A General Schema for Solving Model-Intersection Problems on a Specialization System by Equivalent Transformation

Kiyoshi Akama[1] and Ekawit Nantajeewarawat[2]
[1]*Information Initiative Center, Hokkaido University, Hokkaido, Japan*
[2]*Computer Science, Sirindhorn International Institute of Technology, Thammasat University, Pathumthani, Thailand*

Keywords: Model-Intersection Problem, Query-Answering Problem, Equivalent Transformation, Problem Solving.

Abstract: A model-intersection problem (MI problem) is a pair of a set of clauses and an exit mapping. We define MI problems on specialization systems, which include many useful classes of logical problems, such as proof problems on first-order logic and query-answering (QA) problems in pure Prolog and deductive databases. The theory presented in this paper makes clear the central and fundamental structure of representation and computation for many classes of logical problems by (i) axiomatization and (ii) equivalent transformation. Clauses in this theory are constructed based on abstract atoms and abstract operation on them, which can be used for representation of many specific subclasses of problems with concrete syntax. Various computation can be realized by repeated application of many equivalent transformation rules, allowing many possible computation procedures, for instance, computation procedures based on resolution and unfolding. This theory can also be useful for inventing solutions for new classes of logical problems.

1 INTRODUCTION

This paper introduces a *model-intersection problem* (*MI problem*), which 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. The set of all MI problems constitutes a very large class of problems and is of great importance.

A QA problem 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 such a QA problem $\langle Cs, a \rangle$ is defined as the set of all ground instances of a that are logical consequences of Cs. A QA problem $\langle Cs, a \rangle$ is a MI problem $\langle Cs, \varphi_1 \rangle$, where for any set G of ground user-defined atoms, $\varphi_1(G)$ is the intersection of G and the set of all ground instances of a. 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. Many logic programming languages, including Datalog, Prolog, and other extensions of Prolog, deal with specific subclasses of QA problems.

The class of proof problems is also a subclass of MI problems. In contrast to a QA problem, a proof problem, is a "yes/no" problem; it is concerned with checking whether or not one given logical formula is a logical consequence of another given logical formula. 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.

Historically, proof problems were first solved (Robinson, 1965). Then QA problems on pure Prolog were solved based on the resolution principle, which is a solution for proof problems. This approach is proof-centered. It has been believed that computation of Prolog is an inference process. The theory of SLD resolution was used for the correctness of Prolog computation. Many solutions proposed so far for some other classes of logical problems are also basically proof-centered.

In contrast, it was shown in (Akama and Nantajeewarawat, 2013) that the set of all proof problems can be embedded into the set of all QA problems. This result supports a QA-centered approach to solving proof problems, i.e., first, develop a general solution for QA problems, and then, apply it as a solution for proof problems (Akama and Nantajeewarawat, 2012). Since a QA problem is a MI problem (as will be seen in Theorem 3), we have

$$\text{PROOF} \subset \text{QA} \subset \text{MI},$$

Akama, K. and Nantajeewarawat, E..
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
ISBN: 978-989-758-158-8

where PROOF, QA, and MI denote the class of all proof problems, the class of all QA problems, and the class of all MI problems, respectively. The class of all MI problems is larger than that of all QA problems, and it is a more natural class to be solved by the method presented in this paper. A general solution method for MI problems can be applied to any arbitrary QA problem and any arbitrary proof problem.

MI problems are axiomatically constructed on an abstract structure, called a *specialization system*. It consists of abstract atoms and abstract operations (extensions of variable-substitution operations) on atoms, called *specializations*. These abstract components can be any arbitrary mathematical objects as long as they satisfy given axioms. Abstract clauses can be built on abstract atoms. This is a sharp contrast to most of the conventional theories in logic programming, where concrete syntax is usually used. In Prolog, for example, usual first-order atoms and substitutions with concrete syntax are used, and there is no way to give a foundation for other forms of extended atoms and for various specialization operations other than the usual variable-substitution operation.

An axiomatic theory enables us to develop a very general theory. By instantiating a specialization system to a specific domain and by imposing certain restrictions on clauses, our theory can be applied to many subclasses of MI problems.

We proposed a general schema of solving MI problems by equivalent transformation (ET), where problems are solved by repeated simplification. We introduced the concept of target mapping and proposed three target mappings. Since transformation preserving a target mapping is ET, target mappings provide a strong foundation for inventing many ET rules for solving MI problems on clauses.

An ET-based solution consists of the following steps: (i) formalize an initial MI problem on some specialization system, (ii) prepare ET rules, (iii) construct an ET sequence, (iv) compute a set of models using a target mapping, (v) apply the set-intersection operation to the resulting set of models, and (vi) apply an exit mapping to the intersection result to obtain a solution.

To begin with, Section 2 recalls the concept of specialization system and formalizes MI problems on a specialization system. Section 3 defines the notions of a target mapping and a representative mapping, and introduces a schema for solving MI problems based on equivalent transformation (ET) preserving target mappings. The correctness of this solution schema is shown. Section 4 applies the general ET-based schema in Section 3 to the domain of clause sets with built-in constraint atoms. A target mapping, $\mathbb{MM}$, is introduced for associating with each clause set a collection of its specific models computed in a bottom-up manner. Section 5 shows an example of solution of a MI problem. Section 6 concludes the paper.

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

2 CLAUSES AND MODEL-INTERSECTION PROBLEMS

2.1 Specialization Systems

A substitution $\{X/f(a), Y/g(z)\}$ changes an atom $p(X,5,Y)$ in the term domain into $p(f(a),5,g(z))$. Generally, a substitution in first-order logic defines a total mapping on the set of all atoms in the term domain. Composition of such mappings is also realized by some substitution. There is a substitution that does not change any atom (i.e., the empty substitution). A ground atom in the term domain is a variable-free atom.

Likewise, in the string domain, substitutions for strings are used. A substitution $\{X/\text{“}aYbc\text{”}, Y/\text{“}xyz\text{”}\}$ changes an atom $p(\text{“}X5Y\text{”})$ into $p(\text{“}aYbc5xyz\text{”})$. Such a substitution for strings defines a total mapping on the set of all atoms that may include string variables. Composition of such mappings is also realized by some string substitution. There is a string substitution that does not change any atom (i.e., the empty substitution). A ground atom in the string domain is a variable-free atom.

