

## Incorporating Explanation-Based Generalization with Analogical Reasoning

Hirowatari, Eiju

Research Institute of Fundamental Information Science, Kyushu University

Arikawa, Setsuo

Research Institute of Fundamental Information Science, Kyushu University

<https://doi.org/10.5109/3063>

---

出版情報 : Bulletin of informatics and cybernetics. 26 (1/2), pp.13-33, 1992-03. Research  
Association of Statistical Sciences

バージョン :

権利関係 :



# Incorporating Explanation-Based Generalization with Analogical Reasoning\*

Eizyu Hirowatari<sup>†</sup>      Setsuo Arikawa

Research Institute of Fundamental Information Science,  
Kyushu University 33, Fukuoka 812, Japan  
e-mail: eizyu, arikawa@rifis.sci.kyushu-u.ac.jp

## Abstract

The EBG system builds an explanation and learns a concept definition as its generalization provided a domain theory is complete. It does not work when a domain theory is incomplete. Then we introduce a notion of generalizations by an analogy which makes it possible to construct rules necessary for domain theories. Furthermore, we develop EBG by analogical reasoning which copes with the incompleteness by using generalizations by an analogy. We first formulate EBG in a mathematical way, construct EBG by analogical reasoning in terms of our formulation, and realize EBG by analogical reasoning system as a Prolog program.

## 1 Introduction

EBG(Explanation-Based Generalization) takes as inputs a domain theory, a training example, a goal concept and an operationality criterion. It constructs an explanation in terms of the domain theory that proves how the training example satisfies the goal concept definition. Then it determines a set of operationally sufficient conditions for the goal concept under which the explanation holds, and returns it as an output. To solve many problems efficiently, EBG has been extensively studied in the field of machine learning[3, 10, 11, 16]. However, mathematical discussions for EBG have not yet been developed well. In the present paper we first define the concept of EBG in a mathematical way.

---

\*This work is partly supported by Grants-in-Aid for Scientific Research on Priority Areas from the Ministry of Education, Science and Culture, Japan.

<sup>†</sup>JSPS Fellowship for Japanese Junior Scientists.

EBG does not work when a domain theory is incomplete, for EBG needs a complete domain theory. It is important to solve this problem as pointed out by many authors [1, 3, 4, 8, 11, 12, 15, 20]. In terms of our formulation, we develop EBG by analogical reasoning (EBG-by-AR, for short) which copes with incomplete domain theories, and realize EBG-by-AR system as a Prolog program.

Analogical reasoning [2, 5, 6, 7] is an important paradigm of machine learning. It acquires unknown analogical facts in domains by computing an analogy which gives a similarity between the domains. In analogical reasoning, we detect an analogy, project the well-known rules in the domain into the other domain by transformations of rules, and then acquire analogical facts by usual deductions. We can get ground rules necessary for domains by transformations of rules. However, transformations of rules are not suitable for EBG with incomplete domain theories, because transformed rules are not general at all. Thus we propose generalizations of rules by an analogy and show some properties of them.

In our EBG-by-AR system, we use generalizations of rules by an analogy in stead of transformations of rules. Then we make up rules necessary for domain theories by means of analogical reasoning when a domain theory is incomplete. Furthermore our system automatically constructs an analogy, and outputs it when our system answers a question.

Note here that Numao and Shimura [13] presented a method of analogical inference using explanation-based learning. In contrast, we present a method of EBG with an incomplete domain theory using analogical inference.

This paper is organized as follows: In Section 2 we present a mathematical formulation of EBG. In Section 3 we prepare some concepts on analogical reasoning necessary for our discussion. In Section 4 we introduce the notion of generalizations by an analogy and show some properties of them. In Section 5 we present EBG-by-AR. In Section 6 we describe a realization of EBG-by-AR system as a Prolog program.

## 2 Formulation of EBG

In this section we define the concept of EBG in a mathematical way. See [9] for detailed definitions on first order logic and logic programming.

We simply call an atomic formula an *atom* and an atom without variables a *ground atom*. A *definite clause* is a clause of the form

$$A \leftarrow B_1, \dots, B_n \quad (n \geq 0),$$

where  $A, B_1, \dots, B_n$  are atoms. We call  $A$  the *head* and  $B_1, \dots, B_n$  the *body* of the definite clause. Then we call a definite clause with body a *rule* and a rule without variables a *ground rule*. We define a *domain theory*, denoted by  $D$ , as a finite set of definite clauses and a *training example*, denoted by  $T$ , as a non-empty finite set of ground atoms. Both  $D$  and  $T$  are sets of definite clauses. Let  $P = D \cup T$ . We call  $P$  a *program*.  $P$  has at least one constant symbol, because  $T$  is a non-empty set. We define a *goal concept*, denoted by  $G$ , as an atom. Let  $\Pi_P$  be the set of all predicate symbols in  $P$ . We define an *operationality criterion*, denoted by  $O$ , as a subset of  $\Pi_P$ . Let  $\Pi(O)$  be the set of all

atoms that have predicate symbols in  $O$ . From now on we identify  $O$  with  $\Pi(O)$ . Let  $I = (P, G, O)$ . We call  $I$  an *input*.

Let  $U_P$  be the Herbrand universe for  $P$ , and  $B_P$  be the Herbrand base for  $P$ . Let  $M_P$  be the least Herbrand model for  $P$ , that is, the set of all ground atoms that are logical consequences of  $P$ .

**Definition 1** *An explanation tree for  $I = (P, G, O)$  is a finite tree satisfying the following conditions:*

- (a) *Each node of the tree is an element of  $M_P$ .*
- (b) *The root node is an instance  $g$  of  $G$ .*
- (c) *If a node  $\alpha$  in the tree has children  $\beta_1, \dots, \beta_n$  ( $n \geq 1$ ) in this order, then  $\alpha \leftarrow \beta_1, \dots, \beta_n$  is a ground instance of a rule in  $P$ , and  $\alpha$  is not an element of  $\Pi(O)$ .*
- (d) *Each node without children is an element of  $\Pi(O)$ .*

We call the root node  $g$  a *goal example* of  $G$ .

Standard Prolog systems employ, together with the depth-first search, the computation rule that always selects the leftmost atom in a goal. We assign a number to each rule used in the explanation tree in the following way:

- (a) Assign a number to each node that is not an element of  $\Pi(O)$  in depth-first order starting from 1.
- (b) Let  $\alpha$  be a node which has children  $\beta_1, \dots, \beta_n$  ( $n \geq 1$ ) in this order, and to which number  $j$  assigned. Then assign the number  $j$  to a rule in  $P$ , denoted by  $C_j$ , which is a generalization of  $\alpha \leftarrow \beta_1, \dots, \beta_n$ .

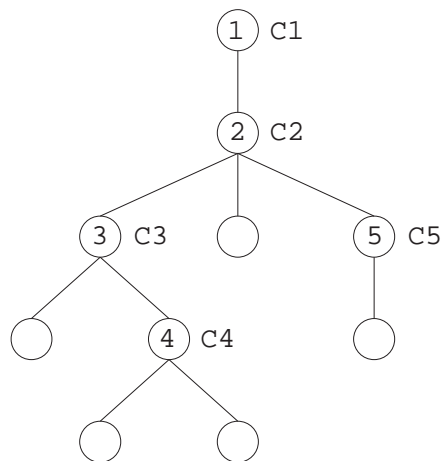


Figure 1: Numbering nodes and rules in an explanation tree.

We use the rules with numbers in the following generalized derivation.

**Definition 2** A *generalized derivation* for  $I = (P, G, O)$  is an SLD-derivation via  $CR$  which consists of a finite sequence  $\leftarrow G_0 (= \leftarrow G), \leftarrow G_1, \dots, \leftarrow G_n$  of goals, a sequence  $C'_1, \dots, C'_n$  of variants of rules in  $P$ , and a sequence  $\theta_1, \dots, \theta_n$  of most general unifiers, where  $CR$  is a computation rule to select the leftmost atom that is not in  $\Pi(O)$ .

