A6), ap(A6,[A4],[1\|A3]), ap(A3,[1],[A4\|A5]),$	
	$\underline{ap([1\|A3],[2],[2,A4\|A5])}\}$	repl-iff(app), eq
p_9	$\{ans([\,]) \leftarrow rv([\,],[\,]), \underline{ap([\,],[2],[2])}\}$	repl-iff(app), eq
p_{10}	$\{ans([\,]) \leftarrow \underline{rv([\,],[\,])}\}$	repl-iff(rev), eq
p_{11}	$\{ans([\,]) \leftarrow\}$	–

6.2.2. *The "pal-pal" proof problem.* The pal-pal proof problem is defined to prove that there are no ground terms s and t such that $[1, s|t]$ and $[2, s|t]$ are palindromes, which is formalized as the following MI problem $\langle Cs_2, E_0, \varphi_2\rangle$.

1) Cs_2 is a singleton set of

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

2) φ_2 is a mapping to give, for any $G \subseteq \mathcal{G}_{\mathrm{u}}$, "yes" if $\{\langle s,t\rangle \mid ans(s,t) \in G\}$ is an empty set, and "no" otherwise.

By the ET computation in Table 2, we have

$$ans_{\mathrm{MI}}(Cs_2, E_0, \varphi_2) = ans_{\mathrm{MI}}(\{\}, E_0, \varphi_2) = \text{"yes"},$$

which shows that the answer to the pal-pal proof problem is "yes".

TABLE 2. Transformation for the pal-pal proof problem

	Problem representation	Rule applied
p'_1	$\{ans(A,X) \leftarrow \underline{pal([1,A\|X])}, pal([2,A\|X])\}$	repl-iff(pal)
p'_2	$\{ans(A,X) \leftarrow rv([1,A\|X],[1,A\|X]), \underline{pal([2,A\|X])}\}$	repl-iff(pal)
p'_3	$\{ans(A,X) \leftarrow \underline{rv([1,A\|X],[1,A\|X])}, rv([2,A\|X],[2,A\|X])\}$	repl-iff(rev), eq
p'_4	$\{ans(A,X) \leftarrow rv([A\|X],A1), ap(A1,[1],[1,A\|X]), \underline{rv([2,A\|X],[2,A\|X])}\}$	repl-iff(rev), eq
p'_5	$\{ans(A,X) \leftarrow \underline{rv([A,X],A1)}, ap(A1,[1],[1,A\|X]), \underline{rv([A\|X],A2)}, \underline{ap(A2,[2],[2,A\|X])}\}$	r_{rev_2}
p'_6	$\{ans(A,X) \leftarrow rv([A\|X],A1), \underline{ap(A1,[1],[1,A\|X])}, ap(A1,[2],[2,A\|X])\}$	repl-iff(app), eq
p'_7	$\{ans(A,X) \leftarrow rv([A\|X],[1\|A3]), ap(A3,[1],[A\|X]), \underline{ap([1\|A3],[2],[2,A\|X])}\}$	repl-iff(rev), eq
p'_8	$\{\}$	–

6.2.3. *ET rules for solving the "pal-pal" problem.* Referring to the set E_0 of *iff*-formulas:

$$\{\mathit{iff}(pal(x)), \mathit{iff}(rev(x,y)), \mathit{iff}(app(x,y,z))\},$$

we have three replacement rules for the *iff*-formulas in E_0. To these three rules, we add two rules, eq and r_{rev_2}, where eq is an equality-solving rule, and r_{rev_2} is the two-head rule explained in Section 6.1. Let

$$R_1 = \{\text{repl-iff}(pal), \text{repl-iff}(rev), \text{repl-iff}(app), eq, r_{rev_2}\}.$$

Then, as shown in Table 1 and Table 2, the pal-pal QA problem and the pal-pal proof problem are solved by the ET rules in R_1. Note that the resolution rule is not used in these solutions.

6.3. **Relations to conventional theories.** We show that the pair computation framework can generate the conventional SLD resolution for proof problems and the conventional SLD resolution for QA problems.

6.3.1. *The conventional SLD resolution for proof problems.* To solve a proof problem to find whether a first-order formula F_1 entails a first-order formula F_2 (i.e., $F_1 \models F_2$), we take Cs_1 as the initial clauses and limit the computation space $\langle Cs, E, \varphi\rangle$ as follows:

- $Cs_1 = \text{MPD}(\neg(\neg F_1 \vee F_2))$,
- $E = \{\}$, and
- φ is a mapping to give, for any $G \subseteq \mathcal{G}_\text{u}$, "yes" if G is an empty set, and "no" otherwise.

If $\langle Cs_1, \{\}, \varphi\rangle$ reaches, by ET, $\langle Cs', \{\}, \varphi\rangle$ such that $\mathit{Models}(Cs') = \{\}$, we can solve the proof problem affirmatively. If we take resolution and factoring as ET rules, we have the conventional SLD resolution for proof problems.

6.3.2. *The conventional SLD resolution for QA problems.* To solve a QA problem $\langle D, q\rangle$, we take Cs_2 as the initial clauses and limit the computation space $\langle Cs, E, \varphi\rangle$ as follows.

