NEW CLASS OF MATHEMATICAL LANGUAGES AND ORGANIZATION OF ADDRESSLESS COMPUTERS

by

ZDZISŁAW PAWLAK

Warsaw

Introduction

This paper sketches some ideas and results concerning the theory of mathematical machines, obtained by the author during the period 1952—1961. A certain type of organization of digital computers, working directly in given mathematical language, is studied in this paper. The study leads to a new class of mathematical languages for functions. Some of them are reported here.

Processes

A process $\mathfrak{A} = \langle A, O, R \rangle$ is a finite set A of objects, a finite set O of operations defined over the set A, and the ordering relation R defined in O.

If the set O of operations is well ordered by the relation R, we say that $\mathfrak{A}$ is a *sequential process*; if the set O is partially ordered by the relation R we say that $\mathfrak{A}$ is a *concurrent process*.

In the following, elements of A are denoted by small Greek letters and elements of O — by capital Greek Letters.

Process $\mathfrak{A}$ is called *simple*, if and only if:

1. For each $\Delta \in O$, there are three elements $L(\Delta)$, $R(\Delta)$, $W(\Delta)$ called *left argument* of the operation Δ, *right argument* of the operation Δ, and *result* or *output* of the operation Δ respectively.

2. For each $\alpha \in A$, there is at least one operation $\Delta \in O$, so that $\alpha = L(\Delta)$ or $\alpha = R(\Delta)$ or $\alpha = W(\Delta)$.

3. There exist exactly one $\alpha \in A$ such, that for no $\Delta \in O$ neither $\alpha = L(\Delta)$ nor $\alpha = R(\Delta)$ holds, α being called *final result* or *output* of the process $\mathfrak{A}$. The final result of the process $\mathfrak{A}$ will be denoted by "$\mathfrak{A}$".

If $\alpha \in A$ and there is no $\Delta \in O$ such that $\alpha = W(\Delta)$, then α is the *initial data* of the process $\mathfrak{A}$.

If $\alpha \in A$ and there are $\Delta, \Omega \in O$ such that $\alpha = W(\Omega)$ and $\alpha = L(\Delta)$ or $\alpha = R(\Delta)$, then α is called a *partial result* of the process $\mathfrak{A}$.

Examples of simple processes

If A is the set of natural numbers and the set O is the set of arithmetical operations such as addition, substraction, multiplication and division, then

$\mathfrak{A}$ is a process of arithmetical computation and "$\mathfrak{A}$" is the final result of the computation.

If A is the set of truth-values and O is the set of logical operations as negation, conjunction and disjunction, then $\mathfrak{A}$ is a process of logical computation and "$\mathfrak{A}$" is the logical value of the process $\mathfrak{A}$.

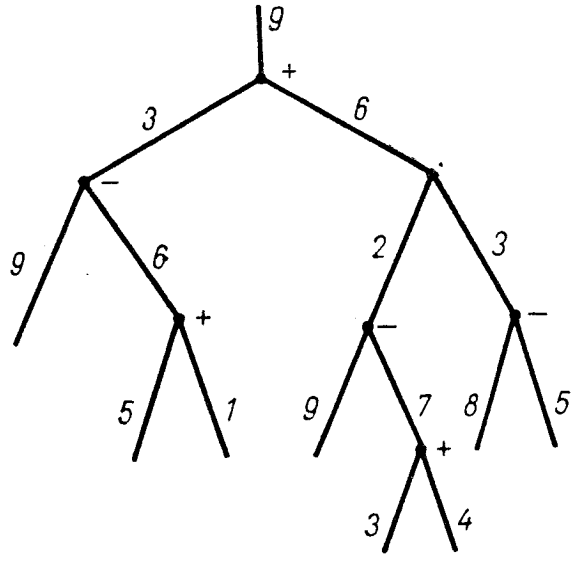


FIG. 1

If A is the set of matrices and O is the set of matrix operations, then $\mathfrak{A}$ is a process of matrix computation.

If A is the set of axioms and of theorems, belonging to a given formal theory, and O is the set of rules of inference in this theory, then $\mathfrak{A}$ is a proof of "$\mathfrak{A}$" in this theory.

The set A may consist of physical objects, O being the set of rules of composition of objects belonging to A; in this case $\mathfrak{A}$ is a process of production.

