REASONING BY ANALOGY AS A PARTIAL IDENTITY BETWEEN MODELS

Haraguchi, Makoto
Research Institute of Fundamental Information Science, Kyushu University

Arikawa, Setsuo
Research Institute of Fundamental Information Science, Kyushu University

https://doi.org/10.5109/13381

出版情報：Bulletin of informatics and cybernetics. 22 (3/4), pp.131-147, 1987-03. 統計科学研究会
バージョン：
権利関係：

Bulletin of Informatics and Cybernetics Vol. 22, No. 3～4, 1987

REASONING BY ANALOGY AS A PARTIAL IDENTITY BETWEEN MODELS

By

Makoto HARAGUCHI* and Setsuo ARIKAWA*

Abstract

We present in this paper a formal theory of reasoning by analogy. We are mainly concerned with three subjects: a formal definition of analogy, a formalization of the reasoning in terms of deduction, and a method for realizing the reasoning in a logic programming system. First we assume that each domain for the reasoning is the least model for logic program. Then we consider an analogy as a partial identity between the models. Secondly we introduce a notion of rule transformation which transforms rules in one domain into those in the other. Then we can formalize the reasoning as a system with three inference rules: instantiation of rules, modus ponens, and the rule transformation. Finally, based on the formalization, we present an extended pure-Prolog interpreter which performs the detection of analogy and the reasoning by the partial identity at the same time.

1. Introduction

We often make use of analogy in solving various problems over various domains. Generally we have two domains. One is concerned with the current problem in inquiry, and the other is assumed to be well-known. Given such domains, we first attempt to detect some analogies between the domains, and to transform the knowledge of the well-known domain into those of the other. The transformed knowledge is not always true. However the detected analogy suggests that the knowledge may be true and that it can be used to infer some facts.

In the present paper, we are mainly concerned with the problem of formalizing the reasoning by analogy so that we can realize it by a computer. Some computational methods for the reasoning have been discussed from various viewpoints. However, since each of them has been developed for and depending on its own domain, any common framework to consider fundamental concepts about the reasoning has not been discussed. It seems to the authors that this should come from the lack of mathematical formulation of the reasoning.

From this viewpoint, one of the authors has tried to formalize the reasoning [4, 5]. The studies are considered to be a step to reasoning by analogy by a computer. However it is true that there have still remained some problems to be overcome.

---

* Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan.

In the previous papers [4, 5], the domains for the reasoning are represented by finite sets of facts (ground atoms). Given such representations, we have discussed an analogy as a partial identity between the domains. Moreover we have defined a notion of rule transformation [5] by which some ground rules in one domain are transformed to those in the other. The transformed rules are used to derive some facts. The notion of partial identity thus defined was purely syntactic, and hence it was not suited for considering the semantic aspects of analogy.

The purpose of the present paper is to modify and extend the notion of partial identity so that we can consider the semantic aspects of analogy to some extent. The new formulation is summed up as follows:

(1) The domain for the reasoning is represented by a logical formula, and is an intended model for it. In this paper, the logical formula and the intended model are assumed to be a logic program and the least (Herbrand) model for it, respectively.

(2) Given two logic programs, an analogy we consider is a partial identity between their least models. The partial identity consists of a pairing of ground terms and a one-to-one relation of facts in the domains. The paired terms are treated just like a single term in order to put two domains together.

(3) To define the reasoning based on the partial identity, we consider Winston's analogy-based reasoning [10, 11], in which "similar" reasons are assumed to lead to "similar" effects. We introduce a notion of rule transformation to deal with the reasoning in a framework of deduction. In fact, we can formally define the analogy-based reasoning in terms of logic programming with the function of transforming rules. Then we precisely define the set of ground atoms which are reasoned based on some partial identity.

To realize the reasoning as in the above, we must design a procedure to detect a partial identity. Such a procedure plays an important role in the reasoning, since the reasoning is dependent on the chosen analogy. Here it should be noticed that the process for detecting the desired partial identity generally requires a large amount of computation. To avoid the increases of computational complexity, it is considered better to have a reasoning procedure which detects and reasons at the same time. Such a reasoning procedure has no information about the partial identity at the beginning of its computation, and it will partially get the information concerning the partial identity at each stage of reasoning process. In fact, we present a Prolog program, which is an extension of pure-Prolog interpreter, to carry out the reasoning in such a way.

In Section 2, we define the notion of partial identity and present a condition for a pairing of terms to define the partial identity. In Section 3, we formally define the reasoning based on the partial identity in terms of logic programming. In Section 4, we introduce the notion of reasoning procedure. Finally, in Section 5, we present an extended Prolog interpreter which carries out the reasoning.

2. Analogy as a Partial Identity

We define in this section a notion of partial identity. Since we assume that each domain for the reasoning is represented by a logic program, we first give some necessary definitions concerning logic programs.

A definite clause is a clause of the form

$$A \leftarrow B_1, \cdots, B_n \quad (n \geqq 0),$$

where A and B_j are positive literals. We call the definite clause a rule. A logic program is a finite set of rules, and is simply called a program.

Since a program P is a set of clauses, any model for P can be considered as the corresponding Herbrand model. (For instance, see [8].) All Herbrand models for P have the same domain $U(P)$, called the Herbrand universe, and the same meaning of function symbol appearing in P. $U(P)$ is defined to be the set of all ground terms whose symbols are all in P. Moreover the set of all terms consisting of symbols in P is denoted by $L(P)$. The meaning of n-ary function symbol f is defined to be a function $I_f : U(P)^n \rightarrow U(P)$, where

$$I_f(t_1, \cdots, t_n) = f(t_1, \cdots, t_n).$$

We also need the notion of Herbrand base. The Herbrand base $B(P)$ of a program P is defined as

$$B(P) = \{p(t_1, \cdots, t_n) \mid p \text{ is an } n\text{-ary predicate symbol appearing in } P, \text{ and } t_j \in U(P)\}.$$

An element in $B(P)$ is called a ground atom.

