BULLETIN DE L'ACADÉMIE
POLONAISE DES SCIENCES
Série des sciences techniques
Volume VIII, No. 1 — 1960

The Organization of Digital Computers and Computable Functions

by

Z. PAWLAK

Presented by P. SZULKIN on November 20, 1959

Introduction

Universal digital computers are constructed, in principle, for the purpose of numerical computation of the values of the functions $f(x_1, ..., x_n, t_1, ..., t_m)$, $m \geqslant 0$, $n > 0$ defined on rational numbers for different values of the parameters $t_1, ..., t_m$. The aim of the present paper is the application of certain elementary notions of the theory of computable functions to the organization of digital computers.

Computable functions may also be used for automatic programming of digital computers. This subject is, however, not discussed here.

General remarks on computable functions

We shall now define a few elementary notions concerning computable functions used in the following.

Computable functions may be defined as functions, the values of which can be calculated, for given arguments, by means of a finite number of operations such as, e.g. addition, subtraction, multiplication and raising to power.

The class K of computable functions is usually determined by giving:

i) the initial functions $f_1, f_2, ..., f_n$;

ii) the operations $O_1, ..., O_m$ enabling, from the functions belonging already to class K, the construction of new functions belonging to the same class.

Depending on the choice of the initial functions and the choice of operations, we obtain various classes of computable functions.

The notion of computable functions has been introduced by Gödel. Definitions of computable functions were given by S. C. Kleene, R. Peter, J. Robinson, A. M. Turing and others. All these definitions are equivalent.

The definition of Turing is based on the notion of a computer. He has shown that every computable function is also computable in the sense of his definition, that is we may build a machine for computing its values. Every Turing machine is composed of a paper tape divided into squares and a device for shifting the tape through any number of squares towards the right or the left; it also writes or cancels a finite number of symbols. Turing's machines are of great importance in mathematics, but seem to be rather useless for practical application.

Another example of a machine for computing the values of computable functions was given by A. Grzegorczyk [2]. The Grzegorczyk machine is, according to the present author's opinion, more closely connected with the computers now in use than Turing's machines. According to Grzegorczyk, a machine for computing values of functions belonging to a class K, determined by the initial functions $f_1, ..., f_n$ and the operations $O_1, ..., O_m$, must contain a device enabling the computation of the values of the initial functions $f_1, ..., f_n$, and the operations $O_1, ..., O_m$ must also be realized in some way. The conception of Grzegorczyk is the starting point of the present paper.

Operations

We shall give a few of the simplest operations *) leading from computable functions to new computable functions.

a) Operation of substitution.

If $f(x_1, ..., x_i, ..., x_n)$ and $h(y_1, ..., y_m)$ are computable functions, then the function $f(x_1, ..., h(y_1, ..., y_m), ..., x_n)$, obtained by substituting for the variable x_i in the function f, the function h is also a computable function.

b) Operation of simple recursion.

If q and h are computable functions, then the function f satisfying the conditions:

$$(1) \qquad \begin{cases} \text{a) } f(0, \bar{y}) = g(\bar{y}), \\ \text{b) } f(x+1, \bar{y}) = h(\bar{y}, x, f(x, \bar{y})), \end{cases}$$

is also computable. The variable x, in relation to which the recurrence occurs will be called the recurrence variable; all the remaining variables $\bar{y} = y_1, ..., y_n$ will be called parameters. Thus, for instance, by means of the functions $x \cdot y$, $x+1$ and the operation of simple recursion we can define the operation of raising to power y^x, as follows

$$(2) \qquad \begin{cases} \text{a) } y^0 = 1, \\ \text{b) } y^{x+1} = y^x \cdot y. \end{cases}$$

Similarly, we can define the factorial function, $x!$

$$(3) \qquad \begin{cases} \text{a) } 0! = 0+1, \\ \text{b) } (x+1)! = x!\,(x+1). \end{cases}$$

*) For accurate definitions see [1].

c) Conditional operation.

