Model-Intersection Problems with Existentially Quantified Function Variables: Formalization and a Solution Schema

Kiyoshi Akama[1] and Ekawit Nantajeewarawat[2]

[1]*Information Initiative Center, Hokkaido University, Sapporo, Hokkaido, Japan*

[2]*Computer Science, Sirindhorn International Institute of Technology, Thammasat University, Pathumthani, Thailand*

Keywords: Model-Intersection Problem, Extended Clause, Function Variable, Equivalent Transformation.

Abstract: Built-in constraint atoms play a very important role in knowledge representation and are indispensable for practical applications. It is very natural to use built-in constraint atoms together with user-defined atoms when formalizing logical problems using first-order formulas. In the presence of built-in constraint atoms, however, the conventional Skolemization in general preserves neither the satisfiability nor the logical meaning of a given first-order formula, motivating us to step outside the conventional Skolemization and the usual space of first-order formulas. We propose general solutions for proof problems and query-answering (QA) problems on first-order formulas possibly with built-in constraint atoms. We map, by using new meaning-preserving Skolemization, all proof problems and all QA problems, preserving their answers, into a new class of model-intersection (MI) problems on an extended clause space, where clauses are in a sense "higher-order" since they may contain not only built-in constraint atoms but also function variables. We propose a general schema for solving this class of MI problems by equivalent transformation (ET), where problems are solved by repeated simplification using ET rules. The correctness of this solution schema is shown. Since MI problems in this paper form a very large class of logical problems, this theory is also useful for inventing solutions for many classes of logical problems.

1 INTRODUCTION

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, and the answer to this problem is defined to be "yes" if E_2 is a logical consequence of E_1, and it is defined to be "no" otherwise. A proof problem $\langle E_1, E_2\rangle$ is solved (Chang and Lee, 1973; Robinson, 1965) 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 (Chang and Lee, 1973; Fitting, 1996), (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.

A *query-answering problem* (*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. In our previous work (Akama and Nantajeewarawat, 2015a), for solving proof problems on first-order formulas and QA problems on clauses, these problems are transformed into *model-intersection problems* (*MI problems*) on the conventional clause space. Such a MI problem is a pair $\langle Cs, \varphi\rangle$, where Cs is a set of clauses and φ is a mapping, called an *exit mapping*, used for constructing the output answer from the intersection of all models of Cs. More formally, the answer to a MI problem $\langle Cs, \varphi\rangle$ is $\varphi(\bigcap Models(Cs))$, where $Models(Cs)$ is the set of all models of Cs and $\bigcap Models(Cs)$ is the inter-

Akama, K. and Nantajeewarawat, E.
Model-Intersection Problems with Existentially Quantified Function Variables: Formalization and a Solution Schema.
DOI: 10.5220/0006056800520063
In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016) - Volume 2: KEOD*, pages 52-63
ISBN: 978-989-758-203-5

section of all elements of $Models(Cs)$. Note that, in this theory, an interpretation is a set of ground user-defined atoms, which is similar to a Herbrand interpretation (Chang and Lee, 1973; Fitting, 1996). Since each element of $Models(Cs)$ is a set of ground user-defined atoms, we can take the intersection of all elements of it.

In this paper, we consider first-order formulas that possibly includes 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. One of the objectives of this paper is to propose general solutions for proof problems and QA problems on $\mathrm{FOL_c}$, which are large problem classes that have never been solved fully so far. 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 (Chang and Lee, 1973; Fitting, 1996). However, satisfiability preservation of a formula does not generally hold for formulas in $\mathrm{FOL_c}$ (Akama and Nantajeewarawat, 2015b). The conventional Skolemization, therefore, does not provide a transformation process towards correct solutions for proof problems and QA problems on $\mathrm{FOL_c}$.

Meaning-preserving Skolemization (MPS) was invented (Akama and Nantajeewarawat, 2008; Akama and Nantajeewarawat, 2011) to overcome the difficulties caused by the conventional Skolemization. MPS 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 MPS. An extended clause may contain function variables and atoms of a special kind called *func*-atoms. The set of all extended clauses is called $\mathrm{ECLS_F}$.

This paper introduces a model-intersection problem (MI problem) on this extended space, which is a pair $\langle Cs, \varphi \rangle$, where Cs is a set of extended clauses and φ is an exit mapping. The set of all MI problems on the extended clauses constitutes a very large class of problems and is of great importance. As outlined by Fig. 1, all proof problems and all QA problems on $\mathrm{FOL_c}$ are mapped, preserving their answers, into MI problems on $\mathrm{ECLS_F}$. By solving MI problems on $\mathrm{ECLS_F}$, we solve proof problems and QA problems on $\mathrm{FOL_c}$. We propose a general schema for solving MI problems on $\mathrm{ECLS_F}$ by equivalent transformation (ET), where problems are solved by repeated problem simplification using ET rules.

The class of MI problems established 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 class of MI problems considered in our previous work (Akama and Nantajeewarawat, 2015a) involves only usual clauses (with no function variable being allowed) and is not sufficient for dealing with proof problems and QA problems on $\mathrm{FOL_c}$ entirely.

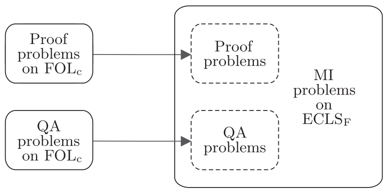
Proof problems on $\mathrm{FOL_c}$
QA problems on $\mathrm{FOL_c}$
Proof problems
QA problems
MI problems on $\mathrm{ECLS_F}$

Figure 1: MI-problem-centered view of logical problems.

The rest of the paper is organized as follows: Section 2 defines extended clauses and $\mathrm{ECLS_F}$, and introduces meaning-preserving Skolemization. Section 3 formalizes MI problems on extended clauses and describes how QA problems and proof problems can be converted into MI problems. Section 4 presents a general schema for solving MI problems by ET. Section 5 demonstrates an application of the general schema. Section 6 concludes the paper.

