A MATHEMATICAL MODEL OF DIGITAL COMPUTERS

Z. PAWLAK

Mathematical model of Stored Program Computers (SPC) have been investigated by many authors (see references).In this note we present a new mathematical description of SPC as well as its programming language (the machine language) in such a way that both these two notions are closely related one to another.

STORED PROGRAM COMPUTERS

The definition of SPC.By SPC we shall mean the system

$$M = \langle C_M, I_M, \rho_M, \gamma_M, \lambda_M \rangle$$

or briefly

$$M = \langle C, I, \rho, \gamma, \lambda \rangle ,$$

where C,I are sets and ρ, γ, λ are functions.

The set C will be called the memory of M ;elements of C are referred to as memory states (or contents) , the set I is called set (or list) of instructions of M.

The function ρ : $I \times C \rightarrow C$, called realization of instructions,describes how each instruction changes memory states.

The function γ: $C \rightarrow I$, referred to as actual instruction selector , describes how an instruction is defined by the actual memory state.

Finaly the function λ : $C \rightarrow C$, called next instruction selector , determines next instruction (via next memory state) to be performed by the computer.

In order to describe real computers we shall specify these sets and functions so that basic properties of computers could be investigated in the model.

The memory.The memory C is defined as the set of partial functions

$$\mathcal{C} \subseteq B^{[A]}$$

satisfying the following conditions:

$$1^\circ.\ A \cap B \neq \Phi,$$
$$2^\circ.\ \bigwedge_{c \in \mathcal{C}} \overline{\overline{D}}_c < \infty$$
$$3^\circ.\ \bigwedge_{c \in \mathcal{C}} \ell \in D_c,$$

where elements of A and B are called <u>addresses</u> and <u>values</u> of C respectively, 1 is distinguished element of A called <u>instruction register or counter</u> . D_c is the domain of c and $\overline{\overline{X}}$ is the cardinality of X.

<u>Instructions</u>. Instructions of M are expressions belonging to a language called here instruction language $\mathcal{J}$. Thus first we define the language $\mathcal{J}$ and then with every computer M we associate the list of instructions I_M, which is a subset of $\mathcal{J}$ $(I_M \subset \mathcal{J})$.

Instructions are finite sequences of symbols of the following alphabet:

$$\mathcal{A} = \{A, f_1, \ldots, f_k, \alpha, \Rightarrow, (,)\},$$

where

A - is the set of addresses

$f_1, \ldots, f_k$ - are names of some two argument functions from B x B to B

$\alpha, \Rightarrow, (,)$ - are auxiliary symbols .

To define instructions , we begin with the definition of <u>terms</u> . The set of terms $\mathcal{T}$ over the alphabet $\mathcal{A}$ is defined as follows:

1°. If $t \in A$, then $t \in \mathcal{T}$,

2°. If $t \in \mathcal{T}$, then $\alpha(t) \in \mathcal{T}$,

3°. If $t, t' \in \mathcal{T}$, then $f_i(t, t') \in \mathcal{T}$ $(i = 1, \ldots, k)$

4°. Nothing else is a term

Now we can define the language of instructions in a very simple way:

$$\mathcal{J} = \{t \Rightarrow t' : t, t' \in \mathcal{T}\} \cup \{Stop\}.$$

Thus each instruction is defined by a pair of terms exept STOP instruction .

Realization of instructions. In this section we define the valuation of terms and the realization of instructions.

Valuation of terms in a memory states is a function

$$(*)\quad \vartheta : \mathcal{T} \times C \to B .$$

Instead of $(*)$ we shall write

$$(**)\quad \vartheta_c : \mathcal{T} \to B ,$$

where c is treated as a parameter and $(**)$ is defined as follows:

$$1^\circ\ \vartheta_c(t) = t ,\ \text{for } t \in A ,$$
$$2^\circ\ \vartheta_c(\alpha(t)) = c(\vartheta_c(t)) ,$$
$$3^\circ\ \vartheta_c(f_i(t,t')) = f_i(\vartheta_c(t), \vartheta_c(t')) .$$

