International Journal of Innovative Computing, Information and Control
Volume **17**, Number **4**, August **2021**

ICIC International ©2021 ISSN 1349-4198
pp. 1103–1117

FORMALIZATION OF LOGICAL PROBLEMS AS MODEL-INTERSECTION PROBLEMS ON AN EXTENDED CLAUSE SPACE

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 January 2021; revised April 2021

Abstract. *Proof problems have long been the main target for logical problem solving. A problem in this class is a "yes/no" problem concerning checking whether one logical formula is a logical consequence of another logical formula. Meanwhile, the importance of another class of problems, query-answering problems (QA problems), has been increasingly recognized. A QA problem is an "all-answers finding" problem concerning finding all ground instances of a query atomic formula that are logical consequences of a given logical formula. In this paper, we introduce a model-intersection problem (MI problem). The set of all MI problems constitutes a very large class of logical problems. We give a general conversion method to map all proof problems and all QA problems on first-order logic into MI problems, which enables us to solve them by equivalent transformation rules with full guarantee of correctness. Formalization as 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.*

Keywords: Model-intersection problem, Extended clause, Function variable, Equivalent transformation

1. Introduction.

Recently, the deep learning community has given increasing attention to integration of logical or symbolic reasoning into neural network architectures [1-10]. Some systems integrate inductive logic programming into neural networks [11]. Such integration usually uses rather small solvers, e.g., SAT or MAXSAT, which are realized in deep learning architectures. It does not mean that deep learning is of conceptually central structure of human intelligence over logical structure. We believe that logical structure should be of fundamental and central importance for human intelligence. However, the current theory of logic is still unsatisfactory. The theory of logic should be reconstructed extensively, seeking appropriate structure to capture human intelligence and to maximize representational and computational power. The conventional theory of logic is proof-centered and inference-based one on the first-order formula space. This paper changes the structure of logic by extending problem classes, computation methods, and the underlying formula space.

DOI: 10.24507/ijicic.17.04.1103

Proof problems are historically the most important problem class in logical problem solving. A resolution-based proof procedure for logical formulas was invented [12-15]. Based on the resolution proof method, automated theorem proving has been extensively investigated [16-20]. A proof problem is a "yes/no" problem; it is concerned with checking whether or not one given logical formula entails another given logical formula. Formally, a proof problem is a pair $\langle E_1, E_2 \rangle$, where E_1 and E_2 are first-order formulas. The answer to this problem is defined to be "yes", if E_2 is a logical consequence of E_1, and "no" otherwise. A proof problem $\langle E_1, E_2 \rangle$ is solved [12, 13] by (i) constructing the formula $E = (E_1 \wedge \neg E_2)$, since the unsatisfiability of E means that the answer of this proof problem is "yes", (ii) conversion of E into a set Cs of clauses using the conventional Skolemization [12, 15], (iii) transformation of the clause set Cs by the resolution and factoring inference rules, and (iv) determining the answer by checking whether an empty clause can be obtained, i.e., if an empty clause is obtained, then Cs is unsatisfiable and the answer to the proof problem is "yes". This solution relies on the preservation of satisfiability. The conversion of E into Cs using the conventional Skolemization preserves the satisfiability of E. Transformation of Cs by using resolution and factoring also preserves the satisfiability of Cs.

Being inspired by the resolution proof method, resolution-based solutions to *query-answering problems* (*QA problems*) were investigated [21-24]. A QA problem on clauses is a pair $\langle Cs, a \rangle$, where Cs is a set of clauses and a is a user-defined query atom. The answer to a QA problem $\langle Cs, a \rangle$ is defined as the set of all ground instances of a that are logical consequences of Cs. 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.

SLD resolution has been a main method for solving proof problems and QA problems. However, many logical problems, e.g., pal-pal proof/QA problems [25] and Agatha proof/QA problems [26], cannot be solved by SLD resolution. SLD resolution is incomplete both for the class of all proof problems that are defined by using first-order formulas, 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. Integration of solution methods for proof problems and QA problems by SLD resolution is not successful.

This paper introduces a *model-intersection problem* (*MI problem*). The set of all MI problems constitutes a very large class of logical problems. MI problems integrate proof problems and QA problems. We give a general conversion method to map all proof problems and all QA problems on first-order logic into MI problems, which enables us to solve them by using equivalent transformation (ET) rules with full guarantee of correctness. Formalization as 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.

Assume that we consider logical problems inside first-order logic. An MI problem is a pair $\langle Cs, \varphi \rangle$, where Cs is a set of clauses and φ is a mapping, called an *extraction mapping*, used for constructing the output answer from the intersection of all models of Cs. More formally, the answer to an MI problem $\langle Cs, \varphi \rangle$ is $\varphi(\bigcap \mathit{Models}(Cs))$, where $\mathit{Models}(Cs)$ is the set of all models of Cs and $\bigcap \mathit{Models}(Cs)$ is the intersection of all elements of $\mathit{Models}(Cs)$. Note that, in this theory, an interpretation is a set of ground user-defined atoms, which is similar to a Herbrand interpretation [12, 15]. Since each element of $\mathit{Models}(Cs)$ is a set of ground user-defined atoms, we can take the intersection of all elements of it.

An MI problem is defined on a space. MI problems defined above on first-order logic (the usual clause space) cannot attain completeness, i.e., many logical problems cannot be solved correctly within first-order logic (the usual clause space) without built-in constraint atoms. To formalize logical problems, we need to use first-order formulas that possibly include built-in constraint atoms. The set of all such formulas is denoted by $\mathrm{FOL_c}$. Built-in constraint atoms play a crucial role in knowledge representation and are essential for practical applications. The classical theorem-proving theory motivates us to transform proof problems and QA problems on $\mathrm{FOL_c}$ into MI problems on clauses by the conventional Skolemization [12, 15]. However, satisfiability preservation of a formula does not generally hold for formulas in $\mathrm{FOL_c}$ [27, 28]. The conventional Skolemization, therefore, does not provide a transformation process towards correct solutions to proof problems and QA problems on $\mathrm{FOL_c}$.

Meaning-preserving decomposition (MPD) was invented [28] to overcome the difficulties caused by the conventional Skolemization. MPD preserves the logical meanings of first-order formulas (and, thus, also preserves their satisfiability) even when they include built-in constraint atoms. Conventional clauses should be extended in order that all first-order formulas in $\mathrm{FOL_c}$ can be equivalently converted by MPD. An extended clause may contain function variables and atoms of a special kind called *func*-atoms. The set of all extended clauses is called $\mathrm{ECLS_F}$.

