数理解析研究所講究録
363巻 1979年 278-291

On Semantic Generalization of Examples

Makoto HARAGUCHI

Department of Mathematics, Kagoshima University

1. Introduction

An effective program synthesis system should have so convenient facilities that the descriptions of functions can be exactly and easily realized in it. The function-descriptions fed to such a system are either complete or incomplete. Although one may prefer the complete descriptions to the incomplete ones, giving them is often as difficult as writing the desired programs. On the other hand, when we use the incomplete descriptions, which are comparatively easy to obtain, we have to give the resulting programs careful examinations in order to verify if they are desired ones.

Examples (input-output pairs) of the functions are most typical of the incomplete descriptions in the sense that even the beginners can understand them and that there are a great many functions which are consistent with the examples. Very active studies [5-8] have been made of the ways programs are mechanically constructed from their example data, and the approaches are called "programmings by examples" or "program syntheses from examples".

Obviously the programming by examples is an induction, and the formalizations of the inductions have been proposed in the literature [1,2,3]. By modifying them, we can put the programming by examples into the mathematical theory, and this is what we concern in this paper.

In Section 2, we give two definitions that specify the power of the automatic programming by examples denoted by APE. One of the definitions is the "syntactic generalization" and the other "semantic generalization". The syntactic one is a special case of the "identification in the limit"[1,2] which is the widely accepted concept on the inductions. However, as the mathematical criterion for APE, we adopt the semantic generalization rather than the syntactic one. In Section 3, we compare the syntactic and semantic generalizations, and it is proved that the semantic generalizations are more powerful than the syntactic ones under the reasonable assumption. In the final section, we show that the class RP of all recursive predicates is outside of the scope of the semantic generalizations.

2. Syntactic and semantic generalizations

We formalize APEs by specifying machines. An APE machine is an effective procedure which takes a finite set of examples of the form $(x,y) \in N \times N$ and returns a program i which is an index of the partial recursive function ϕ_i . Moreover an external agency, who may be a user, feeds the APE machine examples of a function and expects the machine to compute the correct programs of the function. Let us consider the factorial program to understand the well-behaved APE machines. We may feel that the program is "identity" from the example set $\{(1,1),(2,2)\}$. By receiving the additional data (3,6), we are aware of our mistakes. For more example $\{(x,x!): 0 \le x \le 7\}$, we will infer that the function is the factorial one. Moreover, we never change our valid inference so long as received data are of the form (x,x!).

Essential points found in the above is that, from rich enough examples, inferred programs are all the same and correct ones. The following definition is the formalization of this statement.

Notation For a partial function f, D(f) denotes the set of all finite example sets of f. Moreover, for a class C of partial functions, D(C) denotes the union of D(g) for all functions g in C.

Definition 1. (Syntactic generalization)

A partial function f is said to be syntactically generalized (from its examples) by an APE machine M, if there exists a set $c \in D(f)$ such that (1) $\phi_{M(c)} \supseteq f$ and

(2) $M(d) = M(c)$ for each d with $c \subseteq d \in D(f)$.

Moreover a class of partial functions C is said to be syntactically generalizable, if there exists an APE machine M with $D(C) \subseteq \mathrm{domain}(M)$ such that any f in C is syntactically generalized by M.

It follows that if c_1 and c_2 satisfy the condition (1) and (2) then $M(c_1) = M(c_2)$ holds. Thus syntactic generalization requires that a unique correct programs i is related to f by the machine M. For this reason, we call Definition 1 a syntactic generalization. Now let us recall the goal of APE machines. An APE machine M is expected to produce the correct programs of the function. Hence, different programs are allowable as long as they are correct. From this viewpoint, we have the following definition.

Definition 2. (Semantic generalization)

A partial function f is said to be semantically generalized (from its examples) by an APE machine M, if there exists $c \in D(f)$ such that $\phi_{M(d)} \supseteq f$ for each d with $c \subseteq d \in D(f)$.
Moreover, a class of partial functions C is said to be semantically generalizable if there exists an APE machine such that $D(C) \subseteq$ domain(M) and each f in C is semantically generalized by M.

