SYSTEM OF TRANSFORMATIONS OF CERTAIN REPRESENTATIONS OF BOOLEAN FUNCTIONS

B. P. Kuznetsov and A. A. Shalyto

UDC 681.3.06:62-507

Three forms of representation of Boolean functions are considered: the nonrepetitive Boolean formula, the linear binary graph, and the orthogonal disjunctive normal form. A one-to-one correspondence among them is exhibited, and a complete system of equivalent transformations is studied. The role of these transformations in computer-aided programming of logical devices is noted.

The following problems arise in the automation of the programming of logical devices [1]: to estimate the complexity of a binary graph (BG) of a given Boolean formula (BF) in the basis AND, OR, NOT; to translate a BF into a BG; to enumerate the paths in a given BG (testing of a BG); to retranslate a BF of a given BG; to retranslate a BF of a given enumeration of the paths of a BG; to form a BG from a given enumeration of its paths.

Because an arbitrary BF can always be reduced by a change of variables to a nonrepetitive Boolean formula (NBF) represented in the same basis and containing the same number of letters [2], the BG corresponding to it will be linear (LBG) [3-5]. Therefore the above problems involving NBFs and LBGs can be solved, and it is possible to use orthogonal disjunctive normal forms (ODNF) to enumerate the paths of a LBG [6].

Thus to solve the above problems it is necessary to consider a complete system of one-to-one equivalent transformations of the three forms of representation of Boolean functions: NBF, LBG, and ODNF (Fig. 1).

Methods are known for carrying out some of these transformations; for example, a transformation of a NBF into a LBG is studied in [3-5], and a transformation of a NBF into a ODNF in [6]. Transformations of LBGs into NBFs and ODNFs, and transformations of ODNFs into NBFs and LBGs are not considered in practice. Therefore one can say that a complete system of transformations has never been studied from a single point of view.

It is the purpose of this article to remedy this deficiency.

We enumerate the basic peculiarities of a LBG. Its conditional vertices correspond, one-to-one, to the letters of the right-hand side of the NBF and are placed linearly in the same order as the letters in the NBF. A LBG, besides h conditional vertices, also contains two operator vertices, where h is the number of letters in the right-hand side of the NBF. The conditional vertices of the LBG are connected by arcs which are always directed to the right. Adjacent conditional vertices are always connected by an arc. We will always give the operator vertex "y = 1" the number h + 1, and the vertex "y = 0" the number h + 2, where y is the notation for the NBF.

In what follows, we assume that the NBF is positively monotone, and that its structure is fixed.

1. TRANSFORMATION OF A NBF INTO A LBG

In [4, 5] an algorithm is given which is based on the replacement of fragments of a NBF by intermediate variables and the construction of a LBG by the composition of its fragments corresponding to these variables. Therefore, the number of steps in the transformation of the NBF into the LBG is not proportional to the number of letters in the NBF but depends in an essential way on the structure of the NBF. We propose a simple method for accomplishing this transformation with a linear bound for the complexity of the number of steps.

We construct the framework of the LBG from the linear arrangement of the vertices corresponding to the letters of the NBF in order of their appearance from left to right in its

Leningrad. Translated from Avtomatika i Telemekhanika, No. 11, pp. 120-127, November, 1985. Original article submitted November 11, 1984.

Automation and Remote Control. Vol. 46. No 11, Part 2, November, 1985

 0005-1179/85/4611-1450$09.50

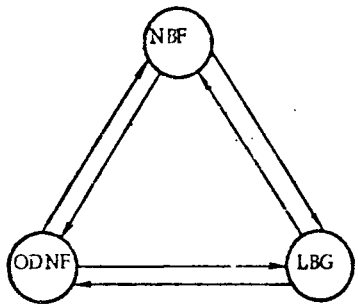

Fig. 1. Complete system of equivalent transformations of the three forms of representation of Boolean functions.

right-hand side. Adjacent conditional vertices are connected by arcs which are oriented to the right. We will call them basic arcs in what follows. We will call the last conditional vertex to the right the output. We will not consider operator vertices in what follows but we will use instead the term "output" (unit or zero).