Then, each Herbrand model (interpretation) is specified by a subset of $B(P)$. The subset consists of all ground atoms which are true under the model (interpretation). By the model intersection property [1], the intersection of all Herbrand models for P is also a model for P. This model is called the least model for P, and is denoted by $M(P)$.

PROPOSITION 2.1. ([1, 8]) *$M(P)$ is the set of all ground atoms which are logical consequences of P.*

From Proposition 2.1, we take $M(P)$ as the formal meaning of P, and call an element in $M(P)$ a fact. In what follows, we consider only the Herbrand models, and simply call them models.

We define the correspondence of analogy by a pairing of elements in domains.

DEFINITION 2.1. Let P_1 and P_2 be logic programs. We call a finite subset of $U(P_1) \times U(P_2)$ a *pairing of terms*. For a pairing ψ, we define the set ψ^+ to be the smallest set satisfying the following conditions:

$$\psi \subseteqq \psi^+, \tag{2.1}$$

$$\text{if } \langle t_1, t_1' \rangle, \cdots, \langle t_n, t_n' \rangle \in \psi^+, \text{ then } \langle f(t_1, \cdots, t_n), f(t_1', \cdots, t_n') \rangle \in \psi^+, \tag{2.2}$$

where f is a function symbol appearing in both P_1 and P_2,

Then relation ψ^+ is not always one-to-one. For example, consider the following pairing ψ:

$$\psi=\{\langle a, a'\rangle, \langle b, b'\rangle, \langle c, on(a', b')\rangle\},$$

where "*on*" is a binary function symbols, and the other symbols appearing in ψ are constant symbols. Then we have

$$\langle c, on(a', b')\rangle, \langle on(a, b), on(a', b')\rangle\in\psi^+,$$

which means that two distinct terms $on(a, b)$ and c in P_1 are related to the same term $on(a', b')$ in P_2.

As mentioned in the introduction, we consider an analogy as a partial identity. The partial identity is a function of putting together two distinct domains for the reasoning. To put them together, we first identify some paired elements, one from each domain, and treat them just like a single element. According to such an identification of elements, we identify some paired facts which hold in each domain. Thus, by the partial identity, two distinct elements in one domain are never paired with the same element in another domain. Since our domains for the reasoning are Herbrand models, distinct terms denote distinct elements. Hence the set ψ^+ of paired terms should be one-to-one. The pairing ψ in the example above is not allowed as a pairing of partial identity. In what follows, a pairing of partial identity is simply called a partial identity.

DEFINITION 2.2. A pairing ψ is called a *partial identity* if ψ^+ is a one-to-one relation of terms.

The paired atoms identified by the partial identity are defined as follows:

DEFINITION 2.3. For a partial identity ψ, two ground atoms $\alpha\in B(P_1)$ and $\alpha'\in B(P_2)$ are said to be *identified by* ψ, denoted by $\alpha\psi\alpha'$, if α and α' are compatible, that is, they can be written as

$$\alpha=p(t_1, \cdots, t_n),$$
$$\alpha'=p(t_1', \cdots, t_n'),$$

for some predicate symbol p, and α and α' are syntactically the same up to the pairing ψ, that is, $\langle t_i, t_i'\rangle\in\psi^+$.

Then the set of paired facts which are identified by ψ is

$$ID(P_1, P_2; \psi)=\{\langle\alpha, \alpha'\rangle \mid \alpha\in M(P_1), \alpha'\in M(P_2), \alpha\psi\alpha'\}.$$

PROPOSITION 2.2. *$ID(P_1, P_2; \psi)$ is a one-to-one relation of facts.*

Now we present a condition for a pairing to define a partial identity. The condition is called *Extended Partial Identity Condition* (*EPIC*, for short).

DEFINITION 2.4. A pairing ψ is said to *violate EPIC* if there exist $\langle t, t'\rangle$, $\langle t_j, t_j'\rangle\in\psi$ and a term $T(X_1, \cdots, X_n)$ in $L(P_1)\cap L(P_2)$, that contains no constant symbols, such that

$$t=T(t_1, \cdots, t_n) \quad\text{and}\quad t'\neq T(t_1', \cdots, t_n'),$$

or

$$t\neq T(t_1, \cdots, t_n) \quad\text{and}\quad t'=T(t_1', \cdots, t_n').$$

Moreover, if otherwise, ϕ is said to *satisfy EPIC*.

THEOREM 2.1. *A pairing ϕ is a partial identity iff ϕ satisfies EPIC.*

To prove the theorem, we need a definition.

DEFINITION 2.5. A pairing $\phi \subseteqq U(P_1) \times U(P_2)$ is said to be a *generator* if there exist no $\langle t, t' \rangle$, $\langle t_j, t'_j \rangle$ in ϕ and a non-variable term $T(X_1, \cdots, X_n)$ containing no constant symbols such that

$$t = T(t_1, \cdots, t_n),$$
$$t' = T(t'_1, \cdots, t'_n).$$

PROOF OF THEOREM 2.1. It is clear that ϕ^+ is not one-to-one if ϕ violates *EPIC*. Hence we prove only that ϕ violates *EPIC* if ϕ^+ is not one-to-one.