If q and h are computable functions, the function f determined by the expressions:

$$(4) \qquad f(x) = \begin{cases} g(x), & \text{if the condition } R(x) \text{ is satisfied,} \\ h(x), & \text{if the condition } R(x) \text{ is not satisfied,} \end{cases}$$

is also computable. For instance,

$$(5) \qquad Sg\,x = \begin{cases} 0 & \text{if} \quad x < 0, \\ 1 & \text{if} \quad x \geqslant 0. \end{cases}$$

In the general case the function f may be defined by means of more than two functions and an appropriate number of relations.

Computable relations

An n-term relation $R(x_1, ..., x_n)$ will be called computable relation of a class K, if there is a function f called here the characteristic function of the relation R, such that for every $x_1, ..., x_n$ (x_i is non-negative integer), we have the identity:

$$(6) \qquad R(x_1, ..., x_n) \equiv f(x_1, ..., x_n) = 0.$$

If $R(x_1, ..., x_n)$ and $S(y_1, ..., y_m)$ are computable relations of class K, there exist functions f and g such that:

$$(7) \qquad R(x_1, ..., x_n) \equiv f(x_1, ..., x_n) = 0,$$

and

$$(8) \qquad S(y_1, ..., y_m) \equiv g(y_1, ..., y_m) = 0.$$

From the elementary laws of arithmetics of non-negative integers:

$$(9) \qquad (a = 0) \cap (b = 0) \equiv a + b = 0,$$

$$(10) \qquad \sim (a = 0) \equiv 1 \dot{-} a = 0 \text{ *)},$$

where $\cap$, $\sim$ denote the logical product and negation, respectively, we can derive the relations:

$$(11) \qquad R(x_1, ..., x_n) \equiv 1 \dot{-} f(x_1, ..., x_n) = 0,$$

$$(12) \qquad R(x_1, ..., x_n) \cap S(y_1, ..., y_m) \equiv f(x_1, ..., x_n) + g(y_1, ..., y_m) = 0.$$

If K is a class closed under the operations of substitution and includes the functions $x+1$, $x+y$, $x \dot{-} y$, then the set of computable relations of the class K is closed under the operation of propositional calculus.

For instance, to the relation $x < y$ corresponds the characteristic function $x \dot{-} (y \dot{-} 1)$, and to the relation $x \leqslant y$ — the function $(x \dot{-} y)$.

*) The function $x \dot{-} y$ is defined thus:

$$x \dot{-} y = \begin{cases} 0 & \text{if} \quad x \leqslant y, \\ x - y & \text{if} \quad x > y. \end{cases}$$

Universal functions

If K is a class of functions, then K_n is the class of n-argument functions belonging to K.

The function $f(x_1, ..., x_n, t)$ of $n+1$ arguments is a universal function of the class K_n, if for every function $g(x_1, ..., x_n)$ of n arguments, function g belongs to K_n, if and only if there exists a number t such that for every $x_1, ..., x_n$,

$$g(x_1, ..., x_n) = f(x_1, ..., x_n, t). \tag{13}$$

Organization of a computer for calculating the functions of a class S

Adresses and addressing function

The memory of the computer may be considered to be a lined sheet of paper. All the squares thus obtained are numbered with successive integers. The numbers of the squares will be called addresses. In the computation process all the initial data have definite adresses (are located in given squares). All the partial results are also located in given squares.

In the present section we shall define functions F and G determining the addresses of partial results depending on the addresses of the arguments; these functions will be called addressing functions.

Formulae and programs

The class S is defined by n initial functions $f_1, ..., f_n$ of two arguments and the operation of substitution. To make the subject more concrete, let us assume four initial functions $x+y$, $x-y$, $x \cdot y$, x/y.

Thus, for instance $(p \cdot q) - (s \cdot r)/(t-u)$ is a function of the S class.

Let F be a class of formulae with three kinds of elementary symbols

i) variables $p, q, r, s, t, u, v, w, x, y, z$,

ii) symbols of dyadic operators,

iii) parentheses "(", ")".

The arbitrary variable will be denoted by X, and the arbitrary operation by Δ. The notion of formula will be defined by recursion scheme:

a) the variable is a formula,

b) $\alpha, \beta \in F \supset (\alpha \Delta \beta) \in F$.