A similar operation can be considered in the class-variable domain. Consider, for example, an atom $p(X\colon animal, Y\colon dog, Z\colon cat)$ in this domain, where $X\colon animal$, $Y\colon dog$, and $Z\colon cat$ represent an animal object, a dog object, and a cat object, respectively. When we obtain additional information that X is a dog, we can restrict $X\colon animal$ into $X\colon dog$ and the atom $p(X\colon animal, Y\colon dog, Z\colon cat)$ into $p(X\colon dog, Y\colon dog, Z\colon cat)$. By contrast, with new information that Z is a dog, we cannot restrict $Z\colon cat$ and the above atom since Z cannot be a dog and a cat at the same time. More generally, such a restriction operation may not be applicable to some atoms, i.e., it defines a partial mapping on the set of all atoms. Composition of such partial mappings is also a partial mapping, and we can determine some composi-

tion operation corresponding to it. An empty substitution that does not change any atom can be introduced. A ground atom in the class-variable domain is a variable-free atom.

In order to capture the common properties of such operations on atoms, the notion of a specialization system was introduced around 1990.

Definition 1. A *specialization system* Γ is a quadruple $\langle \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu \rangle$ of three sets $\mathcal{A}$, $\mathcal{G}$, and $\mathcal{S}$, and a mapping μ from $\mathcal{S}$ to $partialMap(\mathcal{A})$ that satisfies the following conditions:

1. $(\forall s', s'' \in \mathcal{S})(\exists s \in \mathcal{S}) : \mu(s) = \mu(s') \circ \mu(s'')$.
2. $(\exists s \in \mathcal{S})(\forall a \in \mathcal{A}) : \mu(s)(a) = a$.
3. $\mathcal{G} \subseteq \mathcal{A}$.

Elements of $\mathcal{A}$, $\mathcal{G}$, and $\mathcal{S}$ are called *atoms*, *ground atoms*, and *specializations*, respectively. The mapping μ is called the *specialization operator* of Γ. A specialization $s \in \mathcal{S}$ is said to be *applicable* to $a \in \mathcal{A}$ iff $a \in dom(\mu(s))$. □

Assume that a specialization system $\Gamma = \langle \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu \rangle$ is given. A specialization in $\mathcal{S}$ will often be denoted by a Greek letter such as θ. A specialization $\theta \in \mathcal{S}$ will be identified with the partial mapping $\mu(\theta)$ and used as a postfix unary (partial) operator on $\mathcal{A}$ (e.g., $\mu(\theta)(a) = a\theta$), provided that no confusion is caused. Let ε denote the identity specialization in $\mathcal{S}$, i.e., $a\varepsilon = a$ for any $a \in \mathcal{A}$. For any $\theta, \sigma \in \mathcal{S}$, let $\theta \circ \sigma$ denote a specialization $\rho \in \mathcal{S}$ such that $\mu(\rho) = \mu(\sigma) \circ \mu(\theta)$, i.e., $a(\theta \circ \sigma) = (a\theta)\sigma$ for any $a \in \mathcal{A}$.

2.2 User-defined Atoms, Constraint Atoms, and Clauses

Let $\Gamma_u = \langle \mathcal{A}_u, \mathcal{G}_u, \mathcal{S}_u, \mu_u \rangle$ and $\Gamma_c = \langle \mathcal{A}_c, \mathcal{G}_c, \mathcal{S}_c, \mu_c \rangle$ be specialization systems such that $\mathcal{S}_u = \mathcal{S}_c$. Elements of $\mathcal{A}_u$ are called *user-defined atoms* and those of $\mathcal{G}_u$ are called *ground user-defined atoms*. Elements of $\mathcal{A}_c$ are called *constraint atoms* and those of $\mathcal{G}_c$ are called *ground constraint atoms*. Hereinafter, assume that $\mathcal{S} = \mathcal{S}_u = \mathcal{S}_c$. Elements of $\mathcal{S}$ are called *specializations*. Let TCON denote the set of all true ground constraint atoms.

A *clause* on $\langle \Gamma_u, \Gamma_c \rangle$ is an expression of the form

$$a_1, \ldots, a_m \leftarrow b_1, \ldots, b_n,$$

where $m \geq 0$, $n \geq 0$, and each of $a_1, \ldots, a_m, b_1, \ldots, b_n$ belongs to $\mathcal{A}_u \cup \mathcal{A}_c$. It is a *ground* clause on $\langle \Gamma_u, \Gamma_c \rangle$ iff each of $a_1, \ldots, a_m, b_1, \ldots, b_n$ belongs to $\mathcal{G}_u \cup \mathcal{G}_c$. Let CLS denote the set of all clauses on $\langle \Gamma_u, \Gamma_c \rangle$.

2.3 Interpretations and Models

An *interpretation* is a subset of $\mathcal{G}_u$. Unlike ground user-defined atoms, the truth values of ground constraint atoms are predetermined by TCON (cf. Section 2.2) independently of interpretations. A ground constraint atom g is true iff $g \in$ TCON. It is false otherwise.

A ground clause $C = (a_1, \ldots, a_m \leftarrow b_1, \ldots, b_n)$ is *true* with respect to an interpretation $G \subseteq \mathcal{G}_u$ (in other words, G *satisfies* C) iff at least one of the following conditions is satisfied:

1. There exists $i \in \{1, \ldots, m\}$ such that $a_i \in G \cup$ TCON.
2. There exists $j \in \{1, \ldots, n\}$ such that $b_j \notin G \cup$ TCON.

A clause C is *true* with respect to an interpretation $G \subseteq \mathcal{G}_u$ (in other words, G *satisfies* C) iff for any specialization θ such that $C\theta$ is ground, $C\theta$ is true with respect to G. A *model* of a clause set $Cs \subseteq$ CLS is an interpretation that satisfies every clause in Cs.

Note that the standard semantics is taken in this paper, i.e., all models of a formula are considered instead of specific ones, such as those considered in the minimal model semantics (Clark, 1978; Lloyd, 1987) (i.e., the semantics underlying logic programming) and those considered in stable model semantics (Gelfond and Lifschitz, 1988; Gelfond and Lifschitz, 1991) (i.e., the semantics underlying answer set programming).

2.4 Model-Intersection (MI) Problems

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 $Cs \subseteq$ CLS.

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 determine one unique true subset of $\mathcal{G}_u$. 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 $\bigcap Models(Cs)$.

A *model-intersection problem* (*MI problem*) is a pair $\langle Cs, \varphi \rangle$, where $Cs \subseteq \text{CLS}$ 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 $ans_{MI}(Cs, \varphi)$, is defined by

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

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

2.5 Query-Answering (QA) Problems

Let $Cs \subseteq \text{CLS}$. For any $Cs' \subseteq \text{CLS}$, Cs' is a *logical consequence* of Cs, denoted by $Cs \models Cs'$, iff every model of Cs is also a model of Cs'. For any $a \in \mathcal{A}_u$, a is a *logical consequence* of Cs, denoted by $Cs \models a$, iff $Cs \models \{(a \leftarrow)\}$.