First we prove this for a generator ϕ. Assume that ϕ violates *EPIC*, and assume that ϕ^+ is not one-to-one. Then there exist pairs $\langle t_i, t'_i \rangle$ in ϕ^+ $(i=1, 2)$ such that

$$t_1 = t_2 \quad \text{and} \quad t'_1 \neq t'_2, \tag{2.3}$$

or

$$t_1 \neq t_2 \quad \text{and} \quad t'_1 = t'_2. \tag{2.4}$$

Assume that (2.3) holds and $\langle t_j, t'_j \rangle \in \phi^+ \backslash \phi$ $(j=1, 2)$. The proofs for the other cases are similar, and therefore omitted. From the assumption, there exist $\langle s_j, s'_j \rangle$, $\langle u_k, u'_k \rangle$ in ϕ and a non-variable term T_i containing no constants such that

$$t_1 = T_1(s_1, \cdots, s_n), \quad t'_1 = T_1(s'_1, \cdots, s'_n),$$
$$t_2 = T_2(u_1, \cdots, u_m), \quad t'_2 = T_2(u'_1, \cdots, u'_m).$$

Case 1: T_1 and T_2 are variant. In this case, $s_j = u_j$ holds for each j, and there exists i such that $s'_i \neq u'_i$. This implies that ϕ is not one-to-one, and therefore ϕ violates *EPIC*.

Case 2: T_1 and T_2 are not variant. Let

$$T_1 = T_1(X_1, \cdots, X_n), \quad T_2 = T_2(Y_1, \cdots, Y_m).$$

There exists a disagreement $\langle V_1, V_2 \rangle$. Here the disagreement is a symbol position at which T_1 and T_2 have distinct symbols. V_i is the subexpression extracted from T_i at that symbol position. Since $t_1 = t_2$, one of V_j is a variable and the other is a non-variable term. Without loss of generality, we assume that V_1 is the variable X_q. Let

$$V_2 = V_2(Y_{21}, \cdots, Y_{2k}),$$

where $\{Y_{21}, \cdots, Y_{2k}\} \subseteqq \{Y_1, \cdots, Y_m\}$. Then, $t_1 = t_2$ implies

$$s_q = V_2(u_{21}, \cdots, u_{2k}).$$

Since ϕ is a generator, this implies that

$$s'_q \neq V_2(u'_{21}, \cdots, u'_{2k}).$$

Thus ϕ violates *EPIC*.

Finally, for a pairing ϕ which is not a generator, we consider the following generator ϕ_0:

$$\psi_0 \subseteq \psi, \tag{2.5}$$

$$\psi^+ = \psi_0^+. \tag{2.6}$$

Let $\psi^+ = \psi_0^+$ be not one-to-one. Since ψ_0 is a generator, ψ_0 violates *EPIC*. From (2.5), this implies that ψ violates *EPIC*, which completes the proof.

The condition *EPIC* concerns unifiability of terms, and plays an essential role in realizing the reasoning in a logic programming system. The details are discussed in Section 5.

3. Reasoning Based on Partial Identity

In this section, we present a formal reasoning based on a partial identity. First we state the principle of reasoning by analogy:

Assume that, in P_1, the premises $\beta_1, \cdots, \beta_n$ logically imply a fact α. Assume also that the analogous premises $\beta'_1, \cdots, \beta'_n$ hold in P_2. Then we reason an atom α' in P_1 which is analogous to α.

The reasoning above is conceptually due to Winston's analogy-based reasoning [11] based on the causal structures of domains. Since our analogy between $M(P_1)$ and $M(P_2)$ is a partial identity, we can restate the statement as follows:

Assume that $\beta_1, \cdots, \beta_n$ in $M(P_1)$ logically imply α in P_1, and assume that there exist $\beta'_1, \cdots, \beta'_n$ in $M(P_2)$ such that $\beta_j \psi \beta'_j$ for all j. Then we reason an atom α' in $B(P_2)$ such that $\alpha \psi \alpha'$.

The reasoned ground atom α' is not always a logical consequence of P_2. Hence the reasoning is beyond the usual deduction. Our goal is to describe the reasoning in terms of deduction. We need the following definition:

DEFINITION 3.1. Let

$$R_1 = (\alpha \leftarrow \beta_1, \cdots, \beta_n),$$
$$R_2 = (\alpha' \leftarrow \beta'_1, \cdots, \beta'_n)$$

be two ground rules ($n \geqq 1$) whose symbols are all appearing in P_1 and P_2, respectively. Let ψ and I_j be a partial identity and an Herbrand interpretation of P_j, respectively. Then the rules R_1 and R_2 are called ψ-analogous w.r.t. I_1 and I_2, if $\beta_j \in I_1$, $\beta'_j \in I_2$, $\alpha \psi \alpha'$, and $\beta_j \psi \beta'_j$. In this case, R_1 (R_2) is called a ψ-analogue of R_2 (R_1) w.r.t. I_1 and I_2.

We call the conversion of R_1 into R_2, or R_2 into R_1, a rule transformation. In what follows, we represent the transformation by the following schema:

$$\frac{\alpha \leftarrow \beta_1, \cdots, \beta_n}{\alpha' \leftarrow \beta'_1, \cdots, \beta'_n} \ (\psi, I_1, I_2),$$

where $\alpha \psi \alpha'$, $\beta_j \psi \beta'_j$, $\beta_j \in I_1$, $\beta'_j \in I_2$ and the dotted line shows that the upper rule is transformed into the lower rule. By using this schema, we can represent the reasoning as follows:

$$\dfrac{\beta_1', \cdots, \beta_n' \qquad \dfrac{\dfrac{A\leftarrow B_1, \cdots, B_n}{\alpha\leftarrow\beta_1, \cdots, \beta_n}\ (\theta)}{\alpha'\leftarrow\beta_1', \cdots, \beta_n'}\ (\phi, M(P_1), M(P_2))}{\alpha'},$$

where $A\leftarrow B_1, \cdots, B_n$ is a rule in P_1, θ is a ground substitution to obtain a logically true ground rule $\alpha\leftarrow\beta_1, \cdots, \beta_n$, and the second real line shows modus ponens. Thus the reasoning is a combination of the usual deductions and the rule transformations. This schema is said to be *basic*.