Thus, for instance $(((p \cdot q) - (s \cdot r))/(t-u))$ is a formula.

A formula containing at least one pair of parentheses will be called a program.

Components

A component $\sigma(\Phi)$ of the formula Φ will be defined as follows:

a) a formula is a component,

b) if $(\alpha \Delta \beta)$ is a component of a formula Φ, α and β are also components of the formula Φ.

Thus, for instance, the expressions $p, t, (p \cdot q), (((p \cdot q) - (s \cdot r))/(t-u))$ are components of the formula $(((p \cdot q) - (s \cdot r))/(t-u))$.

It is obvious that if a formula has n variables, it has $2n-1$ components.

The order of a component

A number $\langle \sigma(\Phi) \rangle$, called the order of the component, is associated with every component $\sigma(\Phi)$. The order is defined thus

a) if $\sigma(\Phi)$ has a form identical with Φ, then $\langle \alpha(\Phi) \rangle = 1$,

b) if $\langle \sigma(\Phi) \rangle = i$ and if $\sigma(\Phi)$ has the form $(\alpha \Delta \beta)$, then $\langle \alpha \rangle = \langle \beta \rangle = i+1$.

For instance, the component $(p \cdot q)$ of the formula $(((p \cdot q) - (r \cdot s))/(t-u))$ is of the order $\langle (p \cdot q) \rangle = 4$, and the component $((p \cdot q) - (s \cdot r))$ is of the order $\langle ((p \cdot q) - (s \cdot r)) \rangle = 3$.

Adressing of the components

Let $\{\sigma(\Phi)\}$ be the number of a component $\sigma(\Phi)$. The number $\{\sigma(\Phi)\}$ will be called the address of the component $\sigma(\Phi)$. In the following we shall consider the two simplest addressing functions, which we shall use further in this paper.

We shall assume that the addresses have the following properties:

$$\begin{cases} \text{a) } \{\alpha\} = i \text{ and } (\alpha \Delta \beta) \supset \{\beta\} = i+1, \\ \text{b) let } \sigma_1(\Phi) \text{ and } \sigma_2(\Phi) \text{ be components of the formula } \Phi \\ \qquad \langle \sigma_1(\Phi) \rangle > \langle \sigma_2(\Phi) \rangle \supset \{\sigma_1(\Phi)\} > \{\sigma_2(\Phi)\}; \\ \text{c) } \langle \sigma(\Phi) \rangle = 1 \supset \{\sigma(\Phi)\} = 1; \\ \text{d) } \sigma_1(\Phi) \neq \sigma_2(\Phi) \supset \{\sigma_1(\Phi)\} \neq \{\sigma_2(\Phi)\}. \end{cases} \tag{14}$$

Binary addressing

$$\begin{cases} \text{a) } \langle \sigma(\Phi) \rangle = 1 \supset \{\sigma(\Phi)\} = 1; \\ \text{b) } \{(\alpha \Delta \beta)\} = i \supset \{\alpha\} = 2i \text{ and } \{\beta\} = 2i+1. \end{cases} \tag{15}$$

(This way of addressing may very easily be realized in the binary system) *).

The above way of addressing may be made dependent on the structure of the formula Φ.

*) The problem of addressing the components, in general form, is as follows: there are given the addresses of the variables of the formulae Φ, to find the addresses of all the components $\sigma(\Phi)$. The most general form of the component addressing function are pairing functions. A function $F(x, y)$ is a pairing function, if there exist functions G and H such that $G(F(x, y)) = x$, and $H(F(x, y)) = y$. The problem of addressing components by means of pairing functions will not be considered in this paper.

Let $\varrho_i(\Phi)$ denote the i-th symbol of the formula Φ (counting from left to right), and let $F(\varrho_i(\Phi))$ denote a function, such that to every symbol $\varrho_i(\Phi)$ corresponds a natural number n.