We mark off the basic arcs in the following way. We designate the arc as unit (zero) if there is a conjunction (disjunction) sign after the corresponding letter in the NBF.

Since two arcs must issue from each conditional vertex ("vertex" from now on), the second (additional) arc must connect the given vertex with some vertex of the framework whose symbol is as yet unknown. To determine the symbol of this vertex, we introduce the concept of the remainder of the formula (RF) with respect to a letter of the NBF.

The RF of some letter of a NBF is a conjunction (disjunction) transformed by neglecting that part of the NBF located to the left of this letter and by including unpaired parentheses in the remaining part of the NBF consisting of more than one letter. In addition, the conjunction (disjunction) can contain an arbitrary formula in parentheses as a factor (term). For example, for a NBF of the form $y = x_1(x_2 \vee x_3x_4)x_5 \vee x_6x_7$ we get as RFs of its letters: $x_1(x_2 \vee x_3x_4)x_5$; $x_2 \vee x_3x_4$; $x_3x_4x_5$; x_4x_5; $x_5 \vee x_6x_7$; x_6x_7.

The unknown vertex with which the one under consideration is connected by the new arc corresponds to the letter on the right of the RF just obtained. If there is no letter to the right of this RF, then the new arc is connected to a unit output vertex if the RF is a disjunction and to a zero output vertex if the RF is a conjunction.

In the case where the formula is not monotonic, the LBG is constructed disregarding inverse, and then arcs issuing from vertices corresponding to letters with inverses are labeled with the opposite symbols.

Example 1. We construct the LBG for a given NBF of the form $y = (x, \bar{x}_2 \vee x_3)\bar{x}_4$. First we construct the LBG for the NBF without the negations $y = (x_1x_2 \vee x_3)x_4$. We construct the framework and mark the vertices, the basic arcs, and the arcs of the output vertex (Fig. 2a). We list the RFs for y: x_1x_2; $x_2 \vee x_3$; x_3x_4. Hence it follows that vertex x_1 must be connected to x_3 (Fig. 2b), x_2 to x_4 (Fig. 2c), and x_3 to a zero output (Fig. 2d). The arcs issuing from x_2 and x_4 must be relabeled with the opposite symbols because they appear in y with inverses (Fig. 2e).

Thus the LBG of a given NBF can be constructed in h (for a positively monotone formula) or h + 1 (for a nonmonotonic formula) steps. For the nonmonotonic formula in this example, the number of steps in the transformation is five (Fig. 2a-e).

2. TRANSFORMATION OF A NBF INTO AN ODNF

Known methods of orthogonalization (obtaining an ODNF from a given BF) are oriented, as a rule, toward the reduction of the BF to a DNF, from which one ODNF is constructed ([6], for example). Two basic rules are used to do this.

We first introduce the following notation: D is a disjunction, D' is the ODNF of a disjunction, $\bar{D}'$ is the ODNF of a negation of a disjunction, K is a conjunction, K' is the ODNF of a conjunction, and $\bar{K}'$ is the ODNF of a negation of a conjunction.