A *query-answering problem* (*QA problem*) in this paper is a pair $\langle Cs, a \rangle$, where $Cs \subseteq \text{CLS}$ and a is a user-defined atom in $\mathcal{A}_u$. The *answer* to a QA problem $\langle Cs, a \rangle$, denoted by $ans_{QA}(Cs, a)$, is defined by

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

Theorem 1. *For any $Cs \subseteq \text{CLS}$ and $a \in \mathcal{A}_u$,*

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

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

Proof: Let $Cs \subseteq \text{CLS}$ and $a \in \mathcal{A}_u$. By the definition of $\models$, for any ground atom $g \in \mathcal{G}_u$, $Cs \models g$ iff $g \in \bigcap Models(Cs)$. Then

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

Theorem 1 shows the importance of the intersection of all models of a clause set. By this theorem, the answer to a QA problem can be rewritten as follows:

Theorem 2. *Let $Cs \subseteq \text{CLS}$ and $a \in \mathcal{A}_u$. Then $ans_{QA}(Cs, a) = ans_{MI}(Cs, \varphi_1)$, where for any $G \subseteq \mathcal{G}_u$, $\varphi_1(G) = rep(a) \cap G$.*

Proof: It follows from Theorem 1 and the definition of φ_1 that $ans_{QA}(Cs, a) = \varphi_1(\bigcap Models(Cs)) = ans_{MI}(Cs, \varphi_1)$. $\square$

This is one way to regard a QA problem as a MI problem, which can be understood as follows: The set $\bigcap Models(Cs)$ often contains too many ground atoms. The set $rep(a)$ specifies a range of interest in the set $\mathcal{G}_u$. The exit mapping φ_1 focuses attention on the part $rep(a)$ by making intersection with it.

Theorem 3 below shows another way to formalize a QA problem as a MI problem.

Theorem 3. *Let $Cs \subseteq \text{CLS}$ and $a \in \mathcal{A}_u$. Then $ans_{QA}(Cs, a) = ans_{MI}(Cs \cup \{ans(a) \leftarrow a\}, \varphi_2)$, where for any $G \subseteq \mathcal{G}_u$, $\varphi_2(G) = \{x \mid ans(x) \in G\}$.*[1]

Proof: By Theorem 1 and the definition of φ_2,

$$\begin{aligned}
&ans_{QA}(Cs, a) \\
&= rep(a) \cap (\bigcap Models(Cs)) \\
&= \varphi_2(\bigcap Models(Cs \cup \{ans(a) \leftarrow a\})) \\
&= ans_{MI}(Cs \cup \{ans(a) \leftarrow a\}, \varphi_2). \quad \square
\end{aligned}$$

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.

3 SOLVING MI PROBLEMS BY EQUIVALENT TRANSFORMATION

A general schema for solving MI problems based on equivalent transformation is formulated and its correctness is shown (Theorem 8).

3.1 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 3.2.

[1] The expression $ans(a)$ is not an atom in the usual first-order logic space. One way to understand Theorem 3 in the context of the conventional first-order logic is (i) $ans(a)$ is interpreted as $ans(v_1, \ldots, v_n)$, where $v_1, \ldots, v_n$ are the variables occurring in a, and then (ii) $\varphi_2(G) = \{x \mid ans(x) \in G\}$ is interpreted as $\varphi_2(G) = \{a' \mid a' = ans(t_1, \ldots, t_n) \in G\}$.

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.

3.2 Target Mappings

Given a MI problem $\langle Cs,\varphi\rangle$, since $ans_{\mathrm{MI}}(Cs,\varphi) = \varphi(\bigcap Models(Cs))$, the answer to this MI problem is determined uniquely by $Models(Cs)$ and φ. As a result, we can equivalently consider a new MI problem with the same answer by switching from Cs to another clause set Cs' if $Models(Cs) = Models(Cs')$. According to the general terminologies defined in Section 3.1, on condition that $Models(Cs) = Models(Cs')$, transformation from $x = Cs$ into $x' = Cs'$ preserves $f = Models$ and is called ET with respect to $f = Models$, where (i) $x,x' \in pow(\mathrm{CLS})$ and (ii) $Models(x), Models(x') \in pow(pow(\mathcal{G}))$. We can also consider an ET sequence $[Cs_0,Cs_1,\ldots,Cs_n]$ of elements in $pow(\mathrm{CLS})$ with respect to a singleton set $\{Models\}$. MI problems can be transformed into simpler forms by ET preserving *Models*.

In order to use more partial mappings for simplification of MI problems, we extend our consideration from the specific mapping *Models* to a class of partial mappings, called GSETMAP, defined below.

Definition 2. GSETMAP is the set of all partial mappings from $pow(\mathrm{CLS})$ to $pow(pow(\mathcal{G}))$. □

As defined in Section 2.4, $Models(Cs)$ is the set of all models of Cs for any $Cs \subseteq \mathrm{CLS}$. Since a model is a subset of $\mathcal{G}$, *Models* is regarded as a total mapping from $pow(\mathrm{CLS})$ to $pow(pow(\mathcal{G}))$. Since a total mapping is also a partial mapping, the mapping *Models* is a partial mapping from $pow(\mathrm{CLS})$ to $pow(pow(\mathcal{G}))$, i.e., it is an element of GSETMAP.

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

Definition 3. A partial mapping $M \in$ GSETMAP is a *target mapping* iff for any $Cs \in dom(M)$, $\bigcap M(Cs) = \bigcap Models(Cs)$. □

It is obvious that:

Theorem 4. *The mapping Models is a target mapping.* □

Transformation preserving target mappings and computation of a target mapping constitute a method for solving MI problems in this paper.

For more general consideration, we introduce a binary relation $\preceq$ on GSETMAP as follows:

Definition 4. Let $M_1, M_2 \in$ GSETMAP. $M_1 \preceq M_2$ iff the following conditions are satisfied:

1. $dom(M_1) \subseteq dom(M_2)$.
2. For any Cs in $dom(M_1)$,

$$\bigcap M_1(Cs) = \bigcap M_2(Cs). \quad \square$$

Obviously, $\preceq$ is reflexive and transitive. It is also obvious that:

Proposition 1. *For any $M \in$ GSETMAP, M is a target mapping iff $M \preceq Models$.* □

By its definition, a target mapping M satisfies the following two conditions: (i) the domain of M is a subset of $pow(\mathrm{CLS})$, and (ii) for any clause set Cs in the domain of M, the intersection of all ground-atom sets in $M(Cs)$ is equal to the intersection of all models of Cs. By the first condition, since the domain of M can be smaller than that of the mapping *Models*, we can expect a more efficient program for computing $M(Cs)$ for Cs in the domain of M. By the second condition, the correctness of transformation and computation is guaranteed (Theorems 5 and 8).

