数理解析研究所講究録
236巻 1975年 15-20

Recursive Program Schemata and Formal Languages

Takayasu Ito

(Mitsubishi Electric Corp.)

Introduction

Theory of program schemata was initiated by Yanov.
Yanov solved the equivalence problem of Yanov's program schemata, which are equivalent to Graph schemata.
Igarashi and Rutledge reduced Yanov's result into the equivalence problem of finite automata.
Luckham-Park-Paterson, Karp-Miller, Ershov-Itkin and others extended Yanov's result in various ways in connection with automata theory.
In this talk we discuss an application of formal language theory to program schemata.
Applications of formal language theory to program schemata were taken during 1966-1968 while I was in Stanford University.
A part of the results was published in my IEEE symposium on Switching and Automata Theory. In this talk I am going to introduce some of my unpublished results while I was in Stanford University.

Recursive Program Schemata

Example : fact [n] : function defining n !

fact [n] = if n=o then 1 else n $\times$ fact [n-1]

$$\varphi_i(x_1, \cdots, x_n) = \mathcal{E}_i(\varphi_1, \cdots, \varphi_m, x_1, \cdots, x_n)$$

where $\mathcal{E}_i$ is "conditional expression"

containing $\varphi_1, \ldots, \varphi_m, x_1, \ldots, x_n$.

A class of functions defined recursively in this way is called "recursively-defined functions".

If an interpretation is not given for elementary functions in recursively-defined functions, they are called "recursive program schemata".

Example :

$$\varphi[x] = \underline{\text{if}}\ \pi[x]\ \underline{\text{then}}\ a[x]\ \underline{\text{else}}\ h[x, \varphi[S[x]]]$$

$$\downarrow$$

fact [x] = if x=o then 1 else multiply [x, fact [x-1]]

Definition of Equivalence of Recursive Program Schemata

$$\left.\begin{array}{l} \varphi_i^{(1)}(x_1, \cdots, x_n) = \mathcal{E}_i^{(1)}(\varphi_1^{(1)}, \cdots, \varphi_n^{(1)}, x_1, \cdots, x_n) \\ \varphi_i^{(2)}(x_1, \cdots, x_n) = \mathcal{E}_i^{(2)}(\varphi_1^{(2)}, \cdots, \varphi_n^{(2)}, x_1, \cdots, x_n) \end{array}\right\}$$

$$\varphi_1^{(1)}(x_1, \cdots, x_n) \cong \varphi_1^{(2)}(x_1, \cdots, x_n) \Leftrightarrow \forall_I \forall_{\underline{x}} [\varphi_1^{(1)}(x_1, \cdots, x_n) = \varphi_1^{(2)}(x_1, \cdots, x_n)]$$

Theorem 1 (Luckham-Park-Paterson)

"The equivalence problem of recursive program schemata is recursively undecidable"

Proof of Theorem 1 can be made by showing that any LPP schema (Luckham-Park-Paterson schema) can be simulated by a recursive program schema.

Simple Recursive Program Schemata

If a recursive program schema contains only the functions of one argument and one variable, then it is called a simple recursive program schema.

Example :

$$\varphi[x] = \underline{\text{if}}\ p[x]\ \underline{\text{then}}\ f[x]\ \underline{\text{else}}\ \varphi[\varphi[f[x]]]$$

Natation :

(i) $\underline{\text{if}}\ p\ \underline{\text{then}}\ \alpha\ \underline{\text{else}}\ \beta \overset{\text{def}}{\Longleftrightarrow} \lfloor_p\ \alpha \vee \beta \rfloor$

(ii) $\varphi[\varphi[f[x]]] \overset{\text{def}}{\Longleftrightarrow} [f\ \varphi\ \varphi][x]$

Example :

$$\varphi[x] = \lfloor_p f \vee f\varphi\varphi \rfloor[x]$$

$$\varphi \cong \lfloor_p f \vee f\varphi\varphi \rfloor$$

This sort of expressions will be called "simple recursive program schema".

Assume that we have a simple recursive program schema defined by

$$\varphi_i \cong \alpha_i(\varphi_1, \cdots, \varphi_n) \qquad (*)$$

and ϕ is the always undefined function. Then the solution of (*) can be given by $\lim_{k\to\infty} x_i^{(k)}$, the limit of $x_i^{(0)} \equiv \alpha_i(\phi, \cdots, \phi)$, $\cdots$,
$x_i^{(k+1)} \equiv \alpha_i(x_1^{(k)}, \cdots, x_n^{(k)})$

Definition 1

(i) simple recursive program schema defined by

$$x_i \cong \lfloor_{p_{i_1}} a_{i_0} \vee \lfloor_{p_{i_2}} a_{i_1} x_1 \vee \cdots \lfloor_{p_{i_n}} a_{i_{n-1}} x_{n-1} \vee a_{i_n} x_n \rfloor\rfloor \cdots \rfloor$$

is called "right linear".