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

2 AN EXTENDED CLAUSE SPACE

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. Fig. 3 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 Fig. 3.) Let $\mathcal{A}_u$ be the set of all user-defined atoms, $\mathcal{G}_u$ the set of all ground user-defined atoms, $\mathcal{A}_c$ the set of all constraint atoms, and $\mathcal{G}_c$ the set of all ground constraint atoms.

A *func-atom* (Akama and Nantajeewarawat, 2011) is an expression of the form $func(f, t_1, \ldots, t_n, t_{n+1})$, where f is either an n-ary function constant or an n-ary function variable, and the t_i are usual terms. For example, supposing that f_0 is a unary function variable, $func(f_0, x, y)$ is a *func*-atom (cf. the clauses C_{24} and C_{25} in Fig. 3). A *func*-atom $func(f, t_1, \ldots, t_n, t_{n+1})$ is *ground* if f is a function constant and the t_i are ground usual terms.

There are two types of variables: usual variables and function variables. (In Fig. 3, x, y, z, and w are usual variables, while f_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 *FVar* ∪ *FCon*. Each n-ary function constant is associated with a mapping from $\mathcal{G}_t^n$ to $\mathcal{G}_t$, where $\mathcal{G}_t$ denotes the set of all usual ground 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, 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.

Let DCL denote the set of all extended definite clauses with no constraint atom in their left-hand sides. Given a definite clause $C \in$ DCL, the user-defined atom in $lhs(C)$ is called the *head* of C, denoted by $head(C)$, and the set $rhs(C)$ is called the *body* of C, denoted by $body(C)$.

2.3 An Extended Clause Space

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 $\mathrm{ECLS_F}$. The *extended clause space* in this paper is the powerset of $\mathrm{ECLS_F}$.

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.

2.4 Conversion of First-order Formulas into Sets of Extended Clauses

Semantically, an extended clause corresponds to a disjunction of extended literals, and a set of extended clauses corresponds to an extended conjunctive normal form. After explaining the limitations of the conventional Skolemization, conversion of a first-order formula in $\mathrm{FOL_c}$ into a set of extended clauses in $\mathrm{ECLS_F}$ by meaning-preserving Skolemization (Akama and Nantajeewarawat, 2008; Akama and Nantajeewarawat, 2011) is introduced.

2.4.1 Conventional Skolemization

In the conventional proof theory, a first-order formula is usually converted into a conjunctive normal form in the usual first-order formula space. The conversion involves removal of existential quantifications by Skolemization (Chang and Lee, 1973; Fitting, 1996), i.e., by replacement of an existentially quantified variable with a Skolem term determined by its relevant quantification structure. Let $\mathrm{CSK}(E)$ denote the set of usual clauses obtained by applying this conversion to a first-order formula E.

The conventional Skolemization, however, does not generally preserve the logical meaning of a first-order formula in $\mathrm{FOL_c}$, nor the satisfiability thereof. This is precisely shown by Theorem 1 below. Given a first-order formula E in $\mathrm{FOL_c}$ and a set *Cs* of extended clauses in $\mathrm{ECLS}_{\mathrm{F}}$, let $\mathit{Models}(E)$ and $\mathit{Models}(Cs)$ denote the set of all models of E and that of all models of Cs, respectively.

Theorem 1.

1. *There are a first-order formula E in $\mathit{FOL}_{\mathrm{c}}$ and a clause set $Cs \subseteq \mathit{ECLS}_{\mathrm{F}}$ such that $\mathrm{CSK}(E) = Cs$ and $\mathit{Models}(E) \neq \mathit{Models}(Cs)$.*
2. *There are a first-order formula E in $\mathit{FOL}_{\mathrm{c}}$ and a clause set $Cs \subseteq \mathit{ECLS}_{\mathrm{F}}$ such that $\mathrm{CSK}(E) = Cs$, $\mathit{Models}(E) \neq \varnothing$, and $\mathit{Models}(Cs) = \varnothing$.*

Proof: Assume that:

- *noteq* is a predicate for built-in constraint atoms and for any ground terms t_1 and t_2, $\mathit{noteq}(t_1,t_2)$ is true iff $t_1 \neq t_2$.
- F_1, F_2, F_3, and F_4 are the first-order formulas in $\mathrm{FOL}_{\mathrm{c}}$ given by:

F_1: $\forall x \forall y \forall z : [(\mathit{hasChild}(x,y) \wedge \mathit{hasChild}(x,z) \wedge \mathit{noteq}(y,z)) \rightarrow \mathit{TaxCut}(x)]$

F_2: $\mathit{hasChild}(\mathit{Peter},\mathit{Paul})$

F_3: $\exists x : \mathit{hasChild}(\mathit{Peter},x)$

F_4: $\neg(\exists x : \mathit{TaxCut}(x))$

Consider a first-order formula $E = F_1 \wedge F_2 \wedge F_3 \wedge F_4$. Obviously, $\mathit{Models}(E) \neq \varnothing$ and a model of E is $\{\mathit{hasChild}(\mathit{Peter},\mathit{Paul})\}$. Let $\mathrm{CSK}(E) = Cs$. Then Cs consists of the following clauses, where f is a new constant:

$\mathit{TaxCut}(x) \leftarrow \mathit{hasChild}(x,y), \mathit{hasChild}(x,z), \mathit{noteq}(y,z)$
$\mathit{hasChild}(\mathit{Peter},\mathit{Paul}) \leftarrow$
$\mathit{hasChild}(\mathit{Peter},f) \leftarrow$
$\leftarrow \mathit{TaxCut}(x)$

Since f is a constant and $\mathit{noteq}(\mathit{Paul},f)$ is true, the clause set Cs has no model, i.e., $\mathit{Models}(Cs) = \varnothing$. Hence Results 1 and 2 of this theorem hold. □

