Partially Isomorphic Generalization and Analogical Reasoning

Hirowatari, Eiju
Research Institute of Fundamental Information Science, Kyushu University

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

出版情報：RIFIS Technical Report. 76, 1993-10. Research Institute of Fundamental Information Science, Kyushu University
バージョン：
権利関係：

Partially Isomorphic Generalization and Analogical Reasoning*

Eiju Hirowatari† Setsuo Arikawa

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

Abstract

Analogical reasoning is carried out based on an analogy which gives a similarity between a base domain and a target domain. Thus, the analogy plays an important role in analogical reasoning. However, computing such an analogy leads to a combinatorial explosion. This paper introduces partially isomorphic generalizations of atoms and rules which make it possible to carry out analogical reasoning without computing the analogy, and also gives a relationship between our generalization and the analogy. Then, we give a procedure which produces such a generalization in polynomial time with respect to the length of a given atom or rule, and realize it as a Prolog program.

1 Introduction

Analogical reasoning is an important paradigm of machine learning [1, 5, 6]. it acquires unknown knowledge by computing an analogy, which gives a similarity between a base domain and a target domain. In analogical reasoning, we first detect an analogy, and then project the well-known knowledge in the base domain on the target domain by using the analogy. Thus, it is an essential point for analogical reasoning to compute an analogy which maps from a base domain to a target domain. Then, Many authors have extensively studied analogical reasoning from this point of vew [1, 5, 6, 11, 16, 17]. However, there often arises a problem of combinatorial explosion in computing analogies [3].

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

†JSPS Fellowship for Japanese Junior Scientists.

In order to solve this problem, we introduce a new concept of partially isomorphic generalizations of atoms and rules. Then, we present a procedure to compute such generalizations in polynomial time, and show that our generalizations make it possible to carry out analogical reasoning without computing an analogy.

The relationships between analogical reasoning and generalizations have been discussed by many researches [2, 4, 14, 15]. Haraguchi [4] and Furtado [2] dealt with generalizations of two atoms in base and target domains, which are nearly the same as Plotkin's generalization [9, 10]. Russell introduced a notion of single instance generalization [14] in connection with EBG and determinations. Vrain and Lu [15] dealt with a generalization of predicates by using a taxonomy of concepts.

Our partially isomorphic generalization is a method to generalize an atom or a rule as general as possible without destroying its syntactical structure. Then, we use this new generalization in analogical reasoning, and show that ground instantiations of the atoms thus generalized are also derived by the ordinary analogical reasoning by Haraguchi and Arikawa [5, 6]. Hence, our generalization can be justified by their theory of analogical reasoning.

This paper is organized as follows. In Section 2, we briefly recall the analogical reasoning based on an analogy as a function from a base domain to a target domain. In Section 3, we prepare some concepts on generalization and instance of atoms necessary for our discussion. In Section 4, we introduce the notion of partially isomorphic generalizations of atoms and rules, show some properties and justification of them, and describe a procedure to compute our generalizations in polynomial time. In Section 5, we realize our generalization procedure as a Prolog program.

2 Analogical Reasoning

In this paper, we deal with logic programs (programs, for short) as the domains for analogical reasoning. Let P_1 and P_2 be base and target programs, respectively, on which analogical reasoning is carried out.

Assume that, in P_1, premises $\alpha_1, \ldots, \alpha_n$ logically imply a fact α. Assume also that premises $\beta_1, \ldots, \beta_n$ hold in P_2, and α_i and β_i $(1 \leq i \leq n)$ are analogous. Then, analogical reasoning is to derive a fact β in P_2 which is analogous to α. Thus, we follow the principle of analogical reasoning by Haraguchi and Arikawa [6] shown in Figure 1 which is based on Polya [11] and Winston [16, 17].

$$
\begin{array}{lccc}
P_1: & \alpha_1, \ldots, \alpha_n & \rightarrow & \alpha \\
 & \Updownarrow \text{similarity} \quad \varphi & & \Updownarrow \\
P_2: & \beta_1, \ldots, \beta_n & \rightarrow & \underline{\beta}
\end{array}
$$

Figure 1: The principle of analogical reasoning.

Then, φ in Figure 1 denotes an analogy which gives a similarity between the base and the target. Analogical reasoning is carried out by projecting some of the base on

the target by using φ. Hence, we take φ as a mapping from the base to the target. Let U_i be the Herbrand universe for P_i $(i = 1, 2)$. Then, we present some definitions for P_1 and P_2.