The semantic generalization requires that produced programs are all the correct ones if the examples contain sufficiently large informations. Especially, if f is a total recursive function then a unique function related to f by M is just f itself. For this reason, we call Definition 2 a semantic generalization.

Remark that the syntactic and semantic generalizations are modifications of the identification in the limit [1,2] and the "weak identification in the limit" [4] , respectively. By their formulations of the inductions, the generalizations of data are treated as "learning processes" in the environment that the induction machines are fed infinitely long data sequence called the data presentation. However, it is troublesome to consider the occurences of repeated data in the data presentations or the choice of order by which each data id fed to M. Hence, in this paper, we adopt the definition that APE machines really act on data "sets" but not on data "sequences".

3. The class of functions that are same almost everywhere

A syntactic generalization is obviously a semantic one. Is the converse also true? In order to answer the question, we prepare the following recursive function called "degeneration union".

Definition 3. (Degeneration union)

Let du be the recursive function defined by

$$\phi_{du(i,d)}(x) = \text{if } x \in \text{domain}(d) \text{ then value}(x,d) \text{ else } \phi_i(x),$$

where $d \in D$, the class of all finite single-valued subsets of $N \times N$, and value$(x,d) = y$ iff $(x,y) \in d$.

Uses of tables make the structures of programs degenrate. In this sense we call the definition a degeneration union. In general, a generalized object accounts the source of generalization. This means, in the case of APE, that the examples are exactly realized in the APE machine. Formally, an APE machine M is said to be feasible if $\phi_{M(d)} \supseteq d$, whenever M(d) is defined. The proposition bellow is directly from the definition.

Proposition 1. For each M that semantically generalizes a class C, there is a feasible APE machine M' that also semantically generalizes C. Hence, we can assume the feasibility in the case of semantic generalizations, without loss of generality.

Proof. Let us define M' to be M'(d) = du(M(d),d), then M'is clearly feasible. For any f in C, there is a $c \in D(f)$ with $\phi_{M(d)} \supseteq f$ whenever $c \subseteq d \in D(f)$. Because $d \subseteq f$ and $\phi_{M(d)} \supseteq f$, it follows that $\phi_{M'(d)}(x)$ is if $x \in$domain(d) then value(x,d) else $\phi_{M(d)}(x)=f(x)$ for all $x \in$domain(f).

We now give a simple example of the semantic generalization by using the degeneration union. Let DU(i) be the class

$DU(i)=\{\phi_{du(i,d)} : d \in D\}$, then obviously DU(i) is semantically generalized by M defined to be M(d) = du(i,d), and the functions in DU(i) differ each other at most in finite input sets. If domain(ϕ_i) is recursive, we can easily modify M using a decision procedure of domain(ϕ_i) so that it is syntactically generalizable. In case domain(ϕ_i) is not recursive, it seems difficult to generalize DU(i) syntactically. We formally justify this intuition by restricting APE machines to feasible ones. For this purpose we prepare two lemmas-one is a basic result on computability and the other due to Blum [2].

Lemma 1. There exists a 0-1 valued partial recursive function ψ such that any 0-1 valued recursive function cannot be an extension of ψ .

Lemma 2. For any feasible APE machine M with domain(M)=D, there exists a two placed recursive function h such that any partial function is h-honest whenever f is syntactically generalized by M. Here, a partial function f is called h-honest if there is a program j such that (1) $\phi_j \supseteq f$ and

(2) $\Phi_j(x) \leq h(x,f(x))$ almost everywhere on domain(f),

where Φ_i denotes the computational complexity measure (M.Blum,1967).

Proof. Assume that a fixed order < on N×N is given, and let $\Sigma(x,y)$ be the finite family of all s with the properties

$$s \ni (x,y) \quad \text{and} \quad s \subseteq \{(a,b) : (a,b) \leq (x,y)\} .$$

The desired recursive function h is defined by

$$h(x,y) = \max \{\Phi_{M(s)}(x) : s \in \Sigma(x,y)\}.$$

From the assumptions, h is really a recursive function. Let (x_1,y_1), $(x_2,y_2),\ldots$ be the sequence of data of f in the fixed order. Then, for the smallest k with $\Sigma(x_k,y_k) \ni c$ that satisfies the conditions of syntactic generalization, it follows that if $n \geq k$ then $s_n \supseteq s_k \supseteq c$, where $s_i = \{(x_1,y_1),\ldots(x_i,y_i)\}$. Hence, it follows that if $n \geq k$ then $\Phi_{M(s_n)}(x_n) = \Phi_{M(c)}(x_n) \leq h(x_n,y_n)$ holds. This completes the proof.