Let $\langle Cs,\varphi\rangle$ be a MI problem. If M is a target mapping such that $M(Cs)$ is defined, then M can be used for computing the answer to $\langle Cs,\varphi\rangle$. More precisely:

Theorem 5. *Let $\langle Cs,\varphi\rangle$ be a MI problem and $M \in$ GSETMAP. If M is a target mapping and $Cs \in dom(M)$, then $ans_{\mathrm{MI}}(Cs,\varphi) = \varphi(\bigcap M(Cs))$.*

Proof: Assume that M is a target mapping and $Cs \in dom(M)$. Then $\bigcap M(Cs) = \bigcap Models(Cs)$. Consequently, $ans_{\mathrm{MI}}(Cs,\varphi) = \varphi(\bigcap Models(Cs)) = \varphi(\bigcap M(Cs))$. □

3.3 Representative Mappings

The relations "smaller than" and "finer than" on GSETMAP are introduced below.

Definition 5. Let $M_1, M_2 \in$ GSETMAP. M_1 is *smaller* than M_2 iff the following conditions are satisfied:

1. $dom(M_1) \subseteq dom(M_2)$.
2. For any $Cs \in dom(M_1)$, $M_1(Cs) \subseteq M_2(Cs)$. □

Definition 6. Let $M_1, M_2 \in$ GSETMAP. M_1 is *finer* than M_2 iff the following conditions are satisfied:

1. $dom(M_1) \subseteq dom(M_2)$.
2. For any $Cs \in dom(M_1)$ and any $m_2 \in M_2(Cs)$, there exists $m_1 \in M_1(Cs)$ such that $m_1 \subseteq m_2$. □

A smaller target mapping is basically preferable in order to reduce the cost of computing an answer. The concept of representative mapping defined below is useful for constructing small target mappings.

Definition 7. Let $M_1, M_2 \in$ GSETMAP. M_1 is a *representative mapping* of M_2 iff the following conditions are satisfied:

1. M_1 is smaller than M_2.
2. M_1 is finer than M_2. □

Theorem 6. *Let $M_1, M_2 \in$ GSETMAP. If M_1 is a representative mapping of M_2, then $M_1 \preceq M_2$.*

Proof: Suppose that M_1 is a representative mapping of M_2. Let $Cs \in dom(M_1)$. Since M_1 is smaller than M_2, $M_1(Cs) \subseteq M_2(Cs)$. Thus $\bigcap M_1(Cs) \supseteq \bigcap M_2(Cs)$. We show that $\bigcap M_1(Cs) \subseteq \bigcap M_2(Cs)$ as follows: Assume that $g \in \bigcap M_1(Cs)$. Let $m_2 \in M_2(Cs)$. Since M_1 is finer than M_2, there exists $m_1 \in M_1(Cs)$ such that $m_1 \subseteq m_2$. Since $g \in \bigcap M_1(Cs)$, g belongs to m_1. So $g \in m_2$, and thus, $g \in \bigcap M_2(Cs)$. It follows that $\bigcap M_1(Cs) = \bigcap M_2(Cs)$. Hence $M_1 \preceq M_2$. □

3.4 Solving MI Problems by Equivalent Transformation

Next, a schema for solving MI problems based on equivalent transformation (ET) preserving target mappings is formulated. The notions of preservation of target mappings, ET with respect to target mappings, and an ET sequence are obtained by specializing the general definitions in Section 3.1.

Let π be a mapping, called *state mapping*, from a given set STATE to the set of all MI problems. Elements of STATE are called *states*.

Definition 8. Let $\langle S, S' \rangle \in$ STATE $\times$ STATE. $\langle S, S' \rangle$ is an *ET step* with π iff if $\pi(S) = \langle Cs, \varphi \rangle$ and $\pi(S') = \langle Cs', \varphi' \rangle$, then $ans_{\mathrm{MI}}(Cs, \varphi) = ans_{\mathrm{MI}}(Cs', \varphi')$. □

Definition 9. A sequence $[S_0, S_1, \ldots, S_n]$ of elements of STATE is an *ET sequence* with π iff for any $i \in \{0, 1, \ldots, n-1\}$, $\langle S_i, S_{i+1} \rangle$ is an ET step with π. □

We can construct an ET step by using transformation preserving a target mapping. ET steps used for solving MI problems are mainly realized based on target mappings.

Theorem 7. *Let $S, S' \in$ STATE. Assume that $\pi(S) = \langle Cs, \varphi \rangle$, $\pi(S') = \langle Cs', \varphi \rangle$, and M is a target mapping such that $M(Cs) = M(Cs')$. Then $\langle S, S' \rangle$ is an ET step with π.*

Proof:

$$
\begin{aligned}
& ans_{\mathrm{MI}}(Cs, \varphi) \\
&= \varphi(\bigcap Models(Cs)) \\
&= \text{(since } M \text{ is a target mapping)} \\
&= \varphi(\bigcap M(Cs)) \\
&= \text{(since } M(Cs) = M(Cs')\text{)} \\
&= \varphi(\bigcap M(Cs')) \\
&= \text{(since } M \text{ is a target mapping)} \\
&= \varphi(\bigcap Models(Cs')) \\
&= ans_{\mathrm{MI}}(Cs', \varphi) \quad \square
\end{aligned}
$$

As shown below, we can solve MI problems by constructing ET sequences.

Theorem 8. *Assume that:*

- *$\langle Cs, \varphi \rangle$ is a MI problem.*
- *$[S_0, S_1, \ldots, S_n]$ is an ET sequence with π.*
- *$\pi(S_0) = \langle Cs, \varphi \rangle$ and $\pi(S_n) = \langle Cs_n, \varphi_n \rangle$.*
- *M is a target mapping such that $Cs_n \in dom(M)$.*

Then $ans_{\mathrm{MI}}(Cs, \varphi) = \varphi_n(\bigcap M(Cs_n))$.

Proof:

$$
\begin{aligned}
& ans_{\mathrm{MI}}(Cs, \varphi) \\
&= \varphi(\bigcap Models(Cs)) \\
&= \text{(since } [S_0, S_1, \ldots, S_n] \text{ is an ET sequence)} \\
&= \varphi_n(\bigcap Models(Cs_n)) \\
&= \text{(since } M \text{ is a target mapping)} \\
&= \varphi_n(\bigcap M(Cs_n)). \quad \square
\end{aligned}
$$

4 TARGET MAPPINGS FOR CLAUSES AND COMPUTATION

Next, three target mappings are introduced, i.e., τ_1 for sets of positive unit clauses, τ_2 for sets of definite clauses, and $\mathbb{MM}$ for sets of arbitrary clauses in

CLS. Based on these target mappings, an ET solution for MI problems on clauses is given according to the general schema of Section 3.