2.4.2 Meaning-preserving Skolemization

In order to transform a first-order formula equivalently into a set of extended clauses, meaning-preserving Skolemization was invented in (Akama and Nantajeewarawat, 2008; Akama and Nantajeewarawat, 2011). Let $\mathrm{MPS}(E)$ denote the set of extended clauses resulting from applying meaning-preserving Skolemization to a given first-order formula E in $\mathrm{FOL}_{\mathrm{c}}$. $\mathrm{MPS}(E)$ is obtained from E by repeated subformula transformation and conversion into a clausal form. Consider, for example, the first-order formula E in the proof of Theorem 1. $\mathrm{MPS}(E)$ is the clause set Cs' consisting of the following extended clauses, where h is a 0-ary function variable:

$\mathit{TaxCut}(x) \leftarrow \mathit{hasChild}(x,y), \mathit{hasChild}(x,z), \mathit{noteq}(y,z)$
$\mathit{hasChild}(\mathit{Peter},\mathit{Paul}) \leftarrow$
$\mathit{hasChild}(\mathit{Peter},x) \leftarrow \mathit{func}(h,x)$
$\leftarrow \mathit{TaxCut}(x)$

An algorithm for computing $\mathrm{MPS}(E)$ was given in (Akama and Nantajeewarawat, 2011). Each transformation used by this algorithm preserves the logical meaning of an input formula. As a result, the next theorem is obvious.

Theorem 2. *Let E be a first-order formula in $\mathit{FOL}_{\mathrm{c}}$ and $Cs \subseteq \mathit{ECLS}_{\mathrm{F}}$. If $\mathrm{MPS}(E) = Cs$, then*

1. *$\mathit{Models}(E) = \mathit{Models}(Cs)$, and*
2. *$\mathit{Models}(E) = \varnothing$ iff $\mathit{Models}(Cs) = \varnothing$.* □

2.5 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 $\mathit{func}(f,t_1,\ldots,t_n,t_{n+1})$ is true iff $f(t_1,\ldots,t_n) = t_{n+1}$.

A ground clause $C = (a_1,\ldots,a_m \leftarrow b_1,\ldots,b_n,\mathbf{f}_1,\ldots,\mathbf{f}_p) \in \mathrm{ECLS}_{\mathrm{F}}$ 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 $Cs \subseteq \mathrm{ECLS}_{\mathrm{F}}$ 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., $\mathit{Models}(Cs)$ is the set of all models of Cs for any $Cs \subseteq \mathrm{ECLS}_{\mathrm{F}}$.

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 (Clark, 1978; Lloyd, 1987), which underlies definite logic programming, and in the stable model semantics (Gelfond and Lifschitz, 1988; Gelfond and Lifschitz, 1991), which underlies answer set programming.

3 MODEL-INTERSECTION PROBLEMS

3.1 Model Intersection is Important

Assume that a person A and a person B are interested in knowing which atoms in $\mathcal{G}_u$ are true and which atoms in $\mathcal{G}_u$ are false. They want to know the unknown set G of all true ground atoms. Due to shortage of knowledge, A still cannot identify one unique subset of $\mathcal{G}_u$ as the state of the world. The person A can only limit possible subsets of true atoms by specifying a subset Gs of $pow(\mathcal{G}_u)$. The unknown set G of all true atoms belongs to Gs.

One way for A to inform this knowledge to B compactly is to send to B a clause set Cs such that $Gs \subseteq Models(Cs)$. Receiving Cs, B knows that $Models(Cs)$ includes all possible intended sets of ground atoms, i.e., $G \in Models(Cs)$. As such, B can know that each ground atom outside $\bigcup Models(Cs)$ is false, i.e., for any $g \in \mathcal{G}_u$, if $g \notin \bigcup Models(Cs)$, then $g \notin G$. The person B can also know that each ground atom in $\bigcap Models(Cs)$ is true, i.e., for any $g \in \mathcal{G}_u$, if $g \in \bigcap Models(Cs)$, then $g \in G$. This shows the importance of calculating $\bigcap Models(Cs)$.

3.2 Model-Intersection (MI) Problems on the Extended Clause Space

It is natural for us to seek information about the model intersection of given knowledge, which motivates us to introduce a new class of logical problems.

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 $pow(\mathcal{G}_u)$ to some set W. The mapping φ is called an *exit mapping*. The answer to this problem, denoted by $ans_{\mathrm{MI}}(Cs, \varphi)$, is defined by

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

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

Example 1. Consider the Oedipus puzzle described in (Baader et al., 2007). Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Polyneikes also had children, among them Thersandros, who is not a patricide. The problem is to find 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:

$$\begin{array}{ll} isCh(oe,io) \leftarrow & isCh(po,io) \leftarrow \\ isCh(po,oe) \leftarrow & isCh(th,po) \leftarrow \\ pat(oe) \leftarrow & \leftarrow pat(th) \\ \multicolumn{2}{l}{prob(x), pat(y) \leftarrow isCh(z,x), pat(z), isCh(y,z)} \end{array}$$

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

$$\begin{array}{l} app([\,],x,x) \leftarrow \\ app([w|x],y,[w|z]) \leftarrow app(x,y,z) \\ ans(x) \leftarrow app([1,2,3],[4,5],x) \end{array}$$

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. Consider the "tax-cut" problem discussed in (Motik et al., 2005). This problem is to find all persons who can have discounted tax, with the knowledge 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:

$$\begin{array}{l} TaxCut(x) \leftarrow hasChild(x,y), hasChild(x,z), \\ \qquad\qquad notSame(y,z) \\ notSame(x,y) \leftarrow Man(x), Woman(y) \\ \leftarrow notSame(x,x) \\ Woman(x) \leftarrow motherOf(x,y) \\ hasChild(Peter,x) \leftarrow func(f_1,x) \\ motherOf(x,y) \leftarrow func(f_1,x), func(f_2,y) \\ hasChild(Peter,Paul) \leftarrow \end{array}$$

$Man(Paul) \leftarrow$

The fifth and the sixth clauses together represent the fifth statement (i.e., "Peter has a child, who is someone's mother"), where f_1 and f_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 $\langle Cs_3, \varphi_3\rangle$. $\square$

Example 4. Consider the "Dreadsbury Mansion Mystery" problem, 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, f_0 is a 0-ary function variable, and f_1 is a unary function variable:

$live(x,D) \leftarrow func(f_0,x)$
$kill(x,A) \leftarrow func(f_0,x)$
$\leftarrow live(x,D), neq(x,A), neq(x,B), neq(x,C)$
$live(A,D) \leftarrow$
$live(B,D) \leftarrow$
$live(C,D) \leftarrow$
$hate(x,y) \leftarrow kill(x,y)$
$\leftarrow kill(x,y), richer(x,y)$
$\leftarrow hate(A,x), hate(C,x), live(x,D)$
$hate(A,x) \leftarrow neq(x,B), live(x,D)$
$richer(x,A), hate(B,x) \leftarrow$
$hate(B,x) \leftarrow hate(A,x)$
$\leftarrow hate(x,y), func(f_1,x,y), live(x,D)$
$live(y,D) \leftarrow live(x,D), func(f_1,x,y)$
$killer(x) \leftarrow kill(x,A)$

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

Example 5. Let an exit mapping φ_{pr} be given as follows: For any $G \subseteq \mathcal{G}_u$, $\varphi_{pr}(G) =$ "yes" if $G = \mathcal{G}_u$, and $\varphi_{pr}(G) =$ "no" otherwise. Referring to the clause sets Cs_1–Cs_4 in Examples 1–4, 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 $\langle Cs_5, \varphi_{pr}\rangle$ 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 $\langle Cs_6, \varphi_{pr}\rangle$ 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 $\langle Cs_7, \varphi_{pr}\rangle$ represents the problem of proving whether someone gets discounted tax.
- Letting $Cs_8 = Cs_4 \cup \{(\leftarrow killer(A))\}$, the MI problem $\langle Cs_8, \varphi_{pr}\rangle$ represents the problem of proving whether Agatha killed herself. $\square$

3.3 Conversion of Query-Answering (QA) Problems into MI Problems

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

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

In logic programming (Lloyd, 1987), 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 (Baader et al., 2007), a class of problems formulated as conjunctions of DL-based axioms and assertions together with query atoms has been discussed (Tessaris, 2001). These two problem classes can be formalized as subclasses of QA problems considered in this paper.

Theorem 3. *For any closed first-order formula $E \in FOL_c$ and any $a \in \mathcal{A}_u$,*

$$ans_{QA}(E,a) = rep(a) \cap (\bigcap Models(E)),$$

where $rep(a)$ denotes the set of all ground instances of a.

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

$$\begin{aligned} & ans_{QA}(E,a) \\ = \ & \{a\theta \mid (\theta \in \mathcal{S}) \,\&\, (a\theta \in \mathcal{G}_u) \,\&\, (E \models a\theta)\} \\ = \ & \{g \mid (\theta \in \mathcal{S}) \,\&\, (g = a\theta) \,\&\, (g \in \mathcal{G}_u) \,\&\, (E \models g)\} \\ = \ & \{g \mid (g \in rep(a)) \,\&\, (E \models g)\} \\ = \ & \{g \mid (g \in rep(a)) \,\&\, (g \in (\bigcap Models(E)))\} \\ = \ & rep(a) \cap (\bigcap Models(E)). \quad \square \end{aligned}$$

Theorem 4 below shows that a QA problem on $\mathrm{FOL_c}$ can be converted into a MI problem on $\mathrm{ECLS_F}$.

Theorem 4. *Let E be a first-order formula in FOL_c and $a \in \mathcal{A}_u$. Let $Cs \subseteq ECLS_F$. If $Models(E) = Models(Cs)$, then*

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

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 $G \subseteq \mathcal{G}_u$,

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

Proof: Assume that $Models(E) = Models(Cs)$. Then

$ans_{\mathrm{QA}}(E,a)$
$=$ (by Theorem 3)
$= rep(a) \cap (\bigcap Models(E))$
$= rep(a) \cap (\bigcap Models(Cs))$
$=$ (by the definition of φ_{qa})
$= \varphi_{\mathrm{qa}}(\bigcap Models(Cs \cup \{(p(x_1,\ldots,x_n) \leftarrow a)\}))$
$= ans_{\mathrm{MI}}(Cs \cup \{(p(x_1,\ldots,x_n) \leftarrow a)\}, \varphi_{\mathrm{qa}})$. ☐

3.4 Conversion of Proof Problems into MI Problems

A *proof problem* 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 $ans_{\mathrm{Pr}}(E_1,E_2)$, is defined by

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

It is well known that 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) (Chang and Lee, 1973; Fitting, 1996). As a result, $ans_{\mathrm{Pr}}(E_1,E_2)$ can be equivalently defined by

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

Theorem 5 below shows that a proof problem can be converted into a MI problem on $\mathrm{ECLS_F}$.

Theorem 5. *Let $\langle E_1, E_2 \rangle$ be a proof problem, where E_1 and E_2 are first-order formulas in FOL_c. Let $Cs \subseteq ECLS_F$. Let $\varphi_{\mathrm{pr}} : pow(\mathcal{G}_u) \rightarrow \{\text{“yes”}, \text{“no”}\}$ be defined by: for any $G \subseteq \mathcal{G}_u$,*

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

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

Proof: Assume that $Models(E_1 \wedge \neg E_2) = \emptyset$ iff $Models(Cs) = \emptyset$. 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, $m - \{b\} \neq \mathcal{G}_u$. Therefore, (i) if $Models(Cs) \neq \emptyset$, then $\bigcap Models(Cs) \neq \mathcal{G}_u$, and (ii) if $Models(Cs) = \emptyset$, then $\bigcap Models(Cs) = \bigcap\{\} = \mathcal{G}_u$.

Two cases are considered:

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

Hence $ans_{\mathrm{Pr}}(E_1,E_2) = ans_{\mathrm{MI}}(Cs, \varphi_{\mathrm{pr}})$. ☐

4 SOLVING MI PROBLEMS BY EQUIVALENT TRANSFORMATION

A general schema for solving MI problems based on equivalent transformation (ET) is formulated and its correctness is shown (Theorem 10).