Now we can prove the important fact mentioned before.

Theorem 1. For a partial recursive function ϕ_i that satisfies Lemma 1, DU(i) is not syntactically generalized by any feasible APE machine.

Proof. Assume that DU(i) is syntactically generalized by M. Then obviously domain(M) is D, the class of all finite functions. Hence, Lemma 2 implies the existence of a recursive function h such that each function in DU(i), especially ϕ_i, has a program j such that $\phi_i \subseteq \phi_j$ and $\Phi_j(x) \leq h(x,\phi_j(x))$ a.e. on domain(ϕ_i). Evaluating $\{\Phi_j(x) : x \in \text{domain}(\phi_i) \text{ and } \Phi_j(x) > h(x,\phi_j(x))\}$ by a constant k, we have $\Phi_j(x) \leq \max\{k,h(x,\phi_j(x))\}$ for all $x \in \text{domain}(\phi_i)$. Moreover, assuming that h is monotonic with respect to the second argument (replace h by $h' = \lambda x.y.\max\{h(x,z) : z \leq y\}$, for example), we have $\Phi_j(x) \leq \max\{k,h(x,1)\}$ for all $x \in \text{domain}(\phi_i)$. Moreover, ϕ is defined as follows.

```
φ(x) = if $\Phi_j(x) \leq \max\{k,h(x,1)\}$
          then [ if $\phi_j(x) > 1$ then 1
                 else $\phi_j(x)$ ]
          else 1 .
```

Clearly ϕ is a 0-1 valued recursive function and an extension of ϕ_i. This is a contradiction.

Thus, under the assumption that APE machines are feasible, we know that the semantic generalizations are more powerful than the syntactic ones, because the class DU(i) in Theorem 1 is semantically generalizable. If we agree that APE machines should be feasible then this section can be closed; otherwise we have to turn our attention to the possibilities of the existence of an APE machine, which is not feasible, to generalize DU(i) syntactically. Since the definition of DU(i) is not dependent on the individual structure of i, the generalizer of DU(i), if any, may be uniform in i. Recall that the semantic generalizer $M_i(d)=du(i,d)$ of DU(i) is clearly uniform in i. Thus we have the conjecture that if, for each i, DU(i) is syntactically generalizable then the generalizer is uniform in i just like the case of semantic generalizations. However, the next theorem shows that this conjecture is impossible. First we present a method for effectively constructing "counter examples" of a syntactic generalizer of DU(i).

Lemma 3. Assume that M syntactically generalizes DU(i). Then there exists a recursive function ϕ uniform in M and i that is not syntactically generalized by M.

Proof. The desired ϕ is defined by the procedure "def" which generates the graph of ϕ.

```
def :=        let $C = \{(0,k)\}$ ; (k is arbitrary.)
        LOOP: let $C_{0,i}$ = empty set for i=0,1;
              let n=1;
              let $C_{n,i} = C_{n-1,i} \cup \{(x,i): x = \min_z [\, z \notin \mathrm{domain}(C \cup C_{n-1}) \,]$
                                                       for $i=0,1\}$ ;
        STEP: [if $M(X) \neq M(C \cup C_{n,0})$ then let $C = C \cup C_{n,0}$ and goto LOOP
                    else if $M(C) \neq M(C \cup C_{n,1})$ then
                              let $C = C \cup C_{n,1}$ and goto LOOP];
        GENE: generate the graph of $\phi_i$ untill
                        $(m,\phi_i(m))$ with $m \notin \mathrm{domain}(C \cup C_{n,i})$ is found;
              (if such a $(m,\phi_i(m))$ is found)
              let $C_{n+1,i} = C_{n,i} \cup \{(m,\phi_i(m))\}$ for i=0,1;
              let n = n+1 and goto STEP;
```

Of course, C, $C_{n,0}$, and $C_{n,1}$ have their values in D, the class of all finite single-valued subsets of N×N. Moreover, $\phi(x) = y$ if and only if (x,y) is in the set C. Under the assumption that M generalizes DU(i) syntactically, we consider the two cases to verify that ϕ is not syntactically generalized by M.