The function F will be defined by recursion scheme

$$(16)\quad \begin{cases} \text{a) } F(\varrho_1(\Phi)) = 1; \\ \text{b) } F(\varrho_{i+1}(\Phi)) = \begin{cases} F(\varrho_i(\Phi)), \text{ if } \varrho_{i+1}(\Phi) = \Delta; \\ E\left(\dfrac{F(\varrho_i(\Phi))}{2}\right), \text{ if } \varrho_{i+1}(\Phi) = (; \\ 2F(\varrho_i(\Phi)), \text{ if } \varrho_{i+1}(\Phi) = X \text{ or } (\text{ and } \varrho_i(\Phi) = (; \\ F(\varrho_i(\Phi)) + 1, \text{ if } \varrho_{i+1}(\Phi) = X \text{ or } (\text{ and } \varrho_i(\Phi) = \Delta. \end{cases} \end{cases}$$

$E(x)$ denotes the integer part of x.

The values for the function $F(\varrho_i(\Phi))$ for the corresponding parentheses or variables are the addresses of the components of the formula, according to the binary addressing method.

Natural addressing

Let $H(\varrho_i(\Phi))$ be a function defined by recursion scheme

$$(17)\quad \begin{cases} \text{a) } H(\varrho_1(\Phi)) = 1, \\ \text{b) } H(\varrho_{i+1}(\Phi)) = \begin{cases} H(\varrho_i(\Phi)), \text{ if } \varrho_{i+1}(\Phi) = X \text{ or } (\text{ and } \varrho_i(\Phi) = \Delta, \\ \qquad \text{or } \varrho_{i+1}(\Phi) = \Delta \text{ and } \varrho_i(\Phi) = X \text{ or }); \\ H(\varrho_i(\Phi)) + 1, \text{ if } \varrho_{i+1}(\Phi) = (\text{ or } X \text{ and } \varrho_i(\Phi) = (; \\ H(\varrho_i(\Phi)) - 1 \text{ if } \varrho_i(\Phi) =). \end{cases} \end{cases}$$

The values of the function H will be called the order of the symbol ϱ_i.

Let $\varphi_k^p(\Phi)$ denote the k-th symbol of the p-th order of the formula and let $G(\varphi_k^p(\Phi))$ be the addressing function defined by recursion

$$(18)\quad \begin{cases} \text{a) } G(\varphi_1^1(\Phi)) = 1; \\ \text{b) } G(\varphi_{k+1}^p(\Phi)) = \begin{cases} G(\varphi_k^p(\Phi)), \text{ if } \varphi_{k+1}^p(\Phi) =) \text{ or } \Delta; \\ G(\varphi_k^p(\Phi)) + 1, \text{ if } \varphi_{k+1}^p(\Phi) = (\text{ or } X; \end{cases} \\ \text{c) } G(\varphi_1^p(\Phi)) = G(\varphi_{\max}^{p-1}(\Phi)) + 1. \end{cases}$$

$\varphi_{\max}^{p-1}$ denotes the symbol of the $p-1$ order for which $H(\varphi_i^{p-1}(\Phi))$ is maximum. Examples of addressing are given in Tables I and II.

TABLE I

Φ	(	(	(	p	·	q	)	—	(	r	·	s	)	)	/	(	t	—	u	)	)
$\varrho_i(\Phi)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$F(\varrho_i(\Phi))$	1	2	4	8	8	9	4	4	5	10	10	11	5	2	2	3	6	6	7	3	1

TABLE II

Φ	(	(	p	·	q	)	—	(	r	·	(	s	+	t	)	)	)
$\varrho_i(\Phi)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$H(\varrho_i(\Phi))$	1	2	3	3	3	2	2	2	3	3	3	4	4	4	3	2	1
$G(\varphi_k^p(\Phi))$	1	2	4	4	5	2	2	3	6	6	7	8	8	9	7	3	1

Addressing of the components of the formula and numeration of the branches of trees

It may be shown that to every formula corresponds an oriented tree. Then the problem of numeration of the formula corresponds to that of numeration of the branches of that tree.