In order to make the notion of process clearer, let us remark that the definition of the simple process resembles the definition of a tree in the theory of graphs. Operations are interpreted as nodes, and objects as branches of the tree. But, in the definition of the tree there are no ordering relations over the set of nodes, thus the process and the tree are two quite different notions. However, for the sake of simplicity, we shall often use the tree to represent a simple process as shown below.

The figure is obvious and requires no comments.

In the same way we may represent the process of proving theorems and other mathematical processes, but in order to make the subject more concrete, we shall consider arithmetical processes only in the following.

Orders of simple processes

In what follows we shall limit ourselves to simple sequential processes only.

If in the processes $\mathfrak{A} = \langle A, O, R \rangle$ the set O is well ordered by the relation R, we say that the process $\mathfrak{A}$ is ordered by the relation R or, in short, that the process has the order R. There are four important orders

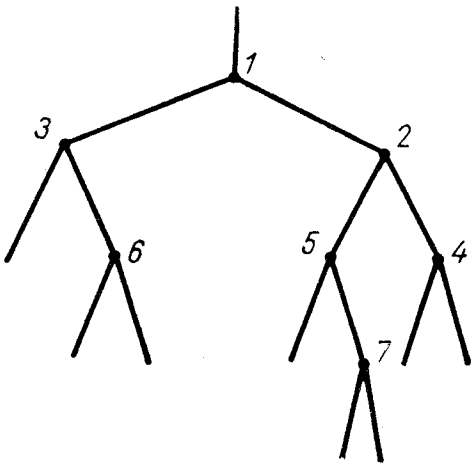


FIG 2. Normal cross order P

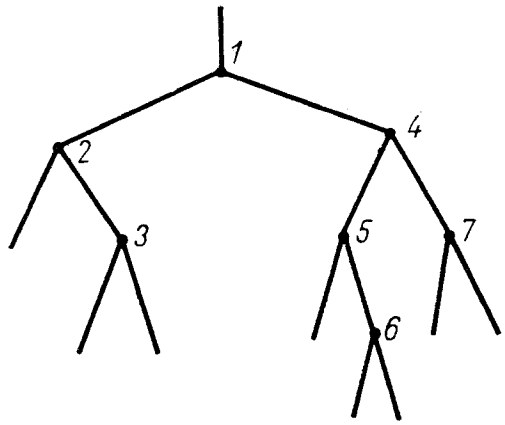


FIG. 3. Normal along order W

in simple sequential processes, namely P, W, $\bar{P}$, $\overline{W}$, called *normal cross* order, *normal along* order, *dual cross* order and *dual along* order respectively. We assume that with each operation $\Delta \in \mathfrak{A}$ there is a natural number $N(\Delta) \neq 0$ associated such that, if Δ, $\Omega \in \mathfrak{A}$ and $\Delta \succ \Omega$, then $N(\Delta) > N(\Omega)$, where $\Delta \succ \Omega$ means that the operation Δ is to be performed before the operation Ω, or the operation Ω is to be performed after the operation Δ. Thus the definition of the order of a process may be regarded as a numeration of the nodes in the corresponding tree. Instead of an exact definition, which would be too long for our purposes, we shall show all four orders by means of examples.

With reference to the previously given example of an arithmetical computation, the operations may be ordered as follows:

Order P	Order W	Order $\bar{P}$	Order $\overline{W}$
$3 + 4 = 7$	$8 - 5 = 3$	$3 + 4 = 7$	$5 + 1 = 6$
$5 + 1 = 6$	$3 + 4 = 7$	$8 - 5 = 3$	$9 - 6 = 3$
$9 - 7 = 2$	$9 - 7 = 2$	$9 - 7 = 2$	$3 + 4 = 7$
$8 - 5 = 3$	$2 \cdot 3 = 6$	$5 + 1 = 6$	$9 - 7 = 2$
$9 - 6 = 3$	$5 + 1 = 6$	$2 \cdot 3 = 6$	$8 - 5 = 3$
$2 \cdot 3 = 6$	$9 - 6 = 2$	$9 - 6 = 3$	$2 \cdot 3 = 6$
$3 + 6 = 9$	$3 + 6 = 9$	$3 + 6 = 9$	$3 + 6 = 9$

Programs of simple processes

Let N be the set of names of objects belonging to the process $\mathfrak{A}$ and let C be the set of names of operations belonging to the process $\mathfrak{A}$. The program of process $\mathfrak{A}$ is any sequence of elements of N and C describing unambiguously process $\mathfrak{A}$. The program process $\mathfrak{A}$ will be denoted by $\mathfrak{A}'$. In other words, program $\mathfrak{A}'$ of process $\mathfrak{A}$ is the linear representation of the corresponding tree. There are many ways of representing the tree in linear form, some of which will be treated here.