4.1 A Target Mapping for Sets of Positive Unit Clauses

A *positive unit clause* is a clause of the form $(a \leftarrow)$, where a is a user-defined atom. Let PUCL denote the set of all positive unit clauses. For any user-defined atom a, let $rep(a)$ denote the set of all ground instances of a.

A partial mapping $\tau_1 \in$ GSETMAP is defined as follows:

1. For any $F \subseteq$ PUCL, $\tau_1(F)$ is the singleton set
$$\{\bigcup\{rep(a) \mid (a \leftarrow) \in F\}\}.$$
2. For any $Cs \subseteq$ CLS such that $Cs \not\subseteq$ PUCL, $\tau_1(Cs)$ is undefined.

Theorem 9. *τ_1 is a representative mapping of Models and is a target mapping.*

Proof: Assume that $F \subseteq$ PUCL. Let $m_F = \bigcup\{rep(a) \mid (a \leftarrow) \in F\}$. Obviously, m_F is a model of F, i.e., $m_F \in Models(F)$. So τ_1 is smaller than *Models*. Now let $m \in Models(F)$. For any $(a \leftarrow) \in F$ and any $g \in rep(a)$, g is true with respect to m, i.e., $g \in m$. Then $m_F \subseteq m$. So τ_1 is also finer than *Models*, whence τ_1 is a representative mapping of *Models*. By Theorem 6 and Proposition 1, τ_1 is a target mapping. □

4.2 A Target Mapping for Sets of Definite Clauses

A *definite clause* is a clause whose left-hand side contains exactly one user-defined atom and no constraint atom. Let DCL denote the set of all definite clauses. Given a definite clause C, the atom in the left-hand side of C is called the *head* of C, denoted by $head(C)$, and the set of all user-defined atoms and constraint atoms in the right-hand side of C is called the *body* of C, denoted by $body(C)$. Assume that D is a set of definite clauses in DCL. The *meaning* of D, denoted by $\mathcal{M}(D)$, is defined as follows:

1. A mapping T_D on $pow(\mathcal{G})$ is defined by: for any set $G \subseteq \mathcal{G}$, $T_D(G)$ is the set
$$\{head(C\theta) \mid (C \in D)\ \&\ (\theta \in \mathcal{S})\ \&\ (\text{each user-defined atom in } body(C\theta) \text{ is in } G)\ \&\ (\text{each constraint atom in } body(C\theta) \text{ is true})\}.$$
2. $\mathcal{M}(D)$ is then defined as the set $\bigcup_{n=1}^{\infty} T_D^n(\varnothing)$, where $T_D^1(\varnothing) = T_D(\varnothing)$ and for each $n > 1$, $T_D^n(\varnothing) = T_D(T_D^{n-1}(\varnothing))$.

Then a partial mapping $\tau_2 \in$ GSETMAP is defined below.

1. For any $D \subseteq$ DCL, $\tau_2(D)$ is the singleton set $\{\mathcal{M}(D)\}$.
2. For any $Cs \subseteq$ CLS such that $Cs \not\subseteq$ DCL, $\tau_2(Cs)$ is undefined.

Theorem 10. *τ_2 is a representative mapping of Models and is a target mapping.*

Proof: Let $D \subseteq$ DCL. Since $\mathcal{M}(D)$ is a model of D, $\{\mathcal{M}(D)\} \subseteq Models(D)$. So τ_2 is smaller than *Models*. Let $m \in Models(D)$. Since $\mathcal{M}(D)$ is the least model of D, $\mathcal{M}(D) \subseteq m$. So τ_2 is finer than *Models*. Then τ_2 is a representative mapping of *Models*, and thus, by Theorem 6 and Proposition 1, it is a target mapping. □

Theorem 11. *For any $F \subseteq$ PUCL, $\tau_2(F) = \tau_1(F)$.*

Proof: For any $F \subseteq$ PUCL, $\tau_2(F) = \{\mathcal{M}(F)\} = \{\bigcup\{rep(a) \mid (a \leftarrow) \in F\}\} = \tau_1(F)$. □

4.3 A Target Mapping for Clause Sets

Given a clause C, the set of all user-defined atoms and constraint atoms in the left-hand side of C is denoted by $lhs(C)$ and the set of all those in the right-hand side of C is denoted by $rhs(C)$. A clause C is said to be *positive* if $lhs(C)$ is not empty; it is said to be *negative* otherwise.

It is assumed henceforth that (i) for any constraint atom c, $not(c)$ is a constraint atom; (ii) for any constraint atom c and any specialization θ, $not(c)\theta = not(c\theta)$; and (iii) for any ground constraint atom c, c is true iff $not(c)$ is not true.

The following notation is used for defining a target mapping $\mathbb{MM}$ for arbitrary clauses in CLS (Definition 10).

1. Let Cs be a set of clauses possibly with constraint atoms. MVRHS(Cs) is defined as the set $\{\text{MVRHS}(C) \mid C \in Cs\}$, where for any clause $C \in Cs$, MVRHS(C) is the clause obtained from C as follows: For each constraint atom c in $lhs(C)$, remove c from $lhs(C)$ and add $not(c)$ to $rhs(C)$.
2. Let Cs be a set of clauses with no constraint atom in their left-hand sides. For any $G \subseteq \mathcal{G}$, GINST(Cs, G) is defined as the set
$$\{\text{RMCON}(C\theta) \mid (C \in Cs)\ \&\ (\theta \in \mathcal{S})\ \&\ (\text{each user-defined atom in } C\theta \text{ is in } G)\ \&\ (\text{each constraint atom in } rhs(C\theta) \text{ is true})\},$$

where for any clause C', $\textsc{rmCon}(C')$ is the clause obtained from C' by removing all constraint atoms from it.

3. Let Cs be a set of clauses possibly with constraint atoms. For any $G \subseteq \mathcal{G}$, $\textsc{Inst}(Cs, G)$ is defined by

$$\textsc{Inst}(Cs, G) \;=\; \textsc{GInst}(\textsc{mvRhs}(Cs), G).$$

4. Let Cs be a set of ground clauses with no constraint atom. We can construct a set of definite clauses from Cs as follows: For each clause $C \in Cs$,

   - if $lhs(C) = \varnothing$, then construct a definite clause the head of which is $\bot$ and the body of which is $rhs(C)$, where $\bot$ is a special symbol not occurring in Cs;
   - if $lhs(C) \neq \varnothing$, then (i) select one arbitrary atom a from $lhs(C)$, and (ii) construct a definite clause the head of which is a and the body of which is $rhs(C)$.

   Let $\textsc{Dc}(Cs)$ denote the set of all definite-clause sets possibly constructed from Cs in the above way.