Thus, for instance, to the formula $(((p \cdot q) - (r \cdot s))/(t - u))$ corresponds the tree:

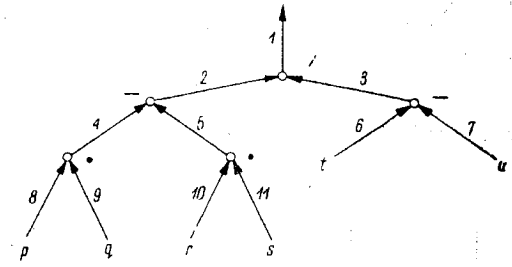

Fig. 1

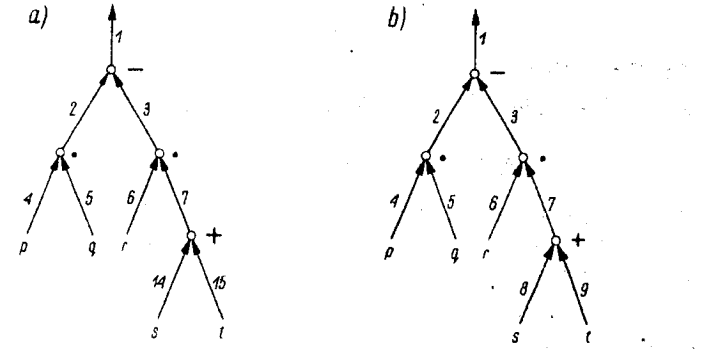

Fig. 2

Binary and natural addressing is, in the given example, identical. To the formula $(((p \cdot q) - (r \cdot s))/(t - u))$ corresponds a tree with branches numbered in the binary manner, shown if Fig. 2a. In natural numeration the

branches have numbers as shown in Fig. 2b. The introduction of the trees brings nothing new and is only an additional illustration of the principles of numeration of components.

Ordering of arguments

From the principles of addressing discussed it follows that all the pairs of arguments, on which the operations are performed, may be addressed, and the address of the result is a function of the address of the arguments. (This fact enables us to organize the computer in another way than it is actually done).

Thus, for instance for the formula $(((p \cdot q) - (r \cdot s))/(t - u))$ we have

TABLE III

10	p	q	11	$\cdot$
8	r	s	9	$\cdot$
6	t	u	7	$-$
4	$(p \cdot q)$	$(r \cdot s)$	5	$-$
2	$((p \cdot q) - (r \cdot s))$	$(t - u)$	3	$/$
		$(((p \cdot q) - (r \cdot s))/(t - u))$	1	

E.g., for the formula $((p - q) - (r \cdot (s + t)))$ we have Table IV.

TABLE IV

8	s	t	9	$+$
6	r	$s + t$	7	$\cdot$
4	p	q	5	$\cdot$
2	$(p \cdot q)$	$(r \cdot (s + t))$	3	$-$
		$((p \cdot q) - (r \cdot (s + t)))$	1	

On the both sides of the Tables III and IV the addresses of the corresponding components and the symbols of operations are given.

Addressless organization

Let Ms denote a machine for calculating functions of class S. The machine Ms is composed of a memory, which may be imagined as a paper tape divided into numbered *) rows. Every row consists of four squares numbered 1, 2, 3, 4. In the squares 1, 2 arguments are located, in 3 — the symbol of operation, in 4 — the number of the row.

If i is the row number, then $2i$ is the number of the first square, $2i + 1$ is the number of the second square. The squares 3 and 4 are not numbered.

*) The number of the row will be called its address.

In addition, the machine has an arithmometer, constituting a realization of the universal function of the initial functions of class S.

The machine has also a device for computing the values of the addressing function F (or G), which computes the address of the result, depending on the row number.

The machine shifts the rows, beginning with the row with the greatest address, address 1 being the last. After shifting the entire tape the computation is finished.

A simplified scheme of such a machine is presented in Fig. 3.