For the rules  $C_1, \dots, C_n$  in the explanation tree, each  $C'_i$  ( $1 \leq i \leq n$ ) is a suitable variant of  $C_i$  in the sense that  $C'_i$  does not have any variables which already appear in the derivation up to  $G_{i-1}$ .

A generalized derivation depends on an explanation tree. An *explanation* is to construct an explanation tree, and a *generalization* is to regress a goal concept through the explanation tree using a generalized derivation.

**Definition 3** *EBG* is to derive a general definition  $R$  of  $G$  from its input  $I = (P, G, O)$  by an explanation and a generalization, and it is denoted by

$$I \xrightarrow{EBG} R,$$

where

$$R = \begin{cases} G & \text{if } n = 0, \\ G\theta_1 \dots \theta_n \leftarrow G_n & \text{otherwise,} \end{cases}$$

and  $(\leftarrow G_n)$  is the goal and  $(\theta_1, \dots, \theta_n)$  is the sequence of most general unifiers in the generalized derivation of depth  $n$ .

Now we can prove the following results:

**Proposition 1** Let  $I = (P, G, O)$  be an input. Then,  $I \xrightarrow{EBG} G$  iff  $G \in \Pi(O)$ .

**Proof.**  $I \xrightarrow{EBG} G$

$\iff$  there exists an explanation tree of depth 0 with the root  $G$

$\iff G \in \Pi(O)$ .  $\square$

**Theorem 1** Let  $I = (P, G, O)$  be an input,  $R$  be a rule and  $F$  be the set of all rules in  $P$  of that the head is not in  $\Pi(O)$ . Then,  $I \xrightarrow{EBG} R$  iff the following conditions hold:

- (a)  $F \vdash R$ .
- (b)  $\{R\} \cup \Pi(O)|_{M_P} \vdash g$  for a ground instance  $g$  of  $G$ .
- (c)  $G \notin \Pi(O)$ .

**Proof.** Put  $R = (A \leftarrow A_1, \dots, A_m)$ .

( $\implies$ ) Suppose  $I \xrightarrow{EBG} R$ . Let  $C_1, \dots, C_n$  be the sequence of rules in the generalized derivation of depth  $n$ . Then we get  $\{C_1\} \cup \dots \cup \{C_n\} \vdash R$ . Furthermore  $C_1, \dots, C_n \in F$ . Hence we have  $F \vdash R$ .

Since  $A_1, \dots, A_m \in \Pi(O)$ , there exists a substitution  $\theta$  such that  $g = G\theta$  and  $A_1\theta, \dots, A_m\theta \in \Pi(O)|_{M_P}$  hold. Hence we have  $\{R\} \cup \Pi(O)|_{M_P} \vdash g$ .

( $\impliedby$ ) Suppose that the conditions (a),(b) and (c) hold. By (c) there exists no substitution  $\gamma$  such that  $G\gamma \in \Pi(O)$ . Then, by (a) and (b), there exists a substitution  $\theta$  such that  $g = G\theta$  and  $A_1\theta, \dots, A_m\theta \in \Pi(O)|_{M_P}$ . Hence there exists an explanation tree with the root  $G\theta$  and the leaves  $A_1\theta \dots, A_m\theta$ . Therefore we have  $I \xrightarrow{EBG} R$ .  $\square$

These results claim that EBG is deductive reasoning from a program which satisfies the input conditions.

### 3 Analogical Reasoning

Haraguchi and Arikawa[2, 5, 6, 7] dealt with logic programs (programs, for short) as domains for analogical reasoning and defined a formal analogy as a relation between elements in domains. This section prepares, according to their works, some concepts on analogical reasoning necessary for our later discussion.

Let  $P_1$  and  $P_2$  be programs on which analogical reasoning is carried out. Let  $U_i, B_i$  and  $M_i$  be the Herbrand universe for  $P_i$ , the Herbrand base for  $P_i$  and the least Herbrand model for  $P_i$  ( $i = 1, 2$ ), respectively.

Assume that, in  $P_1$ , premises  $\beta_1, \dots, \beta_n$  logically imply a ground atom  $\alpha$ . Assume also that premises  $\beta'_1, \dots, \beta'_n$  hold in  $P_2$ , and  $\beta_i$  and  $\beta'_i$  are analogous for each  $i$  ( $1 \leq i \leq n$ ). Then analogical reasoning is to derive a ground atom  $\alpha'$  in  $P_2$  which is analogous to  $\alpha$ . We follow the principle of analogical reasoning by Haraguchi and Arikawa [7] shown in Figure 2, that is based on Polya [14] and Winston [18, 19].

$$\begin{array}{rcl}
 P_1 : & \beta_1, \dots, \beta_n & \rightarrow \alpha \\
 & \Downarrow & \Downarrow \text{ (similarity } \varphi \text{)} \\
 P_2 : & \beta'_1, \dots, \beta'_n & \rightarrow \underline{\alpha'}
 \end{array}$$

Figure 2: The principle of analogical reasoning.

The  $\varphi$  in Figure 2 denotes an analogy which gives a similarity between  $P_1$  and  $P_2$ . In this paper, we take an analogy  $\varphi$  as a partial identity as defined below. We prepare some definitions for  $P_1$  and  $P_2$ .

**Definition 4** A finite subset  $\varphi$  of  $U_1 \times U_2$  is called a *pairing* between  $P_1$  and  $P_2$ . The set  $\varphi^+$  is defined to be the smallest set that satisfies the following conditions:

- (a)  $\varphi \subseteq \varphi^+$ ,
- (b)  $\langle t_1, t'_1 \rangle, \dots, \langle t_n, t'_n \rangle \in \varphi^+ \Rightarrow \langle f(t_1, \dots, t_n), f(t'_1, \dots, t'_n) \rangle \in \varphi^+$ ,

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

**Definition 5** A pairing  $\varphi$  is called a *partial identity* between  $P_1$  and  $P_2$  if  $\varphi^+$  is a one-to-one relation between terms.

An analogy can be extended from terms to atoms and rules in a natural way.

**Definition 6** Let  $\alpha$  and  $\alpha'$  be ground atoms in  $B_1$  and  $B_2$ , respectively.  $\alpha$  and  $\alpha'$  are said to be *identified* by  $\varphi$ , denoted by  $\alpha\varphi\alpha'$ , if  $\alpha$  and  $\alpha'$  are written as

$$\alpha = p(t_1, \dots, t_n) \text{ and } \alpha' = p(t'_1, \dots, t'_n),$$

respectively, for some predicate symbol  $p$  appearing in both  $P_1$  and  $P_2$ , and  $\langle t_i, t'_i \rangle \in \varphi^+$  holds for each  $i$  ( $1 \leq i \leq n$ ).

**Definition 7** Let

$$R = (\alpha \leftarrow \beta_1, \dots, \beta_n) \text{ and } R' = (\alpha' \leftarrow \beta'_1, \dots, \beta'_n)$$

be two ground rules whose symbols appear in  $P_1$  and  $P_2$ , respectively, and  $I_i$  be an Herbrand interpretation of  $P_i$  ( $i = 1, 2$ ). Then  $R$  and  $R'$  are said to be  $(\varphi, I_1, I_2)$ -analogous if  $\alpha\varphi\alpha'$  holds and for each  $j$  ( $1 \leq j \leq n$ ), there exist  $\beta_j$  in  $I_1$  and  $\beta'_j$  in  $I_2$  with  $\beta_j\varphi\beta'_j$ .

A transformation of rules is to transform a ground rule  $R$  into a ground rule  $R'$  such that  $R$  and  $R'$  are  $(\varphi, I_i, I_j)$ -analogous ( $i \neq j$ ). Analogical reasoning can be represented by using the schema in Figure 3.

$$\frac{\frac{A \leftarrow B_1, \dots, B_n \text{ (ground instantiation)}}{\alpha \leftarrow \beta_1, \dots, \beta_n \text{ (transformation of rules)}}}{\frac{\beta'_1, \dots, \beta'_n \quad \alpha' \leftarrow \beta'_1, \dots, \beta'_n \text{ (modus ponens)}}{\alpha'}}$$

Figure 3: The basic schema of analogical reasoning.