Realization of instructions in a memory state is a function

$$\varrho : \mathcal{I} \times C \to C$$

or if we consider instructions as parameters we may write realization in the form

$$\varrho_{t \Rightarrow t'} : C \to C$$

and define it as

$$\varrho_{t \Rightarrow t'}(c) = c'$$
$$\varrho_{Stop}(c) \text{ is undefined for all } c ,$$

where

$$c'(x) = \begin{cases} \vartheta_c(t), & \text{for } x = \vartheta_c(t'), \\ c(x), & \text{for } x \neq \vartheta_c(t'). \end{cases}$$

By means of this definition we can easly classify instructions. For example, if for some $c \in C$ $\vartheta_c(t') = \ell$ then $t \Rightarrow t'$ is called jump instruction , otherwise the instruction $t \Rightarrow t'$ is operational. If in a jump instruction $t \Rightarrow t'$ value $\vartheta_c(t)$ is constant for all $c \in C$, then $t \Rightarrow t'$ is called unconditional jump , otherwise the jump instruction is conditional.

Selectors of instructions. Instructions in SPC are coded by means of elements of B. Therefore we introduce a one-to-one function

$$\varkappa : \mathcal{I} \to B$$

which to each instruction from I associates code from B. The function $\mathcal{K}$ will be called coding function of a computer .

Each memory state c defines uniquely the instruction to be performed by the computer being in the state c. This instruction is determined by the function

$$\mu : \mathcal{C} \to \mathcal{I}$$

defined as

$$\mu(c) = \mathcal{K}^{-1}(c(c(\ell)))$$

and referred to as an actual instruction selector.

Next instruction to be performed by the computer is defined by the function

$$\lambda : \mathcal{C} \to \mathcal{C} ,$$

where

$$\lambda(c) = c'$$

such that

$$c'(x) = \begin{cases} c(x), & \text{for } x \neq \ell \\ h(c), & \text{for } x = \ell, \end{cases}$$

and $h : \mathcal{C} \to \mathcal{B}$ is a function fixed for each computer. This function willbe called next instruction selector.

Control and output function. In order to describe the action of a computer it is useful to introduce a new function π : C $\to$ C called the control of M. The control π of $M = \langle C , I , \varrho, \mu, \lambda \rangle$ is defined as

$$\pi(c) = \lambda(\varrho_{\mu(c)}(c)) .$$

Thus the control perform the instruction $\mu(c)$ pointed out by the instruction register (counter) and afterwards new content of instruction register(counter) is set up by the function λ .

If we wish to describe terminated actions of the computer it is useful to introduce function ω : C $\to$ C called output function of M . This function is defined as follows:

$$\omega(c) = c'$$

iff there exist a finite sequence $c_0, c_1, \ldots, c_k$, $c_i \in C$ such that $c_0 = c$,

$c_k = c$ and $c_{i+1} = \Pi(c_i)$, $i = 0,1,\dots,k-1$, $c_k \notin D_{\Pi}$.

The notions introduced in this section form primitives for theory of computers.

MACHINE PROGRAMMING LANGUAGES

Semiprograms and programs. Programs in SPC are finite sets of instructions lebelled by addresses. Machine programming language is the set of all possible programs in M. Before the definition of program in M let us first define the notion of semiprogram in M.

Let

$$\bar{\Phi} \subseteq \bar{I}^{[A]}$$

be a set of partial functions such that

$$\bigwedge_{\varphi \in \bar{\Phi}} \bar{\bar{D}}_{\varphi} < \infty ,$$

where A, I, are the sets of addresses and instructions of M respectively. Elements of $\bar{\Phi}$ are referred to as semiprograms in M.

Every pair $\langle \varphi, a \rangle$, or briefly φ^a , where $\varphi \in \bar{\Phi}$ and $a \in D_\varphi$ we shall call program in M , and a will be called the start address of φ . The set of all programs in M will be denoted by $\hat{\bar{\Phi}}$. In other words $\hat{\bar{\Phi}}$ is machine programming language of M.