(ii) if the defining equation is

$$x_i \cong \underset{p_{i_1}}{\llcorner} a_{i_0} \vee \underset{p_{i_n}}{\llcorner} a_{i_1} x_1 b_{i_1} \vee \cdots \underset{p_{i_n}}{\llcorner} a_{i_{n-1}} x_{n-1} b_{i_{n-1}} \vee a_{i_n} x_n b_{i_n} \lrcorner \lrcorner \cdots \lrcorner$$

then it is called "linear".

Definition 2

$\mathbb{E}[P]$: program event of

Program event of P is the set of operator sequences of P under schematic interpretation.

In case of schematic interpretation, semantic functions are defined on operator sequences, assigning truth values to predicates and having the interpretation left open to operators.

Example :

$$\varphi \cong \underset{p}{\llcorner} e \vee \underset{q}{\llcorner} f \varphi f \vee g \varphi g \lrcorner$$

$$\mathbb{E}(\varphi) \text{ : solution of } \varphi = E \cup f \varphi f \cup g \varphi g$$

Theorem 2

(i) If a simple recursive program schema is right-linear, there exists a Yanov schema equivalent to it.

(ii) Equivalence problem of right linear simple recursive program schemata is decidable.

(iii) For any simple recursive program schemata

P_1 and P_2, if $P_1 \cong P_2$, then $\mathbb{E}(P_1) = \mathbb{E}(P_2)$

Difinition

For a class of simple recursive program schema $\mathcal{F}_\alpha$, we define the class of program events by $\mathbb{E}(\mathcal{F}_\alpha) = \{\mathbb{E}(P) : P \in \mathcal{F}_\alpha\}$

Theorem 3

For the class of simple recursive program schemata $\mathcal{F}_\alpha$

(i) $\mathcal{F}_\alpha$ is right linear $\Rightarrow \mathbb{E}(\mathcal{F}_\alpha)$ is regular.

(ii) $\mathcal{F}_\alpha$ is linear $\Rightarrow \mathbb{E}(\mathcal{F}_\alpha)$ is linear CFL.

(iii) $\mathcal{F}_\alpha$ is the class of simple recursive program schemata $\Rightarrow \mathbb{E}(\mathcal{F}_\alpha)$ is CFL.

$\underline{\mathrm{Val}}_{\mathrm{I\!I}}[P]$: value of P under a schematic interpretation $\mathrm{I\!I}$

$\underline{\mathrm{Execval}}_{\mathrm{I\!I}}[P] = x_1\tilde{q}_1 \cdots x_n\tilde{q}_n$

for $\underline{\mathrm{Val}}_{\mathrm{I\!I}}[P] = x_1 \cdots x_n$

where · $\tilde{q}_k$: a sequence consisting of 0's and 1's

· $q_{kj} = \mathrm{I\!I}(\tilde{q}_j)(x_k \cdots x_1)$

· $\tilde{q}_k = q_{k1} \cdots q_{km}$

Difinition 3

Execution event : $\tilde{\mathbb{E}}(P)$ = set of $\mathrm{Execval}_{\mathrm{I\!I}}[P]$

Theorem 4

(i) For any simple recursive program schemata P_1 and P_2

$$P_1 \cong P_2 \Longleftrightarrow \tilde{\mathbb{E}}(P_1) = \tilde{\mathbb{E}}(P_2)$$

(ii) For any simple recursive program schema P,

$\tilde{\mathbb{E}}(P)$ defines a deterministic CFL.

(iii) Termination problem for simple recursive program schemata is decidable.

Outline of Proof

(i) ⓐ $P_1 \cong P_2$ and $\mathbb{I}$ is a schematic interpretation

$\Rightarrow \mathrm{Val}_{\mathbb{I}}(P_1) = \mathrm{Val}_{\mathbb{I}}(P_2)$

and $\mathrm{Execval}_{\mathbb{I}}(P_1) = \mathrm{Execval}_{\mathbb{I}}(P_2)$

$\Rightarrow \tilde{\mathbb{E}}(P_1) = \tilde{\mathbb{E}}(P_2)$

ⓑ $\tilde{\mathbb{E}}(P_1) = \tilde{\mathbb{E}}(P_2)$ and $\mathbb{I}$ is a schematic interpretation

$\Rightarrow \mathrm{Execval}_{\mathbb{I}}[P_1] \in \tilde{\mathbb{E}}(P_2)$

$\Rightarrow \mathrm{Execval}_{\mathbb{I}}[P_1] = \mathrm{Execval}_{\mathbb{I}}[P_2]$

(ii) Show that $\tilde{\mathbb{E}}(P)$ is accepted by deterministic pushdown automata.

(iii) Follow from (ii)

The results in this talk and my IEEE paper tell us that some decision problems on program schemata are reducibld to decision problems on formal languages.