The first line shows the ground instantiation to get a logically true ground rule  $\alpha \leftarrow \beta_1, \dots, \beta_n$ . The second line shows that the rule  $\alpha \leftarrow \beta_1, \dots, \beta_n$  is transformed into the rule  $\alpha' \leftarrow \beta'_1, \dots, \beta'_n$ . The third line shows the modus ponens. Thus analogical reasoning is a combination of the usual deductions and the rule transformations.

**Definition 8** The set  $M_i^*$  ( $i = 1, 2$ ) is defined as follows:

$$M_i^* = \bigcup_n M_i^n,$$

$$M_i^0 = M_i,$$

$$M_i^{n+1} = \{ \alpha \mid \alpha \text{ is a ground atom such that } R_i^n \cup M_i^n \cup P_i \vdash \alpha \text{ holds} \} \quad (n \geq 0),$$

$$R_i^n = \left\{ R' = \alpha' \leftarrow \beta'_1, \dots, \beta'_n \left| \begin{array}{l} \text{There exists a ground instance } R \text{ of} \\ \text{a rule in } P_j \text{ (} i \neq j \text{) such that} \\ R \text{ and } R' \text{ are } (\varphi, M_i^n, M_j^n)\text{-analogous} \end{array} \right. \right\}.$$

Let  $R_i^* = \bigcup_n R_i^n$ . Namely,  $R_i^*$  is the set of all transformed rules. Then  $M_i^*$  is the set of all ground atoms reasoned by the basic schema in Figure 3. The ground atoms in  $M_i^*$  are not always logical consequences of  $P_i$ . Thus Haraguch and Arikawa introduced a notion of analogical unions of programs and showed that all ground atoms in  $M_i^*$  are logical consequences of an analogical union of  $P_1$  and  $P_2$  as follows:

**Definition 9** A set  $\text{Pair}(\varphi)$  of rules is defined as follows:

- (a)  $t \sim t'$  for each  $\langle t, t' \rangle \in \varphi$ ,
- (b)  $f(X_1, \dots, X_n) \sim f(Y_1, \dots, Y_n) \leftarrow X_1 \sim Y_1, \dots, X_n \sim Y_n$ ,

where  $\sim$  is a predicate symbol which does not appear in  $P_1$  and  $P_2$  either, and  $f$  is a function symbol appearing in both  $P_1$  and  $P_2$ .

**Theorem 2** [6] Let  $P_1$  and  $P_2$  be programs and  $\varphi$  be a pairing between  $P_1$  and  $P_2$ . Then the following conditions are equivalent:

- (a)  $\langle t, t' \rangle \in \varphi^+$ .
- (b)  $\text{Pair}(\varphi) \vdash t \sim t'$ .

**Definition 10** Let  $S_i$  be a set of clauses in which each predicate symbol appears in  $P_i$ , and  $C_i$  be a clause in  $S_i$  ( $i = 1, 2$ ). A copy of  $C_i$  is defined to be the clause obtained by replacing each predicate symbol  $p$  in  $C_i$  by a predicate symbol  $p_i$ . A copy of  $S_i$ , denoted by  $\text{Copy}(S_i)$ , is defined to be the set of all copies of clauses in  $S_i$ .

**Definition 11** An analogical union of  $P_1$  and  $P_2$ , denoted by  $P_1\varphi P_2$ , is defined as follows:

$$P_1\varphi P_2 = \text{Copy}(P_1) \cup \text{Copy}(P_2) \cup \text{Trans}(P_1) \cup \text{Trans}(P_2) \cup \text{Pair}(\varphi),$$

where

$$\text{Trans}(P_1) = \left\{ \begin{array}{l} p_2(W_1, \dots, W_n) \leftarrow \dots, \\ t_1 \sim W_1, \dots, t_n \sim W_n, \\ q_2(V_1, \dots, V_k), q_1(s_1, \dots, s_k), \\ s_1 \sim V_1, \dots, s_k \sim V_k, \dots \end{array} \middle| \begin{array}{l} p(t_1, \dots, t_n) \leftarrow \dots, \\ q(s_1, \dots, s_k), \\ \dots \\ \in P_1 \end{array} \right\},$$

$$\text{Trans}(P_2) = \left\{ \begin{array}{l} p_1(W_1, \dots, W_n) \leftarrow \dots, \\ W_1 \sim t_1, \dots, W_n \sim t_n, \\ q_1(V_1, \dots, V_k), q_2(s_1, \dots, s_k), \\ V_1 \sim s_1, \dots, V_k \sim s_k, \dots \end{array} \middle| \begin{array}{l} p(t_1, \dots, t_n) \leftarrow \dots, \\ q(s_1, \dots, s_k), \\ \dots \\ \in P_2 \end{array} \right\}.$$

Analogical reasoning is characterized as deduction from an analogical union of programs.

**Theorem 3** [6] Let  $P_1$  and  $P_2$  be programs and  $\varphi$  be a pairing between  $P_1$  and  $P_2$ . Let  $p(t_1, \dots, t_n)$  be an atom. For each  $i$  ( $i = 1, 2$ ), the following conditions are equivalent:

- (a)  $p(t_1, \dots, t_n) \in M_i^*$ .
- (b)  $P_1\varphi P_2 \vdash p_i(t_1, \dots, t_n)$ .

## 4 Generalization by Analogy

In analogical reasoning, we can acquire ground rules into domains by transformations of rules. We need to generalize the transformed rules in EBG with incomplete domain theories. Transformations of rules are not suitable for making up rules necessary for domain theories, because the transformed rules are ground. As a method of obtaining generalizations of rules necessary for domain theories, in this section, we propose generalizations of rules by an analogy.

Let  $P_i$  be a program,  $F_i$  be the set of all function symbols in  $P_i$  and  $V_i$  be the set of all variable symbols in  $P_i$  ( $i = 1, 2$ ). Let  $V$  be a finite set of variable symbols which do not appear in  $V_1$  or  $V_2$ , and  $U$  be the set of all terms that are formed by symbols appearing in  $F_1, F_2$  or  $V$ . We define a *projection mapping*  $\pi_i$  ( $i = 1, 2$ ) as follows:

$$\pi_1(\langle t, t' \rangle) = t \text{ and } \pi_2(\langle t, t' \rangle) = t'.$$

For a pairing  $\varphi$ , let  $\varphi_i$  and  $\varphi_i^+$  be  $\pi_i(\varphi)$  and  $\pi_i(\varphi^+)$  ( $i = 1, 2$ ), respectively.

**Definition 12** We say that a partial identity  $\varphi$  between  $P_1$  and  $P_2$  is *redundant* if  $(\varphi \setminus \{\langle t, t' \rangle\})^+ = \varphi^+$  for some  $\langle t, t' \rangle \in \varphi$ .

When we get a partial identity  $\varphi$ , we construct a partial identity  $\phi$  which is not redundant and satisfies  $\varphi^+ = \phi^+$ . Thus, from now on, by  $\varphi$  we denote a partial identity between  $P_1$  and  $P_2$  which is not redundant.

**Definition 13** We call a one-to-one mapping  $G_{\varphi_i}$  ( $i = 1, 2$ ) from  $\varphi_i$  to  $V$  a *generalization mapping* by  $\varphi$ . We define the one-to-one mapping  $G_{\varphi_i}^+$  from  $\varphi_i^+$  to  $U$  as follows:

- (a)  $G_{\varphi_i}^+(t) = G_{\varphi_i}(t)$  if  $t \in \varphi_i$ ,
- (b)  $G_{\varphi_i}^+(f(t_1, \dots, t_n)) = f(G_{\varphi_i}^+(t_1), \dots, G_{\varphi_i}^+(t_n))$  if  $f(t_1, \dots, t_n) \notin \varphi_i$ .

**Definition 14** Let  $\alpha$  be a ground atom holds in  $P_i$  ( $i = 1, 2$ ). For a generalization mapping  $G_{\varphi_i}$  by  $\varphi$ , we say that  $\alpha'$  is a *generalization* of  $\alpha$  by  $\varphi$ , denoted by  $G_{\varphi_i}(\alpha) = \alpha'$ , if  $\alpha$  and  $\alpha'$  are written as

$$\alpha = p(t_1, \dots, t_n) \text{ and } \alpha' = p(t'_1, \dots, t'_n),$$

respectively, where  $p$  is a predicate symbol appearing in  $P_i$ , and  $G_{\varphi_i}^+(t_j) = t'_j$  holds for each  $j$  ( $1 \leq j \leq n$ ).