Definition 1 *Let P_1 and P_2 be base and target programs, respectively. A finite subset φ of $U_1 \times U_2$ is called a pairing between P_1 and P_2. Then, the set φ^+ is defined to be the smallest set that satisfies the following conditions:*

(a) $\varphi \subseteq \varphi^+$, *and*

(b) $\langle t_1, s_1 \rangle, \ldots, \langle t_n, s_n \rangle \in \varphi^+ \Rightarrow \langle f(t_1, \ldots, t_n), f(s_1, \ldots, s_n) \rangle \in \varphi^+$,

where f is a function symbol appearing in either P_1 or P_2.

If we can take a pairing φ as a function such that

$$\varphi^+(t) = s \quad \text{if } \langle t, s \rangle \in \varphi^+,$$

then we say that φ is an *analogy*.

An analogy can be extended from terms to atoms in a natural way as follws:

Definition 2 *Let P_1 and P_2 be base and target programs, respectively. Let α be a ground atom that holds in P_1, and β be a ground atom that holds in P_2. For an analogy φ, we say that α is analogous to β under φ, if α and β are written as*

$$\alpha = p(t_1, \ldots, t_n) \text{ and } \beta = p(s_1, \ldots, s_n) \ ,$$

respectively, for some predicate symbol p appearing in either P_1 or P_2, and $\varphi^+(t_i) = s_i$ holds for each i $(1 \leq i \leq n)$.

3 Generalization and instance of atoms

First, we prepare some concepts on generalization and instance of atoms necessary for our discussions. For detailed definitions on first order logic, logic programming and algebraic structure of atoms, readers should refer to [8, 9, 12].

An atom without variables is called a *ground atom*. A *definite clause* is a clause of the form

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

where $A, B_1, \ldots, B_n$ are atoms. A is called the *head* and $B_1, \ldots, B_n$ is called the *body* of the definite clause. Then, a definite clause with the body is called a *rule* and a rule without variables is called a *ground rule*.

Definition 3 *Let α and β be atoms. Then, α is a generalization of β or β is an instance of α, denoted by $\beta \leq \alpha$, if there exists a substitution θ such that $\alpha\theta = \beta$. Atoms α and β are variants, denoted by $\alpha \simeq \beta$, if $\beta \leq \alpha$ and $\alpha \leq \beta$ hold.*

Definition 4 *Let S be a set of atoms. Then, an atom γ is a common generalization of S if $\alpha \leq \gamma$ for each atom α in S. An atom δ is a common instance of S if $\delta \leq \alpha$ for each atom α in S. An atom γ is a least common generalization of S, if γ is a common generalization of S and $\gamma \leq \gamma'$ for each common generalization γ' of S. An atom δ is a greatest common instance of S, if δ is a common instance of S and $\delta' \leq \delta$ for each common instance δ' of S.*

Let S be a set of generalizations of an atom, and α and β be atoms in S. Then, by $[\alpha]$ we denote the equivalence class of α under the equivalence relation $\simeq$, and by $[S]$ the set of equivalence classes of all atoms in S induced by $\simeq$.

Now we define two binary functions $\sqcup$ and $\sqcap$ on S as follows: $\alpha \sqcup \beta$ is a least generalization of α and β obtained by Plotkin's algorithm [9]. let α' and β' be variants of α and β, respectively, which do not share any variable. Then, $\alpha \sqcap \beta$ is an atom $\alpha'\theta$, where θ is the most general unifier of α' and β' obtained by Robinson's unification algorithm [13].

Furthermore, we define two binary functions $\sqcup$ and $\sqcap$, and a relation $\leq$ on $[S]$ as follows: For $[\alpha]$ and $[\beta]$ in $[S]$,

$$[\beta] \sqcup [\gamma] = [\beta \sqcup \gamma] \ , \ [\beta] \sqcap [\gamma] = [\beta \sqcap \gamma],$$

and $[\alpha] \leq [\beta]$ when $\alpha \leq \beta$.

