BULLETIN DE L'ACADÉMIE
POLONAISE DES SCIENCES
Série des sciences techniques
Volume IX, No. 9 — 1961

Realization of the Rule of Substitution in Addressless Digital Computer

by

Z. PAWLAK

Presented by P. SZULKIN on June 17, 1961

In paper [1] a certain realization of the rule of substitution was given and here another realization is presented. This realization seems to be more suitable for an addressless digital computer destined for calculation of various problems, it means that the formulae which are to be computed are often changed. Owing to this a more flexible system of substitution is required.

The problem of substitution

Suppose that two functions $z = F(x_1, \ldots\ x_{i-1}, x_i,\ x_{i+1}, \ldots, x_n)$ and $x_i = G(y_1, \ldots, y_k)$ are given and the values of the function $z = \text{sub}\,(F/x_i)\,G$ for different values of arguments $x_1, \ldots, x_n, y_1, \ldots, y_k$ are to be computed. Function $z = \text{sub}\,(F/x_i)\,G$ will be written in the form

$$x_i = G(y_1, \ldots, y_k),$$

$$z = F(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n),$$

and it will be called composite function.

The variable x_i will be called substitutional variable, the function F — the main function with respect to function G, and function G will be called the sub-function with respect to function F. If F is the main function with respect to G, we will write $F \succ G$.

Generally the sequence of functions

$$x_{i_1} = G_1(y_1, \ldots, y_k),$$
$$\ldots\ldots\ldots\ldots$$
$$x_{i_p} = G_p(z_1, \ldots, z_l),$$
$$v = F(x_1, \ldots, x_{i_1}, \ldots, x_{i_p}, \ldots, x_n)$$

denotes the function $v = \text{sub}\,(F/x_{i_1}, \ldots, x_{i_p})\ G_1, \ldots, G_p$, the value of which is obtained by computing first the values of functions $G_1, \ldots, G_p$ and then — the value of function F with $x_{i_1}, \ldots, x_{i_p}$ as values of $G_1, \ldots, G_p$, respectively. Of

course, $F \succ G_1, \ldots, F \succ G_p$. In the general case G_i may, on the other hand, be a main function with respect to other functions, say $K_1, \ldots, K_r$.

The sequence

$$x_1 = F_1,$$
$$x_2 = F_2,$$
$$\ldots\ldots$$
$$x_n = F_n$$

is a composite function if and only if for all $i\,(1 \leqslant i < n)$ there exists at least one such $j\,(i > j \geqslant n)$ that $F_j \succ F_i$.

Example:

$$z = u \cdot v/y,$$
$$x = a \cdot (b - c),$$
$$w = (z - x) \cdot p.$$

Language of the addressless computer

We shall define formal language in which the presented substitution procedure may be expressed, with the precision necessary to study the organization of the computer. As starting point any notation of formulae may be assumed, e.g. ordinary algebraic language, however, for technical reasons we shall accept special parenthesis-free notation which seems to be very valuable in digital computer applications, [2].

Elementary formulae. As primitive symbols of our language we assume:
a) variables — lower case italic letters, eventually with subscripts and upper indices ,'', ''', etc. Variables without indices are interpreted as numbers — with one index ' — as addresses of numbers, with two indices '' — as addresses of addresses of numbers and so on.
b) * — asterisk denoting the partial result,
c) symbols of dyadic arithmetical operations $+, -, \cdot, /$,
d) symbol of a transfer of the result of computation $=$.

i. If α and β are variables, then $\Delta\alpha\beta$ is a term.
Δ denotes one of the symbols $+, -, \cdot, /$.
ii. If α is variable and β is a term, then $\Delta{*}\alpha\beta$ and $\Delta\alpha{*}\beta$ are terms.
iii. If α and β are terms, then $\Delta{*}{*}\alpha\beta$ is a term.
iiii. If α is an address and β is a term, then $\alpha = \beta$ is an elementary formula.

If a formula Φ contains n symbols of arithmetical operations, we shall say that formula Φ is of the length n. We assume that simple formulae are at the most of the length n, where n is a constant number for a given computer, not greater than 30.

The advantage of this limitation lies in avoiding excessively long formulae not readable at one glance.

Example:

$$x' = +{*}{*}/a{*}\cdot bc - pq$$

or

$$y' = \cdot a{*}/cd.$$

As stated in the previous papers this notation has not a unique meaning. Of course, for a given computer the language must have exactly one fixed meaning, by choosing the proper rule of reading the formulae of our language. However, in this paper the interpretation of formulae is not important and may be assumed arbitrary.