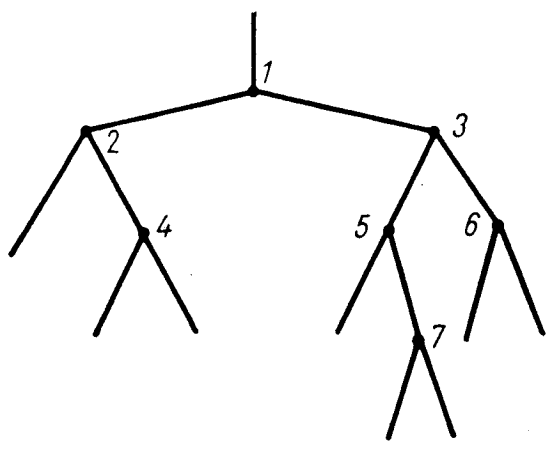


FIG. 4. Dual cross order $\overline{P}$

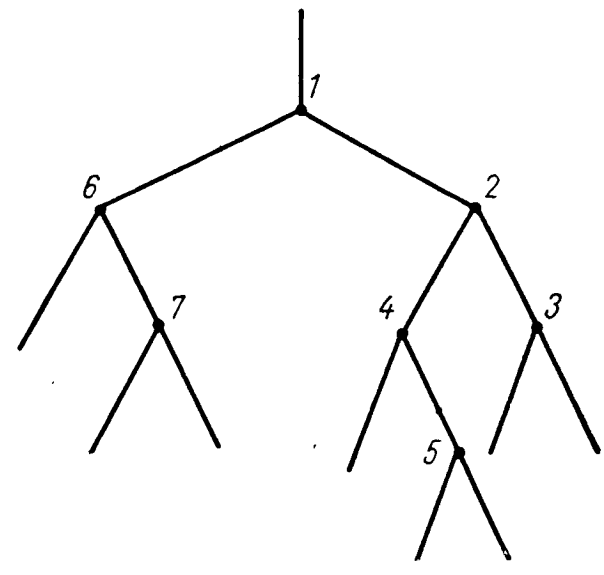


FIG. 5. Dual along order $\overline{W}$

Basis notation of a program. (Languages L_1 and $\bar{L}_1$.) The sequence

$$a_{2n+1}\, a_{2n}\, D_n\, a_{2n-1}\, a_{2n-2}\, D_{n-1} \ldots a_3\, a_2\, D_1\, a_1$$

is a program of process $\mathfrak{A}$ in language L_1 if, for each i $(1 \leqslant i \leqslant n)$, a_{2i+1}, a_{2i} are names of left and right arguments of operation D_i, and $D_{i+1} \succ_N D_i$, where $\succ_N$ denotes the ordering relation according to the normal order, P or W. The sequence

$$a_1\, D_1\, a_2\, a_3 \ldots D_{n-1}\, a_{2n-2}\, a_{2n-1}\, D_n\, a_{2n}\, a_{2n+1}$$

is a program of process $\mathfrak{A}$ in language $\bar{L}_1$ if, for each i $(1 \leqslant i \leqslant n)$, a_{2i}, a_{2i+1} are names of left and right arguments of operation D_i, and $D_i \prec_D D_{i+1}$, where $\prec_D$ denotes the ordering relation according to the dual order, $\overline{P}$ or $\overline{W}$.

We can easily prove the following theorems.

THEOREM. 1 If $\mathfrak{A}'$ is a program of process $\mathfrak{A}$ in L_1 with order P, then the ith symbol of a partial result in program $\mathfrak{A}'$ (counting from right to left) denotes the result of the ith operation D_i.