- $Cs_2 = \{ans(v_1, \ldots, v_n) \leftarrow q\}$, where ans is a predicate that does not appear in D and q, and $v_1, \ldots, v_n$ are the mutually different variables in q.
- E is a set of *iff*-formulas such that $\mathit{completion}(D) = \text{FOL}(E)$.
- φ is a mapping to find the set of all ground atoms $q\theta$ such that $ans(v_1, \ldots, v_n)\theta \in G$ and $\theta \in \mathcal{S}$ for any $G \subseteq \mathcal{G}_\text{u}$.

If $\langle Cs_2, E, \varphi\rangle$ reaches, by ET, $\langle Cs', E, \varphi\rangle$ such that each clause in Cs' has no body atoms, we can have an answer to the QA problem. If we take the replacement rules for *iff*-formulas in E, we can solve many QA problems. This solution method can be basically regarded as the conventional SLD resolution for QA problems, since (i) unfolding with respect to D applied to a body atom a in Cs gives the same transformation as the replacement rule with respect to the *iff*-formulas applied to the atom a in Cs, and (ii) unfolding transformation is similar to repeated SLD resolution.

6.4. **Solution for a class of proof problems on the pair space.** In a proof problem to show $F_1 \models F_2$, assume that F_1 is an *iff*-formula set E_1 and F_2 is a first-order formula $\neg\exists x\colon (A_1, A_2, \ldots, A_n)$. Then, $F_1 \models F_2$ is equivalent to

$$Models(\textsc{Cls}(E_1) \cup \{C_1\}) = Models(\textsc{Cls}(E_1)),$$

where $C_1 = (ans(v_1, \ldots, v_n) \leftarrow A_1, A_2, \ldots, A_n)$ such that ans is a predicate that does not appear in E_1 and $A_1, A_2, \ldots, A_n$, and $v_1, \ldots, v_n$ are the mutually different variables in $A_1, A_2, \ldots, A_n$.

Let Cs_1 be the initial clauses and limit the computation space $\langle Cs, E, \varphi\rangle$ as follows:

- $Cs_1 = \{C_1\}$,
- $E = E_1$, and
- φ is a mapping to give, for any $G \subseteq \mathcal{G}_{\mathrm{u}}$, "yes" if the set of all ground atoms $ans(v_1, \ldots, v_n)\theta$ such that $ans(v_1, \ldots, v_n)\theta \in G$ and $\theta \in \mathcal{S}$ is an empty set, and "no" otherwise.

If $\langle Cs_1, E_1, \varphi\rangle$ reaches, by ET, $\langle\{\}, E_1, \varphi\rangle$, we can solve the proof problem affirmatively.

6.5. **Utility of ET computation and *iff*-rules.** The pal-pal proof problem and the pal-pal QA problem cannot be solved by SLD resolution, which was shown in [26, 27]. Hence, the SLD resolution for proof problems and the SLD resolution for QA problems are not complete. To solve more problems correctly, we need to introduce ET rules other than those the inference-based computation supplies. Due to the limited rules and smaller computation space, computation in the SLD resolution for proof problems and that in the SLD resolution for QA problems are considered as proper subsets of the ET computation in the pair space (Section 6.3).

We explained, in Section 6.2, that using *iff*-replacement rules and a two-head rule, these two pal-pal problems can be solved correctly by the ET-based computation on the pair space. ET computation on the pair space overcomes the limitations of the conventional SLD resolution.

The pal-pal proof problem and the pal-pal QA problem can be represented mainly by the set of the three *iff*-formulas. Hence, it is very natural to apply *iff*-rules to solving the problems. The replacement rule for an *iff*-formula is applicable at any arbitrary atom into which its head can be instantiated. Compared to an increase of the number of clauses in an application of the resolution rule, *iff*-replacement rules are useful to possibly keep or reduce the number of clauses.

The conventional approach consists of the SLD resolution for proof problems and the SLD resolution for QA problems. However, we believe that it is not appropriate to regard "SLD resolution" as the common foundation. In the conventional solution of QA problems in Section 6.3, "SLD resolution" is similar to unfolding with respect to a set D of definite clauses, while in the pair space, the QA problem is formulated by an *iff*-formula set E, and D and E are different formula sets such that $completion(D) = \textsc{Fol}(E)$.

7. **Conclusions.** MI problems constitute one of the largest classes of logical problems and are of fundamental importance. The computation space has been extended from the usual clause space to an extended clause space with function variables, increasing the solvability of MI problems. This paper has further extended the power of representation and computation of MI problems by introduction of *iff*-formulas. We introduced the pair space consisting of pairs of clause sets and *iff*-formula sets based on the extended clause space, where rich representation and computation power can be obtained. We added ET rules dealing with *iff*-formulas. The correctness of the resulting solution method is guaranteed based on the ET principle. The proposed computation framework is not only an integration of SLD resolution for proof problems and QA problems, but also an extension of them, overcoming the limitation of SLD resolution by invention of more ET rules, such as multi-head ET rules.