Case 1, domain(ϕ_i) is infinite. Assume that the def never enters LOOP after some finite time. Because domain(ϕ_i) is infinite, GENE always finds the pair $(m,\phi_i(m))$. Hence, the def begins to repeat the process in which GENE and STEP are executed in turn. Consequently the $C \cup C_{n,0}$ and $C \cup C_{n,1}$ define the functions f_0 and f_1 in the limit, respectively. f_0 and f_1 are in DU(i) and have different values at x. Hence $M(C \cup C_{n,0}) \neq M(C \cup C_{n,1})$ eventually holds, because M generalizes DU(i). This implies that LOOP is entered again. This

is a contradiction. Thus LOOP is infinitely often executed, and this implies that there is a chain of data sets $\{C_n\}$ of ϕ with $C_n \subseteq C_{n+1}$ and $M(C_n) \neq M(C_{n+1})$, that is, ϕ is not syntactically generalized by M.
Case 2. domain(ϕ_i) is finite. ϕ is finite iff the def enters GENE and never exit. However, it is proved that ϕ is really a function with infinite domain and LOOP is infinitely often executed. Thus, Case 2 is similar to Case 1.

Theorem 2. Under the assumption that, for any i, DU(i) is syntactically generalizable, there is no effective method to construct the syntactic generalizer m(i) of DU(i).

Proof. Assume that there is a recursive function m such that DU(i) is syntactically generalized by m(i). Then we can obtain the recursive function g such that g(i) is an index of ϕ , in Lemma 3, which is constructed from m(i) and i. From the recursion theorem, there is a $n \in N$ with $\phi_{g(n)} = \phi_n$. This equation implies that ϕ_n is not syntactically generalized by m(n). This contradicts to the fact that $\phi_n \in DU(n)$, that is ϕ_n is syntactically generalized by m(n).

It will be natural for us to have the new conjecture that the semantic generalizations are properly broader than syntactic ones. Of course the semantic generalizations are not universal. We make sure of this fact in the next section.

4. The class of all recursive predicates

In order to investigate the power of the semantic generalizations theoretically, we consider the class RP of all recursive predicates. Gold[3] had proved that RP is not identifiable in the limit. Hence it is a corollary that RP is not syntactically generalizable. Then, is RP semantically generalizable?

<u>Theorem 3.</u> RP is not semantically generalizable.

<u>Proof.</u> Assume to the contrary that RP is semantically generalized by M. Then a recursive predicate, which is never identified from any rich enough examples, is constructed by deceiving M. First notice that domain(M) contains any 0-1 valued finite functions. Assume that the values of $\phi(x)$ are defined for x with $0 \le x \le m$ and let

$d_{n,i} = \{(0,\phi(0)),\ldots,(m,\phi(m)),(m+1,i),\ldots(m+n,i)\}$ for i=0,1

Then by dovetailing on n, find n and x with

$\phi_{M(d_{n,1})}(x) \ne \phi_{M(d_{n,0})}(x)$. Remark that searching x with $\phi_i(x) \ne \phi_j(x)$ is also done by dovetailing. Because RP is generalized by M and because ψ_i defined by

$\psi_i(x)$ = if $x \le m$ then $\phi(x)$ else i are in RP, there is a set e, which is a finite subset of N, such that $\phi_{M(\psi_i|e)} = \psi_i$.