THEOREM 2. Let σ denote any symbol of program $\mathfrak{A}'$ in language L_1 with order W. Let σ' denote the next symbol in program $\mathfrak{A}'$ and let $F_i(\sigma)$ be the function which attaches to the symbol of operation D_i and further symbols in the program a natural number according to the rule

1. $F_i(D_i) = 1$.
2. $F_i(\sigma') = F_i(\sigma)$, if σ is symbol of data.
3. $F_i(\sigma') = F_i(\sigma) + 1$, if σ symbol of operation.
4. $F_i(\sigma') = F_i(\sigma) - 1$, if σ is symbol of partial result.

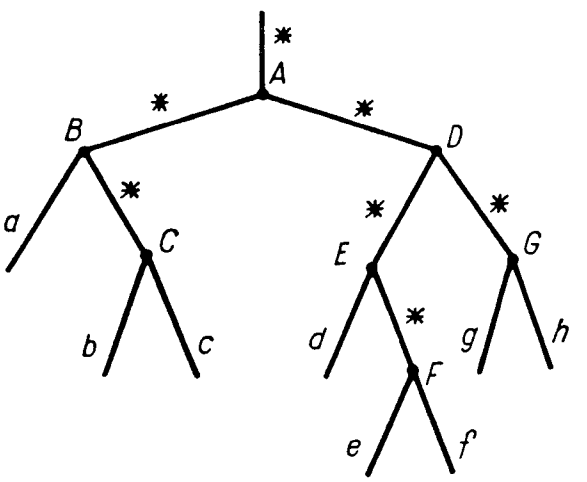


FIG. 6

If $\mathfrak{A}'$ is a program in L_1 of process $\mathfrak{A}$ with order W, then the partial result of operation D_i is denoted in the program by the symbol σ for which $F_i(\sigma) = 0$.

Similar theorems hold for language $\bar{L}_1$.

Some examples will show the application of Theorem 1 and Theorem 2. Let us consider a process as shown below.

Operations are denoted by capital Latin letters, data are denoted by small Latin letters and partial results by asterisk *. Thus programs for different orders in languages L_1 and $\bar{L}_1$ are as follows.

Language L_1:

$$efFbcCd*EghGa*B**D**A* \qquad \text{order } P,$$

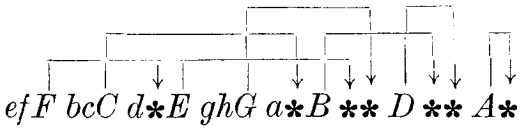
$$ghGefFd*\ E**DbcCa*B**A* \qquad \text{order } W.$$

Language $\bar{L}_1$:

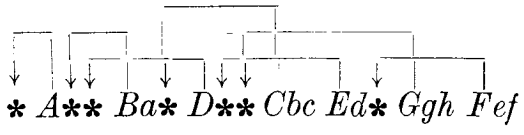
$*A**Ba*D**CbcEd*GghFef$ order $\bar{P}$,

$*A**D**GghEd*FefBa*Cbc$ order $\overline{\bar{W}}$.

According to Theorem 1, symbols of partial results for each operation, by order P and $\bar{P}$, are as shown by arrows below.



order P

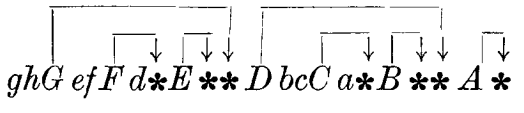


order $\bar{P}$

From theorem 2 for order W we obtain

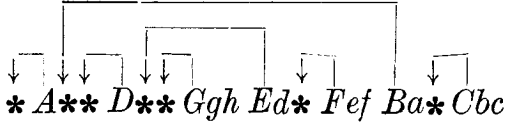
	ghG	efF	d*E	**D	bcC	a*B	**A	*
F_7	1	112	212	10				
F_6		1	10					
F_5			1	0				
F_4				1	112	212	10	
F_3					1	10		
F_2						1	0	
F_1							1	0

Indicating symbols of partial results by arrows, we have



order W.

Proceeding in the similar way for order $\overline{\bar{W}}$, we obtain



order $\overline{\bar{W}}$.

Parenthesis free notation of a program. (Languages L_2 and $\bar{L}_2$.)

In L_2 and $\bar{L}_2$ operations and corresponding partial results are denoted by the same symbols as for example in Fig. 7.

The sequence

$$a_{2n+1}\, a_{2n}\, a_{2n-1}\, a_{2n-2} \ldots a_3\, a_2\, a_1$$

is a program in language L_2 if, for each i $(1 \leq i \leq n)$, a_{2i+1}, a_{2i} are names of left and right arguments of the operation with number i according to order P or W.