A generalization can be extended from atoms to rules in a natural way.

**Definition 15** Let  $R$  be a ground rule holds in  $P_i$  ( $i = 1, 2$ ). For a generalization mapping  $G_{\varphi_i}$  by  $\varphi$ , we say that  $R'$  is a *generalization* of  $R$  by  $\varphi$ , denoted by  $G_{\varphi_i}(R) = R'$ , if  $R$  and  $R'$  are written as

$$R = \alpha \leftarrow \beta_1, \dots, \beta_n \text{ and } R' = \alpha' \leftarrow \beta'_1, \dots, \beta'_n,$$

respectively, and  $G_{\varphi_i}(\alpha) = \alpha'$  and for each  $j$  ( $1 \leq j \leq n$ ),  $G_{\varphi_i}(\beta_j) = \beta'_j$  hold.

**Definition 16** We define a set  $A_i^*$  ( $i = 1, 2$ ) as follows:

$$A_i^* = \bigcup_n A_i^n,$$

$$A_i^0 = M_i,$$

$$A_i^{n+1} = \{\alpha \mid \alpha \text{ is a ground atom such that } H^n \cup A_i^n \cup P_i \vdash \alpha \text{ holds}\} \quad (n \geq 0),$$

$$H^n = \{R \mid R \text{ is a generalization of a rule in } E_1^n \cup E_2^n \text{ by } \varphi.\}$$

$$E_i^n = \left\{ R = \alpha \leftarrow \alpha_1, \dots, \alpha_m \mid \begin{array}{l} R \text{ is a ground instance of a rule in } P_i \\ \text{such that each } \alpha_j \text{ is in } A_i^n \text{ (} 1 \leq j \leq m \text{)} \end{array} \right\}.$$

Let  $H^* = \bigcup_n H^n$ . Namely,  $H^*$  is the set of all generalizations of rules by  $\varphi$ . Then  $A_i^*$  is the set of all ground atoms reasoned by usual deductions and generalizations of rules by  $\varphi$ . From this definition of  $A_i^*$ , we have the following proposition and theorems.

**Proposition 2** Let  $P_1$  and  $P_2$  be programs and  $\varphi$  be a partial identity between  $P_1$  and  $P_2$ . Let  $G_{\varphi_i}$  ( $i = 1, 2$ ) be a generalization mapping by  $\varphi$ . Let  $\alpha$  and  $\alpha'$  be in  $A_1^*$  and  $A_2^*$ , respectively. If  $\alpha$  and  $\alpha'$  are identified by  $\varphi$ , then

$$G_{\varphi_1}(\alpha)\theta = G_{\varphi_2}(\alpha')$$

holds for a variable renaming substitution  $\theta$ .

**Proof.** Clearly it suffices to prove that there exists a variable renaming substitution  $\theta$  such that  $G_{\varphi_1}^+(t)\theta = G_{\varphi_2}^+(t')$  for any  $\langle t, t' \rangle \in \varphi^+$ .

Since  $\varphi$  is a partial identity, for  $s \in \varphi_1$ , there exists only one  $s' \in \varphi_2$  such that  $\langle s, s' \rangle \in \varphi$ . Let  $G_{\varphi_1}(s) = X$  and  $G_{\varphi_2}(s') = Y$ . Then we have  $G_{\varphi_1}(s)\{X := Y\} = G_{\varphi_2}(s')$  for a substitution  $\{X := Y\}$ . Now let  $\theta = \{X := Y \mid \langle s, s' \rangle \text{ is in } \varphi \text{ and } G_{\varphi_1}(s) = X \text{ and } G_{\varphi_2}(s') = Y \text{ hold}\}$ . Hence we have  $G_{\varphi_1}^+(t)\theta = G_{\varphi_2}^+(t')$  for any  $\langle t, t' \rangle \in \varphi^+$ .  $\square$

For atoms  $\alpha$  and  $\beta$ , we write  $\alpha \leq \beta$  when  $\beta\theta = \alpha$  for a substitution  $\theta$ , and  $\alpha \simeq \beta$  when  $\beta \leq \alpha$  and  $\alpha \leq \beta$ . Then let  $[\alpha]$  denote the equivalence class of  $\alpha$  under the equivalence relation  $\simeq$ . We also write  $[\alpha] \leq [\beta]$  when  $\alpha \leq \beta$ , and  $[\alpha] = [\beta]$  when  $\alpha \simeq \beta$ .

By Proposition 2, we have  $[G_{\varphi_1}(\alpha)] = [G_{\varphi_2}(\alpha')]$ . Furthermore we note that  $[G_{\varphi_i}(\alpha)] = [G'_{\varphi_i}(\alpha)]$  holds for any generalization mappings  $G_{\varphi_i}$  and  $G'_{\varphi_i}$  ( $i = 1, 2$ ) by  $\varphi$ .

**Theorem 4** Let  $P_1$  and  $P_2$  be programs and  $\varphi$  be a partial identity between  $P_1$  and  $P_2$ . Let  $G_{\varphi_i}$  ( $i = 1, 2$ ) be a generalization mapping by  $\varphi$ . Let  $\alpha$  and  $\alpha'$  be in  $A_1^*$  and  $A_2^*$ , respectively. If  $\alpha$  and  $\alpha'$  are identified by  $\varphi$ , then

$$G_{\varphi_1}(\alpha)\theta_1 = \alpha' \text{ and } G_{\varphi_2}(\alpha')\theta_2 = \alpha$$

hold for some substitutions  $\theta_1$  and  $\theta_2$ .

**Proof.** We show  $G_{\varphi_1}(\alpha)\theta_1 = \alpha'$  for a substitution  $\theta_1$ . By Proposition 2, we have  $G_{\varphi_1}(\alpha)\theta = G_{\varphi_2}(\alpha')$ , for a variable renaming substitution  $\theta = \{X := Y \mid \langle t, t' \rangle \text{ is in } \varphi \text{ and } G_{\varphi_1}(t) = X \text{ and } G_{\varphi_2}(t') = Y \text{ hold}\}$ . Let  $\delta = \{Y := t' \mid \langle t, t' \rangle \text{ is in } \varphi \text{ and } G_{\varphi_2}(t') = Y \text{ holds}\}$ . By the definition of  $G_{\varphi_2}$ ,  $G_{\varphi_2}(\alpha')\delta = \alpha'$ . Then we have  $G_{\varphi_1}(\alpha)\theta\delta = \alpha'$ . Now let  $\theta_1 = \theta\delta$ . Hence we have  $G_{\varphi_1}(\alpha)\theta_1 = \alpha'$ . The proof for  $\theta_2$  is similar.  $\square$

We have the same results on generalizations of rules as those of atoms.

By Theorem 4, we can get a ground rule  $R'$  from a ground rule  $R$  which holds in  $P_i$  by generalizations of rules by  $\varphi$  and its ground substitution, if we can get  $R'$  from  $R$  by transformations of rules.