4.1 Preservation of Partial Mappings and Equivalent Transformation

Terminologies such as preservation of partial mappings and equivalent transformation are defined in general below. They will be used with a specific class of partial mappings called target mappings, which will be introduced in Section 4.2.

Assume that X and Y are sets and f is a partial mapping from X to Y. For any $x, x' \in dom(f)$, transformation of x into x' is said to *preserve* f iff $f(x) = f(x')$. For any $x, x' \in dom(f)$, transformation of x into x' is called *equivalent transformation* (*ET*) with respect to f iff the transformation preserves f, i.e., $f(x) = f(x')$.

Let $\mathbb{F}$ be a set of partial mappings from a set X to a set Y. Given $x, x' \in X$, transformation of x into x' is called *equivalent transformation* (*ET*) with respect to $\mathbb{F}$ iff there exists $f \in \mathbb{F}$ such that the transformation preserves f. A sequence $[x_0, x_1, \ldots, x_n]$ of elements in X is called an *equivalent transformation sequence* (*ET sequence*) with respect to $\mathbb{F}$ iff for any $i \in \{0, 1, \ldots, n-1\}$, transformation of x_i into x_{i+1} is ET with respect to $\mathbb{F}$. When emphasis is placed on the initial element x_0 and the final element x_n, this sequence is also referred to as an ET sequence *from x_0 to x_n*.

4.2 Target Mappings

We introduce the concept of target mapping, which is useful to devise equivalent transformation (ET) rules in the $\mathrm{ECLS_F}$ space (Theorem 9) or to construct an answer mapping (Theorem 8) for determining an answer from the final state of computation.

The answer to a MI problem $\langle Cs, \varphi\rangle$ is determined uniquely by $\mathit{Models}(Cs)$ and φ. MI problems can thus be transformed into simpler forms by ET preserving the mapping *Models*.

Simplification of MI problems using ET preserving the mapping *Models* can be extended by considering additional partial mappings. A new class of partial mappings, called GSETMAP, will be defined below.

Definition 1. GSETMAP is the set of all partial mappings from $\mathit{pow}(\mathrm{ECLS_F})$ to $\mathit{pow}(\mathit{pow}(\mathcal{G}_\mathrm{u}))$. □

As defined in Section 2.5, $\mathit{Models}(Cs)$ is the set of all models of Cs for any $Cs \subseteq \mathrm{ECLS_F}$. Since a model is a subset of $\mathcal{G}_\mathrm{u}$, *Models* is regarded as a total mapping from $\mathit{pow}(\mathrm{ECLS_F})$ to $\mathit{pow}(\mathit{pow}(\mathcal{G}_\mathrm{u}))$. Since a total mapping is also a partial mapping, the mapping *Models* is a partial mapping from $\mathit{pow}(\mathrm{ECLS_F})$ to $\mathit{pow}(\mathit{pow}(\mathcal{G}_\mathrm{u}))$, i.e., it is an element of GSETMAP.

A partial mapping M in GSETMAP is of particular interest if $\bigcap M(Cs) = \bigcap \mathit{Models}(Cs)$ for any $Cs \in \mathit{dom}(M)$. Such a partial mapping is called a *target mapping*.

Definition 2. A partial mapping $M \in$ GSETMAP is a *target mapping* iff for any $Cs \in \mathit{dom}(M)$, $\bigcap M(Cs) = \bigcap \mathit{Models}(Cs)$. □

It is obvious that:

Theorem 6. *The mapping Models is a target mapping.* □

The next theorem provides a sufficient condition for a mapping in GSETMAP to be a target mapping.

Theorem 7. *Let $M \in$ GSETMAP. M is a target mapping if the following conditions are satisfied:*

1. *$M(Cs) \subseteq \mathit{Models}(Cs)$ for any $Cs \in \mathit{dom}(M)$.*
2. *For any $Cs \in \mathit{dom}(M)$ and any $m_2 \in \mathit{Models}(Cs)$, there exists $m_1 \in M(Cs)$ such that $m_1 \subseteq m_2$.*

Proof: Assume that Conditions 1 and 2 above are satisfied. Let $Cs \in \mathit{dom}(M)$. By Condition 1, $\bigcap M(Cs) \supseteq \bigcap \mathit{Models}(Cs)$. We show that $\bigcap M(Cs) \subseteq \bigcap \mathit{Models}(Cs)$ as follows: Assume that $g \in \bigcap M(Cs)$. Let $m_2 \in \mathit{Models}(Cs)$. By Condition 2, there exists $m_1 \in M(Cs)$ such that $m_1 \subseteq m_2$. Since $g \in \bigcap M(Cs)$, g belongs to m_1. So $g \in m_2$. Since m_2 is any arbitrary element of $\mathit{Models}(Cs)$, g belongs to $\bigcap \mathit{Models}(Cs)$. It follows that $\bigcap M(Cs) = \bigcap \mathit{Models}(Cs)$. Hence M is a target mapping. □

4.3 Answer Mappings

A set of problems that can be solved at low cost is useful to provide a desirable final destination for ET computation. It can also be specified as a partial mapping that is preserved by ET transformation. Such a specification is useful to invent and to justify new ET transformation. This motivates the concept of answer mapping, which is formalized below.

Definition 3. Let W be a set. A partial mapping A from

$$\mathit{pow}(\mathrm{ECLS_F}) \times \mathit{Map}(\mathit{pow}(\mathcal{G}_\mathrm{u}), W)$$

to W is an *answer mapping* iff for any $\langle Cs, \varphi\rangle \in \mathit{dom}(A)$, $\mathit{ans}_\mathrm{MI}(Cs, \varphi) = A(Cs, \varphi)$. □

If M is a target mapping, then M can be used for constructing answer mappings.

Theorem 8. *Let M be a target mapping. Suppose that A is a partial mapping such that*

- *$\mathit{dom}(M) = \{x \mid \langle x, y\rangle \in \mathit{dom}(A)\}$, and*
- *for any $\langle Cs, \varphi\rangle \in \mathit{dom}(A)$,*

$$A(Cs, \varphi) = \varphi(\bigcap M(Cs)).$$

Then A is an answer mapping.