We have the following lemmas:

Lemma 1 *For atoms α and β with the same n-ary predicate symbol, $\alpha \sqcup \beta$ is a least common generalization of $\{\alpha, \beta\}$ with respect to $\leq$.*

Lemma 2 *For atoms α and β which are unifiable, $\alpha \sqcap \beta$ is a greatest common instance of $\{\alpha, \beta\}$ with respect to $\leq$.*

Lemma 3 *For a set of atoms S, the relation $\leq$ is a partial order on $[S]$.*

By these lemmas, we can prove the following proposition:

Proposition 1 *Let S be the set of all generalizations of an atom. Then, $[S]$ is a lattice with a partial order $\leq$, a join operator $\sqcup$, and a meet operator $\sqcap$.*

Proof. By Lemma 3, $\leq$ is a partial order on $[S]$. By Lemma 1 and 2, for $[\beta]$ and $[\gamma]$ in $[S]$, $[\beta] \sqcup [\gamma]$ and $[\beta] \sqcap [\gamma]$ are in $[S]$, and they are the least common generalization and the greatest common instance of $\{[\beta], [\gamma]\}$, respectively. Hence, $\sqcup$ and $\sqcap$ on $[S]$ are join and meet operators, respectively. □

The lattice $[S]$ is said to be a *normal lattice*.

4 Partially Isomorphic Generalization

In this section, we introduce a new concept of partially isomorphic generalization of atoms and rules.

For u, a term or an atom, let C_u be the set of all constants occurring in u, and V_u be the set of all variables. For an atom α and a term t occurring in α, $\alpha[t]$ is an atom obtained by replacing each occurrence of t in α by a new variable which does not occur in α.

Definition 5 *Let α be an atom and t be a term occurring in α. Then, t is a quasi-replaceable term of α, if t is a constant or a term of the form $f(X_1, \cdots, X_n)$, where f is a function symbol and each X_i is a variable symbol. A term t is a replaceable term of α, if t is a quasi-replaceable term and $V_t \cap V_{\alpha[t]} = \emptyset$ holds.*

Example 1 *The set of all replaceable terms of $p(f(a), a, g(b))$ is $\{a, b\}$.*

The set of all replaceable terms of $p(f(X, Z), g(Y, f(X, Z), a))$ is $\{f(X, Z), a\}$.

Let α be an atom and t be a replaceable term of α. Then, we write $\alpha \rightarrow \beta$ for each variant β of $\alpha[t]$. The relation $\rightarrow$ is a binary relation on a set of atoms. Then, we define $\rightarrow^*$ as the reflexive and transitive closure of $\rightarrow$.

Definition 6 *Let α and β be atoms. Then, β is a partially isomorphic generalization (PIG, for short) of α, if $\alpha \rightarrow^* \beta$.*

As shown later, there exists a PIG γ of an atom α such that $\beta \rightarrow^* \gamma$ holds for each PIG β of α. Then, such an atom γ is a *greatest PIG* of α.

Example 2 *For an atom $\alpha = p(f(a), a, g(b))$, the followings atoms are PIGs of α:*

$$p(f(a), a, g(b)) \ , \ p(f(A), A, g(b)) \ ,$$
$$p(f(a), a, g(B)) \ , \ p(f(a), a, C) \ ,$$
$$p(f(A), A, C) \ ,$$

but not $p(A, B, C)$, where A, B and C are new variables. $p(f(A), A, C)$ is a greatest PIG of α.

For an atom $\beta = p(f(X, Z), g(Y, f(X, Z), a))$, the followings atoms are PIGs of β:

$$p(f(X, Z), g(Y, f(X, Z), a)) \ , \ p(A, g(Y, A, a)) \ ,$$
$$p(f(X, Z), g(Y, f(X, Z), B)) \ , \ p(A, g(Y, A, B)) \ ,$$

but not $p(A, B)$, where A and B are new variables. $p(A, g(Y, A, B))$ is a greatest PIG of β.

The following proposition asserts that, for PIGs β and γ of an atom α, a greatest common generalization and a least common instance of $\{\beta, \gamma\}$ are PIGs of α.