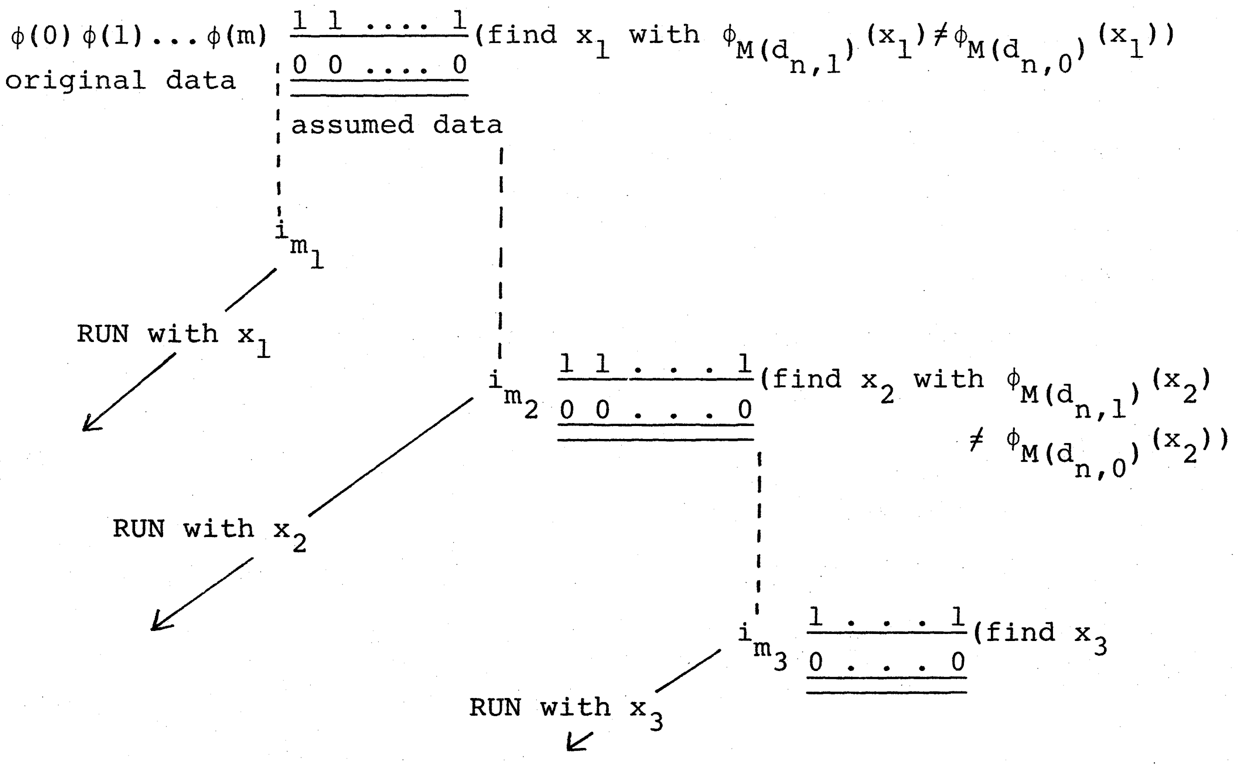
Hence n and x with $\phi_{M(d_{n,1})}(x) \ne \phi_{M(d_{n,0})}(x)$ are really found.

Now execute next two routines in parallel.

RUN: evaluate $\phi_{i_m}(x)$ for $i_m = M(\{\phi(0),\phi(1),\ldots,\phi(m)\})$

ASSUME: "assuming" that ϕ is defined to be $\phi(y)=0$ for $m+1 \le y \le n$, repeat all the processes from the first.

(Refer to the following figure for readers to help to understand the parallelism.)



$\phi(0)\phi(1)\ldots\phi(m)$
original data
$\frac{1\ 1\ \ldots\ 1}{0\ 0\ \ldots\ 0}$ (find x_1 with $\phi_{M(d_{n,1})}(x_1) \neq \phi_{M(d_{n,0})}(x_1)$)
assumed data
i_{m_1}
RUN with x_1
i_{m_2}
$\frac{1\ 1\ .\ .\ .\ 1}{0\ 0\ .\ .\ .\ 0}$ (find x_2 with $\phi_{M(d_{n,1})}(x_2) \neq \phi_{M(d_{n,0})}(x_2)$)
RUN with x_2
i_{m_3}
$\frac{1\ .\ .\ .\ 1}{0\ .\ .\ .\ 0}$ (find x_3
RUN with x_3