In general, reasoning is a process of applications of inference rules. Thus it is natural to consider a process in which the rule transformations and modus ponens are applied successively. Let us consider an example:

EXAMPLE 3.1. Let P_1 and P_2 be the following programs:

$$\begin{aligned} P_1=\{&p(a, b),\\ &q(b)\leftarrow p(a, b),\\ &r(b),\\ &s(b)\leftarrow q(b), r(b)\},\\ P_2=\{&p(a', b'),\\ &r(b')\}. \end{aligned}$$

Then we have the following basic schema:

$$\dfrac{p(a', b') \qquad \dfrac{q(b)\leftarrow p(a, b)}{q(b')\leftarrow p(a', b')}\ (\phi, M(P_1), M(P_2))}{q(b')},$$

where $\phi=\{\langle a, a'\rangle, \langle b, b'\rangle\}$. $q(b')$ is not a logical consequence of P_2. However we use $q(b')$, as if it is a fact, to derive some additional ground atoms. In fact, we can derive $s(b')$ by a basic schema

$$\dfrac{q(b'), r(b') \qquad \dfrac{s(b)\leftarrow q(b), r(b)}{s(b')\leftarrow q(b'), r(b')}\ (\phi, M(P_1), M(P_2)\cup\{q(b')\})}{s(b')}.$$

Thus the successive uses of the basic schemata allow a monotonic extension of models for P_2.

DEFINITION 3.2. For a given partial identity ϕ, we define a set $M_i(*;\phi)$ for $i=1, 2$ as follows:

$$\begin{aligned} &M_i(*;\phi)=\textstyle\bigcup_n M_i(n),\\ &M_i(0)=M(P_i)=\{\alpha\in B(P_i) \mid P_i\vdash\alpha\},\\ &M_i(n+1)=\{\alpha\in B(P_i) \mid R_i(n)\cup M_i(n)\cup P_i\vdash\alpha\}, \end{aligned}$$

where $S\vdash\gamma$ denotes that γ is a logical consequence of S, $R_i(n)$ is the set of all ground rules which are ϕ-analogues of ground instances of rules in P_j $(j\neq i)$ with respect to $M_1(n)$ and $M_2(n)$.

The set $M_i(*;\phi)$ is an Herbrand model for P_i. (See [7].) Hence we can assert that the reasoning based on the partial identity gives us an admissible method to extend the least model for P_i.

4. Reasoning Procedure

In this section, we define a reasoning procedure, and discuss some computational aspects of the reasoning based on partial identities. First we give a definition of reasoning procedure.

DEFINITION 4.1. A *reasoning procedure* is an effective procedure which takes a ground atom α as its input and satisfies the following properties:

(1) $\alpha \in M_i(*;\phi)$ for some partial identity ϕ, if it returns an answer "*yes*", and

(2) $\alpha \notin M_i(*;\phi)$ for any partial identity ϕ, if it returns an answer "*no*".

DEFINITION 4.2. A reasoning procedure M is *complete* if it returns the answer "*yes*" whenever $\alpha \in M_i(*;\phi)$ for some partial identity ϕ.

To consider a complete reasoning procedures, we present, without proof, a theorem which characterizes the reasoning defined in Section 3 in terms of deducibility. The proof is found in [7]. For a program P_i, and a partial identity ϕ, we introduce a new predicate symbol p_i for each predicate symbol p in P_i. Moreover we define a program $copy(P_i)$ which is obtained from P_i by simultaneously replacing each occurrence of p in P_i by the corresponding p_i. Then we have

THEOREM 4.1. ([7]) *Given P_i and ϕ, there exists a program P, denoted by $P_1\phi P_2$, such that*

(1) $copy(P_i) \subseteqq P$ for $i=1, 2$, *and*

(2) $p(t_1, \cdots, t_n) \in M_i(*;\phi)$ *iff* $P \vdash p_i(t_1, \cdots, t_n)$.

Based on this theorem, we present a complete reasoning procedure M_G. Let $\phi_1, \phi_2, \cdots$ be an effective enumeration of all partial identities. Let $\langle dov1(n), dov2(n)\rangle$ be the n-th pair of natural numbers in an effective enumeration. Moreover, for a program Q and an atom β, $Q \vdash_m \beta$ denotes that β is proved to be a logical consequence of Q in at most m steps of computations. Note that, in order to realize $\vdash_m$, it suffices to consider a complete SLD-refutation procedure [1, 8] with a step-counting function.

Reasoning Procedure M_G

input: ground atom $\alpha = p(t_1, \cdots, t_n)$ in $B(P_1) \cup B(P_2)$
begin
$\alpha' :=$ **if** $\alpha \in B(P_1)$ **then** $p_1(t_1, \cdots, t_n)$
else $p_2(t_1, \cdots, t_n)$;
$n := 1$;
while $P_1\phi_{dov1(n)}P_2 \nvdash_{dov2(n)} \alpha'$ **do** $n = n+1$;
output the answer "*yes*" with $\phi_{dov1(n)}$
end

Then it is clear that M_G is complete and that the set defined as

$$succ(P_1\,;P_2)=\bigcup_n\{\beta \mid P_1\psi_n P_2\vdash\beta\}$$

is recursively enumerable.

In the next section, we present a more concrete reasoning procedure, but it is not complete.