Proposition 2 *For PIGs β and γ of an atom α, there exist atoms $\beta \sqcup \gamma$ and $\beta \sqcap \gamma$ which are PIGs of α.*

Proof. Let β' and γ' be variants of β and γ, respectively, which do not share any variable. Let $\Phi(\beta', \gamma')$ be the set of pairs $\langle t_1, t_2 \rangle$ of terms t_1 and t_2 which occur in the same place in β' and γ', respectively, and of which the left most symbols are distinct. Then, for $\langle t_1, t_2 \rangle$ in $\Phi(\beta', \gamma')$, at least one of t_1 and t_2 is a variable. Let

$$\theta_1 = \{t_1 := t_2 \mid t_1 \text{ is a variable and } \langle t_1, t_2 \rangle \text{ is in } \Phi(\beta', \gamma')\},$$
$$\theta_2 = \{t_2 := t_1 \mid t_1 \text{ is not a variable and } \langle t_1, t_2 \rangle \text{ is in } \Phi(\beta', \gamma')\},$$

and put $\theta = \theta_1 \cup \theta_2$. Then, $\alpha \rightarrow^* \beta'\theta$ and $\beta'\theta \simeq \gamma'\theta \simeq \beta \sqcap \gamma$. Hence, $\beta \sqcap \gamma$ is a PIG of α.

Furthermore, for $\langle t_1, t_2 \rangle$ in $\Phi(\beta', \gamma')$, let β'' be an atom obtained by replacing each occurrence of t_1 in β' by t_2 when t_1 is not a variable, and γ'' be an atom obtained by replacing each occurrence of t_2 in γ' by t_1 when t_2 is not a variable. Then, $\beta \rightarrow^* \beta''$ and $\beta'' \simeq \gamma'' \simeq \beta \sqcup \gamma$. Hence, $\beta \sqcup \gamma$ is a PIG of α. □

For a set S of atoms, the relation $\rightarrow^*$ is not a partial order but a quasi-order on S. Thus, we investigate the set $[S]$ of equivalence classes of all atoms in S. For $[\alpha]$ and $[\beta]$ in $[S]$, we also write $[\alpha] \rightarrow^* [\beta]$ when $\alpha \rightarrow^* \beta$. Then, we can easily prove the following proposition:

Proposition 3 *For a set S of atoms, the relation $\rightarrow^*$ is a partial order on $[S]$.*

Just as we have done with $\leq$ in Section 3, we have the following theorem:

Theorem 1 *Let α be an atom and S be the set of all PIGs of an atoms. Then, $[S]$ is a lattice with a partial order $\rightarrow^*$, a join operator $\sqcup$, and a meet operator $\sqcap$.*

Proof. By Proposition 3, $\rightarrow^*$ is a partial order on $[S]$. By Lemma 1 , 2 and Proposition 2, for $[\beta]$ and $[\gamma]$ in $[S]$, there exist the least common generalization $[\beta \sqcup \gamma]$ and the greatest common instance $[\beta \sqcap \gamma]$ of $\{[\beta], [\gamma]\}$ in $[S]$, respectively. Hence, $\sqcup$ and $\sqcap$ on $[S]$ are join and meet operators, respectively. □

The lattice $[S]$ is said to be a *PIG lattice* . From Theorem 1, for any atom α, there exists a greatest PIG of α. The PIG lattice and the normal lattice is illustrated in Figure 2.

For a ground atom α and an analogy φ, let $Ana(\alpha, \varphi)$ be the set of all ground atoms to which α is analogous under φ, and $G(\alpha)$ be the set of all ground instances of the greatest PIG of α. From now on, we identify an atom with its equivalence class. Then, we have the following theorem:

Theorem 2 *Let α be a ground atom. For each atom β in $G(\alpha)$, there exists an analogy φ such that β is in $Ana(\alpha, \varphi)$.*

Proof. Let α'' be the greatest PIG of α and $X_1, \cdots, X_n$ be all the variables occurring in α''. Then, there exist substitutions

$$\theta = \{X_1 := s_1, \cdots, X_n := s_n\} \quad , \quad \delta = \{X_1 := t_1, \cdots, X_n := t_n\}$$