An MI problem proposed in this paper is on $\mathrm{ECLS_F}$. It is a pair $\langle Cs, \varphi\rangle$, where Cs is a set of extended clauses and φ is an extraction mapping. The set of all MI problems on $\mathrm{ECLS_F}$ constitutes a very large class of problems and is of great importance. As shown in Section 4, all proof problems and all QA problems on $\mathrm{FOL_c}$ are mapped, preserving the answers to them, into MI problems on $\mathrm{ECLS_F}$. By solving MI problems on $\mathrm{ECLS_F}$, we solve proof problems and QA problems on $\mathrm{FOL_c}$. MI problems on $\mathrm{ECLS_F}$ are solved by equivalent transformation (ET), where problems are simplified by repeated problem transformation by application of ET rules. The class of MI problems proposed in this paper is the largest and the first one that enables structural embedding of the full class of proof problems on $\mathrm{FOL_c}$ and the full class of QA problems on $\mathrm{FOL_c}$.

The main advantages of our work are the extension of a class of logical problems and computation by many ET rules.

- Formalization: MI problems in this paper form a superset of proof problems and QA problems on first-order logic.
- Computation: Various ET rules can be used, which can solve more problems.

Existing techniques are applicable only to existing classes of problems. Extension to MI problems on the extended space increases the solvability of logical problems. There is no existing technique that can be generally applicable to this part of extension, which includes mapping problems into MI problems and dealing with function variables.

The rest of the paper is organized as follows. Section 2 defines extended clauses and $\mathrm{ECLS_F}$. Section 3 formalizes MI problems on extended clauses. Section 4 describes how proof problems and QA problems can be converted into MI problems. Section 5 demonstrates formalization of and a solution to a logical problem, called the "may-do-thesis" problem. Section 6 concludes the paper.

2. An Extended Clause Space.

We introduce an extended clause that may include universally quantified usual variables and existentially quantified function variables. The set of all extended clauses forms a space, on which MI problems are constructed.

2.1. User-defined atoms, built-in constraint atoms, and *func*-atoms.

An extended formula space is introduced, which 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. Supposing that *teach*, *St*, and *FM* are user-defined predicates, *teach*(*john*, *ai*), *St*(*paul*), and *FM*(x) are user-defined atoms (cf. Figure 1 in Section 5). 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. Typical examples of built-in constraint atoms are *eq*(x, x) and *neq*($1, 2$), where *eq* and *neq* are predefined constraint predicates that stand for "equal" and "not equal", respectively. (No built-in constraint atom appears in Figure 1.) Let $\mathcal{A}_\mathrm{u}$ be the set of all user-defined atoms, $\mathcal{G}_\mathrm{u}$ the set of all ground user-defined atoms, $\mathcal{A}_\mathrm{c}$ the set of all constraint atoms, and $\mathcal{G}_\mathrm{c}$ the set of all ground constraint atoms.

A *func-atom* [28, 29] is an expression of the form $\mathit{func}(h, t_1, \ldots, t_n, t_{n+1})$, where h is either an n-ary function constant or an n-ary function variable, and the t_i are usual terms. For example, supposing that h_0 is a unary function variable, $\mathit{func}(h_0, x, y)$ is a *func*-atom (cf. the clauses C_{24} and C_{25} in Figure 1). A *func*-atom $\mathit{func}(h, t_1, \ldots, t_n, t_{n+1})$ is *ground* if h is a function constant and the t_i are ground usual terms.

There are two types of variables: usual variables and function variables. (In Figure 1, x, y, z, and w are usual variables, while h_0 is a function variable.) A function variable is instantiated into a function constant or a function variable, but not into a usual term. Let *FVar* be the set of all function variables and *FCon* the set of all function constants. A substitution for function variables is a mapping from *FVar* to $\mathit{FVar} \cup \mathit{FCon}$. Each n-ary function constant is associated with a mapping from $\mathcal{G}_\mathrm{t}^n$ to $\mathcal{G}_\mathrm{t}$, where $\mathcal{G}_\mathrm{t}$ denotes the set of all ground usual terms.

2.2. **Extended clauses.**

User-defined atoms and built-in constraint atoms are used in usual clauses, which are extended by allowing *func*-atoms to appear in their right-hand sides. An *extended clause* 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 or a built-in constraint atom, $\mathbf{f}_1$, $\ldots$, $\mathbf{f}_p$ are *func*-atoms, and m, n, and p are non-negative integers. 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. Let *userLhs*(C) denote the number of user-defined atoms in the left-hand side of C. When $\mathit{userLhs}(C) = 0$, C is called a *negative extended clause*. When $\mathit{userLhs}(C) = 1$, C is called an *extended definite clause*. When $\mathit{userLhs}(C) > 1$, 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.

A conjunction of a finite or infinite number of extended clauses is used for knowledge representation and also for computation. As usual, such a conjunction is usually dealt with by regarding it as a set of (extended) clauses. The set of all extended clauses is denoted by ECLS_F.

Let *Cs* be a set of extended clauses. Implicit existential quantifications of function variables and implicit clause conjunction are assumed in *Cs*. Function variables in *Cs* are all existentially quantified and their scope covers all clauses in *Cs*. With occurrences of function variables, clauses in *Cs* are connected through shared function variables. After instantiating all function variables in *Cs* into function constants, clauses in the instantiated set are totally separated.

2.3. **Interpretations and models.**

A state of the world is represented by a set of true ground atoms in $\mathcal{G}_\mathrm{u}$. A logical formula is used to impose a constraint on possible states of the world. Hence, an *interpretation* is a subset of $\mathcal{G}_{\mathrm{u}}$. A ground user-defined atom g 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 TCON denote the set of all true ground constraint atoms, i.e., a ground constraint atom g is true iff $g \in$ 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 $C = (a_1, \ldots, a_m \leftarrow b_1, \ldots, b_n, \mathbf{f}_1, \ldots, \mathbf{f}_p) \in \mathrm{ECLS_F}$ 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 $a_i \in I \cup$ TCON.
2) There exists $i \in \{1, \ldots, n\}$ such that $b_i \notin I \cup$ TCON.
3) There exists $i \in \{1, \ldots, p\}$ such that $\mathbf{f}_i$ is false.

An interpretation I is a *model* of a clause set $Cs \subseteq \mathrm{ECLS_F}$ 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.

Let *Models* be a mapping that associates with each clause set the set of all of its models, i.e., $Models(Cs)$ is the set of all models of Cs for any $Cs \subseteq \mathrm{ECLS_F}$.