5. Realization of Reasoning in a Logic Programming System

In this section, we present a reasoning procedure which is an extension of pure-Prolog interpreter. Given P_1 and P_2, the set $M_i(*\,;\psi)$ of atoms includes the least model M_i for P_i, and is the smallest set which is closed under applications of rules in P_i and $R_i(n)$. The rules in $R_i(n)$ are obtained by transforming some rules in P_j $(j\neq i)$.

Hence it suffices to have the reasoning procedure satisfying the following properties:

(P1) It interpretes each rule in P_i procedurally.

(P2) It performs the rule transformation based on some partial identity.

Since the domains for the reasoning are represented by logic programs P_i, a pure-Prolog interpreter satisfies (P1). For this reason, we design the reasoning procedure as an extension of pure-Prolog interpreter. A standard interpreter generally takes a goal of the form

$$\leftarrow A_1, \cdots, A_n,$$

and tries to refute it by successively deriving subgoals. The rules used in the refutation are usually those in P_i. To perform the reasoning, our extended interpreter is allowed to use additional rules which should be ψ-analogues of rules in P_j $(j\neq i)$.

As mentioned in the introduction, a reasoning procedure should be designed so that it has no information about the partial identity ψ at the beginning of its computation. Hence it must find a possible partial identity ψ such that $\alpha\in M_i(*\,;\psi)$.

However, given P_1 and P_2, the set (search space) of all partial identities is generally very large. Moreover, even if we introduce an ordering of partial identies, and even if we use an optimal partial identity, we still have the problem of choosing one of them.

To avoid this difficulty, we design our interpreter not to pre-compute any partial identity, but to compute it partially during the whole reasoning process. The basic idea of realizing such a reasoning process by the interpreter is briefly summed up in Section 5.1 below.

5.1. Behavior of the extended interpreter

Let α' be a ground atom to be verified that $\alpha'\in M_2(*\,;\psi)$ for some partial identity ψ. The purpose is to find a rule in P_1, a substitution θ, and a partial identity ψ in the basic schema

$$\cfrac{\beta'_1, \cdots, \beta'_n \quad \cfrac{\cfrac{A\leftarrow B_1, \cdots, B_n}{\alpha\leftarrow\beta_1, \cdots, \beta_n}\,(\theta)}{\alpha'\leftarrow\beta'_1, \cdots, \beta'_n}\,(\psi, M_1(*\,;\psi)\,;\ M_2(*\,;\psi))}{\alpha'},$$

To obtain θ, we give the interpreter a goal

$$\leftarrow [B_1, \cdots, B_n \text{ in } P_1]$$

which means the question: "Is there θ such that $B_j\theta \in M_1(*, \psi)$ for some ψ?".

Suppose that our interpreter has found θ. It should be noticed that, to show $B_j\theta \in M_1(*, \psi)$ for some ψ, some partial identity ψ_1 is also computed as a side effect of the interpreter. ψ_1 is empty only if $B_j\theta \in M(P_1)$. Let

$$\alpha = A\theta = p(t_1, \cdots, t_m),$$
$$\alpha' = p(t'_1, \cdots, t'_m),$$
$$B_j\theta = \beta_j = q_j(t_{j1}, \cdots, t_{jk(j)}).$$

Since a partial identity ψ has not been computed yet, we introduce variables X_{ij} and consider the following transformation schema:

$$\frac{p(t_1, \cdots, t_m) \leftarrow q_1(t_{11}, \cdots, t_{1k(1)}), \cdots, q_n(t_{n1}, \cdots, t_{nk(n)})}{p(t'_1, \cdots, t'_m) \leftarrow q_1(X_{11}, \cdots, X_{1k(1)}), \cdots, q_n(X_{n1}, \cdots, X_{nk(n)})}$$

The variable X_{ij} denotes something in $U(P_2)$ to be paired afterward with $t_{ij} \in U(P_1)$. It can be instantiated by using the constraint that a set of paired term

$$\zeta = \psi_1 \cup \{\langle t_j, t'_j \rangle \mid n \geqq j \geqq 1\}$$
$$\cup \{\langle t_{ij}, X'_{ij} \rangle \mid k(i) \geqq j \geqq 1, n \geqq i \geqq 1\} \quad (5.1)$$

should satisfy the condition *EPIC*. Details of checking *EPIC* are discussed in Section 5.2. Let σ be a substitution thus obtained. Then we now have a pairing $\psi_2 = \zeta\sigma$ which is an extension and a refinement of ψ_1. From the definition of the reasoning, the next thing to do is to prove that there exists a substitution θ_2 such that

$$q_j(X_{j1}\sigma\theta_2, \cdots, X_{jk(j)}\sigma\theta_2) \in M_2(*, \psi).$$

For this purpose, we generate a goal

$$\leftarrow [q_j(X_{j1}\sigma, \cdots, X_{jk(j)}\sigma) \text{ in } P_2]$$

and continue the whole process we have just described.

As a result, we obtain a sequence $\{\psi_z\}$ of sets of paired terms such that ψ_{i+1} is an extension and a refinement of ψ_i. The last ψ_z in the sequence is the desired partial identity.

5.2. Checking the condition EPIC

As we have discussed in Section 2, *EPIC* is a condition for a pairing (Definition 2.4). For the sake of simplicity, we assume here that all the function symbols but constants are unary. Then *EPIC* can be restated as follows:

EPIC under the assumption: Given a pairing B, let $\langle t_i, t'_i \rangle \in B$, $term(X) \in L(P_1) \cap L(P_2)$. Then

$$(EPIC1)\ term(t'_0) = t'_1 \text{ whenever } \langle term(t_0), t'_1 \rangle \in B,$$
$$(EPIC2)\ term(t_0) = t_1 \text{ whenever } \langle t_1, term(t'_0) \rangle \in B.$$

To check the condition (*EPIC*1), we consider the following nondeterministic procedure which computes an equation from B. It is completely similar to check ($EPIC2$), so the details of checking ($EPIC2$) is omitted.

