BULLETIN DE L'ACADÉMIE
POLONAISE DES SCIENCES
Série des sciences techniques
Volume VIII, No. 6 — 1960

Automatic Programming of Arithmetical Formulae in Parenthesis Notation by the Addressing Function

by

Z. PAWLAK

Presented by P. SZULKIN on April 4, 1960

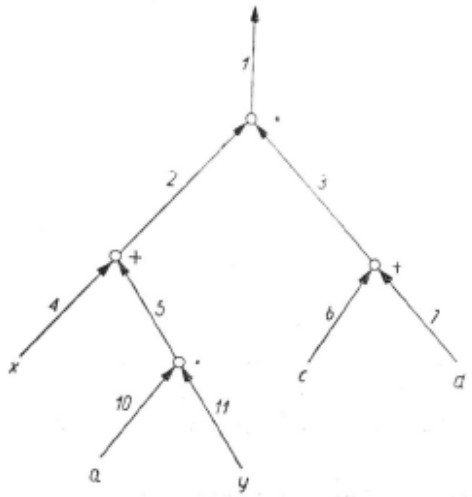
A certain type of organization for a digital computer specially suited for automatic programming has been described in the paper [1]. Results mentioned there can be applied for automatic programming of arithmetical formulae.

Let L be the language with following primitive symbols: variables (denoted by Latin letters), symbols of dyadic operations $\Delta_1, \Delta_2, \ldots, \Delta_n$ and the parenthesis (,).

The formula Φ in L is defined as follows:

i. the variable is the formula in L,

ii. if Φ and Ψ are formulae in L, then the expression $(\Phi \Delta_i \Psi)$ is a formula in L.



Let $\varrho_i(\Phi)$ denote the i-th symbol of the formula Φ, and let $F(\varrho_i(\Phi))$ be the addressing function which attaches to each symbol $\varrho_i(\Phi)$ the natural number $F(\varrho_i(\Phi))$.

Let us adopt the following definition of the function $F(\varrho_i(\Phi))$:

i. $F(\varrho_i(\Phi)) = 1$,

ii. $F(\varrho_{i+1}(\Phi)) = \begin{cases} E(F\varrho_i(\Phi))/2) & \text{if } \varrho_i(\Phi) = \Delta_i \text{ cr}), \\ 2F(\varrho_i(\Phi)) & \text{if } \varrho_i(\Phi) = (, \\ 2F(\varrho_i(\Phi)) + 1 & \text{if } \varrho_i(\Phi) = \Delta_i. \end{cases}$

For example, for the formula $((x + (a \cdot y)) \cdot (c + d))$ the values of the function $F(\varrho_i(\Phi))$ are given in Table I (see the Figure).

TABLE I

$\varrho_i(\Phi)$	(	(	x	+	(	a	·	y	)	)	·	(	c	+	d	)	)
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$F(\varrho_i(\Phi))$	1	2	4	2	5	10	5	11	5	2	1	3	6	3	7	3	7

It is easy to note that if $\varrho_i(\Phi)$ is a symbol of the operation, then the value of the function $F(\varrho_i(\Phi))$ is the address of the result of the operation carried out on arguments with addresses $2F$ and $2F + 1$. Thus a "one-address" instruction may be adopted:

operation	address

which means: carry out operation given in the operational part of the command on arguments, with addresses $2n$ and $2n + 1$, and set the result under address n, where n is the address given in the address part of the command *).

A programming programme should carry out the following operations:

1. Scan subsequent symbols of the formula and compute the addressing function $F(\varrho_i(\Phi))$;

2. If the scanned symbol is that of the operation, form an arithmetical command, the operational part of which is the number of the scanned symbol, and the address part is the value of function F for the scanned symbol, and store the formed command into the subsequent memory register, starting from place p, $p > \text{Max } F(\varrho_i(\Phi))$;

3. If the scanned symbol is a variable, form the transfer command, the address part of which is the value of function F for the scanned symbol and store the transfer command in the subsequent location of the memory starting from k, where $k > p + r - l$ and r is a number of variables in Φ.

4. Order the arithmetical commands in respect to the increasing address parts of these commands.

*) Such a command will be called a "pseudo-one address" command. Obviously, it is possible to design a computer working with "pseudo-one-address" instructions.

5. Store by means of transfer commands previously formed values of variables. (It is assumed for the sake of simplicity that the data are introduced in the sequence, in which they occur in the formula).

6. Carry out the subsequent arithmetical commands applying the programme interpreting the arithmetical commands in the code of the computer.

It is more convenient sometimes in small computers to write immediately in the formula, instead of the variables, their numerical values. The programme thus becomes simplified, as placing transfer commands in the memory is superfluous, whereas the numbers from the input are directly transferred to the memory under the address just computed.