**Theorem 5** *Let  $P_1$  and  $P_2$  be programs and  $\varphi$  be a partial identity between  $P_1$  and  $P_2$ . Then*

$$M_i^* \subseteq A_i^* \quad (i = 1, 2).$$

**Proof.** We prove the theorem by an induction on  $n$ . Suppose first that  $n = 0$ . Then we have  $M_i^0 = M_i = A_i^0$  ( $i = 1, 2$ ). Suppose also that  $M_i^n \subseteq A_i^n$  for  $n \geq 0$  ( $i = 1, 2$ ). For any  $\alpha \in M_1^{n+1}$ , if  $M_1^n \cup P_1 \vdash \alpha$ , then  $\alpha \in A_1^{n+1}$  by the induction hypothesis. If not, there exists a rule  $C = (\alpha \leftarrow \beta_1, \dots, \beta_m) \in R_1^n$  such that  $M_1^n \cup P_1 \vdash \beta_1, \dots, \beta_m$  holds. Then, there exists a ground instance  $C' = (\alpha' \leftarrow \beta'_1, \dots, \beta'_m)$  of a rule in  $P_2$  such that  $C$  and  $C'$  are  $(\varphi, M_1^n, M_2^n)$ -analogous. Since  $\beta'_1, \dots, \beta'_m \in M_2^n$ , we have  $\beta'_1, \dots, \beta'_m \in A_2^n$  by the induction hypothesis. Consequently,  $C'$  is in  $E_2^n$ . Furthermore there exists  $R$  in  $H^{n+1}$  which is a generalization of  $C'$  by  $\varphi$ , since  $C$  and  $C'$  are  $(\varphi, M_1^n, M_2^n)$ -analogous. By Theorem 4, we have  $R\theta = C$  for a substitution  $\theta$ . Since  $M_1^n \cup P_1 \vdash \beta_1, \dots, \beta_m$ , we have  $A_1^n \cup P_1 \vdash \beta_1, \dots, \beta_m$  by the induction hypothesis. Consequently,  $\alpha$  is in  $A_1^{n+1}$ . Hence  $M_1^* \subseteq A_1^*$ . The proof for  $M_2^*$  is similar.  $\square$

## 5 EBG by Analogical Reasoning

In EBG it is assumed that a domain theory is sufficient to prove that the inferred generalizations deductively follow from what the learner already knows[11]. In this section, we take off this assumption, and define a notion of EBG with incomplete domain theories.

We consider two programs  $P_1$  and  $P_2$  with analogous training examples  $T_1$  and  $T_2$  and possibly incomplete domain theories  $D_1$  and  $D_2$ , respectively. First, we give an analogy  $\varphi$  as a partial identity between  $P_1$  and  $P_2$ . Then using generalizations of rules by  $\varphi$  we make up rules necessary for domain theories. In this way, we construct EBG by analogical reasoning.

Let  $G$  and  $O$  be a goal concept and an operability criterion, respectively, and let  $I_i = (P_i, P_j, G, O)$  ( $i \neq j$ ). Then we call  $I_i$  an *input*.

**Definition 17** *An explanation tree for  $I_i = (P_i, P_j, G, O)$  ( $i \neq j$ ) is a finite tree satisfying the following conditions:*

- (a) *Each node of the tree is an element of  $M_i^*$ .*

- (b) The root node is an instance  $g$  of  $G$ .
- (c) If a node  $\alpha$  in the tree has children  $\beta_1, \dots, \beta_n$  ( $n \geq 1$ ) in this order, then  $\alpha \leftarrow \beta_1, \dots, \beta_n$  is a ground instance of a rule in  $P_i$  or  $R_i^*$ , and  $\alpha$  is not an element of  $\Pi(O)$ .
- (d) Each node without children is an element of  $\Pi(O)$ .

We call node  $g$  a *goal example* of  $G$ .

We define a *reformation program* of  $P_i$  for  $g$ , denoted by  $P_i(\varphi : g)$ , to be the set of all rules  $C_1, C_2, \dots$  used in the explanation tree as follows:

- (a) Assign a number to each node that is not an element of  $\Pi(O)$  in depth-first order starting from 1.
- (b) Let  $\alpha$  be a node which has children  $\beta_1, \dots, \beta_n$  ( $n \geq 1$ ) in this order, and to which number  $j$  assigned. Then, if there exists a rule in  $P_i$  which is a generalization of  $\alpha \leftarrow \beta_1, \dots, \beta_n$ , assign the number  $j$  to this rule. Otherwise assign the number  $j$  to the generalization of  $\alpha \leftarrow \beta_1, \dots, \beta_n$  by  $\varphi$ . The rule assigned the number  $j$  is denoted by  $C_j$ .

**Definition 18** A *generalized derivation* for  $I_i = (P_i, P_j, G, O)$  ( $i \neq j$ ) is an SLD-derivation via  $CR$  which consists of a finite sequence  $\leftarrow G_0 (= \leftarrow G), \leftarrow G_1, \dots, \leftarrow G_n$  of goals, a sequence  $C'_1, \dots, C'_n$  of variants of rules in  $P_i(\varphi : g)$  and a sequence  $\theta_1, \dots, \theta_n$  of most general unifiers, where  $CR$  is a computation rule to select the leftmost atom that is not in  $\Pi(O)$ .

$C'_k$  ( $1 \leq k \leq n$ ) is a suitable variant of  $C_k$  in  $P_i(\varphi : g)$ , just as in EBG.

An *explanation* is to construct an explanation tree, and a *generalization* is to regress a goal concept through the explanation tree using a generalized derivation.

**Definition 19** *EBG by analogical reasoning (EBG-by-AR, for short)* is to derive a general definition  $R$  of  $G$  from its input  $I_i = (P_i, P_j, G, O)$  ( $i \neq j$ ) and an analogy  $\varphi$  between  $P_1$  and  $P_2$  by an explanation and a generalization, and it is denoted by

$$I_i \xrightarrow[\varphi]{EBG} R,$$

where

$$R = \begin{cases} G & \text{if } n = 0, \\ G\theta_1 \dots \theta_n \leftarrow G_n & \text{otherwise,} \end{cases}$$

and  $(\leftarrow G_n)$  is the goal and  $(\theta_1, \dots, \theta_n)$  is the sequence of most general unifiers in the generalized derivation of depth  $n$ .

In EBG-by-AR, We deal a reformation program constructed from incomplete programs as a complete program necessary for EBG.

We can also deal with an analogical union of the incomplete programs instead of a reformation program. Now we consider which is better to use, an analogical union of incomplete programs or a reformation program.

**Example 1** Let  $\varphi = \{\langle \text{john}, \text{abel} \rangle, \langle \text{car}, \text{motorbike} \rangle, \langle \text{money}, \text{spirit} \rangle\}$  and  $P_i = D_i \cup T_i$  ( $i = 1, 2$ ), where

$$D_1 = \{get(\text{john}, Y) \leftarrow want(\text{john}, Y), have(\text{john}, \text{money}), mighty(\text{money})\},$$

$$T_1 = \left\{ \begin{array}{l} need(\text{john}, \text{car}), \\ have(\text{john}, \text{money}), \\ mighty(\text{money}) \end{array} \right\},$$

$$D_2 = \{want(X, Y) \leftarrow need(X, Y)\},$$

$$T_2 = \left\{ \begin{array}{l} need(\text{abel}, \text{motorbike}) \\ have(\text{abel}, \text{spirit}) \\ mighty(\text{spirit}) \end{array} \right\}.$$

Let  $I_2 = (P_2, P_1, G, O)$  be an input, where  $O$  is  $\{need, have, mighty\}$  and  $G$  is  $get(X, Y)$ . As is easily seen, we cannot construct the explanation tree only from an input  $(P_2, G, O)$  because  $P_2$  is not complete. We consider  $P_1 \varphi P_2$  and  $P_2(\varphi : g)$  in stead of  $P_2$ , where  $g = get(\text{abel}, \text{motorbike})$  is a goal example of  $G$ .