The standard semantics is taken in this theory in the sense that all models of a formula are considered instead of specific ones, such as those considered in the minimal model semantics [21, 30], which underlies definite logic programming, and in the stable model semantics [31, 32], which underlies answer set programming.

3. Model-Intersection Problems on the Extended Clause Space.

A *model-intersection problem* (*MI problem*) on $\mathrm{ECLS_F}$ is a pair $\langle Cs, \varphi \rangle$, where $Cs \subseteq \mathrm{ECLS_F}$ and φ is a mapping from the power set of $\mathcal{G}_{\mathrm{u}}$ to some set W. The mapping φ is called an *extraction mapping*. The answer to this problem, denoted by $ans_{\mathrm{MI}}(Cs, \varphi)$, is defined by

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

where $\bigcap Models(Cs)$ is the intersection of all models of Cs. Note that when $Models(Cs)$ is the empty set, $\bigcap Models(Cs) = \mathcal{G}_{\mathrm{u}}$.

Examples illustrating how logical problems are formalized as MI problems on $\mathrm{ECLS_F}$ are given below. Example 3.1 includes a multi-head clause; Example 3.2 contains definite clauses; Example 3.3 includes definite clauses with *func*-atoms; and Example 3.4 contains multi-head clauses with *func*-atoms and negative clauses. Existence of multi-head clauses, negative clauses, and *func*-atoms makes it more difficult to solve problems. Except for Example 3.2, the conventional resolution-based method cannot generally solve all these problems with guarantee of correctness.

Example 3.1. *Consider the Oedipus puzzle described in* [33]. *Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Polyneikes also had children, among them Thersandros, who is not a patricide. The problem is to find all persons who have a patricide child who has a non-patricide child.*

Assume that (i) "oe", "io", "po" and "th" stand, respectively, for Oedipus, Iokaste, Polyneikes and Thersandros, (ii) for any terms t_1 and t_2, $isCh(t_1, t_2)$ denotes "t_1 is a child of t_2", and (iii) for any term t, $pat(t)$ denotes "t is a patricide" and $prob(t)$ denotes "t is an answer to this puzzle". To formalize this puzzle, let Cs_1 consist of the following seven clauses:

$isCh(oe, io) \leftarrow$ $isCh(po, io) \leftarrow$ $isCh(po, oe) \leftarrow$
$isCh(th, po) \leftarrow$ $pat(oe) \leftarrow$ $\leftarrow pat(th)$
$prob(x), pat(y) \leftarrow isCh(z, x), pat(z), isCh(y, z)$

Let φ_1 be defined by $\varphi_1(G) = \{x \mid prob(x) \in G\}$ for any $G \subseteq \mathcal{G}_u$. Then $\langle Cs_1, \varphi_1\rangle$ is an MI problem representing this puzzle.

Example 3.2. *Consider a problem of finding all lists obtained by concatenating the list* $[1, 2, 3]$ *with the list* $[4, 5]$*. Let Cs_2 consist of the following clauses:*

$app([\,], x, x) \leftarrow$
$app([w|x], y, [w|z]) \leftarrow app(x, y, z)$
$ans(x) \leftarrow app([1, 2, 3], [4, 5], x)$

Let φ_2 be defined by $\varphi_2(G) = \{x \mid ans(x) \in G\}$ for any $G \subseteq \mathcal{G}_u$. This problem is then formalized as the MI problem $\langle Cs_2, \varphi_2\rangle$.

Example 3.3. *Consider the "tax-cut" problem discussed in* [34]*. This problem is to find all persons who can have discounted tax, with the knowledge consisting of the following statements. (i) Any person who has two children or more can get discounted tax. (ii) Men and women are not the same. (iii) It is false that a person is not the same as himself/herself. (iv) A person's mother is always a woman. (v) Peter has a child, who is someone's mother. (vi) Peter has a child named Paul. (vii) Paul is a man. These statements are represented by the following eight extended clauses:*

$TaxCut(x) \leftarrow hasChild(x, y), hasChild(x, z), notSame(y, z)$
$notSame(x, y) \leftarrow Man(x), Woman(y)$
$\leftarrow notSame(x, x)$
$Woman(x) \leftarrow motherOf(x, y)$
$hasChild(Peter, x) \leftarrow func(h_1, x)$
$motherOf(x, y) \leftarrow func(h_1, x), func(h_2, y)$
$hasChild(Peter, Paul) \leftarrow$
$Man(Paul) \leftarrow$

The fifth and the sixth clauses together represent the fifth statement (i.e., "Peter has a child, who is someone's mother"), where h_1 and h_2 are 0-ary function variables. Let Cs_3 consist of the above eight clauses. Let φ_3 be defined by $\varphi_3(G) = \{x \mid TaxCut(x) \in G\}$ for any $G \subseteq \mathcal{G}_u$. The "tax-cut" problem is then formulated as the MI problem $\langle Cs_3, \varphi_3\rangle$.

Example 3.4. *Consider the "Dreadsbury Mansion Mystery" problem* [17]*, which was given by Len Schubelt and can be described as follows: Someone who lives in Dreadsbury Mansion killed Aunt Agatha. Agatha, the butler, and Charles live in Dreadsbury Mansion, and are the only people who live therein. A killer always hates his victim, and is never richer than his victim. Charles hates no one that Aunt Agatha hates. Agatha hates everyone except the butler. The butler hates everyone not richer than Aunt Agatha. The butler hates everyone Agatha hates. No one hates everyone. The problem is to find who is the killer.*

Assume that neq is a predefined binary constraint predicate and for any ground usual terms t_1 and t_2, $neq(t_1, t_2)$ is true iff $t_1 \neq t_2$. The background knowledge of this mystery is formalized as a set Cs_4 consisting of the following clauses, where the constants A, B, C, and D denote "Agatha", "the butler", "Charles", and "Dreadsbury Mansion", respectively, h_0 is a 0-ary function variable, and h_1 is a unary function variable:

$live(x, D) \leftarrow func(h_0, x)$
$kill(x, A) \leftarrow func(h_0, x)$
$\leftarrow live(x, D), neq(x, A), neq(x, B), neq(x, C)$

$live(A, D) \leftarrow$
$live(B, D) \leftarrow$
$live(C, D) \leftarrow$
$hate(x, y) \leftarrow kill(x, y)$
$\leftarrow kill(x, y), richer(x, y)$
$\leftarrow hate(A, x), hate(C, x), live(x, D)$
$hate(A, x) \leftarrow neq(x, B), live(x, D)$
$richer(x, A), hate(B, x) \leftarrow$
$hate(B, x) \leftarrow hate(A, x)$
$\leftarrow hate(x, y), func(h_1, x, y), live(x, D)$
$live(y, D) \leftarrow live(x, D), func(h_1, x, y)$
$killer(x) \leftarrow kill(x, A)$

Let φ_4 be defined by $\varphi_4(G) = \{x \mid killer(x) \in G\}$ for any $G \subseteq \mathcal{G}_{\rm u}$. This problem is then represented as the MI problem $\langle Cs_4, \varphi_4\rangle$.