Proposition 2. *Let $Cs \subseteq \textsc{Cls}$. For any $m \subseteq \mathcal{G}$, m is a model of Cs iff m is a model of $\textsc{Inst}(Cs, \mathcal{G})$.*

Proof: $\textsc{Inst}(Cs, \mathcal{G})$ is obtained from Cs by (i) moving constraint atoms in the left-hand sides of clauses into their right-hand sides, (ii) instantiation of variables into ground terms, (iii) removal of clauses containing false constraint atoms in their right-hand sides, and (iv) removal of true constraint atoms from the remaining clauses. Each of the operations (i), (ii), (iii), and (iv) preserves models. □

A mapping $\mathbb{MM}$ is defined below.

Definition 10. A mapping $\mathbb{MM} \in \textsc{GSetMap}$ is defined by

$$\mathbb{MM}(Cs) \;=\; \{\mathcal{M}(D) \mid (D \in \textsc{Dc}(\textsc{Inst}(Cs, \mathcal{G}))) \;\&\; (\bot \notin \mathcal{M}(D))\}$$

for any $Cs \subseteq \textsc{Cls}$. □

Theorem 12. *$\mathbb{MM}$ is a representative mapping of Models and is a target mapping.*

Proof: First, we show that $\mathbb{MM}$ is smaller than *Models*. Let $Cs \subseteq \textsc{Cls}$. Suppose that $m \in \mathbb{MM}(Cs)$. Let $Cs' = \textsc{Inst}(Cs, \mathcal{G})$. Then there exists D such that $m = \mathcal{M}(D)$, $D \in \textsc{Dc}(Cs')$, and $\bot \notin \mathcal{M}(D)$. We show that m is a model of Cs' as follows:

- Let C_P be a positive clause in Cs'. Since $D \in \textsc{Dc}(Cs')$, there exists $C \in D$ such that $head(C) \in lhs(C_P)$ and $body(C) = rhs(C_P)$. Since m satisfies C, m also satisfies C_P. Hence m satisfies every positive clause in Cs'.
- Let C_N be a negative clause in Cs'. Since $D \in \textsc{Dc}(Cs')$, there exists $C' \in D$ such that $head(C') = \bot$ and $body(C') = rhs(C_N)$. Since $\bot \notin \mathcal{M}(D)$, m does not include $body(C')$. So $rhs(C_N) \not\subseteq m$, whence m satisfies C_N. Hence m satisfies every negative clause in Cs'.

So m is a model of Cs'. By Proposition 2, m is a model of Cs, i.e., $m \in Models(Cs)$.

Next, we show that $\mathbb{MM}$ is finer than *Models*. Let $Cs \subseteq \textsc{Cls}$. Suppose that $m' \in Models(Cs)$, i.e., m' is a model of Cs. Let $Cs' = \textsc{Inst}(Cs, \mathcal{G})$. By Proposition 2, m' is also a model of Cs'. Let D be a set of definite clauses obtained from Cs' by constructing from each positive clause C in Cs' a definite clause C' as follows:

1. Select an atom a from $lhs(C)$ as follows:
   (a) If $rhs(C) \subseteq m'$, then select an atom $a \in lhs(C) \cap m'$.
   (b) If $rhs(C) \not\subseteq m'$, then select an arbitrary atom $a \in lhs(C)$.
2. Construct C' as a definite clause such that $head(C') = a$ and $body(C') = rhs(C)$.

It is obvious that m' is a model of D. Let $m'' = \mathcal{M}(D)$. Since m'' is the least model of D, $m'' \subseteq m'$. Since m' is a model of Cs', m' satisfies all negative clauses in Cs'. Since $m'' \subseteq m'$, m'' also satisfies all negative clauses in Cs'. It follows that $\bot \notin \mathcal{M}(D)$. Hence $m'' \in \mathbb{MM}(Cs)$.

So $\mathbb{MM}$ is a representative mapping of *Models*. By Theorem 6 and Proposition 1, $\mathbb{MM}$ is a target mapping. □

Theorem 13. *For any $D \subseteq \textsc{Dcl}$, $\mathbb{MM}(D) = \tau_2(D)$.*

Proof: Let $D \subseteq \textsc{Dcl}$. Then $\textsc{Dc}(\textsc{Inst}(D, \mathcal{G}))$ is the singleton set $\{\textsc{Inst}(D, \mathcal{G})\}$. Obviously, $\mathcal{M}(D) = \mathcal{M}(\textsc{Inst}(D, \mathcal{G}))$ and $\bot \notin \mathcal{M}(\textsc{Inst}(D, \mathcal{G}))$. It follows that

$$\begin{aligned}\mathbb{MM}(D) \;&=\; \{\mathcal{M}(D') \mid (D' \in \textsc{Dc}(\textsc{Inst}(D, \mathcal{G}))) \;\&\; (\bot \notin \mathcal{M}(D'))\}\\ &=\; \{\mathcal{M}(\textsc{Inst}(D, \mathcal{G}))\}\\ &=\; \{\mathcal{M}(D)\}\\ &=\; \tau_2(D). \quad \square\end{aligned}$$

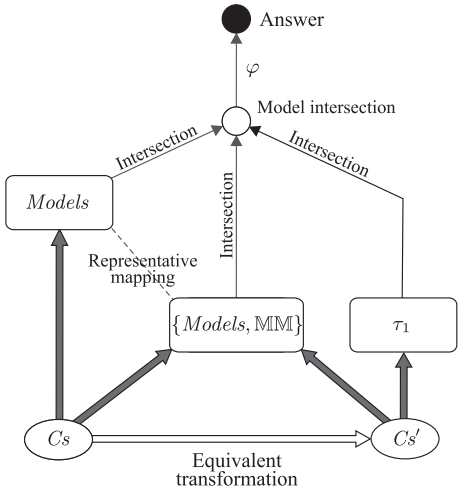

Figure 1: Target mappings and ET computation paths.

4.4 Computation Cost for Solving MI Problems

Given a set Cs of clauses, a user-defined atom a, and an exit mapping φ, 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. 1.

By Theorems 4, 9, and 12, each of *Models*, τ_1, and $\mathbb{MM}$ is a target mapping. By Theorem 8, with $M = \tau_1$, $ans_{\mathrm{MI}}(Cs, \varphi)$ can be obtained as follows:

1. Construct $S_0 = \langle Cs, \varphi \rangle$.
2. Construct an ET sequence based on *Models* and $\mathbb{MM}$ starting with S_0 and ending with $S_n = \langle Cs_n, \varphi_n \rangle$ such that $Cs_n \in dom(\tau_1)$.
3. $ans_{\mathrm{MI}}(Cs, \varphi) = \varphi_n(\bigcap \tau_1(Cs_n))$.