Procedure $mkeq_1$
input: a finite set B of paired terms, and a term t such that $\langle t, t'\rangle \in B$ for some t;
begin
choose a subterm t_0 of t such that
$t = term(t_0)$ for some $term(X) \in L(P_1) \cap L(P_2)$, and $\langle t_0, t_0'\rangle \in B$ for some t_0';
if such a subterm is not found
then do nothing
else output an equation $t' = term(t_0')$
end

Let $Eq_1(t, B)$ be the set of all equations in $U(P_2)$ obtained by the executions of $mkeq_1(t, B)$, and let

$$Eq_1(B) = \cup \{Eq_1(t, B) \mid t \text{ appears in the first argument of some pair in } B\}.$$

$Eq_2(B)$ is similarly defined by scanning the second arguments of pairs in B.

EXAMPLE 5.1. Suppose that $L(P_1) = L(P_2)$, and let

$$B = \{\langle f(h(a)), f(V)\rangle, \tag{5.2}$$
$$\langle g(b), g(b')\rangle, \tag{5.3}$$
$$\langle a, X\rangle, \tag{5.4}$$
$$\langle h(a), Y\rangle\}. \tag{5.5}$$

From (5.2) and (5.5), we have $f(Y) = f(V)$. From (5.2) and (5.4), we have $f(h(X)) = f(V)$. Finally, from (5.4) and (5.5), we have $h(X) = Y$. As a result, we have

$$Eq_1(B) = \{h(X) = Y,\ f(h(X)) = f(Y) = f(V)\}.$$

Note that $Eq_2(B) = \phi$ in this case.

Since we consider Herbrand interpretations, the predicate symbol "=" denotes the identity relation (on each Herbrand universe). Formally we say that Eq_i has a solution if there exists a ground substitution θ such that, for any equation $t = t'$ in Eq_i, $t\theta$ and $t'\theta$ are the same term in $U(P_j)$, where $i \neq j$. Then we have

PROPOSITION 5.1. *If there exists a substitution θ such that $B\theta \in U(P_1) \times U(P_2)$ and that $B\theta$ is a partial identity, then both $Eq_1(B)$ and $Eq_2(B)$ have solutions.*

PROOF. Let B and θ be a set of paired terms and a substitution as in the condition part of the proposition, respectively. Without loss of generality, suppose that there exist two equations in $Eq_1(B)$:

$$t' = \mathrm{term}_1(t_0'), \tag{5.6}$$
$$s' = \mathrm{term}_2(s_0'), \tag{5.7}$$

where $\langle t, t'\rangle$, $\langle t_0, t_0'\rangle$, $\langle s, s'\rangle$, $\langle s_0, s_0'\rangle \in B$, $t = \mathrm{term}_1(t_0)$, and $s = \mathrm{term}_2(s_0')$. Since $B\theta$ satisfies ($EPIC1$),

$$t'\theta = \mathrm{term}_1(t'_0\theta).$$
$$s'\theta = \mathrm{term}_2(s'_0\theta)$$

hold. This completes the proof.

The converse of Proposition 5.1 does not hold in general. In fact, consider a set

$$B = \{\langle a, b\rangle, \langle g(X), Y\rangle, \langle f(g(X)), Z\rangle\},$$

where we assume that P_1 has only a, g and f as its function symbols and that P_2 has only b and f. Then we have

$$Eq_1(B) = \{Z = f(Y)\}.$$

This equation has a solution

$$\theta_n = \{f^{(n+1)}(b)/Z,\ f^{(n)}(b)/Y\}.$$

For any θ such that $B\theta \in U(P_1) \times U(P_2)$,

$$\langle g(X\theta), f^{(n)}(b)\rangle, \langle f^{(n)}(a), f^{(n)}(b)\rangle \in (B\theta)^+$$

for some n. Thus $B\theta$ is not partial identity.

We use Proposition 5.1 to compute a partial identity. Let ζ be the set of paired terms in (5.1), $\zeta\theta$ be a partial identity, and $Eq_1(\zeta) = \{s_i = t_i \mid 1 \leqq i \leqq n\}$. From the proof of Proposition 5.1, the following two lists

$$list_s = [s_1, \cdots, s_n],$$
$$list_t = [t_1, \cdots, t_n]$$

are unifiable by the substitution θ. Let $\theta = \sigma\tau$, where σ is the most general unifier of $list_s$ and $list_t$. Then it is clear that finding θ from ζ is reduced to finding τ from $\zeta\sigma$, which is the next set of paired terms in the sequence $\{\psi_z\}$.

Conversely, if $list_s$ and $list_t$ are not unifiable, then there exist no θ such that $\zeta\theta$ is a partial identity. Hence we can reject ζ, and we must find another set of paired terms. The search of alternative sets of paired terms is realized in our extended interpreter by using Prolog's backtracking.