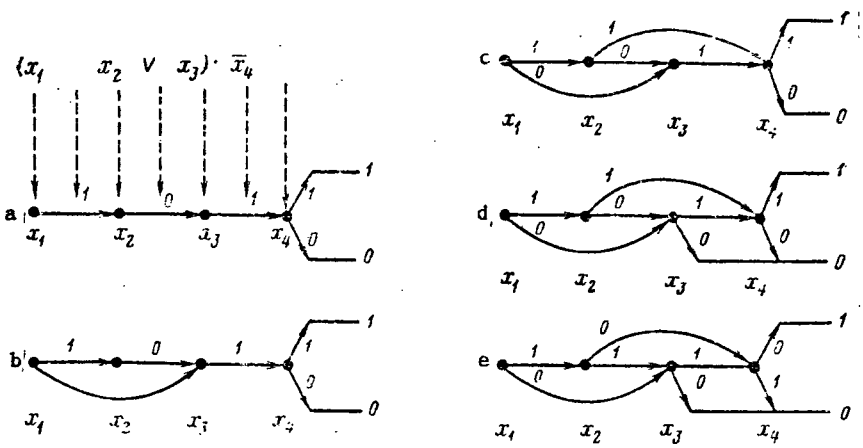

Fig. 2. Construction of a LBG for a given NBF. a) Construction of the framework; construction of the additional arcs originating at the vertices: b) x_1; c) x_2; d) x_3; e) renaming of the arcs.

Rule 1. If $D = x_1 \vee x_2 \vee \ldots \vee x_{n-1} \vee x_n$, then $D' = x_1 \vee \bar{x}_1 x_2 \vee \ldots \vee \bar{x}_1\bar{x}_2 \ldots \bar{x}_{n-1}x_n$ and $\bar{D}' = \bar{x}_1\bar{x}_2 \ldots \bar{x}_{n-1}\bar{x}_n$.

Rule 2. If $K = x_1x_2 \ldots x_{n-1}x_n$, then $K' = x_1x_2 \ldots x_{n-1}x_n$ and $\bar{K}' = \bar{x}_1 \vee x_1\bar{x}_2 \vee \ldots \vee x_1x_2 \ldots x_{n-1}\bar{x}_n$.

The transformation of a NBF into an ODNF is carried out in the following way. Every disjunction and conjunction in the NBF is replaced by an intermediate variable, and Rules 1 and 2 are applied to each of these. The operation of substitution is applied to the ODNF of intermediate variables. Then the parentheses are removed to obtain the desired ODNF. In the course of a transformation, the order of the letters and the intermediate variables must not be changed.

The ODNF obtained in this way is a listing of paths, beginning at the leftmost vertex and ending at a unit output corresponding to the given NBF. Consequently the ODNF is a listing of the "unit" paths of the LBG obtained without constructing it. The ODNF constructed from the inverse of an NBF is a listing of the "zero" paths of the same LBG.

Thus if we obtain the ODNFs from a given NBF and its inverse, it is possible to estimate the complexity of the binary program (BP) which realizes the given NBF, since we find in [7] that the number and the sum of the lengths of the paths in the graph corresponding to it are related to the basic characteristics of an estimate of the complexity of a program. These indices are especially important for a NBF since the number of vertices in the LBG does not properly define its complexity. For h = const, all LBGs have the same number of vertices, h + 2, but they can differ significantly in the indices introduced above. Even for formulas obtained from each other by a permutation of the disjunctions and/or conjunctions, these indices can differ.

Example 2. We wish to construct the ODNF for $y_1 = (x_1\bar{x}_2 \vee x_3)\bar{x}_4$ and its inverse.

We introduce the intermediate variables:

$$a = x_1\bar{x}_2; \quad b = a \vee x_3; \quad c = b\bar{x}_4; \quad y_1 = c.$$

We apply Rules 1 and 2:

$$a' = x_1\bar{x}_2; \quad b' = a' \vee \bar{a}'x_3; \quad c' = b'\bar{x}_4; \quad y_1' = c';$$

$$\bar{a} = \bar{x}_1 \vee x_1x_2; \quad \bar{b}' = \bar{a}'\bar{x}_3; \quad \bar{c}' = \bar{b}' \vee b'x_4; \quad \bar{y}_1' = \bar{c}'.$$