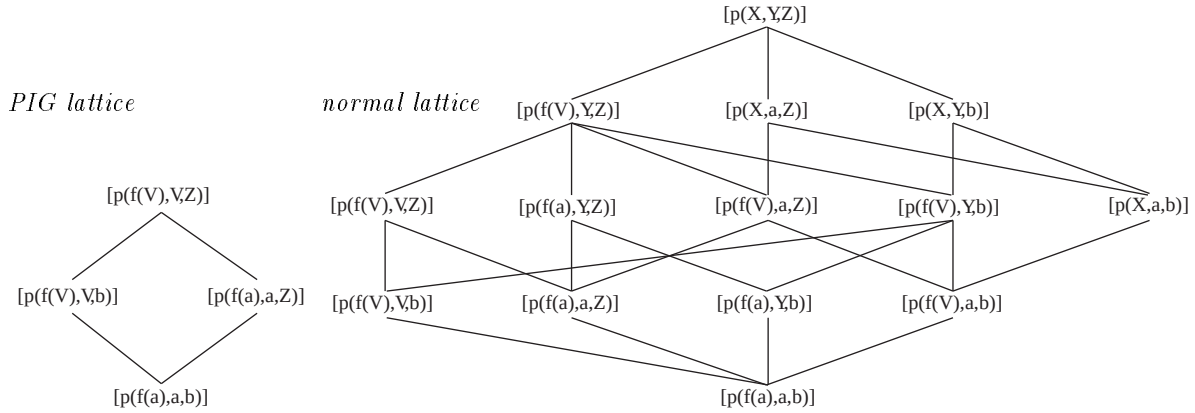

Figure 2: The PIG lattice and the normal lattice for $p(f(a), a, b)$.

such that $\alpha = \alpha''\theta$ and $\beta = \alpha''\delta$ for any β in $G(\alpha)$. Then, $C_{s_i} \cap C_{s_j} = \emptyset$ and $C_{\alpha''} = \emptyset$, where $i \neq j$ for $1 \leq i, j \leq n$. We define a pairing φ as follows:

$$\varphi = \{\langle s_1, t_1 \rangle, \cdots, \langle s_n, t_n \rangle\}.$$

Then, φ is an analogy and $\alpha\varphi\beta$ holds. □

A ground atom α is analogous to all ground instances of the greatest PIG of α under an analogy φ. Hence, in order to obtain an analogy φ and a ground atom to which α is analogous under φ, it suffices to compute the greatest PIG of α. Then, we show an algorithm which computes the greatest PIG of an atom in Figure 3.

Algorithm: G_{PIG}
  input: an atom α
  output: the greatest PIG A of α
begin
  $A := \alpha$;
  $N := 0$;
  while there exists the $(N+1)$-st quasi-replaceable term t
      from the right in A **do begin**
    if t is a replaceable term **then**
      replace each occurrence of t in A by a new variable Z
    else $N := N + 1$
  end;
  output A and halt
end.

Figure 3: The greatest PIG algorithm for an atom.

For an atom α, the *length* of α, denoted by $|\alpha|$, is the number of occurrences of constant, variable and function symbols in α. Then, we have the following propositions:

Proposition 4 *Let α be an atom of length n, and e be Napier's number. The number of nodes in the normal lattice for α is at most $e\ n!$.*

Proof. Let $N(\alpha)$ be the number of nodes of the normal lattice for α. $N(\alpha)$ is the number of combinations of terms which can be replaced by new variables in computing a generalization of α. Thus, $N(\alpha)$ is maximum in case α is of the form $p(a, \cdots, a)$, where p is n-ary predicate symbol and a is a constant symbol. Let $B(i)$ be Bell number and S^i_j $(0 \leq i, j)$ be Stirling number of the first kind. Then, we have

$$\begin{aligned} N(\alpha) &= \sum_{i=0}^{n} {}_nC_i B(i) \\ &\leq \sum_{i=0}^{n} {}_nC_i \sum_{j=0}^{i} S^i_j = \sum_{i=0}^{n} {}_nC_i\ i! = \sum_{i=0}^{n} \frac{n!\ i!}{i!\ (n-i)!} \\ &= n! \left(\frac{1}{0!} + \frac{1}{1!} + \cdots + \frac{1}{n!} \right) = n! \left(e - \frac{e^\theta}{(n+1)!} \right) \\ &< e\ n! \end{aligned}$$

for some $0 < \theta < 1$. □

Proposition 5 *Let α be an atom of length n. The number of nodes in the PIG lattice for α is at most 2^n.*

Proof. Let $P(\alpha)$ be the number of nodes in the PIG lattice for α. $P(\alpha)$ is the number of combinations of terms which can be replaced by new variables in computing a PIG of α. Thus, $P(\alpha)$ is maximum in case α is of the form $p(a_1, \cdots, a_n)$, where p is n-ary predicate symbol and $a_1, \cdots, a_n$ are n distinct constant symbols. Hence, we have

$$\begin{aligned} P(\alpha) &= \sum_{i=0}^{n} {}_nC_i \\ &= 2^n. \end{aligned}$$

□

The following theorem guarantees that we can compute the greatest PIG of an atom in polynomial time.