The sequence

$$a_1\, a_2\, a_3 \ldots a_{2n-2}\, a_{2n-1}\, a_{2n}\, a_{2n+1}$$

is a program in language $\bar{L}_2$ if, for each i $(1 \leq i \leq n)$, a_{2i}, a_{2i+1} are names of the left and right arguments of the operation with number i according to order $\bar{P}$ or $\overline{W}$.

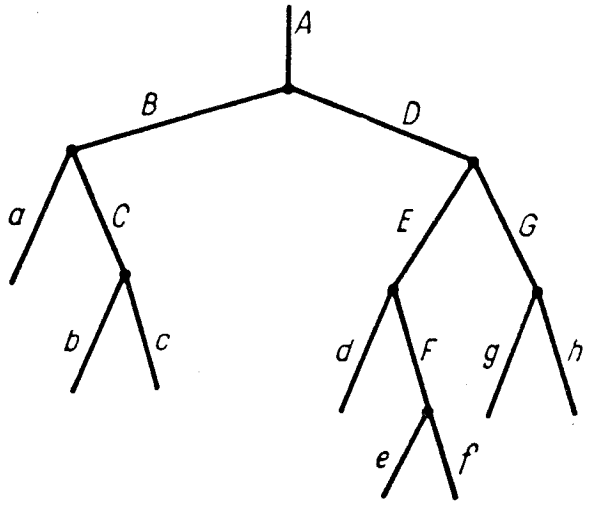


FIG. 7

Theorems 1 and 2 after suitable modification are also valid in languages L_2 and $\bar{L}_2$. Examples of programs in L_2 and $\bar{L}_2$ for different orders have the following form:

Language L_2:

efbcdFghaCEGBDA	order P,
ghefdFEGbcaCBDA	order W.

Language $\bar{L}_2$:

ABDaCEGbcdFghef	order $\bar{P}$,
ABDEGghdFefaCbc	order $\overline{W}$.

In each language finding the proper symbol of operation for each pair of arguments presents no difficulty.

Łukasiewicz notation of a program. (Languages L_3 and $\bar{L}_3$.)

Let us establish numeration of objects in the simple process in order W and $\overline{W}$.

The sequence

$$a_1\, a_2\, a_3 \ldots a_{2n-2}\, a_{2n-1}\, a_{2n}\, a_{2n+1} \qquad \text{order } W$$

is a program in language L_3 if a_i is the name of objects of process $\mathfrak{A}$ with number i by numeration with order $\overline{W}$.

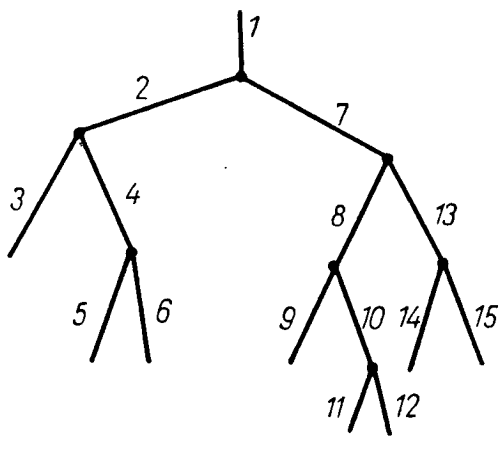


FIG. 8. Order W

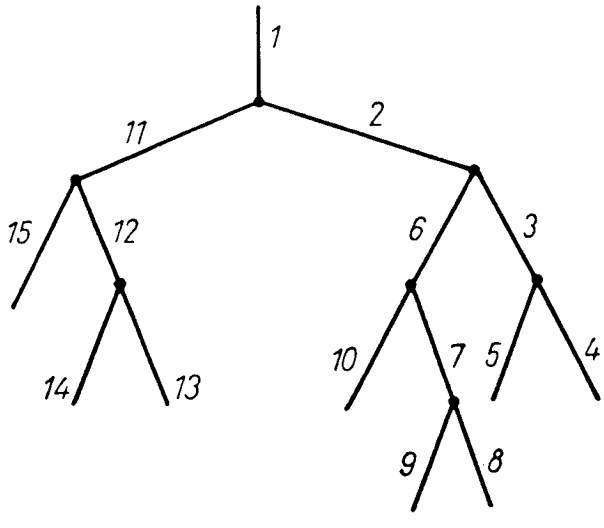


FIG. 9. Order $\overline{W}$

The sequence

$$a_{2n+1}\, a_{2n}\, a_{2n-1}\, a_{2n-2} \ldots a_3\, a_2\, a_1$$

is a propram in language $\bar{L}_3$ if a_i is the name of objects of process with number i, by numeration with order $\overline{W}$.

