International Symposium and Summer School on

MATHEMATICAL FOUNDATIONS OF COMPUTER SCIENCE

Warsaw, Jablonna

August 21-27, 1972

FUNDAMENTALS FOR COMPUTER SCIENCE

by

Zdzisław Pawlak

Computation Centre of the PAS

Computation Centre of the Polish Academy of Sciences

Institute of Computing Machines of Warsaw University

1. Introduction

Design and application of digital computers have given rise to many mathematical problems. Some of them lead to the creation of mathematical disciplines such as theory of switching networks, automata theory, mathematical linguistics, complexity of algorithms, theory of programming and many others (we shall not discuss here the problems related to numerical methods).

These disciplines are in different stages of development as yet, e.g. mathematical linguistics and automata theory attained a high degree of promotion. Much effort is devoted recently to the theory of programming and complexity of algorithms but we can find only a few papers on theory of digital computers, theory of translation and translators, theory of operating systems, theory of information retrieval, etc.

It seems that the time has come to develop more systematic study of phenomena related to computers and computing. The first step towards such systematic theory might come from the investigation of primitives for theoretical computer science. Our belief is that as a root of such systematic study of computer science one should take the notion of a computer itself. This notion should be so wide that within it the definition of simple computer with instruction counter (von Neumann computer) could

be defined with better approximation to real computers as by means of finite automata or Turing machines. Having this definition one can give the definition of a program (internal program), than programming language, translator, etc., in such a way that all this notions are related to each other creating whole, systematic mathematical theory.

Why mathematical ?

Without mathematical tool we are unable as yet to give precise enough definitions of investigated objects so that we can prove something about their properties. Nowadays there is no theory without mathematics. Mathemtics has shown already its usefulness in computer science, however, it is worthwhile to mention also the difficulties in application of mathematics in this field. It seems that there are two main difficulties in mathematical investigation of computer science.

Firstly - the nature of problems in computer science is highly complex and structured in such a way that the mathematical formulation requires as a rule a great simplification in original problems so that results obtained in this way are for many computer scientists too far from the real world.

The second difficulty is due to the fact that there are not adequate mathematical techniques as yet, to deal with the questions of computer science. This means that theorems in other branches of mathematics are often useless for mathematically minded computer scientist. He has to develop his own, complete mathematical theory !

There is a very big gap between practice and theory in computer science. This gap is a real restraintin development of this field and mathematical foundations of this discipline might bridge partially this gap.

As I have said before our belief is that as basis for the mathematical foundation of computer science one should take the notion of a computer and therefore this notion will be mainly discussed in the rest of the paper.

2. The general notion of a computer

We are going to claim in this paper, that the definition of a computer should be as simple as possible (from mathematical point of view) allowing, however, to define at least very simple structures of digital computers with better approximation than finite automaton or Turing machines.

Thus any computer M is assumed to be defined by its memory structure and the control - describing how the computer acts during performance of some calculations. We shall identify in what follows the memory of the computer with some set X_M; elements of X_M are referred to as memory states. Control of M will be identified with the function $\mathcal{T}_M : X_M \to X_M$ (partial) - called also transition function of M.

Memory is meant here in a very broad sense and includes input, output, registers in the computer, etc. If we need to specify some special kinds of memories we have to structure the set X_M in an appropriate way but this problem will be not discussed here. So the computer is defined as an ordered pair $M = \langle X_M, \mathcal{T}_M \rangle$ (or briefly $M = \langle X, \mathcal{T} \rangle$).

None the less the definition is extremely simple one can ask many questions of general nature concerning computers which are of some use in the case when more specific computers are investigated (for result see Bartol 3, Kwasowiec 7, Raś 11).

The next notion basic to our consideration is that of computation of a computer.

By a computation in M we mean any sequence (finite or infinite)

$$(*) \qquad x_0, x_1, \dots, x_k, \dots, x_i \in X_M, \quad i > 0$$

such that

$$x_{i+1} = \mathcal{T}(x_i),$$

for all $i \geqslant 0$.