Because enlarging the original data of ϕ (i.e. $\phi(0)\phi(1)\ldots\phi(m)$) by $\phi(y)=0$ eventually forces M to produce an index of some total function, it must occure that the call of RUN at some level returns with $\phi_{i_m}(x)\downarrow$. Let k be the number of 0's just before the RUN at this level is called. Since x is chosen so that

$$\phi_{M(\phi(0)---\phi(m)0^k1^n)}(x) \neq \phi_{M(\phi(0)---\phi(m)0^k0^n)}(x),$$

we can select i=o or i=1 so that

$\phi_{i_{m+k}}(x) \neq \phi_{M(\phi(0)---\phi(m)0^ki^n)}(x)$. Now we enlarge the definition of ϕ to be

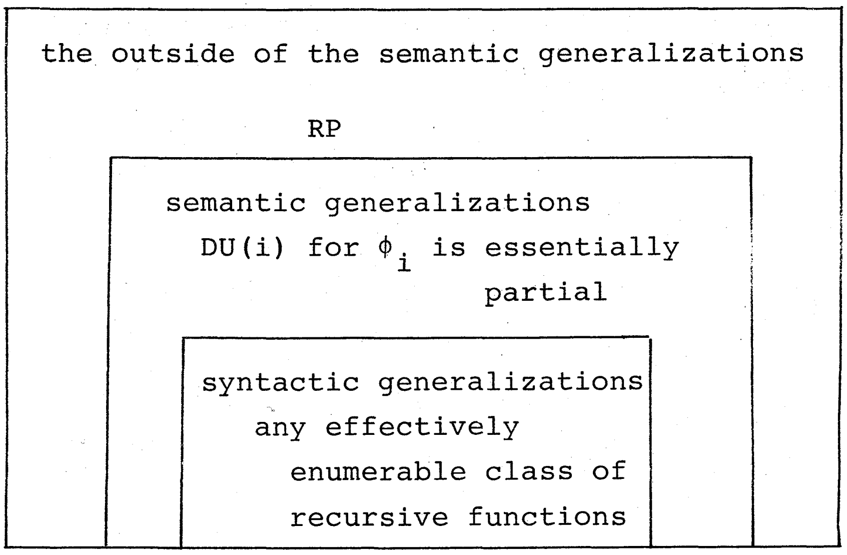
$\phi(m+1)=\phi(m+2)=------=\phi(m+k)=0$ and

$\phi(m+k+1)=\phi(m+k+2)=------=\phi(m+k+n)=i$

Finally, ϕ is defined by the enlargement procedure described as above. Clearly, for any rich enough example set c of ϕ , there exist d_1 and d_2 with $c \sqsubseteq d_i \in D(\phi)$ such that $\phi_{M(d_1)}$ and $\phi_{M(d_2)}$ have defferent values. This implies that ϕ is not semantically generalized by M.

5. Conclusion

Considering the goal of APE machines we have adopted the semantic generalizations as successes of APEs and compared them with the syntactic generalizations. Theorem 1 show that the semantic generalizations have a little wider scope if we consider the feasible generalizations only. However, the class of functions DU(i), that has been considered throughout in Section 2, is a purely theoretical class. It still remains as a question whether there is a practical class that is semantically generalizable but not syntactically generalizable. We close this report by drawing the position of semantic generalizations in the following figure.


the outside of the semantic generalizations
RP
semantic generalizations
DU(i) for ϕ_i is essentially partial
syntactic generalizations
any effectively enumerable class of recursive functions

REFERENCES

[1] E.M. Gold, Language identification in the limit, Information and Control, 10 (1967), 244-262.

[2] L. Blum and M. Blum, Toward a mathematical theory of inductive inference, Information and Control, 38 (1975), 125-155.

[3] E.M. Gold, Limiting recursion, Journal of Symbolic Logic, 20, (1964), 28-48.

[4] M. Haraguchi, A theoretical foundation of a programming by examples, Mem.Fac.Sci.Kyushu Univ.Ser.A., to appear.

[5] A.W. Biermann, Approaches to automatic programming, Advances in Computers Vol. 5 (1976), 1-63.

[6] P.D. Summers, A methodology for LISP program construction from examples, JACM, 17 (1977), 161-175.

[7] S. Hardy, Synthesis of LISP functions from examples, Advanced papers of the 4th IJCAI (1975), 240-245.

[8] D. Show, W. Swartout, and C. Green, Inferring LISP programs from examples, Advanced papers of the 4th IJCAI (1975), 260-267.