For the discussion below, the following notation is assumed:

- For any S, $S' \in$ STATE, let $trans(S, S')$ denote the transformation of S into S', and $time(trans(S, S'))$ denote the computation time required for this transformation step.
- Let π be a state mapping. For any target mapping τ and $S \in$ STATE, let $comp(\tau, S)$ denote the computation of $\varphi(\bigcap \tau(Cs))$, where $\pi(S) = \langle Cs, \varphi \rangle$, and let $time(comp(\tau, S))$ denote the amount of time required for this computation.

Using this notation, the time of the above solution by the ET sequence $[S_0, S_1, \ldots, S_n]$ with τ_1 above is evaluated by

$$T_{\tau_1} = \Sigma_{i=1}^{n} time(trans(S_{i-1}, S_i)) + time(comp(\tau_1, S_n)).$$

By the definition of τ_1, $time(comp(\tau_1, S_n))$ is very small. Assuming each transformation step in the ET sequence from S_0 to S_n is also very small, the value T_{τ_1} is small enough and the solution by this ET sequence with τ_1 can be efficient. This is a basic strategy to obtain an efficient solution for a MI problem.

In order to use τ_1 after repeated equivalent transformation, the clause set Cs_n determined by $\pi(S_n)$, where S_n is the final state obtained from the ET sequence $[S_0, S_1, \ldots, S_n]$, must be inside $dom(\tau_1)$. In other words, the role of the ET sequence $[S_0, S_1, \ldots, S_n]$ is to construct Cs_n that enters $dom(\tau_1)$ starting from S_0.

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}: $mayDoThesis(x,y) \leftarrow curr(x,z), expert(y,z), St(x), Tp(z), FP(y), AC(w), teach(y,w)$
C_{11}: $mayDoThesis(x,y) \leftarrow St(x), NFP(y)$
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}: $teach(y,x), NFP(y) \leftarrow FP(y), funcf_0(y,x)$
C_{25}: $Co(x), NFP(y) \leftarrow FP(y), funcf_0(y,x)$
C_{26}: $funcf_0(john, ai) \leftarrow$
C_{27}: $\leftarrow funcf_0(mary, ai)$

Figure 2: Clauses representing the background knowledge of the modified *mayDoThesis* problem.

5 EXAMPLE

Usual first-order atoms are used for illustration below. To apply the proposed theory in this section, a specialization system $\langle \mathcal{A}_{\mathrm{u}}, \mathcal{G}_{\mathrm{u}}, \mathcal{S}, \mu_{\mathrm{u}} \rangle$ corresponding to the usual first-order space is used, where $\mathcal{A}_{\mathrm{u}}$ is the set of all first-order atoms, $\mathcal{G}_{\mathrm{u}}$ is the set of all ground first-order atoms, $\mathcal{S}$ is the set of all substitutions on

$\mathcal{A}_u$, and μ_u provides the specialization operation corresponding to the usual application of substitutions in $\mathcal{S}$ to atoms in $\mathcal{A}_u$.

5.1 Problem Description

Let Cs be the set consisting of the clauses C_1–C_{27} in Fig. 2. These clauses are obtained from the *mayDoThesis* problem given in (Donini et al., 1998) with some modification.[2] All atoms appearing in Fig. 2 belong to $\mathcal{A}_u$. 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 $mayDoThesis(s,p)$, $curr(s,t)$, $expert(p,t)$, $exam(s,c)$, and $subject(c,t)$ 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.

Let a be the atom $mayDoThesis(paul,x)$. We consider the QA problem $\langle Cs,a\rangle$, which is to find all students who may do their theses with *paul*. Let φ be defined by: for any $G \subseteq \mathcal{G}_u$,

$$\varphi(G) = \{mayDoThesis(paul,x) \mid ans(x) \in G\},$$

where *ans* is a unary predicate denoting "answer." The QA problem $\langle Cs,a\rangle$ above can then be transformed into a MI problem $\langle Cs \cup \{C_0\},\varphi\rangle$, where C_0 is the clause given by:

$$C_0\colon\ \ ans(x) \leftarrow mayDoThesis(paul,x)$$

Using rules for transformation of clauses given in Sections 5.2–5.4, how to compute the answer to the MI problem $\langle Cs \cup \{C_0\},\varphi\rangle$ is illustrated in Section 5.5.

5.2 Unfolding Operation

Assume that:

- $Cs \subseteq$ CLS.
- D is a set of definite clauses in CLS.
- *occ* is an occurrence of an atom b in the right-hand side of a clause C in Cs.

[2] To represent the original *mayDoThesis* problem in a clausal form, extended clauses with function variables are used. To change atoms with function variables into user-defined atoms, the $funcf_0$ predicate is used in the clauses C_{24}–C_{27}.

By unfolding Cs using D at *occ*, Cs is transformed into

$$(Cs - \{C\}) \cup (\bigcup\{resolvent(C,C',b) \mid C' \in D\}),$$

where for each $C' \in D$, $resolvent(C,C',b)$ is defined as follows, assuming that ρ is a renaming substitution for usual variables such that C and $C'\rho$ have no usual variable in common:

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

The resulting clause set is denoted by UNFOLD(Cs, D, occ).

5.3 ET by Unfolding and Definite-clause Removal

For any predicate p, let $Atoms(p)$ denote the set of all atoms having the predicate p. Equivalent transformation (ET) of clauses using unfolding and using definite-clause removal are formulated below.

Theorem 14. *Let $Cs \subseteq$ CLS and $a \in \mathcal{A}_u$. Assume that:*

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

Then the following two sets are equal:

- $(\bigcap Models(Cs)) \cap rep(a)$
- $(\bigcap Models(\text{UNFOLD}(Cs,D,occ))) \cap rep(a)$. □

Theorem 15. *Let $Cs \subseteq$ CLS and $a \in \mathcal{A}_u$. Assume that:*

1. *q is the predicate of the query atom a.*

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

Then the following two sets are equal:

- $(\bigcap Models(Cs)) \cap rep(a)$
- $(\bigcap Models(Cs - D)) \cap rep(a)$ □

5.4 Other Transformations

5.4.1 Elimination of Subsumed Clauses and Elimination of Valid Clauses

A clause C_1 is said to *subsume* a clause C_2 iff there exists a substitution θ for usual variables such that $lhs(C_1)\theta \subseteq lhs(C_2)$ and $rhs(C_1)\theta \subseteq rhs(C_2)$. If a clause set *Cs* contains clauses C_1 and C_2 such that C_1 subsumes C_2, then *Cs* can be transformed into $Cs - \{C_2\}$.

A clause is *valid* iff all of its ground instances are true. Given a clause *C*, if some atom in $rhs(C)$ belongs to $lhs(C)$, then *C* is valid. A valid clause can be removed.