The described system of automatic programming does not utilize the memory to a sufficient degree in the case of certain formulae. The memory utilization coefficient is defined as

$$a = \frac{m}{\operatorname{Max} F(\varrho_i(\Phi))}, \tag{2}$$

where m is the number of data and partial results (or the number of branches of the tree corresponding to the given formula), $\operatorname{Max} F(\varrho_i(\Phi))$ — maximum address occurring in formula Φ. It is easy to note that the Horner formula for which a has the form

$$a = \frac{2n-1}{2^n-1} \tag{3}$$

is the least advantageous for the adopted addressing system, symbol n is the number of variables in the formula. The values a for different n are given in Table II.

TABLE II

n	a	%
2	3/5	100
3	5/7	71
4	7/15	46
5	9/31	29
6	11/63	17
7	13/127	10
8	15/255	5
9	17/511	3

It is convenient for better utilization of the memory to use the function G defined as follows:

Let $H(\varrho_i(\Phi))$ be a function called the order of the symbol $\varrho_i(\Phi))$ defined by recurrence

$$\text{(4)}\qquad \begin{aligned} &\text{i.}\quad H(\varrho_i(\Phi)) = 1,\\ &\text{ii.}\quad H(\varrho_{i+1}(\Phi)) = \begin{cases} H(\varrho_i(\Phi)) + 1; & \text{if } \varrho_{i+1}(\Phi) = X \text{ or } (,\\ H(\varrho_i(\Phi)) - 1, & \text{if } \varrho_{i+1}(\Phi) =) \text{ or } \Delta_i. \end{cases} \end{aligned}$$

Let $\varphi_k^p(\Phi)$ denote the k-th symbol of the order p of the formula Φ; the addressing function G will then be defined as follows:

$$\text{(5)}\qquad \begin{aligned} &\text{i.}\quad G(\varphi_1^1(\varphi)) = 1,\\ &\text{ii.}\quad G(\varphi_{k+1}^p(\Phi)) = \begin{cases} G(\varphi_k^p(\varphi)), & \text{if } \varphi_{k+1}^p(\Phi) =) \text{ or } \Delta_i,\\ G(\varphi_k^p(\Phi)) + 1, & \text{if } \varphi_{k+1}^p(\Phi) = (\text{ or } X, \end{cases}\\ &\text{iii.}\quad G(\varphi_1^p(\Phi)) = \operatorname{Max} G(\varphi_j^p(\Phi)) + 1. \end{aligned}$$

Table III gives the values of functions H and G for the previously given formula.

TABLE III

$\varrho_i(\Phi)$	(	(	x	+	(	a	·	y	)	)	·	(	c	+	d	)	)
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$H(\varrho_i(\Phi))$	1	2	3	2	3	4	3	4	3	2	1	2	3	2	3	2	1
$G(\varrho_i(\Phi))$	1	2	4	2	5	8	5	9	5	2	1	3	6	3	7	3	1

Addressing with the aid of function G gives $a = 1$.

In this case the programming programme carries out identical operations as before, with the only exception that instead of function F, function G is computed.

Addressing by way of function G permits addresses of arguments to be computed not on the basis of addresses of the results but from the addresses of the arithmetical commands from the formulae:

$$\text{(6)}\qquad a_1 = i - p,\ a_2 = i - p + 1,$$

where a_1 and a_2 are addresses of both arguments of the operation, i means address of the arithmetical command being carried out, p — address of the first arithmetical command. The arithmetical command has thus the following meaning: carry out an operation given in the operational part of the arithmetical command on arguments with addresses obtained from formulae (6), and place the result under address given in the address part of the command performed.

A comparison of automatic programming suggested in the present paper on conventionally designed computers with a computer organized as in [2] leads to the following conclusions:

1. Digital computers now in use are not suited to automatic programming.

2. The concept of a digital computer described in [2] seems to be more suitable for automatic programming.

The digital computer described in [2] seems to be more convenient for direct operation with a consequent mathematical language and expressions such as $5 + 4$, or $8 + 7 \cdot 8$ cause at once proper performance of the computer without the intermediary of commands and programme. Every well-formed arithmetical expression is immediately a programme of operation of the computer under consideration. Thus, the conception of a command and a programme seems superfluous from this point of view.

Recently, an experimental digital computer, specially devised for automatic programming on the bases given in [2], is under construction.

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] Z. Pawlak, *The organization of digital computers and computable functions*, Bull. Acad. Polon. Sci., Sér. sci. techn., 8 (1960), 41.

[2] — *Organization of address-free digital computer for calculating simple arithmetical expressions*, Bull. Acad. Polon. Sci., Sér. sci. techn., 8 (1960), 253.