Proof: Let $\langle Cs, \varphi\rangle \in \mathit{dom}(A)$. Since $\mathit{dom}(M) = \{x \mid \langle x, y\rangle \in \mathit{dom}(A)\}$, Cs belongs to $\mathit{dom}(M)$. Since M is a target mapping, $\bigcap M(Cs) = \bigcap \mathit{Models}(Cs)$. So

$$\begin{aligned} \mathit{ans}_\mathrm{MI}(Cs, \varphi) &= \varphi(\bigcap \mathit{Models}(Cs)) \\ &= \varphi(\bigcap M(Cs)) \\ &= A(Cs, \varphi). \end{aligned}$$

Thus A is an answer mapping. □

4.4 ET Steps and ET Rules

A schema for solving MI problems based on equivalent transformation (ET) preserving answers is formulated. The notions of preservation of answers/target mappings, ET with respect to answers/target mappings, and an ET sequence are obtained by specializing the general definitions in Section 4.1.

Let STATE be the set of all MI problems. Elements of STATE are called *states*.

Definition 4. Let $\langle S, S'\rangle \in$ STATE × STATE. $\langle S, S'\rangle$ is an *ET step* iff if $S = \langle Cs, \varphi\rangle$ and $S' = \langle Cs', \varphi'\rangle$, then $\mathit{ans}_\mathrm{MI}(Cs, \varphi) = \mathit{ans}_\mathrm{MI}(Cs', \varphi')$. □

Definition 5. A sequence $[S_0, S_1, \ldots, S_n]$ of elements of STATE is an *ET sequence* iff for any $i \in \{0, 1, \ldots, n-1\}$, $\langle S_i, S_{i+1}\rangle$ is an ET step. $\square$

The role of ET computation constructing $[S_0, S_1, \ldots, S_n]$ is to start with S_0 and to reach S_n from which the answer to the given problem can be easily computed.

The concept of ET rule on STATE is defined by:

Definition 6. An *ET rule* r on STATE is a partial mapping from STATE to STATE such that for any $S \in dom(r)$, $\langle S, r(S)\rangle$ is an ET step. $\square$

We also define ET rules on $pow(\mathrm{ECLS_F})$ as follows:

Definition 7. An *ET rule* r with respect to a target mapping M is a partial mapping from $pow(\mathrm{ECLS_F})$ to $pow(\mathrm{ECLS_F})$ such that for any $Cs \in dom(r)$, $M(Cs) = M(r(Cs))$. $\square$

We can construct an ET rule on STATE from an ET rule with respect to a target mapping.

Theorem 9. *Assume that M is a target mapping and r is an ET rule with respect to M. Suppose that $\overline{r}$ is a partial mapping from* STATE *to* STATE *such that*

- $dom(r) = \{x \mid \langle x, y\rangle \in dom(\overline{r})\}$*, and*
- $\overline{r}(S) = \langle r(Cs), \varphi\rangle$ *if* $S = \langle Cs, \varphi\rangle \in dom(\overline{r})$.

Then $\overline{r}$ is an ET rule on STATE.

Proof: Assume that $S \in dom(\overline{r})$. Then there exist a clause set Cs and an exit mapping φ such that $S = \langle Cs, \varphi\rangle$ and $Cs \in dom(r)$. For such Cs and φ,

$$\begin{array}{lcl}
ans_{\mathrm{MI}}(Cs, \varphi) & = & \varphi(\bigcap Models(Cs)) \\
& = & \text{(since } M \text{ is a target mapping)} \\
& = & \varphi(\bigcap M(Cs)) \\
& = & \text{(since } M(Cs) = M(r(Cs))\text{)} \\
& = & \varphi(\bigcap M(r(Cs))) \\
& = & \text{(since } M \text{ is a target mapping)} \\
& = & \varphi(\bigcap Models(r(Cs))) \\
& = & ans_{\mathrm{MI}}(r(Cs), \varphi).
\end{array}$$

Since $S = \langle Cs, \varphi\rangle$ and $\overline{r}(S) = \langle r(Cs), \varphi\rangle$, $\langle S, \overline{r}(S)\rangle$ is an ET step. Hence $\overline{r}$ is an ET rule on STATE. $\square$

4.5 Correct Solutions based on ET Rules

Given a set Cs of extended clauses and an exit mapping φ, the MI problem $\langle Cs, \varphi\rangle$ can be solved as follows:

1. Let A be an answer mapping.

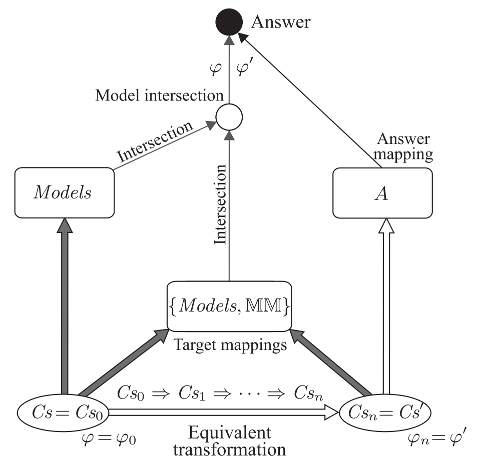

Figure 2: Target mappings and answer mappings yield many correct computation paths.

2. Prepare a set R of ET rules on STATE.
3. Take S_0 such that $S_0 = \langle Cs, \varphi\rangle$ to start computation from S_0.
4. Construct an ET sequence $[S_0, \ldots, S_n]$ by applying ET rules in R, i.e., for each $i \in \{0, 1, \ldots, n-1\}$, S_{i+1} is obtained from S_i by selecting and applying $r_i \in R$ such that $S_i \in dom(r_i)$ and $r_i(S_i) = S_{i+1}$.
5. Assume that $S_n = \langle Cs_n, \varphi_n\rangle$. If the computation reaches the domain of A, i.e., $\langle Cs_n, \varphi_n\rangle \in dom(A)$, then compute the answer by using the answer mapping A, i.e., output $A(Cs_n, \varphi_n)$.

