S. N. Razumovskii

1957

Abstract

Full Text

S. N. Razumovskii

ON THE QUESTION OF AUTOMATING THE PROGRAMMING OF TRANSLATION TASKS FROM ONE LANGUAGE INTO ANOTHER

(Presented by Academician M. A. Lavrent' ev, 25 X 1956)

Let us call a variable quantity X that enters into some logical condition of a translation problem a logical variable of the given problem. Translation problems are characterized by the presence of a large number of logical variables. For example, in the problem of translation from English into Russian their number may reach a thousand or more. In connection with this there arises the necessity of storing several logical variables in one cell of the memory device. The possibility of such storage is ensured by the fact that, as a rule, logical variables assume only a small number of discrete values. In order to specify the coordinates of a logical variable, it is necessary to indicate not only the address of the cell of the memory device containing this variable, but also the position of the variable within the given cell.

Let us renumber in some way the variables within a memory cell and establish a one-to-one correspondence between the number of the variable quantity and its position. We shall specify the coordinates of logical variables by two indices i, k, where i is the ordinal number of the quantity within a memory cell and k is the number of the memory cell.

A characteristic feature of programs of this type of logical problems is that the programs operate not on the entire contents of a memory cell, but only on some part of it. Another characteristic feature of these programs, associated with the large number of logical variables appearing in them, is the presence of a large number of connections between their different parts. If the size of the programs is so large that it is necessary to enter the program into the operative memory in parts, then the optimal choice of these parts is a problem consisting in combining in the best way the condition of maximum use of memory with the condition of minimum intersection of connections.

The schemes of the problem of translation from English into Russian can be represented in the form of sequences of three types of operators: logical, identity, and arithmetic. The same representation is also possible in translation from other languages, for example from German, Japanese, and Chinese.

A **logical operator** is constructed with the aid of the connectives of the propositional calculus: $\wedge$ conjunction; $\vee$ disjunction; $\rightarrow$ implication; $-$ negation.

Let A_1 and A_2 be always true propositions. X is a proposition, true or false.

By the definition of implication, $A_1 \to X$ is true when X is true, and false otherwise. The disjunction of implications

$$(A_1 \to X) \vee (A_2 \to \bar{X})$$

is always true.

Let the proposition A_1 represent the instruction: execute the operator numbered A_1, and A_2, respectively, the instruction: execute the operator A_2.

The truth of the implication $A_1 \to X$ means that one should proceed to the execution of the operator A_1, and the falsity of this implication (and consequently, the truth-

ness $A_2 \to \bar{X}$) means that one should proceed to execute the operator numbered A_2. Thus, a logical operator establishes a correspondence between the set of statements $X, \bar{X}$ and the set A_1, A_2.

We shall agree to write a logical operator in the following way: $\{XA_1; A_2\}$.

Let us associate with the statement X some formula f, constructed by means of the signs $=, <, >$, etc., and having in its structure a constant C and a logical variable x_{ik}. The truth of the statement X corresponds to satisfaction of the condition of the formula (for the given C and x_{ik}); the falsity of X corresponds to nonsatisfaction of the condition of the formula. The logical operator is written in the form $\{f(x_{ik}, C)A_1; A_2\}$.

The **identity operator** establishes the correspondence $x_{i_1 k_1} = c_1, \ldots, x_{i_{nk} n} = c_n$ between the elements of two sets: $x_{i_1 k_1}, \ldots, x_{i_{nk} n}$, and $c_1, \ldots, c_n$. The operator contains in its structure the sign ";" . After it is indicated the number of the operator to which one should proceed after executing the given operator: $[x_{i_1 k_1} = c_1, \ldots, x_{i_{nk} n} = c_n; A]$.

The **arithmetic operator** is an operator on the quantities y_k, y_{k-1}, and m, namely: $y_k = y_{k-1} \pm m$. The arithmetic operator is written in the form $|y \pm m|$. After execution of the arithmetic operator, a transition is made to the operator immediately following the closing vertical bracket.

The adopted system of notation for logical, identity, and arithmetic operators contains all the information needed to construct the parts of the program implementing these operators.

The **information coding system**. We shall call information about one sign of the operator notation of a scheme elementary information. Each item of elementary information is accompanied by an indication of its belonging to one of three types: 1) operator numbers; 2) letters, indices, and operator signs; 3) numbers. Numbers and operator numbers are written as the corresponding binary numbers. A special coding system has been developed for writing letters and signs.

The **programming program** consists of several programs: the main program, the program for synthesizing logical operators, the program for synthesizing identity operators, the program for synthesizing arithmetic operators, the program for compressing the constructed program, and the program for dividing the constructed program into parts.

The operation of the program proceeds in the following order. The coded information is entered into the machine. Referral to the synthesis of definite types of operators is carried out by the main program. After a definite operator has been constructed, a return is made to the main program. This makes it possible to process complex structures (for example {[{‖}]}).

The compression program operates after the entire program has been synthesized. Compression is carried out by eliminating possible unnecessary transfers of control and by eliminating certain recurring parts of the program. If necessary, the program for dividing the constructed program into parts operates.

Institute of Precision Mechanics and Computer Engineering
Academy of Sciences of the USSR

Received
10 X 1956

Note: Figure translations are in progress. See original paper for figures.