Theorem 3 *Let α be an atom of length n. The greatest PIG of α can be computed in time $O(n^2)$.*

Proof. To prove this theorem, we evaluate the total run-time of the algorithm G_{PIG} in Figure 3. The majority of the time is spent in the **while** loop. Let k be the number of terms in α which are not variables. Then, $0 \leq k \leq n$. The **while** loop is repeated at most k times. Let A_0 be α, and A_i $(1 \leq i \leq k)$ be the components of A at the end of

the i-th execution of the **while** loop. G_{PIG} checks if there exists the $(N+1)$-st quasi-replaceable term t_i from the right in A_i in time $O(n)$ for $0 \leq N \leq k$. The operation in the **while** loop involves (a) checking if t_i is a replaceable term and (b) replacing each occurrence of t_i in A_i by a new variable Z when t_i is a replaceable term. The operation (a) takes $O(|t_i|n)$ steps in simply comparing the variables in t_i with A_i. The operation (b) takes $O(n)$ steps. Thus, the total run-time of G_{PIG} is $O(\sum_{i=1}^{k} |t_i|n)$. Furthermore, for the sequence $A_0 = \alpha, A_1, \cdots, A_k$, if t_i occurs in A_{i+1} then $|A_{i+1}| = |A_i|$ otherwise $|A_{i+1}| \leq |A_i| - |t_i| + 1$, for $0 \leq i \leq k$. Let $t'_1, \cdots t'_l$ be all replaceable terms in $\{t_1, \cdots t_k\}$ $(0 \leq l \leq k)$. Then, we have

$$\begin{aligned} \sum_{i=1}^{k} |t_k| &\leq |A_k| + \sum_{j=1}^{l} |t'_j| \\ &\leq |A_0| + k = |\alpha| + k \\ &= n + k, \end{aligned}$$

and

$$\sum_{i=1}^{k} |t_i|n \leq n(n+k).$$

Hence, the total run-time of G_{PIG} is $O(n^2)$. □

Just as we have done with PIGs of atoms, we can define PIGs of rules, and we have the same results on rules as those of atoms.

Now we discuss reasoning by PIGs. Let P_1 and P_2 be programs. For each rule C in $P_1 \cup P_2$, we compute the greatest PIG R of C in polynomial time, and then learn a new program P obtained by replacing each C in $P_1 \cup P_2$ by R. Thus, we can acquire the fact derived from P without computing an analogy which often leads to a combinatorial explosion. The fact thus acquired can be derived from P_1 and P_2 by analogical reasoning. Hence, reasoning by PIGs of rules is more useful than the analogical reasoning as far as time complexity is concerned.

5 Prolog implementation

The PIG system we have realized as a Prolog program takes atom as input, constructs PIGs of α in an ordering by the relation $\rightarrow$ and then returns the greatest PIG of α as output. It carries out a rightmost and depth-first search based on the algorithm G_{PIG} in Figure 3. Then, we show the PIG system.

```
(C1) pig(Atom1,Atom):-
        nonvar(Atom1),functor(Atom1,F,N),
        N>0,!,functor(Atom,F,N),
        pig1(N,Atom1,Atom,0).
(C2) pig1(0,_,_,_):-!.
(C3) pig1(N,Atom1,Atom,Cnt1):-
        Cnt1=0,arg(N,Atom1,Arg1),
```

```
        search(Arg1,Arg,Atom1,Cnt1,Cnt),
        (nonvar(Arg)->pig_A(N,Arg,Atom1,Atom,Cnt);
                      pig_B(N,Arg1,Atom1,Atom)),!.
(C4) pig1(N,Atom1,Atom,Cnt1):-
        arg(N,Atom1,Arg1),
        search1(Arg1,Arg2,Atom1,Cnt1,Cnt2),
        search(Arg2,Arg,Atom1,Cnt1,Cnt),
        (nonvar(Arg)->pig_A(N,Arg,Atom1,Atom,Cnt);
                      pig_B(N,Arg1,Atom1,Atom)),!.
(C5) pig_A(N,Arg,Atom1,Atom,Cnt):-
        replace_term(Atom1,Arg,Var,Atom2),
        pig1(N,Atom2,Atom,Cnt).
(C6) pig_B(N,Arg,Atom1,Atom):-
        M is N-1,arg(N,Atom,Arg),
        pig1(M,Atom1,Atom,0).
```