The answer to the MI problem $\langle Cs, \varphi\rangle$, i.e., $ans_{\mathrm{MI}}(Cs, \varphi) = \varphi(\bigcap Models(Cs))$, can be directly obtained by the computation shown in the leftmost path in Fig. 2. Instead of taking this computation path, the above solution takes a different one, i.e., the lowest path (from Cs to Cs') followed by the rightmost path (through A) in Fig. 2.

The selection of r_i in R at Step 4 is nondeterministic and there may be many possible computation paths for each MI problem. Every output computed by using any arbitrary computation path is correct.

Theorem 10. *When an ET sequence starting from $S_0 = \langle Cs, \varphi\rangle$ reaches S_n in $dom(A)$, the above procedure gives the correct answer to $\langle Cs, \varphi\rangle$.*

Proof: Since $[S_0, \ldots, S_n]$ is an ET sequence, $ans_{\mathrm{MI}}(Cs, \varphi) = ans_{\mathrm{MI}}(Cs_n, \varphi_n)$. Since A is an answer mapping, $ans_{\mathrm{MI}}(Cs_n, \varphi_n) = A(Cs_n, \varphi_n)$. Hence $ans_{\mathrm{MI}}(Cs, \varphi) = A(Cs_n, \varphi_n)$. $\square$

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

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

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

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

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

C_{17}: $\mathit{AC}(x) \leftarrow \mathit{teach}(\mathit{mary},x)$
C_{18}: $\leftarrow \mathit{AC}(x),\mathit{BC}(x)$
C_{19}: $\mathit{AC}(x),\mathit{BC}(x) \leftarrow \mathit{Co}(x)$
C_{20}: $\mathit{Co}(x) \leftarrow \mathit{AC}(x)$
C_{21}: $\mathit{Co}(x) \leftarrow \mathit{BC}(x)$
C_{22}: $\mathit{FP}(x) \leftarrow \mathit{NFP}(x)$
C_{23}: $\leftarrow \mathit{NFP}(x),\mathit{teach}(x,y),\mathit{Co}(y)$
C_{24}: $\mathit{teach}(x,y),\mathit{NFP}(x) \leftarrow \mathit{FP}(x),\mathit{func}(f_0,x,y)$
C_{25}: $\mathit{Co}(y),\mathit{NFP}(x) \leftarrow \mathit{FP}(x),\mathit{func}(f_0,x,y)$

Figure 3: Background knowledge for the *mdt* problem on $\mathrm{ECLS_F}$.

5 EXAMPLE

5.1 Problem Description

The clauses in Fig. 3 are obtained from the "may-do-thesis" problem (for short, the *mdt* problem) given in (Donini et al., 1998) with some modification. All atoms appearing in Fig. 3 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 a MI problem $\langle Cs, \varphi \rangle$, where *Cs* consists of the clauses C_1–C_{25} in Fig. 3 and φ is defined by: for any $G \subseteq \mathcal{G}_\mathrm{u}$,

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

C_{26}: $\mathit{teach}(\mathit{john},\mathit{ai}) \leftarrow$
C_{27}: $\mathit{AC}(\mathit{ai}) \leftarrow$
C_{28}: $\mathit{AC}(x) \leftarrow \mathit{teach}(\mathit{mary},x)$
C_{29}: $\leftarrow \mathit{AC}(x),\mathit{BC}(x)$
C_{30}: $\mathit{AC}(x),\mathit{BC}(x) \leftarrow \mathit{Co}(x)$
C_{31}: $\mathit{Co}(x) \leftarrow \mathit{AC}(x)$
C_{32}: $\mathit{Co}(x) \leftarrow \mathit{BC}(x)$
C_{33}: $\leftarrow \mathit{NFP}(x),\mathit{teach}(x,y),\mathit{Co}(y)$
C_{34}: $\mathit{mdt}(\mathit{paul},\mathit{mary}) \leftarrow \mathit{AC}(x),\mathit{teach}(\mathit{mary},x),$
$\mathit{Co}(\mathit{ai})$
C_{35}: $\mathit{mdt}(\mathit{paul},\mathit{john}) \leftarrow \mathit{AC}(x),\mathit{teach}(\mathit{john},x),$
$\mathit{Co}(\mathit{ai})$
C_{36}: $\mathit{mdt}(\mathit{paul},x) \leftarrow \mathit{NFP}(x)$
C_{37}: $\mathit{teach}(\mathit{mary},x),\mathit{NFP}(\mathit{mary}) \leftarrow \mathit{func}(f_0,\mathit{mary},x)$
C_{38}: $\mathit{teach}(\mathit{john},x),\mathit{NFP}(\mathit{john}) \leftarrow \mathit{func}(f_0,\mathit{john},x)$
C_{39}: $\mathit{Co}(x),\mathit{NFP}(\mathit{mary}) \leftarrow \mathit{func}(f_0,\mathit{mary},x)$
C_{40}: $\mathit{Co}(x),\mathit{NFP}(\mathit{john}) \leftarrow \mathit{func}(f_0,\mathit{john},x)$

Figure 4: Clauses obtained by application of ET rules.

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

5.2 ET Computation

The clause set *Cs* consisting of C_1–C_{25} given in Section 5.1 (Fig. 3) 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 Fig. 4.
- 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 Fig. 5.
- 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 C_{62}–C_{77} in Fig. 6.
- 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} in Fig. 7.