The analogical union  $P_1 \varphi P_2$  is

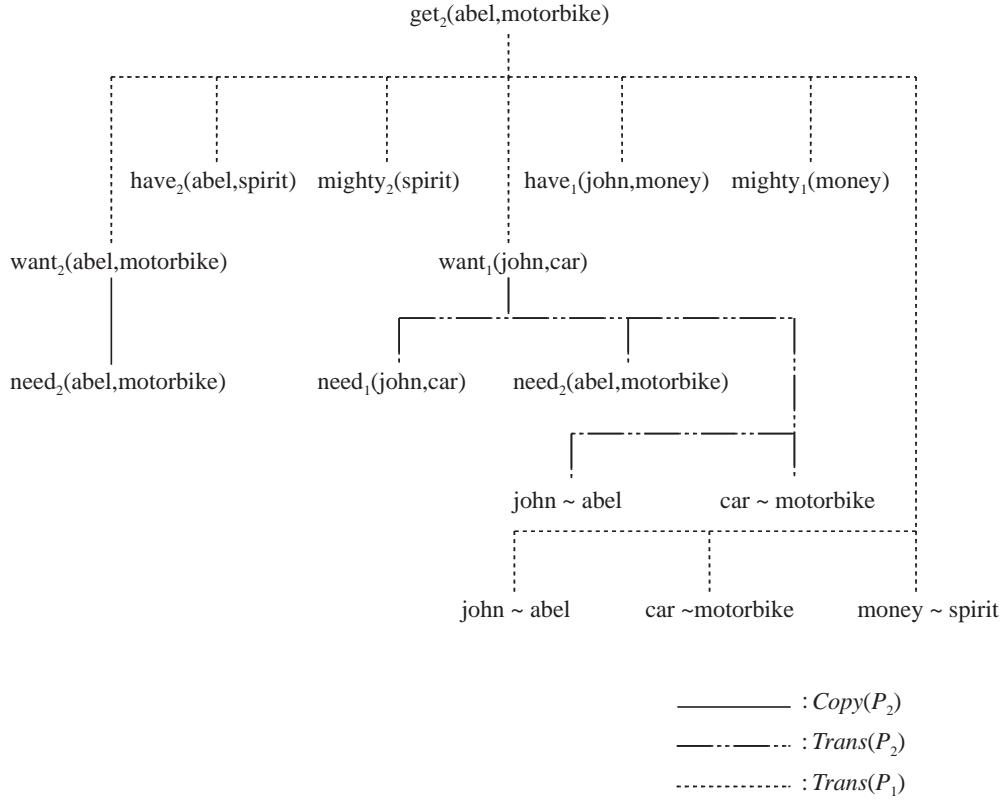
$$\left\{ \begin{array}{l} get_1(\text{john}, Y) \leftarrow want_1(\text{john}, Y), have_1(\text{john}, \text{money}), mighty_1(\text{money}), \\ want_1(V, W) \leftarrow need_1(V, W), \\ \quad V \sim X, W \sim Y, \\ \quad need_2(X, Y), \\ need_1(\text{john}, \text{car}), \\ have_1(\text{john}, \text{money}), \\ mighty_1(\text{money}), \\ want_2(X, Y) \leftarrow need_2(X, Y), \\ get_2(U, W) \leftarrow want_2(U, V), have_2(U, W), mighty_2(W), \\ \quad john \sim U, Y \sim V, money \sim W, \\ \quad want_1(\text{john}, Y), have_1(\text{john}, \text{money}), mighty_1(\text{money}), \\ need_2(\text{abel}, \text{motorbike}), \\ have_2(\text{abel}, \text{spirit}), \\ mighty_2(\text{spirit}), \\ john \sim \text{abel}, \\ car \sim \text{motorbike}, \\ money \sim \text{spirit} \end{array} \right\}.$$

The reformation program  $P_2(\varphi : g)$  is

$$\left\{ \begin{array}{l} want(X, Y) \leftarrow need(X, Y), \\ get(U, W) \leftarrow want(U, V), have(U, W), mighty(W), \\ need(\text{abel}, \text{motorbike}), \\ have(\text{abel}, \text{spirit}), \\ mighty(\text{spirit}) \end{array} \right\}.$$

$P_2(\varphi : g)$  is much simpler than  $P_1\varphi P_2$ , that essentially includes  $\varphi$  itself in atoms with predicate symbol  $\sim$ . The tree for  $P_2(\varphi : g)$  is also much simpler than that for  $P_1\varphi P_2$  as in Figure 4. Moreover, all the leaves in the tree for  $P_2(\varphi : g)$  satisfy the operability criterion  $O$ , i.e., instances of some atoms in  $\Pi(O)$ , while do not the leaves for  $P_1\varphi P_2$ .

### Tree for $P_1\varphi P_2$



### Tree for $P_2(\varphi : g)$

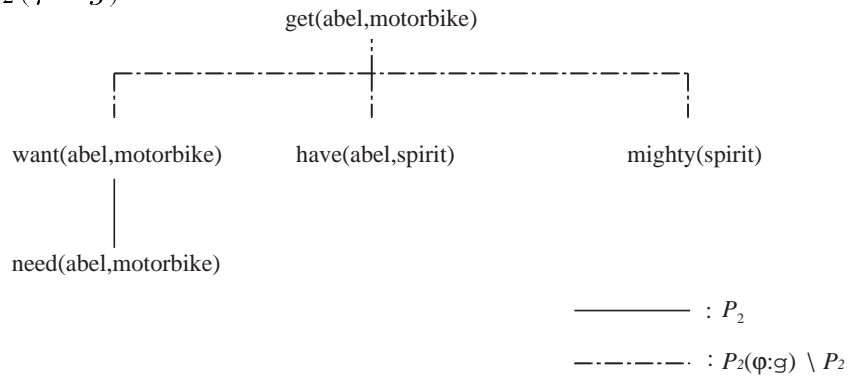


Figure 4: Trees for  $P_1\varphi P_2$  and  $P_2(\varphi : g)$  in Example 1.

By Theorem 4 and 5, the reformation program  $P_i(\varphi : g)$  in EBG-by-AR is a finite subset of a union set of  $P_i$  and  $H^*$ . In general, the following proposition holds for  $P_1\varphi P_2$  and  $H^* \cup P_i$ .

**Proposition 3** *Let  $P_1$  and  $P_2$  be programs,  $\varphi$  be a partial identity between  $P_1$  and  $P_2$  and  $p(t_1, \dots, t_n)$  be a ground atom. If  $P_1\varphi P_2 \vdash p_i(t_1, \dots, t_n)$  for each  $i$  ( $i = 1, 2$ ), then there exists a finite subset  $F$  of  $H^* \cup P_i$  such that  $F \vdash p(t_1, \dots, t_n)$ .*

**Proof.**  $P_1\varphi P_2 \vdash p_i(t_1, \dots, t_n)$  iff  $p(t_1, \dots, t_n) \in M_i^*$  by Theorem 3. Then we have  $p(t_1, \dots, t_n) \in A_i^*$  by Theorem 5. Hence the proof completes by the definition of  $A_i^*$ .  $\square$

For these reasons, we use the reformation program  $P_i(\varphi : g)$  instead of the analogical union  $P_1\varphi P_2$ .

## 6 A Realization of EBG by AR

The EBG-by-AR system is a unification of the EBG system and the analogical reasoning system. The unified system carries out EBG and analogical reasoning simultaneously. If there are no missing rules in a domain theory, the system carries out EBG. If there are some, it carries out making up the missing rules by means of generalizations of rules by an analogy. Unifiability of terms is essential for a realization of EBG-by-AR. Hence, we require the analogy to be a partial identity.

Now we present a definition of EBG-by-AR as a Prolog program according to [2, 5, 10, 16].

```
(C1) ebg_ana(Goal, GenGoal, Leaves, Pairing, Wa):-
    setof(Pair, reason(Goal, GenGoal, Leaves, Pair, Wa), Pair1),
    complete(Pair1, Pairing).
(C2) reason(Goal, GenGoal, Leaves, Pairing, Wa):-
    reason_ebg_ana(Goal, GenGoal, Leaves, [], Pair, Wa),
    reason_ana(Goal, [], Pairing, Wa), check(Pair, Pairing).
(C3) reason_ebg_ana(Leaf, GenLeaf, GenLeaf, Eq, Eq, Wa):-
    operational(Leaf), !, r_ana(Leaf, Wa).
(C4) reason_ebg_ana(A, GA, L, Eq, Eq1, Wa):-
    fact(Wa, (GA:-GAs)), copy((GA:-GAs), (A:-As)),
    reason_ebg_ana(As, GAs, L, Eq, Eq1, Wa).
(C5) reason_ebg_ana((A, As), (G, Gs), (L, Ls), Eq, Eq2, Wa):-
    reason_ebg_ana(A, G, L, Eq, Eq1, Wa),
    reason_ebg_ana(As, Gs, Ls, Eq1, Eq2, Wa).
(C6) reason_ebg_ana(A, GA, L, Eq, Eq2, Wa):-
    prematch(A, (B:-Bs), Wa), world(Wa, Wb), r_ana(Bs, Wb),
    ground(B:-Bs), make_clause(Wa-Wb, (GA:-GAs), (B:-Bs), Pair),
    compact(Pair, Pair1), epic(Pair1), consistency(Eq, Pair1, Eq1, Wa),
    copy((GA:-GAs), (A:-As)), reason_ebg_ana(As, GAs, L, Eq1, Eq2, Wa),
    epic(Eq2).
```

In our system, an atom  $A$  and a rule  $C \leftarrow B_1, \dots, B_n$  in a program  $P_i$  ( $i = 1, 2$ ) are represented by the following Prolog clauses:

$$fact(w_i, (A \leftarrow true))$$

and

$$fact(w_i, (C \leftarrow B_1, \dots, B_n)),$$

respectively, and stored in a Prolog database, where  $w_i$  is a world name for  $P_i$ .