If the sequence $(*)$ is finite we say that the computation is <u>finite</u>. If in finite computation

$$(**) \qquad x_0, \dots, x_k$$

$x_k \notin D_{\mathcal{T}}$, where $D_{\mathcal{T}}$ is the domain of $\mathcal{T}$, we say that the computation $(**)$ is <u>terminated</u>.

Thus with every computer M we may associate the set of all computations in M. Let the set of all computations in M be denoted by C_M.

Thus the properties of computer M we can represent by some properties of the pair $\langle X, \mathcal{T} \rangle$ (algebra with one partial operation $\mathcal{T}$) or by the properties of of the set C_M. For the illustration we make some simple remarks concerning the structure of the set C_M. (For more advanced result see 3, 7, 11.)

To every computer $M = \langle X, \mathcal{T} \rangle$ we may associate transition diagram (graph) G_M defined as follows: points of the graph G_M are memory states of M. States $x, y \in X_M$ are in the relation G_M (are joint by the arrow directed from x to y in the diagram) iff $y = \mathcal{T}(x)$.

In practice we are interested with finite computers, i.e. computers with finite number of memory states, however, for some theoretical reasons it is useful to investigate infinite computers too.

By means of simple inspection of diagram of computers one can easily observe that all computers are some combinations of tree kinds of elementary computers with transition diagrams as shown in Fig. 1 a), b), c) called linear, tree, and loop computers respectively.

Of course, such rough description of computer structure may be of some use only for very simple computers like counters, shifting registers, etc., but it is not good enough for more complex computer structures.*)

If we want the computer to compute something we put the computer in some initial state and the computer starts to change its states as long as it is possible and possibly stops in some state. The resulting state "contains" in some sense results of the computations. To express this fact we introduce the notion of an output function of a computer M. The output function of M will be denoted by π^*_M and is defined as follows:

$\pi^*_M(x) = y$ iff there exists terminal computation $x_0, \ldots, x_k$ in M such that $x_0 = x$ and $x_k = y$.

The notion of a computer, computation, the set of all computations in M and the output function of M form a collection of basic notions allowing to express and investigate many facts about computers. Slight generalizations of the notion of a computer is useful in some other applications, e.g. theory of shifting registers (see Grodzki 5), formal languages, theory of programming (Blikle 4 , Mazurkiewicz 9), and theory of analog computation (see Konikowska 6).

---

*) Note that the transition diagram in the sense of our definition may be assigned to any function $f: X \to X$.

3. Digital computer

As I have said before we are going to define any computer by specifying its memory and control.

In the case of digital computer of v. Neumann type (with instruction register) we have also to specify the memory and the control.

Let us denote in that case the memory of the computer by C (abbreviation of "content"). The memory is now defined as

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

where

A - is the set of all addresses of the memory

B - is the set of all symbols we are allowed to write in memory cells

and $B^{[A]}$ denotes the set of all partial functions from A to B.

We assume that the following conditions are satisfied by the memory

- 1^0. $\bar{D}_c < \infty$, for all $c \in C$

2^0. For all $c \in C$, $1 \in D_c$, $c(1) \in A$

where 1 is some distuinguished element of A, called in what follows instruction register.

In order to define control we have first to introduce some auxiliary notions.

Let $C_b = \{ c \in C : c(c(1)) = b \}$.

Let $\pi^b = \pi / C_b$, where f/Y denotes restriction of f to the set Y.

Thus we may define the control of a digital computer with instruction register as follows :

$$(*) \qquad \Pi(c) = \begin{cases} \pi^{b_0}, \text{ if } c(c(1)) = b_0 \\ \pi^{b_1}, \text{ if } c(c(1)) = b_1 \\ \cdots \\ \pi^{b_k}, \text{ if } c(c(1)) = b_k \\ \cdots\cdots \end{cases}$$

In other words the next state of the memory is uniquely defined by the content of the instruction register.

We may also write control in some more intuitive form. Let us observe that by $(*)$ the following mapping is defined

$$\mathcal{H} : B \to \Gamma$$

where

Γ is the set of all possible restriction of Π, and $\mathcal{H}$ is called coding function. This function associates with each symbol b of B some function π^b. π^b will be called elementary instruction of M and b is referred to as code of the instruction π^b.