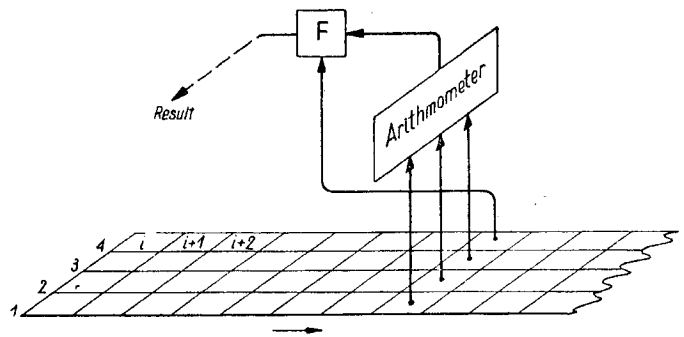

Fig. 3

The organization is called addressless because the order has no address. The order is composed of the symbol of operation and arguments, on which operation is carried out:

operation	argument a	argument b

This organization allows also, in a simple way, an automatic pogramming.

This organization applied to a series machine permits besides automatic programming an increase of the speed of computing by several tens of times and eliminates completely the waiting time. By applying this organization in the digital computer, constructed at the Warsaw Technical University, we should obtain about 9000 operations per second (as compared to 100 operations per second at present). The cost of construction would increase 2—3 times (see, e.g. [4]).

Two-argument organization

If we give up automatic programming, we can omit the device computing the functions F (or G) (the remainder of the machine being unchanged),

and introduce into the order the address of the result. The order will then have the form:

address of result	operation	argument a	argument b

This conception was accepted as starting point in the doctoral thesis of Jaworski, [3].

Organization of a machine for calculating functions of class R

Class R

The class R will be defined by the following initial functions:

a) $x+y$, $x-y$, $x \cdot y$, x/y, $x \dot{-} y$ two-argument arithmetic functions;
b) $N(x)$, $P(x)$, $E(x)$, $Sg(x)$ one-argument arithmetic functions, which will be read: successor x, preceder x, integer part of x (or overflow x) signum x;
c) The characteristic functions of the relations $x<y$, $x \geqslant y$, $x=y$, $x \neq y$. The characteristic functions will be denoted by $(x<y)$, $(x \geqslant y)$, $(x=y)$, $(x \neq y)$, respectively,

as well as by the operations of substitution, simple recursion and the conditional operation.

Formalized language

The construction of a formalized language in which we could write down functions of class R presents no difficulty. However, bearing in mind technical reasons, the realization of a formalized language requires a more detailed investigation. It will be the subject of a subsequent paper.

Conclusion

It would be interesting to investigate the usefullness — from the technical point of view — of the extension of class R by the introduction of the operation of effective minimum. However, this problem has not been dealt with by the author.

In a simple way we can give also addressing functions for the parentheseless notation of Łukasiewicz, however, this system of notation seems to be less convenient in routine use. By introducing the known rules of omitting superfluous parentheses, the notation of the function might be simplified considerably.

We should then incorporate in the computer a device (or a program) computing addressing functions from a formula not containing all the parentheses.

The problem of independence of the program of the location in the memory has not been considered in the present paper. It seems that in the system of addressing just presented it should involve no difficulties.

The results of the present paper were communicated at a meeting of the group of Mathematic Apparatus at the Institute of Mathematics in 1954.

INSTITUTE OF MATHEMATICS, POLISH ACADEMY OF SCIENCES
(INSTYTUT MATEMATYCZNY, PAN)
DEPARTMENT OF TELE- AND RADIOPHONIC CONSTRUCTIONS, WARSAW TECHNICAL UNIVERSITY
(ZAKŁAD KONSTRUKCJI TELE- I RADIOFONII, POLITECHNIKA WARSZAWSKA)

REFERENCES

[1] A. Grzegorczyk, *Some classes of recursive functions*, Rozpr. Matem., 1953.

[2] — *On the computable functions and possibilities of their application to mathematical machines* (lecture held at the Institute of Mathematics, in 1957).

[3] Z. Pawlak, *Teoria programowania bezadresowego cyfrowej maszyny automatycznie liczącej* (typescript pp. 47), 1953.

[4] — *An electronic digital computer based on the "—2" system*, Bull. Acad. Polon. Sci., Sér. sci. techn., 7 (1959), 713.

[5] W. Jaworski, *Programowana maszyna cyfrowa z nową budową instrukcji*, Arch. Automat. i Telemech., 3 (1958), No. 3.