The predicate `ebg_ana` takes a goal example as its first argument, a goal concept as its second argument, and a world name as its fifth argument, and returns the body of a general definition in the third argument, and the set of partial identities between  $P_1$  and  $P_2$  in the fourth argument. The predicate `setof` returns the set of all instances of `Pair` in the third argument if `reason(Goal, GenGoal, Leaves, Pair, Wa)` is successful. The predicate `reason` returns a partial identity in the fourth argument if `reason_ebg_ana(Goal, GenGoal, Leaves, [], Pair, Wa)` is successful. The predicate `reason_ana` computes a partial identity. The predicate `r_ana` carries out analogical reasoning. The predicate `reason_ebg_ana` carries out EBG, if possible; otherwise carries out EBG making up rules necessary for domain theories. The ordered set of clauses (C3), (C4) and (C5) works as a pure Prolog interpreter and the clause (C6) is a proper rule to our EBG-by-AR.

The partial identity is constructed when the system is answering the question just as in the analogical reasoning system[7]. So we do not need to give it in advance.

Now we exemplify the whole reasoning process carried out by the system. Consider the input  $I_2$  in Example 1.

The goals fed to the system are

$$\leftarrow get(X, Y) \quad (\text{in } P_2)$$

and

$$\leftarrow get(abel, motorbike) \quad (\text{in } P_2).$$

For these goals, the system first tries to prove  $get(abel, motorbike) \in M_2$ . Clearly the system fails, and then tries to find a rule in  $P_2$  which has the predicate symbol `get` in the head. The system again fails, and tries to find a rule in  $P_1$  which has the predicate symbol `get` in the head. In this case,

$$get(john, Y) \leftarrow want(john, Y), have(john, money), mighty(money)$$

is chosen. Then the system generates the subgoal

$$\leftarrow want(john, X_1), have(john, money), mighty(money) \quad (\text{in } P_1).$$

Then the system carries out analogical reasoning and then proves that  $want(john, car), have(john, money), mighty(money) \in M_1^*$  under the partial identity

$$\{\langle john, abel \rangle, \langle car, motorbike \rangle\}.$$

Therefore, we have the rule

$$get(john, car) \leftarrow want(john, car), have(john, money), mighty(money)$$

and the system produces a new generalized rule

$$get(X_2, X_3) \leftarrow want(X_2, X_3), have(X_2, X_4), mighty(X_4) \quad (\text{in } P_2)$$

and computes the pairing

$$\varphi_1 = \{\langle john, X_2 \rangle, \langle car, X_3 \rangle, \langle money, X_4 \rangle\}.$$

The system generates the subgoals

$$\leftarrow want(X, Y), have(X, X_4), mighty(X_4) \quad (\text{in } P_2)$$

from the goal  $\leftarrow get(X, Y)$  and the rule

$$get(X_2, X_3) \leftarrow want(X_2, X_3), have(X_2, X_4), mighty(X_4),$$

and

$$\leftarrow want(abel, motorbike), have(abel, X_4), mighty(X_4) \quad (\text{in } P_2)$$

from the goal  $\leftarrow get(abel, motorbike)$  and the rule

$$get(X_2, X_3) \leftarrow want(X_2, X_3), have(X_2, X_4), mighty(X_4).$$

For the former goal, the system tries to find a rule in  $P_2$  which has the predicate symbol  $want$  in the head. Then,

$$want(X, Y) \leftarrow need(X, Y)$$

is chosen. The system generates the subgoals

$$\leftarrow need(X, Y),$$

$$\leftarrow have(X, X_4)$$

and

$$\leftarrow mighty(X_4),$$

and then stops the reasoning since  $need, have, mighty \in O$ . For the latter goal, the system proves  $want(abel, motorbike), have(abel, spirit), mighty(spirit) \in M_2$ . Consequently the system computes the partial identity

$$\varphi = \{\langle john, abel \rangle, \langle car, motorbike \rangle, \langle money, spirit \rangle\}$$

obtained from  $\varphi_1$ , and completes to prove  $get(abel, motorbike) \in M_2^*$ .

The system completes to compute the operationally sufficient condition

$$need(X, Y), have(X, X_5), mighty(X_5)$$

of the goal concept  $get(X, Y)$ . Thus we have the general definition

$$R = (get(X, Y) \leftarrow need(X, Y), have(X, X_5), mighty(X_5))$$

of the goal concept  $get(X, Y)$  and the partial identity

$$\varphi = \{\langle john, abel \rangle, \langle car, motorbike \rangle, \langle money, spirit \rangle\}$$

such that  $I_2 \xrightarrow{EBG} \varphi R$ .

The total system has been implemented in K-Prolog on Sparc Station 10.

Finally we give an other example of EBG-by-AR.

**Example 2**  $W1$  consists of the following clauses:

Domain Theory  $D_1$ :        `great_grandfather(X,Z):-`  
                                          `grandfather(X,Y),father(Y,Z).`  
                                          `parent(X,Y):-mother(X,Y).`  
                                          `parent(X,Y):-father(X,Y).`  
 Training Example  $T_1$ :    `father(yoshihito,hirohito).`  
                                          `father(hirohito,akihito).`  
                                          `father(akihito,naruhito).`

$W2$  consists of the following clauses:

Domain Theory  $D_2$ :    `grandfather(edwardVII,Z):-`  
                                          `father(edwardVII,Y),parent(Y,Z).`  
                                          `parent(X,Y):-mother(X,Y).`  
                                          `parent(X,Y):-father(X,Y).`  
 Training Example  $T_2$ : `father(edwardVII,georgeV).`  
                                          `father(georgeV,georgeVI).`  
                                          `father(georgeVI,elizabethII).`

We put the goal concept  $G$ , the goal example  $g$  and the operational criterion  $O$  as follows:

Goal concept  $G$ :                `great_grandfather(X,Y)`  
 Goal example  $g$ :                `great_grandfather(yoshihito,naruhito)`  
 Operationality Criterion  $O$ : `operational(G)`  
                                          `:-member(G,[father(-,-),mother(-,-)]).`