Example 3.5. *Let an extraction mapping $\varphi_{\rm pr}$ be given as follows: For any $G \subseteq \mathcal{G}_{\rm u}$, $\varphi_{\rm pr}(G) =$ "yes" if $G = \mathcal{G}_{\rm u}$, and $\varphi_{\rm pr}(G) =$ "no" otherwise. Referring to the clause sets Cs_1-Cs_4 in Examples 3.1-3.4, we illustrate that proof problems can be represented as MI problems as follows:*

- *Letting $Cs_5 = Cs_1 \cup \{(\leftarrow prob(io))\}$, the MI problem $\langle Cs_5, \varphi_{\rm pr}\rangle$ represents the problem of proving whether $prob(io)$ is true.*
- *Letting $Cs_6 = Cs_2 \cup \{(\leftarrow ans([1, 2, 3, 4, 5]))\}$, the MI problem $\langle Cs_6, \varphi_{\rm pr}\rangle$ represents the problem of proving whether the resulting list is $[1, 2, 3, 4, 5]$.*
- *Letting $Cs_7 = Cs_3 \cup \{(\leftarrow TaxCut(x))\}$, the MI problem $\langle Cs_7, \varphi_{\rm pr}\rangle$ represents the problem of proving whether someone gets discounted tax.*
- *Letting $Cs_8 = Cs_4 \cup \{(\leftarrow killer(A))\}$, the MI problem $\langle Cs_8, \varphi_{\rm pr}\rangle$ represents the problem of proving whether Agatha killed herself.*

4. Conversion of Logical Problems into MI Problems.

We propose procedures for converting QA problems and proof problems into MI problems, thereby showing the generality of the class of MI problems, i.e., QA problems and proof problems can be integrated into MI problems.

4.1. Conversion of QA problems into MI problems.

A *query-answering problem* (*QA problem*) on FOL$_{\rm c}$ is a pair $\langle E, a\rangle$, where E is a closed first-order formula in FOL$_{\rm c}$ and a is a user-defined atom in $\mathcal{A}_{\rm u}$. Let $\mathcal{S}$ be the set of all substitutions for usual variables. The answer to a QA problem $\langle E, a\rangle$, denoted by $ans_{\rm QA}(E, a)$, is defined by

$$ans_{\rm QA}(E, a) = \{a\theta \mid (\theta \in \mathcal{S}) \;\&\; (a\theta \in \mathcal{G}_{\rm u}) \;\&\; (E \models a\theta)\}.$$

In logic programming [21], a problem represented by a pair of a set of definite clauses and a query atom has been intensively discussed. In the description logic (DL) community [33], a class of problems formulated as conjunctions of DL-based axioms and assertions together with query atoms has been discussed [35]. These two problem classes can be formalized as subclasses of QA problems considered in this paper.

Theorem 4.1 shows that a QA problem on FOL$_{\rm c}$ can be converted towards an MI problem on FOL$_{\rm c}$.

Theorem 4.1. *For any closed first-order formula $E \in FOL_{\rm c}$ and any $a \in \mathcal{A}_{\rm u}$,*

$$ans_{\rm QA}(E, a) = \textsc{Inst}(a) \cap \left(\bigcap Models(E)\right),$$

where $\textsc{Inst}(a)$ denotes the set of all ground instances of a.

Proof: Let E be a closed first-order formula in $\mathrm{FOL_c}$ and $a \in \mathcal{A}_\mathrm{u}$. By the definition of $\models$, for any ground atom $g \in \mathcal{G}_\mathrm{u}$, $E \models g$ iff $g \in \bigcap \mathit{Models}(E)$. Then

$$\begin{aligned} \mathit{ans}_\mathrm{QA}(E,a) &= \{a\theta \mid (\theta \in \mathcal{S}) \;\&\; (a\theta \in \mathcal{G}_\mathrm{u}) \;\&\; (E \models a\theta)\} \\ &= \{g \mid (\theta \in \mathcal{S}) \;\&\; (g = a\theta) \;\&\; (g \in \mathcal{G}_\mathrm{u}) \;\&\; (E \models g)\} \\ &= \{g \mid (g \in \textsc{Inst}(a)) \;\&\; (E \models g)\} \\ &= \{g \mid (g \in \textsc{Inst}(a)) \;\&\; (g \in \bigcap \mathit{Models}(E))\} \\ &= \textsc{Inst}(a) \cap (\bigcap \mathit{Models}(E)). \qquad \square \end{aligned}$$

Theorem 4.2 (with the help of Theorem 4.1) shows that a QA problem on $\mathrm{FOL_c}$ can be converted into an MI problem on $\mathrm{ECLS_F}$.

Theorem 4.2. *Let E be a first-order formula in $\mathrm{FOL_c}$ and $a \in \mathcal{A}_\mathrm{u}$. Let $Cs \subseteq \mathrm{ECLS_F}$. If $\mathit{Models}(E) = \mathit{Models}(Cs)$, then*

$$\mathit{ans}_\mathrm{QA}(E,a) = \mathit{ans}_\mathrm{MI}(Cs \cup \{(p(x_1,\ldots,x_n) \leftarrow a)\}, \varphi_\mathrm{qa}),$$

where p is a predicate that appears in neither Cs nor a, the arguments $x_1,\ldots,x_n$ are all the mutually different variables occurring in a, and for any $G \subseteq \mathcal{G}_\mathrm{u}$,

$$\varphi_\mathrm{qa}(G) = \{a\theta \mid (\theta \in \mathcal{S}) \;\&\; (p(x_1,\ldots,x_n)\theta \in G)\}.$$

Proof: Assume that $\mathit{Models}(E) = \mathit{Models}(Cs)$. Then

$$\begin{aligned} \mathit{ans}_\mathrm{QA}(E,a) &= \text{(by Theorem 4.1)} \\ &= \textsc{Inst}(a) \cap (\bigcap \mathit{Models}(E)) \\ &= \textsc{Inst}(a) \cap (\bigcap \mathit{Models}(Cs)) \\ &= \text{(by the definition of } \varphi_\mathrm{qa}) \\ &= \varphi_\mathrm{qa}(\bigcap \mathit{Models}(Cs \cup \{(p(x_1,\ldots,x_n) \leftarrow a)\})) \\ &= \mathit{ans}_\mathrm{MI}(Cs \cup \{(p(x_1,\ldots,x_n) \leftarrow a)\}, \varphi_\mathrm{qa}). \qquad \square \end{aligned}$$

To satisfy the condition $\mathit{Models}(E) = \mathit{Models}(Cs)$ in Theorem 4.2, we use MPD, i.e.,

$$\mathrm{MPD}(E) = Cs.$$

Theorem 4.2 then gives a correct procedure for converting QA problems on $\mathrm{FOL_c}$ into MI problems on $\mathrm{ECLS_F}$.