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}: $mdt(paul, mary) \leftarrow AC(x), teach(mary, x), func(f_0, mary, ai), notNFP(mary)$
C_{46}: $mdt(paul, mary) \leftarrow AC(x), teach(mary, x), func(f_0, john, ai), notNFP(john)$
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}: $mdt(paul, john) \leftarrow AC(x), teach(john, x), func(f_0, mary, ai), notNFP(mary)$
C_{50}: $mdt(paul, john) \leftarrow AC(x), teach(john, x), func(f_0, john, ai), notNFP(john)$
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}: $teach(mary, x) \leftarrow func(f_0, mary, x), notNFP(mary)$
C_{55}: $teach(john, x) \leftarrow func(f_0, john, x), notNFP(john)$
C_{56}: $notNFP(x) \leftarrow teach(x, y), func(f_0, mary, y), notNFP(mary)$
C_{57}: $notNFP(x) \leftarrow teach(x, y), func(f_0, john, y), notNFP(john)$
C_{58}: $notNFP(x) \leftarrow teach(x, y), BC(y)$
C_{59}: $notNFP(x) \leftarrow teach(x, y), AC(y)$
C_{60}: $AC(x), BC(x) \leftarrow func(f_0, mary, x), notNFP(mary)$
C_{61}: $AC(x), BC(x) \leftarrow func(f_0, john, x), notNFP(john)$

Figure 5: Clauses obtained by application of ET rules.

- By definite-clause removal for *notBC*, C_{78}–C_{83} are transformed into C_{84}–C_{87} in Fig. 8.
- 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} in Fig. 9. 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} in Fig. 9.

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

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

C_{62}: $teach(john, ai) \leftarrow$
C_{63}: $notBC(ai) \leftarrow$
C_{64}: $notBC(x) \leftarrow teach(mary, x)$
C_{65}: $notNFP(x), notBC(y) \leftarrow teach(x, y)$
C_{66}: $notNFP(x) \leftarrow teach(x, y), func(f_0, john, y), notNFP(john)$
C_{67}: $notNFP(x) \leftarrow teach(x, y), func(f_0, mary, y), notNFP(mary)$
C_{68}: $mdt(paul, mary) \leftarrow teach(mary, x)$
C_{69}: $mdt(paul, john) \leftarrow teach(john, x), teach(mary, x)$
C_{70}: $mdt(paul, john) \leftarrow teach(john, ai)$
C_{71}: $mdt(paul, john) \leftarrow teach(john, x), func(f_0, mary, x), notNFP(mary), notBC(x)$
C_{72}: $mdt(paul, john) \leftarrow teach(john, x), func(f_0, john, x), notNFP(john), notBC(x)$
C_{73}: $mdt(paul, x), notNFP(x) \leftarrow$
C_{74}: $teach(mary, x) \leftarrow func(f_0, mary, x), notNFP(mary)$
C_{75}: $teach(john, x) \leftarrow func(f_0, john, x), notNFP(john)$
C_{76}: $notNFP(x) \leftarrow teach(x, ai)$
C_{77}: $notNFP(x) \leftarrow teach(x, y), teach(mary, y)$

Figure 6: Clauses obtained by application of ET rules.

C_{78}: $notBC(x) \leftarrow func(f_0, mary, x), notNFP(mary)$
C_{79}: $mdt(paul, x), notNFP(x) \leftarrow$
C_{80}: $notBC(ai) \leftarrow$
C_{81}: $mdt(paul, john) \leftarrow$
C_{82}: $mdt(paul, mary) \leftarrow func(f_0, mary, x), notNFP(mary)$
C_{83}: $notNFP(john) \leftarrow$

Figure 7: Clauses obtained by application of ET rules.

C_{84}: $mdt(paul, x), notNFP(x) \leftarrow$
C_{85}: $mdt(paul, john) \leftarrow$
C_{86}: $mdt(paul, mary) \leftarrow func(f_0, mary, x), notNFP(mary)$
C_{87}: $notNFP(john) \leftarrow$

Figure 8: Clauses obtained by application of ET rules.

C_{88}: $mdt(paul, mary) \leftarrow$
C_{89}: $mdt(paul, john) \leftarrow$

Figure 9: Clauses obtained by application of ET rules.

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 exit mapping used for constructing the output answer from the intersection of all models of *Cs*. Many logical problems, including proof problems and query-answering (QA) problems, can be transformed into MI problems preserving their answers. The theory in this paper therefore provides a foundation for many kinds of logical problem solving.

We introduced the concepts of target mapping and answer mapping, which are useful for inventing many kinds of ET rules for solving MI problems on extended clauses. The proposed solution schema for MI problems comprises the following steps: (i) formalize a given problem as a MI problem or map it into a MI problem, (ii) prepare ET rules from answers/target mappings, (iii) construct an ET sequence preserving answers/target mappings, and (iv) compute the answer by using some answer mapping (possibly constructed on some target mapping).

Many logical problems, among others, all proof problems and all QA problems on FOL_c, are mapped, by using new meaning-preserving Skolemization (Akama and Nantajeewarawat, 2011), into MI problems with function variables, and solved by ET computation proposed in this paper. When only conventional clauses without function variables are used, meaning-preserving Skolemization 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.

The ET-based solution method together with meaning-preserving Skolemization is very general and fundamental, since any combination of ET steps forms correct computation and the correctness of the method for a very large class of problems has been shown in this paper. By its generality, the theory developed in this paper makes clear a fundamental and central structure of representation and computation for logical problem solving.

ACKNOWLEDGEMENTS

This research was partially supported by JSPS KAKENHI Grant Numbers 25280078 and 26540110.

REFERENCES

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

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

Akama, K. and Nantajeewarawat, E. (2015a). A General Schema for Solving Model-Intersection Problems on a Specialization System by Equivalent Transformation. In *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015), Volume 2: KEOD*, pages 38–49, Lisbon, Portugal.

Akama, K. and Nantajeewarawat, E. (2015b). Function-variable Elimination and Its Limitations. In *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015), Volume 2: KEOD*, pages 212–222, Lisbon, Portugal.

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

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

Clark, K. L. (1978). Negation as Failure. In Gallaire, H. and Minker, J., editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York.

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

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

Gelfond, M. and Lifschitz, V. (1988). The Stable Model Semantics for Logic Programming. In *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press.

Gelfond, M. and Lifschitz, V. (1991). Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–386.

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

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

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

Tessaris, S. (2001). *Questions and Answers: Reasoning and Querying in Description Logic*. PhD thesis, Department of Computer Science, The University of Manchester, UK.