5.4.2 Side-change Transformation

Assume that *p* is a predicate occurring in a clause set *Cs* and *p* does not appear in a query atom under consideration. The clause set *Cs* can be transformed by changing the clause sides of *p*-atoms as follows: First, determine a new predicate *not p* for *p*. Next, move all *p*-atoms in each clause to their opposite side in the same clause (i.e., from the left-hand side to the right-hand side and vice versa) with their predicates being changed from *p* to *not p*. Side-change transformation is useful for decreasing the number of atoms in a multi-head clause (i.e., a clause whose left-hand side contains more than one atom) in *Cs* when (i) every negative clause in *Cs* has at most one *p*-atom in its right-hand side and (ii) every non-negative clause in *Cs* has more *p*-atoms in its left-hand side than those in its right-hand side.

C_{28}: $teach(john, ai) \leftarrow$
C_{29}: $AC(ai) \leftarrow$
C_{30}: $AC(x) \leftarrow teach(mary, x)$
C_{31}: $\leftarrow AC(x), BC(x)$
C_{32}: $AC(x), BC(x) \leftarrow Co(x)$
C_{33}: $Co(x) \leftarrow AC(x)$
C_{34}: $Co(x) \leftarrow BC(x)$
C_{35}: $\leftarrow NFP(x), teach(x,y), Co(y)$
C_{36}: $ans(y) \leftarrow NFP(x)$
C_{37}: $ans(john) \leftarrow AC(x), teach(john, x), Co(ai)$
C_{38}: $ans(mary) \leftarrow AC(x), teach(mary, x), Co(ai)$
C_{39}: $ans(john) \leftarrow AC(x), teach(john, x),$
$NFP(john), Co(ai)$
C_{40}: $ans(mary) \leftarrow AC(x), teach(mary, x),$
$NFP(mary), Co(ai)$
C_{41}: $teach(john, ai), NFP(john) \leftarrow$
C_{42}: $Co(ai), NFP(john) \leftarrow$

Figure 3: Clauses obtained by application of unfolding and application of basic transformation rules.

C_{43}: $ans(x), notNFP(x) \leftarrow$
C_{44}: $notNFP(john) \leftarrow$
C_{45}: $ans(john) \leftarrow$
C_{46}: $\leftarrow BC(ai)$

Figure 4: Clauses obtained by further application of transformation rules.

5.5 ET Computation

The clause set $Cs \cup \{C_0\}$, consisting of C_0–C_{27}, given in Section 5.1 is transformed using ET rules provided by Sections 5.2–5.4 as follows:

- By (i) unfolding using the definitions of the predicates *mayDoThesis*, *FP*, *Tp*, *curr*, *subject*, *expert*, *St*, *exam*, $funcf_0$, and *FM*, (ii) removing these definitions using definite-clause removal, and (iii) removal of valid clauses, the clauses C_0–C_{27} are transformed into the clauses C_{28}–C_{42} in Fig. 3.
- Side-change transformation for *NFP* enables (i) unfolding using the definitions of *teach*, *Co*, and *AC*, (ii) elimination of these definitions using definite-clause removal, (iii) removal of valid clauses, and (iv) elimination of subsumed clauses. By such side-change transformation followed by transformation of these four types, C_{28}–C_{42} are transformed into the clauses C_{43}–C_{46} in Fig. 4.
- Side-change transformation for *notNFP* enables unfolding using the definitions of *BC* and *NFP*. By unfolding and definite-clause removal, C_{43}–C_{46} are transformed into C_{45}, i.e., $(ans(john) \leftarrow)$.

As a result, the MI problem $\langle Cs \cup \{C_0\}, \varphi \rangle$ in Section 5.1 is transformed equivalently into the MI problem $\langle \{(ans(john) \leftarrow)\}, \varphi \rangle$. Hence

$$
\begin{aligned}
& ans_{\mathrm{MI}}(Cs \cup \{C_0\}, \varphi) \\
&= ans_{\mathrm{MI}}(\{(ans(john) \leftarrow)\}, \varphi) \\
&= \varphi(\bigcap Models(\{(ans(john) \leftarrow)\})) \\
&= \{mayDoThesis(paul, john)\}.
\end{aligned}
$$

6 CONCLUSIONS

A model-intersection problem (MI problem) is a pair $\langle Cs, \varphi \rangle$, where Cs is a set of clauses and φ is an exit mapping used for constructing the output answer from the intersection of all models of Cs. The proposed ET-based solution for MI problems consists of the following steps: (i) formalize a given problem as an MI problem on some specialization system, (ii) prepare ET rules from clauses, (iii) construct an ET sequence, (iv) compute a set of models using a target mapping, (v) apply the set-intersection operation to the resulting set of models, and (vi) apply an exit mapping to the intersection result to obtain a solution.

The class of MI problems considered in this paper has many parameters, such as abstract atoms, specializations, restriction on forms of clauses, etc. By instantiating these parameters, we can obtain theories for subclasses of QA and proof problems corresponding to conventional clause-based theories, such as datalog, Prolog, and many other extensions of Prolog.

We introduced the concept of target mapping and proposed three target mappings, i.e., τ_1 for sets of positive unit clauses, τ_2 for sets of definite clauses, and $\mathbb{MM}$ for arbitrary sets of clauses. These target mappings provide a strong foundation for inventing many ET rules for solving MI problems on clauses. Most kinds of ET rules, including the resolution and factoring ET rules, are realized by transformations that preserve these target mappings. For instance, a proof based on the resolution principle can be regarded as ET computation using the resolution and factoring ET rules. By introducing new ET rules, we can devise a new proof method (Akama and Nantajeewarawat, 2013). By inventing additional ET rules, we have been successful in solving a large class of QA problems (Akama and Nantajeewarawat, 2014).

By instantiation, the class of MI problems on specialization systems produces, among others, one of the largest classes of logical problems with first-order atoms and substitutions. The ET solution has been proved to be very general and fundamental since its correctness for such a large class of problems has been shown in this paper. By its generality, the theory developed in this paper makes clear the 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. (2012). Proving Theorems Based on Equivalent Transformation Using Resolution and Factoring. In *Proceedings of the Second World Congress on Information and Communication Technologies*, WICT 2012, pages 7–12, Trivandrum, India.

Akama, K. and Nantajeewarawat, E. (2013). Embedding Proof Problems into Query-Answering Problems and Problem Solving by Equivalent Transformation. In *Proceedings of the 5th International Conference on Knowledge Engineering and Ontology Development*, pages 253–260, Vilamoura, Portugal.

Akama, K. and Nantajeewarawat, E. (2014). Equivalent Transformation in an Extended Space for Solving Query-Answering Problems. In *Proceedings of the 6th Asian Conference on Intelligent Information and Database Systems*, LNAI 8397, pages 232–241, Bangkok, Thailand.

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.

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.

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.

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.