Programs of the process given above in languages L_3 and $\bar{L}_3$ are

Language L_3:

$$ABaCbcDEdFefGgh \qquad \text{order } W.$$

Language $\bar{L}_3$:

$$abcCBdefFEgkGDA \qquad \text{order } \overline{W}.$$

Parenthesis notation of a program. (Language L_4.)

The well known parenthesis notation may also be assumed to be the program of a simple process. Relation between the tree and parenthesis formula follows from Fig. 10.

Symbols along the dotted line arranged in a string form a parenthesis program in L_4. Thus the program of the above process in L_4 is

$$((aB(bCc))A((d\ E(eFf))D(gGh))).$$

Let us define function $G(\sigma)$ which number all operations of program $\mathfrak{A}'$ in L_4 according to order $\bar{P}$.

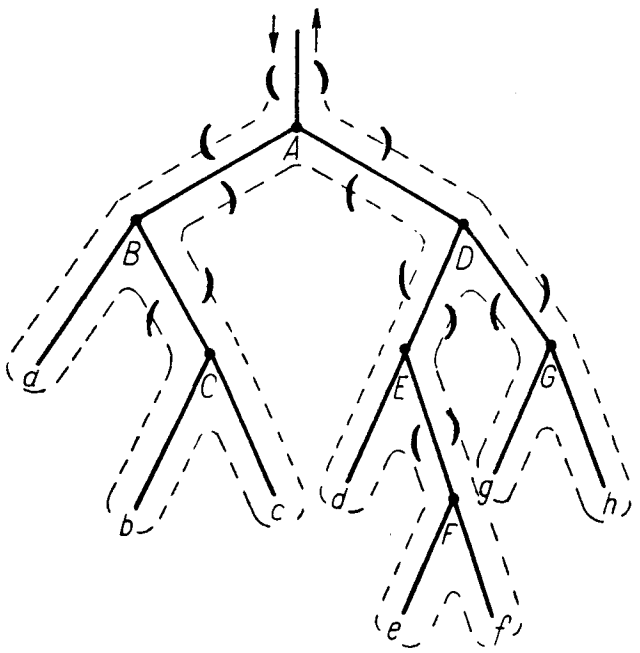


FIG. 10

Let $H(\sigma)$ be the function defined by recurrence

1. $H(\sigma) = 1$, if σ is the first symbol in the program.
2. $H(\sigma') = \begin{cases} H(\sigma) + 1, & \text{if } \sigma \text{ is the symbol of data or left parenthesis,} \\ H(\sigma) - 1, & \text{if } \sigma \text{ is the symbol of operation or right parenthesis.} \end{cases}$

$H(\sigma)$ will be called the depth of the symbol σ in the program.

Let σ_p denote the symbol of depth p. Function $G(\sigma)$ will then be defined as follows:

$G(\sigma) = 0$, if σ is first symbol in the program.

$G(\sigma_p') = \begin{cases} G(\sigma_p), & \text{if } \sigma_p \text{ is left or right parenthesis or symbol of data,} \\ G(\sigma_p) + 1, & \text{if } \sigma_p \text{ is symbol of operation.} \end{cases}$

$G(\sigma_{p+1}^1) = G(\sigma_p^*)$, where σ_{p+1}^1 denotes first symbol of depth $p+1$ in program, and σ_p^* denotes latest symbol of depth p, in the program.

Table I gives the values of functions H and G for the previous example.

TABLE I

	(	(	a	B	(	b	C	c	)	)	A	(	(
$H(\sigma)$	1	2	3	2	3	4	3	4	3	2	1	2	3
$G(\sigma)$	0	1	3	2	3	6	4	6	4	2	1	2	4

d	E	(	e	F	f	)	)	D	(	g	G	h	)	)	)
4	3	4	5	4	5	4	3	2	3	4	3	4	3	2	1
6	5	6	8	7	8	7	5	3	8	7	6	7	6	3	1

A similar definition may be given for the determination of order P.

