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

On the Utility of Arithmetical Formalisms in Digital Computers

by

Z. PAWLAK

Presented by P. SZULKIN on June 17, 1961

It is necessary to distinguish two essential, alternately recurring, steps in the process of multiple computation of the values of the function $F(x_1, ..., x_n)$, namely: computation of its values for a prescribed set of arguments, and substitution of the new values of the variables and parameters. The present paper contains certain elementary remarks relating to the rendering of either process by mathematical formalisms. These considerations can be of use in the construction of an external language for digital computers and for that of machines operating directly in a language resembling the one generally prevailing in mathematical symbolics.

Computation of the values of a function

The language describing the process of computing the values of a function should convey more or less precise information on the order of the operations, on finding the arguments of the latter, and on localization the partial results. In the usual parenthesis notation, this information is not presented in sufficiently precise form; such notation, if applied to digital computers, requires more precise formulation by means of additional algorithms. E.g., the distribution of parentheses in a formula is in itself insufficient even for prescribing the order, in which the operations should be carried out. In fixing the latter, we can prescribe an algorithm for presenting the partial results; quite generally, for each ordering of the operations, a different algorithm will be required (see, e.g., [1]). However, independently of the order in which the operations are carried out, the final result is the same, i.e. the meaning of a formula in parenthesis notation does not depend on how we read it. It is noteworthy that, once we have univocally fixed the positions of the arguments corresponding to a given operation, the variables are no longer required to be written in the formula. Thus, e.g., the formula $(((a+b)\cdot c)/(d\cdot e))$ can be rewritten in brief as follows: $(((+)\cdot)/(\cdot))$. The latter simplified formula is sufficient for prescribing the order of the operations, that of presenting the partial results, and that of finding the arguments of an operation. Omission of the left or right parenthesis in simplified parenthesis notation also yields a description of the computation process. (Semi-parenthesis notation

was proposed independently by Ehrenfeucht and Kalmar. The respective results have as yet not been published). Thus, the foregoing example can be written in abbreviated form as follows: $(((+\cdot/(\cdot \text{ or } +)\cdot)/\cdot))$.

The process of computation applying parenthesis-free notation is described similarly. This notation is not univocal, i.e. the meaning of a formula depends on the way in which it is read [2]. Omission of the variables, as in parenthesis notation, is here impossible. E.g. the foregoing formula in parenthesis-free notation is $/\cdot+abc\cdot de$; if read according to a different algorithm [2], this could mean $((a\cdot b)/(c+(d\cdot e)))$ in parenthesis notation.

It would seem that modified parenthesis-free notation presents the greatest interest for application to computers. This notation is defined as follows:

Let Δ denote an arbitrary dyadic operator.

i. If α and β are variables, $\Delta\alpha\beta$ is a formula.

ii. If α is a variable and β is a formula, $\Delta*\alpha\beta$ and $\Delta\alpha*\beta$ are formulae.

iii. If α and β are formulae, $\Delta**\alpha\beta$ is a formula.

This language, too, is non-unique as the meaning of a formula depends on how we read it or, more exactly, on the way in which the partial results are set. This language would seem to be the simplest for technical realizations. The above example, in this language, assumes the form of either $/**\cdot*c+ab\cdot de$ or else — with a different algorithm for setting the partial results — $/**\cdot de\cdot* c+ab$. Neither are the variables required here for describing the process of computation; it is sufficient to know whether a given figure is a partial result or initial data. Thus, in order to describe the process of computation in this language, any single symbol, such as 1, can be used to replace the different variables. The foregoing examples now become $/**\cdot* 1+11\cdot 11$ or $/**\cdot 11\cdot* 1+11$.

Substitution of values of the variables

On substituting in, e.g., the expression $x+y$ of 7 for x and 3 for y, we have the expression $7+3$. An attempt to effect a second substitution, e.g., of 4 for x and of 5 for y would lead us into difficulties, as by the first substitution the initial expression $x+y$ is destroyed and we no longer know where new values should be substituted. Hence, a formalism admitting multiple substitution of values of the variables should present a form, which allows to preserve the symbols of variables, for which we substitute the given values. Such substitution, from the point of view of the computer, does not consist in replacing a variable by a figure but rather in inscribing a figure at the position whose name is the given variable. Thus, from the point of view of the computer, the variable constitutes the name of a position wherein different symbols can be inscribed; moreover, on substitution of the symbol, this name cannot undergo "destruction". In the construction of digital computers, two interpretations of the variables seem to be of practical significance. In order to explain them we assume the formulae to be written on a paper tape divided into

"squares". To each square is related its name, i.e. the variable. If the variable x has the value n, we inscribe the figure n in the square denoted by the variable x*).

If the formula is written down on the paper tape in such a manner that each variable appearing therein is inscribed in the square whose name it constitutes, the system of notation is said to be addressless. If, on the other hand, the formula is written down so that each of its variables is inscribed into a square whose name it does not constitute, we have address notation. Thus, in addressless notation, the value of the variable is inscribed into the same square as the variable. In address notation, the variable and its value are written down in different squares. If one and the same variable occurs n times in a formula, there are n squares in the addressless system denoted by it, whereas in address notation only one square bears ts mark.

In the case of a simple digital computer, the addressless system would seem to present considerable advantages over address notation because of the simplicity of its technical realization. In more highly developed computers, the addressless system can be used for constructing the arithmometer, i.e., the latter would not serve for the computation of separate arithmetical operations but rather of entire arithmetical formulae, so that with such a computer the order could be of the form: Compute the value of the function F.

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

REFERENCES

[1] Z. Pawlak, *Organization of address-free computer with separate memory of partial results*, Bull. Acad. Polon. Sci., Sér. sci. techn., 9 (1961), 123.

[2] — , *New method of parenthesis-free notation of formulae*, ibid., 8 (1960), 197.

*) Substitution thus defined excludes substitution of a function for a variable, e.g., in the expression $x+y$ we cannot substitute $z \cdot u$ for y since, from the point of view of the computer, this would mean an operation consisting in the addition of the figure substituted for x and the expression $z \cdot u$, which obviously makes no sense as addition is here understood to be an operation on figures.