4.2. Conversion of proof problems into MI problems.

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

$$\mathit{ans}_\mathrm{Pr}(E_1,E_2) = \begin{cases} \text{“yes”} & \text{if } E_1 \models E_2, \\ \text{“no”} & \text{otherwise.} \end{cases}$$

It is well known that E_2 is a logical consequence of E_1 iff $E_1 \wedge \neg E_2$ is unsatisfiable (i.e., $E_1 \wedge \neg E_2$ has no model) [12, 15]. As a result, $\mathit{ans}_\mathrm{Pr}(E_1,E_2)$ can be equivalently defined by

$$\mathit{ans}_\mathrm{Pr}(E_1,E_2) = \begin{cases} \text{“yes”} & \text{if } \mathit{Models}(E_1 \wedge \neg E_2) = \emptyset, \\ \text{“no”} & \text{otherwise.} \end{cases}$$

Theorem 4.3 below shows that a proof problem on $\mathrm{FOL_c}$ can be converted into an MI problem on $\mathrm{ECLS_F}$.

Theorem 4.3. *Let $\langle E_1, E_2\rangle$ be a proof problem, where E_1 and E_2 are first-order formulas in $\mathrm{FOL_c}$. Let $Cs \subseteq \mathrm{ECLS_F}$. Let φ_pr be a mapping from the power set of $\mathcal{G}_\mathrm{u}$ to { “yes”, “no”}* *defined by: for any* $G \subseteq \mathcal{G}_{\mathrm{u}}$,

$$\varphi_{\mathrm{pr}}(G) = \begin{cases} \text{“yes”} & \text{if } G = \mathcal{G}_{\mathrm{u}}, \\ \text{“no”} & \text{otherwise.} \end{cases}$$

If the conditions $\mathit{Models}(E_1 \wedge \neg E_2) = \emptyset$ *and* $\mathit{Models}(\mathit{Cs}) = \emptyset$ *are equivalent, then* $\mathit{ans}_{\mathrm{Pr}}(E_1, E_2) = \mathit{ans}_{\mathrm{MI}}(\mathit{Cs}, \varphi_{\mathrm{pr}})$.

Proof: Assume that $\mathit{Models}(E_1 \wedge \neg E_2) = \emptyset$ iff $\mathit{Models}(\mathit{Cs}) = \emptyset$. Let b be a ground user-defined atom that is not an instance of any user-defined atom occurring in Cs. If m is a model of Cs, then $m - \{b\}$ is also a model of Cs. Obviously, $m - \{b\} \neq \mathcal{G}_{\mathrm{u}}$. Therefore, (i) if $\mathit{Models}(\mathit{Cs}) \neq \emptyset$, then $\bigcap \mathit{Models}(\mathit{Cs}) \neq \mathcal{G}_{\mathrm{u}}$, and (ii) if $\mathit{Models}(\mathit{Cs}) = \emptyset$, then $\bigcap \mathit{Models}(\mathit{Cs}) = \bigcap \{\} = \mathcal{G}_{\mathrm{u}}$.

Two cases are considered:

1) Suppose that $\mathit{Models}(E_1 \wedge \neg E_2) = \emptyset$. Consequently, $\mathit{Models}(\mathit{Cs}) = \emptyset$. So $\bigcap \mathit{Models}(\mathit{Cs}) = \mathcal{G}_{\mathrm{u}}$, and, therefore, $\mathit{ans}_{\mathrm{MI}}(\mathit{Cs}, \varphi_{\mathrm{pr}}) =$ “yes”.
2) Suppose that $\mathit{Models}(E_1 \wedge \neg E_2) \neq \emptyset$. In this case, $\mathit{Models}(\mathit{Cs}) \neq \emptyset$, and, thus, $\bigcap \mathit{Models}(\mathit{Cs}) \neq \mathcal{G}_{\mathrm{u}}$. So $\mathit{ans}_{\mathrm{MI}}(\mathit{Cs}, \varphi_{\mathrm{pr}}) =$ “no”.

Hence $\mathit{ans}_{\mathrm{Pr}}(E_1, E_2) = \mathit{ans}_{\mathrm{MI}}(\mathit{Cs}, \varphi_{\mathrm{pr}})$. □

To satisfy that the conditions $\mathit{Models}(E_1 \wedge \neg E_2) = \emptyset$ and $\mathit{Models}(\mathit{Cs}) = \emptyset$ in Theorem 4.3 are equivalent, we use MPD, i.e.,

$$\mathrm{MPD}(E_1 \wedge \neg E_2) = \mathit{Cs}.$$

Theorem 4.3 then gives a correct procedure for converting proof problems on $\mathrm{FOL}_{\mathrm{c}}$ into MI problems on $\mathrm{ECLS}_{\mathrm{F}}$.

5. **Example: Formalization and Solution.**

We take a small example of an MI problem and give a solution to it by using equivalent transformation (ET), suggesting the usefulness of formalization by MI problems and that of the transformational approach based on unfolding and other ET rules. More examples can be found in [26, 37, 38].

5.1. **Problem description.**

The clauses in Figure 1 are obtained by applying the conversion procedure in Section 4.1 to a first-order formula representing the background knowledge (with some modification) of the “may-do-thesis” problem (for short, the *mdt* problem) given in [36]. All atoms appearing in Figure 1 belong to $\mathcal{A}_{\mathrm{u}}$. The unary predicates *NFP*, *FP*, *FM*, *Co*, *AC*, *BC*, *St*, and *Tp* denote “non-teaching full professor”, “full professor”, “faculty member”, “course”, “advanced course”, “basic course”, “student”, and “topic”, respectively. The clauses C_9-C_{11} together provide the conditions for a student to do his/her thesis with a professor, where $\mathit{mdt}(s, p)$, $\mathit{curr}(s, t)$, $\mathit{expert}(p, t)$, $\mathit{exam}(s, c)$, and $\mathit{subject}(c, t)$ are intended to mean “s may do his/her thesis with p”, “s studied t in his/her curriculum”, “p is an expert in t”, “s passed the exam of c”, and “c covers t”, respectively, for any student s, any professor p, any topic t, and any course c.