We substitute the variables

$$y_1' = c' = b'\bar{x}_4 = (a' \vee \bar{a}'x_3)x_4 = (x_1\bar{x}_2 \vee (\bar{x}_1 \vee x_1x_2)x_3)\bar{x}_4 =$$
$$= x_1\bar{x}_2\bar{x}_4 \vee \bar{x}_1x_3\bar{x}_4 \vee x_1x_2x_3\bar{x}_4;$$
$$\bar{y}_1' = \bar{c}' = \bar{b}' \vee b'x_4 = \bar{a}'\bar{x}_3 \vee (a' \vee \bar{a}'x_3)x_4 =$$
$$= (\bar{x}_1 \vee x_1x_2)\bar{x}_3 \vee (x_1\bar{x}_2 \vee (\bar{x}_1 \vee x_1x_2)x_3)x_4 =$$
$$= \bar{x}_1\bar{x}_3 \vee x_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_4 \vee \bar{x}_1x_3x_4 \vee x_1x_2x_3x_4.$$

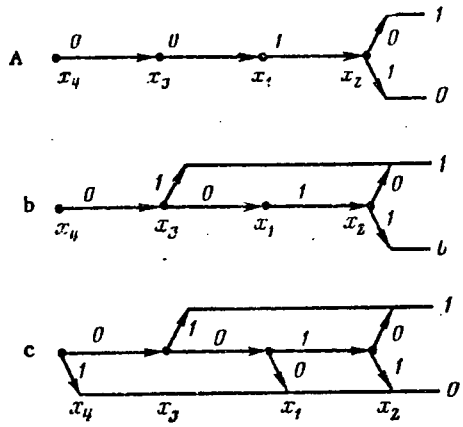

Fig. 3. Construction of the LBG of a given ODNF. a) Construction of the framework; b) construction of the additional arc for the second conjunction; c) construction of the remaining additional arcs.

If we compare y' and $\bar{y}'$ with the list of unit and zero paths of the LBG depicted in Fig. 2e, we can see that they coincide.

If we apply this method to the formula $y_2 = \bar{x}_4(x_3 \vee x_1\bar{x}_2)$ obtained from y_1 by permuting fragments, we get that $y_2' = x_4x_3 \vee x_4x_3x_1x_2$ and $\bar{y}_2' = x_4 \vee \bar{x}_4\bar{x}_3\bar{x}_2 \vee x_4x_3x_1x_2$.

3. TRANSFORMATION OF AN ODNF INTO A LBG

We consider the ODNF corresponding to the unit paths of a LBG, and we construct this graph. The conjunction in the ODNF with the largest number of letters corresponds to the framework of the LBG. Therefore we can construct the framework of the conjunction by marking the basic arcs appropriately, depending on the absence or presence of an inversion sign over the corresponding letter in the conjunction. Then we carry out a step-by-step process in which, at each step, we construct an additional arc for the next conjunction. If the conjunction coincides (without taking inverses into account) in i letters to the left and K letters to the right with the conjunction forming the framework, then the next additional arc starts at the i-th vertex and ends at the (h − K + 1)-th vertex. For K = 0 this arc ends at the unit output of the LBG. After considering all of the conjunctions of the ODNF, we construct the remaining additional arcs which close on the zero output of the LBG.

Example 3. We construct the LBG for the ODNF $y_2' = \bar{x}_4x_3 \vee \bar{x}_4\bar{x}_3x_1\bar{x}_2$. The conjunction $\bar{x}_4x_3x_1\bar{x}_2$ corresponds to the framework (Fig. 3a). The conjunction $\bar{x}_4x_3$ coincides in the first two letters with the first one (i = 2, K = 0); therefore, we connect the vertex x_3 with the unit output by an additional arc (Fig. 3b). We connect the vertices x_4 and x_1 with the zero output by means of additional arcs (Fig. 3c).

4. TRANSFORMATION OF AN ODNF INTO A NBF

To begin with, we consider the difference between a NBF and an ODNF. We will call the letters in a NBF, including those with inverse signs, basic symbols. Besides the basic symbols, an ODNF contains supplementary symbols, negations of basic symbols. Therefore an ODNF contains an excess in comparison with the DNF obtained from the given NBF by expanding the parentheses, at the expense of the supplementary symbols.