The question to our EBG-by-AR system is

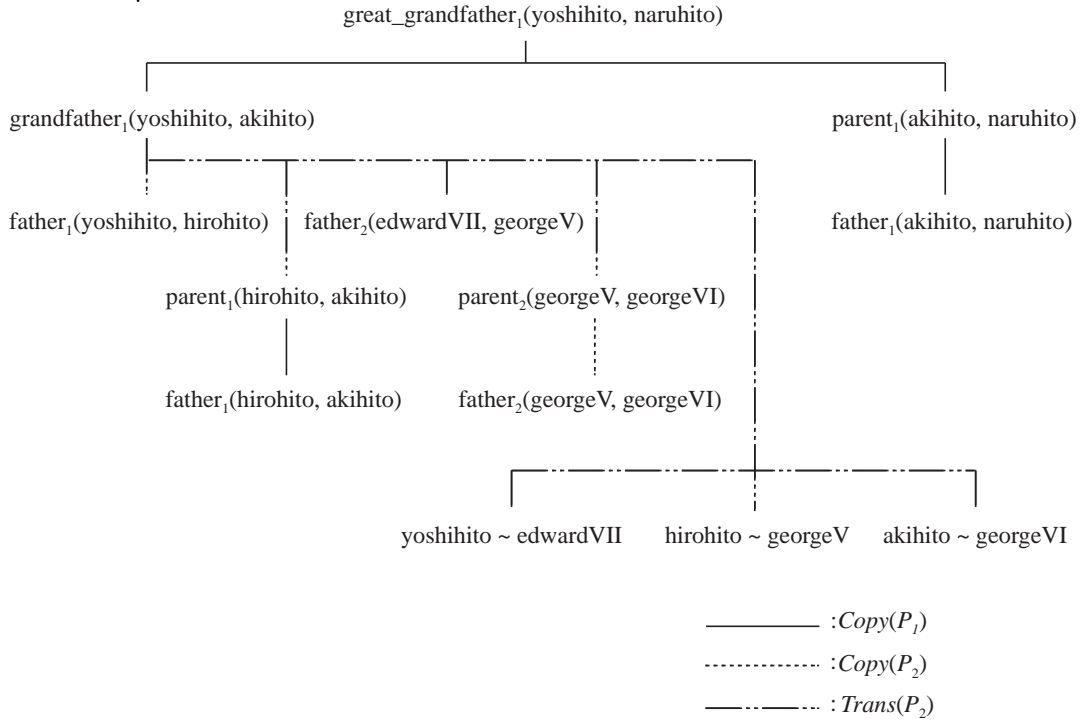
```
?- ebg_ana(great_grandfather(yoshihito,naruhito),
           great_grandfather(X,Y),Leaves,Pairing,w1).
```

The answers from the system are

```
X = X,
Y = Y,
Leaves = father(X,_474),father(_474,_482),father(_482,Y)
Pairing = [[yoshihito-edwardVII,akihito-georgeVI,hirohito-georgeV]]
```

As in the Pairing above, the set of pairings are obtained when the system has made an answer to the question. Figure 5 shows the trees for  $P_1\varphi P_2$  and  $P_1(\varphi : g)$  in Example 2. The tree for  $P_1(\varphi : g)$  is the explanation tree for  $W1$ . We can not construct any explanation tree for  $W1$  in the usual EBG system, but can construct an explanation tree in our EBG-by-AR system.

Tree for  $P_1\varphi P_2$



Tree for  $P_1(\varphi : g)$  (Explanation Tree for  $W1$ )

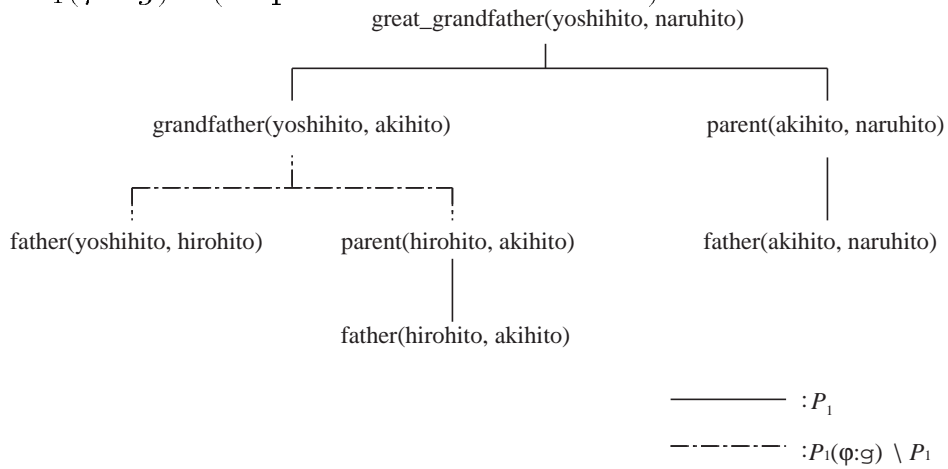


Figure 5: Trees for  $P_1\varphi P_2$  and  $P_1(\varphi : g)$  in Example 2.

## 7 Conclusion

We have formulated EBG in a mathematical way. Then we have shown that EBG is a deductive reasoning from a program which satisfies the input conditions. The ordinary method by analogical reasoning is not suitable for acquiring general rules necessary for domains. Thus we have proposed generalizations by an analogy. In terms of our

formulation, we have formulated EBG-by-AR and realized it, by unifying EBG and analogical reasoning. Our EBG-by-AR carries out EBG even when domain theories are incomplete.

The analogy we treated in the present paper is a pairing between terms. We can generalize our EBG-by-AR for an analogy as a pairing between function symbols and predicate symbols[17].

## References

- [1] K. M. Ali. Augmenting domain theory for explanation-based generalization. In *Proceedings of the Sixth International Workshop on Machine Learning*, pp. 40–42, 1989.
- [2] S. Arikawa and M. Haraguchi. *A Theory of Analogical Reasoning*. Ohm-sha, 1987.
- [3] T. Ellman. Explanation-based learning: A survey of programs and perspectives. *ACM Computing Surveys*, Vol. 21, pp. 163–221, 1989.
- [4] J. Genest, S. Matwin, and B. Plante. Explanation-based learning with incomplete theories: A three-step approach. In *Proceedings of the seventh International Conference on Machine Learning*, pp. 286–294, 1990.
- [5] M. Haraguchi and S. Arikawa. A formulation of analogical reasoning and its realization. *Journal of Japanese Society for Artificial Intelligence* (in Japanese), Vol. 1, pp. 132–139, 1986.
- [6] M. Haraguchi and S. Arikawa. A foundation of reasoning by analogy – analogical union of logic programs. In *Proceedings of Logic Programming Conference 1986 (Lecture Notes in Computer Science 264 Springer-Verlag)*, pp. 58–69. Springer-Verlag, 1987.
- [7] M. Haraguchi and S. Arikawa. Reasoning by analogy as a partial identity between models. In *Proceedings of First International Conference on Analogical and Inductive Inference, (Lecture Notes in Computer Science 265 Springer-Verlag)*, pp. 61–87. Springer-Verlag, 1987.
- [8] M. Koppel. ESBL: an integrated method for leaning from partial information. *Annals of Mathematics and Artificial Intelligence*, Vol. 4, pp. 323–343, 1991.
- [9] J. W. Lloyd. *Foundation of logic programming (second edition)*. Springer-Verlag, 1987.
- [10] H. Matsubara, K. Hanada, and S. Akibe. Explanation-based generalization  $\neq$  partial computation. *Journal of Japanese Society for Artificial Intelligence* (in Japanese), Vol. 6, pp. 276–279, 1991.
- [11] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: a unifying view. *Machine Learning*, Vol. 1, pp. 47–80, 1986.

- [12] M. Numao. Explanation-based learning — approaches by using domain-specific knowledge —. *Journal of Japanese Society for Artificial Intelligence* (in Japanese), Vol. 3, pp. 704–711, 1988.
- [13] M. Numao and M. Shimura. Analogical reasoning by decomposing explanation structure. *Journal of Japanese Society for Artificial Intelligence* (in Japanese), Vol. 6, pp. 716–724, 1991.
- [14] G. Polya. *Induction and Analogy in Mathematics*. Princeton University Press, 1954.
- [15] J. W. Shavlik and G. G. Towell. Combining explanation-based learning and artificial neural networks. In *the sixth International Workshop on Machine Learning*, pp. 90–92, 1989.
- [16] F. van Harmelen and A. Bundy. Explanation-based generalization = partial evaluation. *Artificial Intelligence*, Vol. 36, pp. 401–412, 1988.
- [17] R. Wakizono. A study on analogical reasoning and abstraction. In *JSAI, SIG-FAI-8904-2* (in Japanese), 1990.
- [18] P. H. Winston. Learning and reasoning by analogy. *Communications of ACM*, Vol. 23, pp. 689–703, 1980.
- [19] P. H. Winston. Learning new principles from precedents and exercises. *Artificial Intelligence*, Vol. 19, pp. 321–350, 1983.
- [20] M. Yamamura and S. Kobayashi. An augmented EBL and its application to the utility problem. In *Proceedings of Twelfth International Joint Conference on Artificial Intelligence*, pp. 623–629, 1991.