Suppose that we want to find all professors with whom *paul* may do his thesis. This problem is formulated as an MI problem $\langle \mathit{Cs}, \varphi \rangle$, where Cs consists of the clauses C_1-C_{25} in Figure 1 and φ is defined by: for any $G \subseteq \mathcal{G}_{\mathrm{u}}$,

$$\varphi(G) = \{x \mid \mathit{mdt}(\mathit{paul}, x) \in G\}.$$

How to compute the answer to this MI problem using many kinds of clause transformation rules is demonstrated in Section 5.2.

C_1: $FM(x) \leftarrow FP(x)$ C_2: $FP(john) \leftarrow$
C_3: $FP(mary) \leftarrow$ C_4: $teach(john, ai) \leftarrow$
C_5: $St(paul) \leftarrow$ C_6: $AC(ai) \leftarrow$
C_7: $Tp(kr) \leftarrow$ C_8: $Tp(lp) \leftarrow$

C_9: $curr(x, z) \leftarrow exam(x, y), subject(y, z), St(x), Co(y), Tp(z)$

C_{10}: $mdt(x, y) \leftarrow curr(x, z), expert(y, z), St(x), Tp(z), FP(y), AC(w), teach(y, w)$

C_{11}: $mdt(x, y) \leftarrow St(x), NFP(y)$

C_{12}: $exam(paul, ai) \leftarrow$ C_{13}: $subject(ai, kr) \leftarrow$
C_{14}: $subject(ai, lp) \leftarrow$ C_{15}: $expert(john, kr) \leftarrow$
C_{16}: $expert(mary, lp) \leftarrow$

C_{17}: $AC(x) \leftarrow teach(mary, x)$
C_{18}: $\leftarrow AC(x), BC(x)$
C_{19}: $AC(x), BC(x) \leftarrow Co(x)$
C_{20}: $Co(x) \leftarrow AC(x)$
C_{21}: $Co(x) \leftarrow BC(x)$
C_{22}: $FP(x) \leftarrow NFP(x)$
C_{23}: $\leftarrow NFP(x), teach(x, y), Co(y)$
C_{24}: $teach(x, y), NFP(x) \leftarrow FP(x), func(h_0, x, y)$
C_{25}: $Co(y), NFP(x) \leftarrow FP(x), func(h_0, x, y)$

FIGURE 1. Background knowledge for the *mdt* problem on $\mathrm{ECLS_F}$

5.2. **ET computation.** The clause set *Cs* consisting of C_1-C_{25} given in Section 5.1 (Figure 1) is transformed as follows:

- By (i) unfolding using the definitions of the predicates *FP*, *Tp*, *curr*, *subject*, *expert*, *St*, and *exam*, (ii) removing these definitions along with the definition of *FM* using definite-clause removal, (iii) removal of valid clauses, and (iv) removal of subsumed clauses, the clauses C_1-C_{25} are transformed into the clauses C_{26}-C_{40} in Figure 2.

C_{26}: $teach(john, ai) \leftarrow$
C_{27}: $AC(ai) \leftarrow$
C_{28}: $AC(x) \leftarrow teach(mary, x)$
C_{29}: $\leftarrow AC(x), BC(x)$
C_{30}: $AC(x), BC(x) \leftarrow Co(x)$
C_{31}: $Co(x) \leftarrow AC(x)$
C_{32}: $Co(x) \leftarrow BC(x)$
C_{33}: $\leftarrow NFP(x), teach(x, y), Co(y)$
C_{34}: $mdt(paul, mary) \leftarrow AC(x), teach(mary, x), Co(ai)$
C_{35}: $mdt(paul, john) \leftarrow AC(x), teach(john, x), Co(ai)$
C_{36}: $mdt(paul, x) \leftarrow NFP(x)$
C_{37}: $teach(mary, x), NFP(mary) \leftarrow func(h_0, mary, x)$
C_{38}: $teach(john, x), NFP(john) \leftarrow func(h_0, john, x)$
C_{39}: $Co(x), NFP(mary) \leftarrow func(h_0, mary, x)$
C_{40}: $Co(x), NFP(john) \leftarrow func(h_0, john, x)$

FIGURE 2. Clauses obtained by application of ET rules

- Side-change transformation for *NFP* enables (i) unfolding using the definition of *Co*, (ii) elimination of this definition using definite-clause removal, and (iii) removal of valid clauses. By such side-change transformation followed by transformation of these three types, C_{26}-C_{40} are transformed into the clauses C_{41}-C_{61} in Figure 3.

C_{41}: $teach(john, ai) \leftarrow$
C_{42}: $AC(ai) \leftarrow$
C_{43}: $AC(x) \leftarrow teach(mary, x)$
C_{44}: $\leftarrow AC(x), BC(x)$
C_{45}: $mdt(paul, mary) \leftarrow AC(x), teach(mary, x), func(h_0, mary, ai), notNFP(mary)$
C_{46}: $mdt(paul, mary) \leftarrow AC(x), teach(mary, x), func(h_0, john, ai), notNFP(john)$
C_{47}: $mdt(paul, mary) \leftarrow AC(x), teach(mary, x), BC(ai)$
C_{48}: $mdt(paul, mary) \leftarrow AC(x), teach(mary, x), AC(ai)$
C_{49}: $mdt(paul, john) \leftarrow AC(x), teach(john, x), func(h_0, mary, ai), notNFP(mary)$
C_{50}: $mdt(paul, john) \leftarrow AC(x), teach(john, x), func(h_0, john, ai), notNFP(john)$
C_{51}: $mdt(paul, john) \leftarrow AC(x), teach(john, x), BC(ai)$
C_{52}: $mdt(paul, john) \leftarrow AC(x), teach(john, x), AC(ai)$
C_{53}: $mdt(paul, x), notNFP(x) \leftarrow$
C_{54}: $teach(mary, x) \leftarrow func(h_0, mary, x), notNFP(mary)$
C_{55}: $teach(john, x) \leftarrow func(h_0, john, x), notNFP(john)$
C_{56}: $notNFP(x) \leftarrow teach(x, y), func(h_0, mary, y), notNFP(mary)$
C_{57}: $notNFP(x) \leftarrow teach(x, y), func(h_0, john, y), notNFP(john)$
C_{58}: $notNFP(x) \leftarrow teach(x, y), BC(y)$
C_{59}: $notNFP(x) \leftarrow teach(x, y), AC(y)$
C_{60}: $AC(x), BC(x) \leftarrow func(h_0, mary, x), notNFP(mary)$
C_{61}: $AC(x), BC(x) \leftarrow func(h_0, john, x), notNFP(john)$