A numeration of operations in the parenthesis program $\mathfrak{A}$ with order W may be obtained in a similar way to that presented in the paragraph concerning languages L_1 and $\bar{L}_1$. Let us number successive left parentheses in program $\mathfrak{A}$ with numbers $1, 2, \ldots$. With each left parenthesis we associate function $K_i(\sigma)$ defined recursively:

1. $K_i(\sigma) = 1$, if σ is the left parenthesis with number i.
2. $K_i(\sigma') = K_i(\sigma) + 1$, if σ' is the left parenthesis (or symbol of data.
3. $K_i(\sigma') = K_i(\sigma) - 1$, if σ' is the right parenthesis) or symbol of operation.

It may be shown that if $\mathfrak{A}'$ is the program in language L_4 of a simple process with order W, the symbol of operation, for which $K_i(\sigma) = 1$, has the number i according to order W. Thus function $K_i(\sigma)$ associates to each left parenthesis one symbol of operation with the same number as the corresponding parenthesis. A similar definition may be given for order $\overline{W}$.

Table II gives values of function K_i for the example discussed.

TABLE II

Program		(	(	a	B	(	b	C	c	)	)	A
Number of parenthesis		1	2			3						
K_1		1	2	3	2	3	4	3	4	3	2	1
K_2			1	2	1							
K_3						1	2	1				
K_4												
K_5												
K_6												
K_7												
number of operation					2			3				1

TABLE 2 (CONTINUED)

(	(	*d*	*E*	(	*e*	*F*	*f*	)	)	*D*	(	*g*	*G*	*h*	)	)	)
4	5			6							7						
1	2	3	2	3	4	3	4	3	2	1							
	1	2	1														
				1	2	1											
											1	2	1				
			5			6				4			7				

Organization of the machine

Our task is to give the general scheme of the machine for the realization of simple sequential processes according to a given language. There are several solutions of this problem. One solution will be briefly reported here. Some others are to be published elsewhere.

Let us consider a machine consisting of:

R = Partial result memory
D = Data memory
P = Program memory
O = Operator
C = Control

Each memory consists of *cells* denoted by $r_1, r_2, \ldots, d_1, d_2, \ldots, p_1, p_2, \ldots$ for memory R, D and P respectively. Each cell of data memory may hold one number (initial data). Data are located in memory D in the same order in which they occur in the program. Program memory P holds the program of the process in any of the relevant languages. Each cell of memory P holds one symbol of the program. Memory of partial results R stores the result of each operation according to rules described later. Operator O performs operations occurring in the process. Control scans symbols of operations in the program stored in memory P, in the predetermined order $P, \bar{P}, W, \bar{W}$, and investigates symbols of arguments associated with the scanned symbol of operation. Next the contents of memories D or R is read and sent to the operator O. The result of the operation is located in memory R.

Let us define two rules of location of partial results in memory R.

R u l e 1. Partial results are located in consecutive cells of memory R, beginning with cell r_1, and read in the same order beginning also with the first cell r_1.

R u l e 2. Partial results are located in consecutive free cells of memory R, and read beginning from the last occupied cell. After reading, the cell is free.

It may be shown that for order P or $\bar{P}$ the memory of partial result working according to Rule 1 is necessary and for orders W and $\overline{W}$ the memory of partial results have to work according to Rule 2.

Thus the location of partial results in the computer with assumed organization does not depend on the language used for programming but only on the order in which operations are performed. However for different languages discussed here the method of scanning the formula in memory P is dissimilar. For instance, when the basic language is used, the operations in the formula occur in the same order as they are to be obeyed by the computer, thus the control scans all symbols of the formula in consecutive order. But when another language is used, e.g. parenthesis notation, the symbols of the formula are to be read not in the consecutive order but in such a sequence that after operation i operation $i + 1$ is scanned, where i is the number of the operation according to the numeration $P, \bar{P}, W, \overline{W}$. Thus special means must be provided in this case in order to read the formula in the appropriate order. This causes some complications in the organization of the computer, so the proposed scheme is rather more suitable to basic language then to Łukasiewicz or parenthesis language. For these two last-mentioned languages some different solution of the organization of the computer seems to be reasonable. This is studied in some detail in the author's book [1].

REFERENCES

[1] Pawlak, Z., *Organization of addressless computers*, in press.