Standard formulae. If some elementary formulae are very often used in calculations, we introduce usually abbreviations such as sin, cos, ln, etc. These formulae will be called standard formulae. In order to simplify the problem, we adopt only one letter abbreviations, what limits the total number of standard formulae to about 30. Further, for the sake of simplicity we shall consider only standard functions of one argument, the rest of independent variables being treated as parameters. Of course, the list of standard formulae for a given computer is fixed. In all standard formulae the one argument will be denoted by the same letter u'', with two upper indices. However, in the abbreviation any letter with one upper index may be used as an argument.

Thus, the standard formula may have the form

$$x' = /u''* - u''a,$$

or in the abbreviated form

$$x' = z'y',$$

where y' denotes the argument u'' and z' is the name of the formula. Since the abbreviation has the nature of an address, as it will be shown in the next paragraph, an index is used by z. Since there is no danger of misunderstanding, whether the given expression presents an elementary or standard formula, we shall omit all upper indices in standard formulae abbreviation, so the above example will be

$$x = zy.$$

Also in the elementary formulae the address of the final result will be written without indices.

Composite formulae. Let $x = \Phi$ and $y = \Psi$ be an elementary formula or abbreviation of a standard formula. If Ψ contains the variable x', we will write $x = \Phi \prec y = \Psi$. The sequence $\alpha_1, \alpha_2, \ldots, \alpha_n$ — where α_i is an elementary formula or abbreviation of standard formula—will be a composite formula if and only if for all i, $1 \leqslant i < n$ there exists at least one such j, $i < j \leqslant n$ that $\alpha_i \prec \alpha_j$.

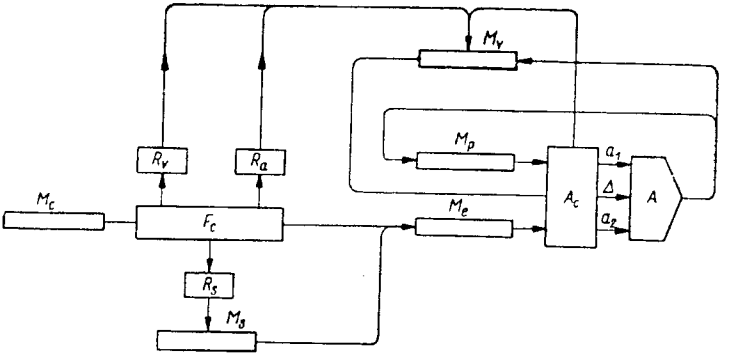
Example:

$$x = /**+ab\cdot cd,$$
$$y = +c*\cdot x'a,$$
$$z = hu,$$
$$w = /*y'+az'.$$

Organization of the computer

A simplified scheme of computer realizing the above described language is shown in the Figure. The computer consists of composite formulae memory M_c, standard formula memory M_s, elementary formula memory M_e (working memory), partial results memory M_p, and elementary functions value memory M_v, arithmo-

meter A, arguments control circuits A_c, formuale control circuits F_c and three address registers, namely: standard function name register R_s, function value address register R_v, and standard function argument address register R_a. Suppose that in the M_c memory there is the composite formula $\alpha_1, \alpha_2, \ldots, \alpha_n$. Formulae control unit F_c selects consecutive expressions α_i, and puts the address of function value to the register R_v. If the selected expression is an elementary formula, then the rest of the formula is transferred to the working memory M_e; in the case when



Simplified diagram of the computer

the selected expression is an abbreviation of the standard formula, the name of the formula is transferred to the register R_s, and the address of the independent variable to the register R_v. The contents of the R_s register cause that the proper standard function is sent from the standard functions memory M_s — to the working memory M_e. Argument control circuit A_c scanes subsequent operations in the working memory M_e, sets the arithmometer to proper operation and selects both arguments from the memory M_e, M_p, or M_v, in dependence whether, the arguments are data, partial results or the value of a previously computed function. The result of each operation is memorized in the memory M_p. The final result of the computation of the formula is in the memory M_v under the address previously set in the function value address register R_v.

INSTITUTE OF MATHEMATICS, POLISH ACADEMY OF SCIENCES
(INSTYTUT MATEMATYCZNY, PAN)

REFERENCES

[1] Z. Pawlak, *On the application of the rule of substitution in the organization of an address-free computer*, Bull. Acad. Polon. Sci., Sér. sci. techn., **8** (1960), 685.

[2] — , *Organization of the address-free digital computer for calculating simple arithmetical expressions*, ibid., **8** (1960), 193.