Realization of programs. Similarly to the realization of instructions we define the realization of programs. The realization of program is the function

$$\hat{\varrho} : \hat{\bar{\Phi}} \times C \to C$$

or

$$\hat{\varrho}_{\varphi a} : C \to C .$$

In order to define $\hat{\varrho}_{\varphi a}$ we introduce some auxiliary notions. Let

$$Q \subseteq \bar{\Phi} \times C$$

be a binary relation defined as follows:

$$Q(\varphi, c) \Leftrightarrow \bigwedge_{x \in D_\varphi} c(x) = \mathcal{H}(\varphi(x))$$

and

$$\hat{\mathcal{Q}} \subseteq \hat{\Phi} \times C$$

be a relation such that

$$\hat{\mathcal{Q}}(\varphi^a, c) \Leftrightarrow \mathcal{Q}(\varphi, c) \,\&\, c(c(\ell)) = a.$$

If $\mathcal{Q}(\varphi, c)$ then we say that φ is stored in c , and if $\hat{\mathcal{Q}}(\varphi^a, c)$ we say that φ^a is prepared to realization in c. Let

$$C^{\varphi} = \{c \in C : \mathcal{Q}(\varphi, c)\}$$

and

$$C^{\varphi a} = \{c \in C : \hat{\mathcal{Q}}(\varphi^a, c)\}.$$

Now we are able to give the definition of realization:

$$\hat{\delta}_{\varphi a}(c) = \omega/_{C^{\varphi a}}(c),$$

where $\omega/_X$ denotes restriction of ω to the set X.

It is easy to prove that for every SPC M and every program φ^b in M

$$\hat{\delta}_{\varphi b} \neq \omega$$

where ω is the output function of M.

<u>Some properties of programs</u>. Let

$$N \subseteq C \times A$$

be a binary relation such that

$$N(c, x) \Leftrightarrow \bigvee_{k>0} \pi_c^k(\ell) = x,$$

where π_c is to mean $\pi(c)$ and π^k is the k-fold composition of π .

We shall say that the program φ^a is <u>open</u> iff

$$\bigvee_{c \in C^{\varphi a}} \bigvee_{x \in A - D_\varphi} N(c, x),$$

otherwise the program φ^a is <u>closed</u>.

One can show that for every open program φ^a in M there exist closed program φ^b in M such that

$$\hat{\delta}_{\varphi a} = \hat{\delta}_{\varphi b}.$$

Program φ^a will be called <u>self-modifying</u> iff

$$\bigvee_{c \in C^{\varphi a}} \bigvee_{k>0} \pi^k(c) \notin C^{\varphi},$$

otherwise the program φ^a is <u>fixed</u>.

One can show that for very shelf-modifying program φ^a in M does not exist fixed program φ^b in M such that

$$\overset{\neg}{\rho}_{\varphi a} = \overset{\neg}{\rho}_{\varphi b} .$$

Thanks are due to dr.A.Mazurkiewicz for many helpful disscusions.

References

[1] Van Ba N.,Address machines and their programs,Warszawa,1972

[2] Barański H.,About computations in address machines ,Warszawa,1972

[3] Bartol W., Dynamic programs of computations,Warszawa,1973

[4] Elgot C.C.,Robinson A., Random -access stored -program machines,Journal of ACM 11(1964), 365 - 399 .

[5] Engeler E., Notes for Math.University of Minnesota .

[6] Hotz G., Grundlagen einer Theorie der Programmierungsprachen ,Universität des Saarlandes ,Saarbrücken .

[7] Kwasowiec W.,Instructions of address machines,Warszawa,1972 .

[8] Maurer W.D., A theory of computer instructions, Journal of ACM 13(1966),226-235 .

[9] Orgass R.J.,Fitch F.B., A theory of computing machines ,Studium Generale, 22 (1969)83 -104 .

[10] Pawlak Z.,On the notion of a computer ,Logic Math. and Phil.Sci., 3(1968)255-267 .

[11] Pawlak Z., Stored program machines , Algorytmy 10 (1969),5-19 .

[12] Raś Z., Algebraization of the notion of semiprogram in an address machine , Warszawa,1972

[13] Rozenberg G., Constant -program machines are universal for a class of countable machines,manuscript

[14] Wagner E.G.,Bounded action machines ,Journal of Computer and System Sciences, 2(1968),13-75 .