FIGURE 3. Clauses obtained by application of ET rules

- Side-change transformation for *BC* enables unfolding using the definition of *AC*. By (i) unfolding, (ii) definite-clause removal, (iii) removal of duplicate atoms, (iv) removal of valid clauses, and (v) removal of subsumed clauses, C_{41}-C_{61} are transformed into the clauses C_{62}-C_{77} in Figure 4.
- By (i) unfolding using the definition of *teach*, (ii) definite-clause removal, (iii) removal of duplicate atoms, (iv) removal of valid clauses, and (v) removal of subsumed clauses, C_{62}-C_{77} are transformed into C_{78}-C_{83}:
  C_{78}: $notBC(x) \leftarrow func(h_0, mary, x), notNFP(mary)$
  C_{79}: $mdt(paul, x), notNFP(x) \leftarrow$
  C_{80}: $notBC(ai) \leftarrow$
  C_{81}: $mdt(paul, john) \leftarrow$
  C_{82}: $mdt(paul, mary) \leftarrow func(h_0, mary, x), notNFP(mary)$
  C_{83}: $notNFP(john) \leftarrow$
- By definite-clause removal for *notBC*, C_{78}-C_{83} are transformed into C_{84}-C_{87}:
  C_{84}: $mdt(paul, x), notNFP(x) \leftarrow$
  C_{85}: $mdt(paul, john) \leftarrow$
  C_{86}: $mdt(paul, mary) \leftarrow func(h_0, mary, x), notNFP(mary)$
  C_{87}: $notNFP(john) \leftarrow$

C_{62}: $\mathit{teach}(\mathit{john}, \mathit{ai}) \leftarrow$
C_{63}: $\mathit{notBC}(\mathit{ai}) \leftarrow$
C_{64}: $\mathit{notBC}(x) \leftarrow \mathit{teach}(\mathit{mary}, x)$
C_{65}: $\mathit{notNFP}(x), \mathit{notBC}(y) \leftarrow \mathit{teach}(x, y)$
C_{66}: $\mathit{notNFP}(x) \leftarrow \mathit{teach}(x, y), \mathit{func}(h_0, \mathit{john}, y), \mathit{notNFP}(\mathit{john})$
C_{67}: $\mathit{notNFP}(x) \leftarrow \mathit{teach}(x, y), \mathit{func}(h_0, \mathit{mary}, y), \mathit{notNFP}(\mathit{mary})$
C_{68}: $\mathit{mdt}(\mathit{paul}, \mathit{mary}) \leftarrow \mathit{teach}(\mathit{mary}, x)$
C_{69}: $\mathit{mdt}(\mathit{paul}, \mathit{john}) \leftarrow \mathit{teach}(\mathit{john}, x), \mathit{teach}(\mathit{mary}, x)$
C_{70}: $\mathit{mdt}(\mathit{paul}, \mathit{john}) \leftarrow \mathit{teach}(\mathit{john}, \mathit{ai})$
C_{71}: $\mathit{mdt}(\mathit{paul}, \mathit{john}) \leftarrow \mathit{teach}(\mathit{john}, x), \mathit{func}(h_0, \mathit{mary}, x), \mathit{notNFP}(\mathit{mary}), \mathit{notBC}(x)$
C_{72}: $\mathit{mdt}(\mathit{paul}, \mathit{john}) \leftarrow \mathit{teach}(\mathit{john}, x), \mathit{func}(h_0, \mathit{john}, x), \mathit{notNFP}(\mathit{john}), \mathit{notBC}(x)$
C_{73}: $\mathit{mdt}(\mathit{paul}, x), \mathit{notNFP}(x) \leftarrow$
C_{74}: $\mathit{teach}(\mathit{mary}, x) \leftarrow \mathit{func}(h_0, \mathit{mary}, x), \mathit{notNFP}(\mathit{mary})$
C_{75}: $\mathit{teach}(\mathit{john}, x) \leftarrow \mathit{func}(h_0, \mathit{john}, x), \mathit{notNFP}(\mathit{john})$
C_{76}: $\mathit{notNFP}(x) \leftarrow \mathit{teach}(x, \mathit{ai})$
C_{77}: $\mathit{notNFP}(x) \leftarrow \mathit{teach}(x, y), \mathit{teach}(\mathit{mary}, y)$

FIGURE 4. Clauses obtained by application of ET rules

- Refer to the following clauses:
  C_{88}: $\mathit{mdt}(\mathit{paul}, \mathit{mary}) \leftarrow$
  C_{89}: $\mathit{mdt}(\mathit{paul}, \mathit{john}) \leftarrow$
  Application of the resolution rule to C_{84} and C_{86}, followed by removal of independent *func*-atoms and removal of duplicated atoms, yields the clause C_{88}. By removal of subsumed clauses, C_{84} and C_{86} are removed. By definite clause removal, C_{87} is removed. Then C_{84}-C_{87} are transformed into C_{88}-C_{89}.

As a result, the MI problem $\langle Cs, \varphi\rangle$ in Section 5.1 is transformed equivalently into the MI problem $\langle\{C_{88}, C_{89}\}, \varphi\rangle$. Hence

$$\begin{aligned} \mathit{ans}_{\mathrm{MI}}(Cs, \varphi) &= \mathit{ans}_{\mathrm{MI}}(\{C_{88}, C_{89}\}, \varphi) \\ &= \varphi(\bigcap \mathit{Models}(\{C_{88}, C_{89}\})) \\ &= \varphi(\{\mathit{mdt}(\mathit{paul}, \mathit{mary}), \mathit{mdt}(\mathit{paul}, \mathit{john})\}) \\ &= \{\mathit{mary}, \mathit{john}\}. \end{aligned}$$

5.3. **Robustness of the ET-based method.** Observing the representation and computation of this example, we can find important features of the ET-based method as follows:

- The clauses in Figure 1 are obtained by applying the conversion procedure in Section 4.1 to a first-order formula representing the "may-do-thesis" problem. The clauses in Figure 1 are not in the usual clause space, but in the extended space $\mathrm{ECLS}_{\mathrm{F}}$. The conventional method cannot convert the "may-do-thesis" problem into the usual clause space with guarantee of correctness, i.e., an incorrect answer is obtained by conventional Skolemization-based conversion. To satisfy the condition $\mathit{Models}(E) = \mathit{Models}(Cs)$ in Theorem 4.2, MPD must be chosen with the extended space that is augmented with function variables.
- In the computation of the "may-do-thesis" problem, unfolding is applied many times. The correctness of the unfolding transformation is guaranteed on the preservation of model-intersection in the extended space. Conventional unfolding cannot be used since it is not designed to be applied to clause sets that may include non-definite clauses. Conversion by resolution in SLD resolution makes a similar transformation

to conventional unfolding, and cannot be applied safely (non-explosively) in the extended space.