5.3. An example of the whole reasoning process

Now let us exemplify the whole reasoning process carried out by our extended interpreter. For this purpose, consider the following logic programs:

$$\begin{aligned}
P_1 = \{&f(b, c),\\
&m(a, b),\\
&gf(X, Z) \leftarrow p(X, Y), f(Y, Z),\\
&p(X, Y) \leftarrow f(X, Y),\\
&p(X, Y) \leftarrow m(X, Y)\},\\
P_2 = \{&m(a', b'),\\
&f(b', c')\}.
\end{aligned}$$

The initial goal fed to the interpreter is

$$\leftarrow[gf(a', c') \text{ in } P_2].$$

For this goal, the interpreter firstly tries to prove

$$gf(a', c') \in M_2 = M(P_2).$$

Clearly the interpreter fails to prove it, and therefore tries to find some ϕ-analogue of a rule in P_1. In this case,

$$gf(X, Z) \leftarrow p(X, Y), f(Y, Z)$$

is chosen, since the head has the predicate symbol gf. The interpreter then generates a subgoal

$$\leftarrow[p(X, Y), f(Y, Z) \text{ in } P_1]$$

to find the values of variables X, Y and Z. It turns out that X, Y and Z are instantiated to a, b and c, respectively, since $p(a, b)$, $f(b, c) \in M_1$. Hence, at this point, the first set ϕ_1 of paired terms is empty. Based on the substitution above, the interpreter produces a rule transformation.

$$\frac{gf(a, c) \leftarrow p(a, b), f(b, c)}{gf(a', c') \leftarrow p(X_1, X_2), f(X_3, X_4)}$$

where the variables X_j are introduced to compute a partial identity. Then we have a set

$$B_1 = \{\langle a, a'\rangle, \langle c, c'\rangle, \langle a, X_1\rangle, \langle b, X_2\rangle, \langle b, X_3\rangle, \langle c, X_4\rangle\},$$

and ask then the existence of a substitution θ such that $B_1\theta$ is a partial identity. Our extended interpreter computes

$$Eq_1 = \{a' = X_1, X_2 = X_3, c' = X_4\},$$
$$Eq_2 = \phi$$

from B_1. From these equations, we have two lists

$$[a', X_2, c'], \qquad [X_1, X_3, X_4].$$

Unifying them, we have the most general unifier

$$\theta = \{a'/X_1, X_3/X_2, c'/X_4\}.$$

Hence the next set ϕ_2 of paired terms is

$$\phi_2 = \{\langle a, a'\rangle, \langle b, X_3\rangle, \langle c, c'\rangle\}.$$

We now have a rule transformation

$$\frac{gf(a, c) \leftarrow p(a, b), f(b, c)}{gf(a', c') \leftarrow p(a', X_3), f(X_3, c')}$$

The value of X_3 is still unknown, so the interpreter generates the next subgoal

$$\leftarrow [p(a', X_3),\ f(X_3, c') \text{ in } P_2] \tag{5.8}$$

to find the value. Since there exist no instance of X_3 such that $p(a', X_3) \in M_2$, the interpreter similarly tries to find ψ-analogue of a rule in P_1. In this case, there exist two rules

$$(\text{R1}):\ p(X, Y) \leftarrow f(X, Y),$$

$$(\text{R2}):\ p(X, Y) \leftarrow m(X, Y)$$

which have the predicate symbol p in their head parts. These two possibilities are examined alternatively. First suppose that (R1) is chosen. Our interpreter instantiates X and Y, and a rule transformation:

$$\frac{p(b, c) \leftarrow f(b, c)}{p(a', X_3) \leftarrow f(X_4, X_5)}$$

where X_4 and X_5 are new variables. From the schema above, we have

$$\sigma_1 = \{\langle b, a'\rangle, \langle c, X_3\rangle, \langle b, X_4\rangle, \langle c, X_5\rangle\},$$

and the next set of paired terms

$$\sigma_1 \cup \psi_2 = \{\langle a, a'\rangle, \langle b, X_3\rangle, \langle c, c'\rangle, \langle b, a'\rangle, \langle c, X_3\rangle, \langle b, X_4\rangle, \langle c, X_5\rangle\}.$$

In this case,

$$a = b \in Eq_2(\sigma_1 \cup \psi_2),$$

which is not unifiable. Hence our extended interpreter goes back to (5.8). Now it chooses (R2) instead of (R1).

Similarly we have a rule transformation:

$$\frac{p(a, b) \leftarrow m(a, b)}{p(a', X_3) \leftarrow m(X_4, X_5)}$$

with the new set of paired terms

$$\sigma_2 = \{\langle a, a'\rangle, \langle b, X_3\rangle, \langle a, X_4\rangle, \langle b, X_5\rangle\}.$$

Then we have

$$\psi_2 \cup \sigma_2 = \{\langle a, a'\rangle, \langle b, X_3\rangle, \langle c, c'\rangle, \langle a, X_4\rangle, \langle b, X_5\rangle\},$$

and

$$Eq_1(\psi_2 \cup \sigma_2) = \{a' = X_4,\ X_3 = X_5\}.$$

Unifying $[a', X_3]$ and $[X_4, X_5]$, we have a rule transformation

$$\frac{p(a, b) \leftarrow m(a, b)}{p(a', X_5) \leftarrow m(a', X_5)},$$

and

$$\psi_3 = \{\langle a, a'\rangle, \langle b, X_5\rangle, \langle c, c'\rangle\}.$$

To find the value of X_5, the goal

$$\leftarrow [m(a', X_5) \text{ in } P_2]$$

is generated, and it succeeds with b'/X_5, since $m(a', b') \in M_2$. Thus we have a partial identity

$$\psi = \{\langle a, a' \rangle, \langle b, b' \rangle, \langle c, c' \rangle\}$$

and complete to prove $gf(a', c') \in M_2(*; \psi)$.