Let α be an atom as input to the system. The predicate `pig` takes α as its first argument and returns the greatest PIG of α as its second argument. The clauses (C2), (C3) and (C4) are proper rules to the system. The predicate `pig1` takes a number N as its first argument, a PIG A of α as its third argument and a number M as its forth argument, and returns a replaceable term t if it exists in the N-th argument of A as its second argument. The predicate `search1` searches the N-th argument of A for the M-th quasi-replaceable term s. The predicate `search` searches the N-th argument of A for the rightmost quasi-replaceable term u to the left of s, and checks if u is a replaceable term. If u is a replaceable term, the clause (C3) or (C4) calls the predicate `pig_A`. Otherwise, the clause calls the predicate `pig_B`. The clause (C5) replaces each occurrence of u in A by a new variable.

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

Example 3 *Let* α *be* $p(f(a),a,g(b))$,
and β *be* $p(f(X),g(a,f(X),Y),b,h(c,Z,c),a)$.

The question to our PIG system is

```
?- pig(p(f(a),a,g(b)),Atom).
```

The answer from the system is

```
Atom = p(f(_212),_212,_210).
```

The question to our PIG system is

```
?- pig(p(f(X),g(a,f(X),Y),b,h(c,Z,c),a),Atom).
```

The answer from the system is

```
Atom = p(_272,g(_264,_272,Y),_268,_266,_264).
```

6 Conclusion

The ordinary methods of generalization [9] of examples often cause non-valid and over generalization, and sometimes they need vast search-spaces. To overcome these difficulties, we have considered syntactic analogies and introduced the notion of PIG. Our PIGs are all valid generalizations in the sense that they are justified by the theory of analogical reasoning. Moreover, each PIG can be computed in polynomial time.

Now we are considering a declarative definition of PIGs, and a kind of completeness of PIGs with respect to the analogical reasoning. Also we are improving our previous work on EBG by analogical reasoning [7] using PIGs.

References

[1] J. Arima. A logical analysis of relevance in analogy. In *Proceedings of the 2nd Workshop on Algorithmic Learning Theory*, pp. 255–265, 1991.

[2] A. L. Furtado. Analogy by generalization–and the quest of the grail. *ACM SIGPLAN Notices*, Vol. 27, pp. 105–113, 1992.

[3] S. Furuya and S. Miyano. Analogy is NP-hard. In *Proceedings of the 2nd Workshop on Algorithmic Learning Theory*, pp. 207–212, 1991.

[4] M. Haraguchi. Towards a mathematical theory of analogy. *Bulletin of Informatics and Cybernetics*, Vol. 21, pp. 29–56, 1985.

[5] 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.

[6] 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.

[7] E. Hirowatari and S. Arikawa. Incorporating explanation-based generalization with analogical reasoning. *Bulletin of Informatics and Cybernetics*, Vol. 26, pp. 13–33, 1994.

[8] J. W. Lloyd. *Foundations of logic programming (second edition)*. Springer-Verlag, 1987.

[9] G. D. Plotkin. A note on inductive generalization. *Machine Intelligence*, Vol. 5, pp. 153–216, 1970.

[10] G. D. Plotkin. A further note on inductive generalization. *Machine Intelligence*, Vol. 6, pp. 101–124, 1971.

[11] G. Polya. *Induction and Analogy in Mathematics*. Princeton University Press, 1954.

[12] J. C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. *Machine Intelligence*, Vol. 5, pp. 135–151, 1970.

[13] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, Vol. 12, pp. 23–41, 1965.

[14] S. J. Russell. Analogy and single-instance generalization. In *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 390–397, 1987.

[15] C. Vrain and Cheng-Ren Lu. An analogical method to do incremental learning of concepts. In *Proceedings of the Third European Working Session on Learning*, pp. 227–235, 1988.

[16] P. H. Winston. Learning and reasoning by analogy. *Communications of ACM*, Vol. 23, pp. 689–703, 1980.

[17] P. H. Winston. Learning new principles from precedents and exercises. *Artificial Intelligence*, Vol. 19, pp. 321–350, 1983.