- Other transformations except unfolding take important roles for non-explosive computation. In a simpler space like definite clause space used by Prolog, many problems can be solved by using unfolding (resolution). However, in the extended space, correct unfolding is a partial mapping, and repeated application of unfolding may result in non-applicability states. One of the important roles of other transformations is to make a new computation state to which unfolding can be applied safely. The ET-based method supports this by admitting all necessary ET rules. Limited-rules approaches like SLD resolution or an unfolding-only method cannot detect incorrect transformation steps, nor prompt unfolding steps by using various rules.

The ET-based method is more robust than existing resolution-based methods, since we have rich representation with function variables (representation robustness), and rich computation by many ET rules (computation robustness).

6. **Conclusions.** We have defined a class of model-intersection (MI) problems on extended clauses possibly with constraint atoms and *func*-atoms, each of which is a pair of a set Cs of extended clauses and an extraction mapping used for constructing the output answer from the intersection of all models of Cs. Many logical problems can be transformed into MI problems, with the answers to them being preserved. The theory in this paper therefore provides a foundation for many kinds of logical problem solving.

We have defined two mappings that convert all proof problems and all QA problems on $\mathrm{FOL_c}$, by using MPD, into MI problems on a new logic that includes function variables. We have proved the correctness of the conversion based on the preservation of logical meanings by MPD.

When only conventional clauses without function variables are used, MPD is impossible. In the presence of built-in constraint atoms, the classical theory, which uses the conventional Skolemization, cannot guarantee the correctness of the conversion of logical formulas into clauses. This is the fundamental reason why a general method for solving all proof problems and all QA problems cannot be established on the conventional first-order logic.

The ET-based solution method together with MPD is very general and fundamental. Any combination of ET steps forms correct computation and the correctness of the method is guaranteed for a very large class of logical problems. Integration of proof problems and QA problems on $\mathrm{FOL_c}$ into MI problems on $\mathrm{ECLS_F}$ matches the future development of a fundamental and central structure of representation and computation for logical problem solving, which was partly shown theoretically and experimentally in [26, 37, 38].

Acknowledgment. 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] 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.

[2] R. B. Palm, U. Paquet and O. Winther, Recurrent relational networks, *arXiv Preprint*, arXiv: 1711.08028, 2017.

[3] 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.

[4] N. Cingillioglu and A. Russo, Deeplogic: End-to-end logical reasoning, *arXiv Preprint*, arXiv: 1805.07433, 2018.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] R. Evans and E. Grefenstette, Learning explanatory rules from noisy data, *Journal of Artificial Intelligence Research*, vol.61, pp.1-64, 2018.

[12] C. L. Chang and R. C. T. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973.

[13] J. A. Robinson, A machine-oriented logic based on the resolution principle, *Journal of the ACM*, vol.12, pp.23-41, 1965.

[14] K. Doets, *From Logic to Logic Programming*, The MIT Press, 1994.

[15] M. Fitting, *First-Order Logic and Automated Theorem Proving*, 2nd Edition, Springer-Verlag, 1996.

[16] C. Walther, A mechanical solution of Schubert's steamroller by many-sorted resolution, *Artificial Intelligence*, vol.26, no.2, pp.217-224, 1985.

[17] F. J. Pelletier, Seventy-five problems for testing automatic theorem provers, *Journal of Automated Reasoning*, vol.2, no.2, pp.191-216, 1986.

[18] M. Stickel, Schubert's steamroller problem: Formulations and solution, *Journal of Automated Reasoning*, vol.2, no.2, pp.89-104, 1986.

[19] T. C. Wang and W. W. Bledsoe, Hierarchical deduction, *Journal of Automated Deduction*, vol.3, no.1, pp.35-77, 1987.

[20] 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.

[21] J. W. Lloyd, *Foundations of Logic Programming*, 2nd Edition, Springer-Verlag, 1987.

[22] R. A. Kowalski, Predicate logic as a programming language, *Proc. of the 6th IFIP Congress 1974*, North-Holland, Amsterdam, pp.569-574, 1974.

[23] R. A. Kowalski, Algorithm = logic + control, *Communications of the ACM*, vol.22, pp.424-435, 1979.

[24] J. Minker, *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann Publishers Inc., 1988.

[25] K. Akama, E. Nantajeewarawat and H. Koike, Program generation in the equivalent transformation computation model using the squeeze method, *Lecture Notes in Computer Science*, vol.4378, pp.41-54, 2007.

[26] 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.

[27] K. Akama and E. Nantajeewarawat, Function-variable elimination and its limitations, *Proc. of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, KEOD, Lisbon, Portugal, pp.212-222, 2015.

[28] 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.

[29] 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.

[30] K. L. Clark, Negation as failure, in *Logic and Data Bases*, H. Gallaire and J. Minker (eds.), New York, Plenum Press, 1978.

[31] M. Gelfond and V. Lifschitz, The stable model semantics for logic programming, *Proc. of International Logic Programming Conference and Symposium*, pp.1070-1080, 1988.

[32] M. Gelfond and V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New Generation Computing*, vol.9, pp.365-386, 1991.

[33] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi and P. F. Patel-Schneider, *The Description Logic Handbook*, 2nd Edition, Cambridge University Press, 2007.

[34] 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.

[35] S. Tessaris, *Questions and Answers: Reasoning and Querying in Description Logic*, Ph.D. Thesis, Department of Computer Science, The University of Manchester, UK, 2001.

[36] F. M.Donini, M. Lenzerini, D. Nardi and A. Schaerf, $\mathcal{AL}$-log: Integrating datalog and description logics, *Journal of Intelligent Information Systems*, vol.16, pp.227-252, 1998.

[37] 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*, KEOD, Seville, Spain, pp.84-95, 2018.

[38] 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.