The transformation of an ODNF into a NBF can be carried out in three steps: the elimination of the supplementary symbols in the ODNF; making all reductions in the resulting DNF; taking symbols out of parentheses.

We consider the first step in more detail. We make a list of the conjunctions of the ODNF. Using this list, we carry out the following recursive procedure to determine the next basic symbol:

If the list contains only one letter, then this is regarded as the last of the basic symbols, and the procedure is terminated;

the letter farthest to the right in the conjunction with the largest number of letters in the list is taken as the basic symbol of the required NBF;

if all of the conjunctions of the list contain a found basic symbol, then this symbol is eliminated from all of the conjunctions in the list, and the procedure is repeated;

if conjunctions containing a found basic symbol include letters which are absent in conjunctions not containing a found basic symbol, then this basic symbol is eliminated from the corresponding conjunctions, and the procedure is applied again;

all of the conjunctions containing found basic symbols are deleted, and the procedure is repeated.

As a result of the completed procedure, we get a list of basic symbols. The corresponding supplementary symbols are eliminated from the original ODNF. We get a DNF, and the first step is completed.

Example 4. We obtain the NBF of the ODNF $y_1' = x_1\bar{x}_2\bar{x}_4 \vee \bar{x}_1x_3\bar{x}_4 \vee x_1x_2x_3\bar{x}_4$. We make a list of the conjunctions: $x_1x_2x_4$; $x_1x_3x_4$; $x_1x_2x_3x_4$. We apply the above procedure. We identify the basic symbol x_4. After this, the list becomes: x_1x_2; x_1x_3; $x_1x_2x_3$. We identify the basic symbol x_3, and the list becomes: $x_1\bar{x}_2$. We identify the basic symbol x_2; the list then degenerates to x_1. We identify the basic symbol x_1. The list of conjunctions is then empty. The basic symbols of the required NBF are: x_1, $\bar{x}_2$, x_3, $\bar{x}_4$. We delete the supplementary symbols from the ODNF, and we get a DNF of the form $y_1 = x_1x_2x_4 \vee x_3x_4 \vee x_1x_3x_4$. If we carry out a reduction, we get $y_1 = x_1x_2x_4 \vee x_3x_4$. We take x_4 out of the parentheses. The NBF has the form $y_1 = (x_1x_2 \vee x_3)x_4$.

5. TRANSFORMATION OF A LBG INTO AN ODNF

We consider a correspondence between an enumeration of the parths of an LBG and an ODNF. We will denote the unit and zero arcs issuing from the vertex x_i by x_i and $\bar{x}_i$, respectively. Then any path in the LBG beginning at the leftmost vertex and ending at the output of the LBG can be written as conjunctions having as terms symbols which deviate arcs along which the path extends.

The enumeration of paths coincides with the enumeration of the conjunctions in the ODNF corresponding to the given LBG. Therefore the transformation of the LBG into an ODNF is equivalent to an enumeration of its paths (unit and zero).

The enumeration of the paths of a LBG is a special case of the problem of the enumeration of the paths of graphs, which is treated in many articles, in [7], for example. However, the well-known methods of solving this problem are based on the application of combinatorial operations on graphs. We consider an analytic method of enumerating the paths of a LBG. To do this, we must represent the LBG by means of expressions which describe the connection between its conditional vertices.

We define the right subgraph of an LBG with respect to a vertex i to be the subgraph of the LBG which includes the vertex i and all of the vertices to the right of i, together with all of the arcs issuing from vertices in this subgraph. We will denote the right subgraph with respect to the vertex i by e_i, and henceforth call the subgraph e_i. The subgraph e_i corresponds to the LBG in purpose. We note that a basic arc issuing from the vertex i connects with the vertex i + 1, and an additional arc with the node K, where $K \geq i + 2$, $i \leq h$.