By SLD resolution, we can only solve proof problems on clauses (under the standard semantics), and can only solve QA problems on definite clauses (under the closed-world semantics). We can formalize and solve all logical problems (MI problems) on the pair space (under the standard semantics). This extension is indispensable for future development of a theory of logical problem solving.

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] H. Aït-Kaci and R. Nasr, LOGIN: A logic programming language with built-in inheritance, *J. Logic Programming*, vol.3, pp.185-215, 1986.
[3] M. Fitting, *First-Order Logic and Automated Theorem Proving*, 2nd Edition, Springer-Verlag, 1996.
[4] J. W. Lloyd, *Foundations of Logic Programming*, 2nd Edition, Springer-Verlag, 1987.
[5] D. W. Loveland, *Automated Theorem Proving: A Logical Basis*, North-Holland Publishing, 1978.
[6] B. Motik, U. Sattler and R. Studer, Query answering for OWL-DL with rules, *Journal of Web Semantics*, vol.3, no.1, pp.41-60, 2005.
[7] A. Garcez, T. R. Besold, L. D. Raedt, P. Földiák, P. Hitzler, T. Icard, K.-U. Kühnberger, L. Lamb, R. Miikkulainen and D. Silver, Neural-symbolic learning and reasoning: Contributions and challenges, *Proc. of the AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches*, Stanford, 2015.
[8] R. B. Palm, U. Paquet and O. Winther, Recurrent relational networks, *arXiv Preprint*, arXiv: 1711.08028, 2017.
[9] F. Yang, Z. Yang and W. W. Cohen, Differentiable learning of logical rules for knowledge base reasoning, *Advances in Neural Information Processing Systems*, pp.2319-2328, 2017.
[10] N. Cingillioglu and A. Russo, DeepLogic: End-to-end logical reasoning, *arXiv Preprint*, arXiv: 1805.07433, 2018.
[11] W.-Z. Dai, Q.-L. Xu, Y. Yu and Z.-H. Zhou, Tunneling neural perception and logic reasoning through abductive learning, *arXiv Preprint*, arXiv: 1802.01173, 2018.
[12] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester and L. De Raedt, DeepProbLog: Neural probabilistic logic programming, *Advances in Neural Information Processing Systems*, pp.3749-3759, 2018.
[13] G. Sourek, V. Aschenbrenner, F. Zelezny, S. Schockaert and O. Kuzelka, Lifted relational neural networks: Efficient learning of latent relational structures, *Journal of Artificial Intelligence Research*, vol.62, pp.69-100, 2018.

[14] J. Xu, Z. Zhang, T. Friedman, Y. Liang and G. V. den Broeck, A semantic loss function for deep learning with symbolic knowledge, *Proc. of the 35th International Conference on Machine Learning*, Stockholm, Sweden, pp.5502-5511, http://proceedings.mlr.press/v80/xu18h.html, 2018.

[15] Z. Hu, X. Ma, Z. Liu, E. Hovy and E. Xing, Harnessing deep neural networks with logic rules, *Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, pp.2410-2420, 2016.

[16] D. Selsam, M. Lamm, B. Bunz, P. Liang, L. de Moura and D. L. Dill, Learning a SAT solver from single-bit supervision, *arXiv Preprint*, arXiv: 1802.03685, 2018.

[17] 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.

[18] 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.

[19] K. Akama and E. Nantajeewarawat, Solving query-answering problems with if-and-only-if formulas, *Proc. of the International Conference on Knowledge Engineering and Ontology Development*, Rome, Italy, pp.333-344, 2014.

[20] K. Akama and E. Nantajeewarawat, Proving theorems based on equivalent transformation using resolution and factoring, *Proc. of the 2nd World Congress on Information and Communication Technologies*, Trivandrum, India, pp.7-12, 2012.

[21] K. Akama, E. Nantajeewarawat and H. Koike, Solving query-answering problems for the semantic web using equivalent transformation, *Proc. of the 2nd International Conference on Software and Computer Applications*, Paris, France, vol.1, no.3, pp.249-253, 2013.

[22] K. Akama, E. Nantajeewarawat and H. Koike, Program generation in the equivalent transformation computation model using the squeeze method, in *Perspectives of Systems Informatics. PSI 2006. Lecture Notes in Computer Science*, I. Virbitskaite and A. Voronkov (eds.), Berlin, Heidelberg, Springer, 2007.

[23] K. Akama and E. Nantajeewarawat, Meaning-preserving skolemization, *Proc. of the 3rd International Conference on Knowledge Engineering and Ontology Development*, Paris, France, pp.322-327, 2011.

[24] 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.

[25] K. Akama and E. Nantajeewarawat, Model-intersection problems and their solution schema based on equivalent transformation, *Communications in Computer and Information Science*, vol.631, pp.1-12, 2016.

[26] 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.

[27] K. Akama and E. Nantajeewarawat, Solving query-answering problems based-on equivalent transformation, *International Journal of Innovative Computing, Information and Control*, vol.18, no.5, pp.1547-1558, 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.