5.4. Control structure of the extended interpreter

We now present our interpreter called "*ana*" in a form of Prolog program. First suppose that a fact of assertion A and a rule $C \leftarrow B_1, \cdots, B_n$ in P_1 are represented by Prolog's clauses

$$fact_1(A).$$
$$fact_1(C) :- fact_1(B_1), \cdots, fact_1(B_n).,$$

respectively, and stored in Prolog database, where $fact_1$ is a reserved symbol to show that the fact and the rule are in P_1. For a clause in P_2, we assume a similar representation using a symbol $fact_2$. Now we list the Prolog program *ana*.

```
(C1) ana(Goal) :- reason(Goal, [ ], Epic).
(C2) reason(true, Epic, Epic) :- !.
(C3) reason((Goal, Goals), Epic, Epic2)
                 :- !, reason(Goal, Epic, Epic1),
                       reason(Goals, Epic1, Epic2).
(C4) reason(Goal, Epic, Epic1)
                 :- clause(Goal, Goals),
                      reason(Goals, Epic, Epic1).
(C5) reason(Goal, Epic, Epic3)
                 :- prematch(Goal, Sgoal, Sgoals),
                      reason(Sgoal, Sgoals, Tgoals, Bind),
                      trans(Goal, Sgoal, Sgoals, Tgoals, Bind),
                      epic(Bind),
                      append(Epic1, Bind, Epic2),
                      epic(Epic2),
                      reason(Tgoals, Epic2, Epic3),
                      epic(Epic3).
```

A goal $\leftarrow$[A in P_1] fed to our interpreter is represented by a Prolog's term $fact_i(A)$. Hence the initial goal to the Prolog interpreter is

$$\leftarrow ana(fact_i(A)).$$

Some set of paired terms are to be assigned to the second and the third arguments of the predicate *reason*. A set of paired terms $\{\langle t_j, t'_j \rangle | 1 \leqq j \leqq n\}$ is represented by a pair-list $[[t_1, t'_1], \cdots, [t_n, t'_n]]$. The empty list is denoted by [ ]. Given a goal and a pair-list $list_0$ assigned to the variable *Epic*, *reason* computes a pair-list $list_1$ which is an extension and a refinement of $list_0$. The third variable *Epic'* of *reason* is instantiated to $list_1$, when the goal *reason*(*Goal*, *Epic*, *Epic'*) succeeds. The process of finding the sequence of $\{\psi_k\}$ is thus realized by a recursion.

The ordered set of clauses (C2), (C3) and (C4) works as a pure-Prolog interpreter. The last clause (C5) carries out the rule transformation and the construction of pair-list. The predicate *prematch* in the body of (C5) searches a possible rule in $P_j(j \neq i)$ to be transformed.

The predicate *trans* produces a pair-list necessary to perform transformations. The produced pairing is assigned to the variable *Bind.*

The predicate *epic* computes a set of equations, and tries to unify them, as we have discussed. Hence, after the execution of *epic*(*Bind*), some variables in *Bind* are instantiated to terms as long as *Bind* does not violate the condition *EPIC*. If it turns out that *Bind* violates *EPIC*, *reason* goes back, and tries to find another pair-list and another rule to be transformed.

Finally we present an example (due to Winston [11]) of the reasoning performed by the Program "*ana*". In the example, the question asked to *ana* is

$$\leftarrow [kill(X, noble\_a) \ in \ P_2].$$

Equivalently it has the form

$$ana(fact_2(kill(X, noble\_a)))$$

as an initial goal to the Prolog interpreter. The underlying logic programs for the reasoning are as follows:

$P_1 = \{greedy(ladymac).$
$marry(mac, ladymac).$
$weak(mac).$
$wtbqueen(ladymac).$
$king(duncan).$
$loyal(macduff, duncan).$
$evil(mac) :- weak(mac), marry(mac, X), greedy(X).$
$infl(X, Y) :- marry(Y, X), weak(Y), greedy(X).$
$wtbking(mac) :- infl(ladymac, mac), wtbqueen(ladymac).$
$murder(Murder, King) :- king(King), wtbking(Murder), evil(Murder).$
$kill(Loyal, Murder) :- murder(Murder, Someone), loyal(Loyal, Someone).\}$
$P_2 = \{noble(noble\_a).$
$lady(lady\_a).$
$king(king\_a).$
$noble(noble\_b).$
$loyal(noble\_b, king\_a).$
$marry(noble\_a, lady\_a).$
$weak(noble\_a).$
$greedy(lady\_a).$
$wtbqueen(lady\_a).\}.$

The extended interpreter has been developed for DCL U-station in CProlog, and returned the answer

$$X = noble\_b$$

for the question above after about 15 seconds.

References

[1] APT, K. R. and VAN EMDEN, M. H.: *Contribution to the theory of logic programming*, JACM, **29**, 3 (1982), 841-862.
[2] CLARK, K. L.: *Negation as Failure*, Logic and Databases, H. GALLAIRE and J. MINKER (Eds.), Plenum Press, New York (1978), 293-322.
[3] GENTNER, P.: *Are scientific analogies metaphors?*, Metaphor: Problems and Perspectives, D. S. MIALL (Ed.), The Harvester Press, Sussex (1982), 106-132.
[4] HARAGUCHI, M.: *Towards a mathematical theory of analogy*, Bull. Inform. Cybernetics, **21** (1985), 29-56.
[5] HARAGUCHI, M.: *Analogical reasoning using transformations of rules*, LNCS 221 (1986), 56-65.
[6] HARAGUCHI, M. and ARIKAWA, S.: *A formulation and a realization of analogical reasoning*, Journal of JSAI, 1, 1 (1986), 132-139 (in Japanese).
[7] HARAGUCHI, M. and ARIKAWA, S.: *Analogical Union of Logic Programs*, Proc. Logic Programming Conference '86 (1986), 103-110.
[8] LLOYD, J. W.: Foundations of logic programming, Springer-Verlag (1984).
[9] POLYA, G.: Induction and analogy in mathematics, Princeton University Press (1954).
[10] WINSTON, P. H.: *Learning and reasoning by analogy*, CACM, **23** (1980), 689-703.
[11] WINSTON, P. H.: *Learning new principles from precedents and exercises*, Artificial Intelligence, **19** (1983), 321-350.

Received September 29, 1986