Thus we may write the control of digital computer as

$$\Pi(c) = \mathcal{H}[c(c(1))](c) .$$

Such definition of a computer with instruction allows to formulate more specific problems concerning digital computers (for some results see Van Ba 1, Barański 2, Kwasowiec 8, Pawlak 10, Raś 11, Skowron 12).

4. Programs

Programs (algorithms) are some description of action of a computer. Thus programs are linguistic entities (programs are to be distinguished carefully from computers even if they are performing "the same actions").

Let M be a computer with the set of addresses A and the alphabet B, and let $\Phi \subseteq B^{[A]}$ denotes the set of all partial functions satisfying the conditions :

1^0 For all $\varphi \in \Phi$, $1 \notin D_\varphi$

2^0 $\overline{\overline{D}}_\varphi < \infty$ for all $\varphi \in \Phi$.

For each φ we distinguish some address $a \in D_\varphi$, called the initial address of φ. Every pair $\langle \varphi, a \rangle$, in short φ_a, will be called program in M.

Now we can define well formed programs in M. Let us first introduce two notions necessary for this definition.

Let

$$C_{\varphi_a} = \{ c \in C : \bigvee_{A' \subset A} c/A' = \varphi \text{ and } c(1) = a \}.$$

C_{φ_a} is referred to as the set of all states "containing" program φ_a. By $N \subset C \times A$ we shall denote the <u>next instruction</u> relation in M,

$$N(c,a) \equiv \bigvee_{k>0} \pi_c^k(1) = a ,$$

where π_c is to mean $\pi(c)$ and π^k is the abbreviation of $\underbrace{\pi \ldots \pi}_{k}$.

We say that φ_a is well formed in M iff

$$\bigwedge_{a' \in D_{\varphi_a}} \bigvee_{c \in C_{\varphi_a}} N(c,a') .$$

This definition may be used for classification of programs. For example, well formed program φ_a in M is closed iff

$$R_N^{\varphi_a} \subseteq D_{\varphi_a} ,$$

where

$$R_N^{\varphi_a} = R_N / C_{\varphi_a}$$

and R_N is the range of the relation N. Otherwise the program φ_a is open. In other words, program φ_a is closed if during all possible executions of φ_a there are no references to other instructions but only from φ_a.

One can also very easy define program with modifications of its instruction and program without modifications. We say that the program φ_a is program with modifications if

$$\bigvee_{c \in C_{\varphi_a}} \bigvee_{k>0} \pi_c^k \notin C_\varphi ,$$

where

$$C_\varphi = \{ c \in C : \bigvee_{A' \subset A} c/A' = \varphi \} .$$

With every program φ_a well formed in M we may associate a function

$$\sigma_{\varphi_a} : C \to C ,$$

defined as

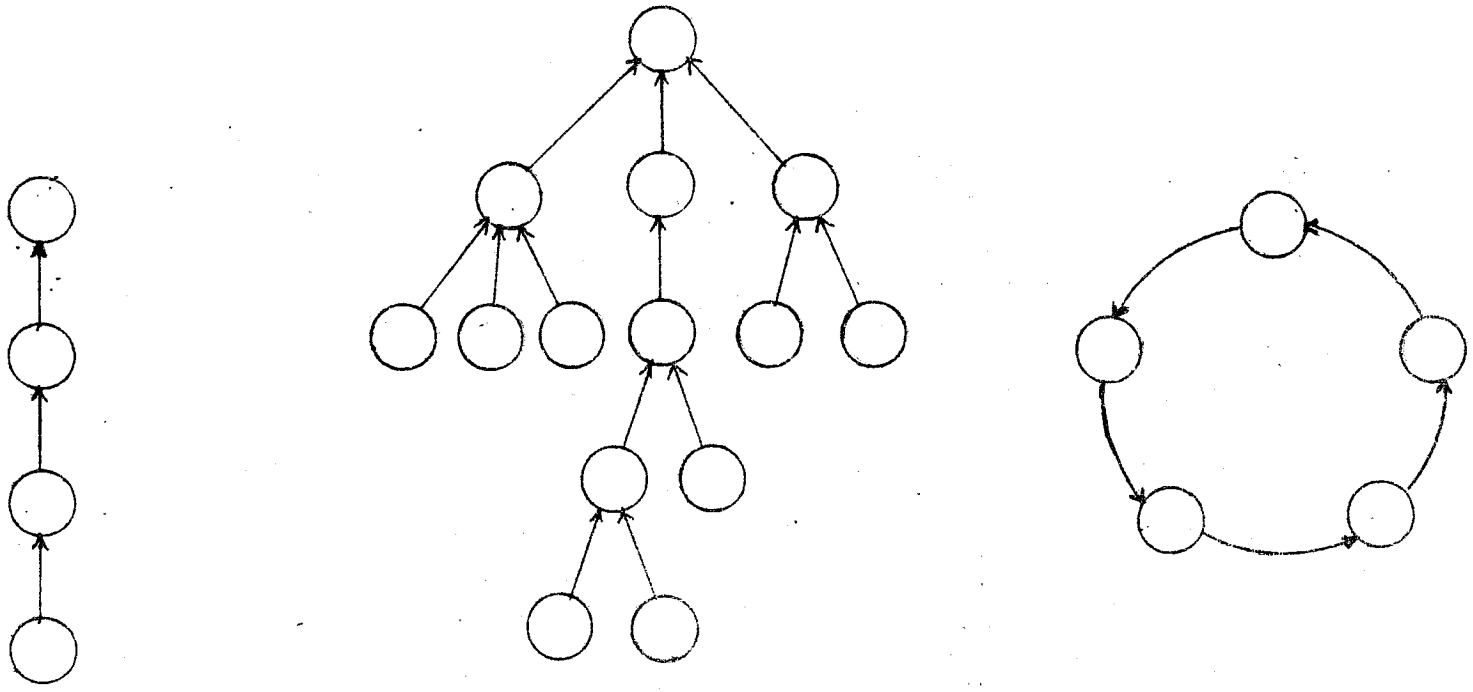
$$\sigma_{\varphi_a} = \pi_M^* / C_{\varphi_a} .$$

This function describes all terminated actions of the program φ_a, and can be considered as "output function" for the program φ_a. This output function may be regarded as a kind of semantics of program, describing all possible actions (terminal) of the machine by executing the given program.

I want to claim here that computers and programs (algorithms) are two different kinds of notions. In order to define any algorithm we have to have first in mind some computer (eventually human "computer") which is able to perform it. So computers are not algorithms ! This distinction leads to better understanding the differences and similarities between hardware and software, but this subject will be discussed in detail somewhere else.

References

1. N. van Ba, Doctoral Dissertation, Warszawa, 1971
2. H. Barański, Doctoral Dissertation, Warszawa, 1971
3. W. Bartol, This issue
4. A. Blikle, This issue
5. Z. Grodzki, This issue
6. B. Konikowska, This issue
7. W. Kwasowiec, Generable sets, Information and Control, No 17(1970), 257-264
8. W. Kwasowiec, This issue
9. A. Mazurkiewicz, This issue
10. Z. Pawlak, Programmed Machines, Algorytmy, No 10(1969), 5-19
11. Z. Raś, Doctoral dissertation
12. A. Skowron, This issue



a)

b)

c)

Fig. 1

8^8 (insert) A variable which has its defining occurrence in the labelled branch $lb \in LB_A$ is called <u>output variable of lb</u>, if after the last defining occurrence (from left to right) of this variable in lb either no its occurrence appear or there appear only its applied occurrences in decisional commands.

Note. In rows 8^6 - 8^8 should be omitted all words which correspond to the definition of output variable (which is formulated separately and inserted as a whole).

10^{15} the end of this row "$y \in Outp_A$, where" should be replaced by "$y \in Outp_{lb}$ where $lb \in LB_{AA}(A_1 \sigma_0)$ and"

25_8 there was omitted "Π_ν"; probably, only "ν" appeared in the text