Each subgraph e_i can be expressed in terms of a subgraph of vertices adjoining i with respect to the arcs issuing from it. Hence e_i can be written in one of the forms:

$$e_i = x_i e_{i+1} \vee \bar{x}_i e_k, \tag{1a}$$

$$e_i = \bar{x}_i e_{i+1} \vee x_i e_k. \tag{1b}$$

The expression (1a) is used if the basic arc is a unit arc, while (1b) is used if it is a zero arc. To enumerate the unit paths, we put $y = e_1$, $e_{h+1} = 1$, $e_{h+2} = 0$. To enumerate the zero paths, we put $y = e_1$, $e_{h+1} = 0$, $e_{h+2} = 1$, and to enumerate all of the paths, we put $y = e_1$, $e_{h+1} = e_{h+2} = 1$. Then we successively substitute the expressions for each subgraph in the expression obtained in the preceding step. When all of the substitutions have been made and the parentheses expanded, we have the required ODNF. If the variable e_i is encountered several times in some step of the transformation, we may take it out of the parentheses and then replace it by the corresponding subgraph. We note that the substitution of expressions for subgraphs must be done in order of increasing indices, which makes it possible

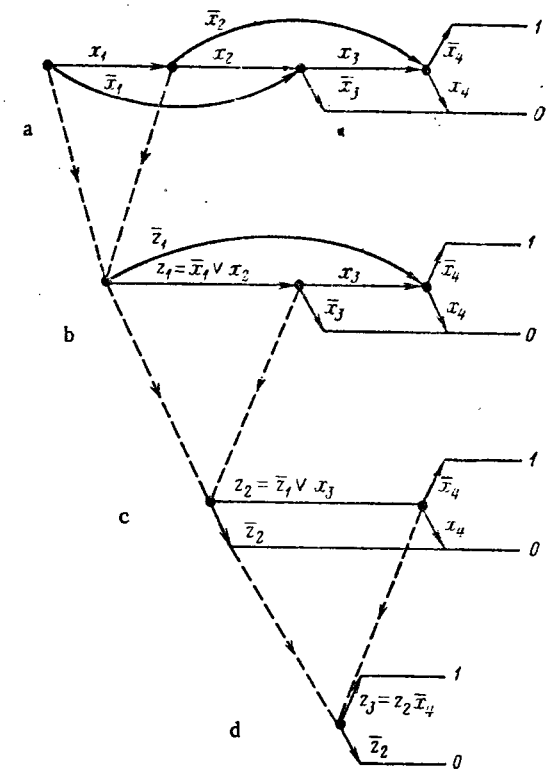

Fig. 4. Construction of the NBF of a given LBG. a) Original LBG; b) step 1; c) step 2; d) step 3.

to simplify the transformation by eliminating multiple substitutions of an expression for the same subgraph.

Example 5. For the LBG in Fig. 4a, we obtain the ODNF corresponding to the enumeration of the unit paths.

The given LBG is described in the following way:

$$e_1 = x_1 e_2 \vee \bar{x}_1 e_3, \quad e_3 = x_3 e_5 \vee \bar{x}_3 e_6,$$
$$e_2 = x_2 e_3 \vee \bar{x}_2 e_4, \quad e_4 = \bar{x}_4 e_5 \vee x_4 e_6. \tag{2}$$

We put $y_1' = e_1$, $e_5 = 1$, $e_6 = 0$.

Then $y_1' = e_1 = \underline{x}_1 e_2 \vee \bar{x}_1 e_3 = x_1 (\underline{x}_2 e_3 \vee \bar{x}_2 \underline{e}_4) \vee \bar{x}_1 e_3 = (x_1 x_2 \vee \bar{x}_1) e_3 \underline{\vee} x_1 \underline{\bar{x}}_2 e_4 = (x_1 x_2 \vee \bar{x}_1)(x_3 e_4 \underline{\vee} x_3 e_6) \vee \underline{x}_1 \underline{x}_2 e_4 = [(x_1 x_2 \vee x_1) x_3 \vee x_1 x_2] \underline{e}_4 = [(x_1 x_2 \vee x_1) x_3 \vee x_1 x_2](x_4 e_5 \vee x_4 e_6) = [(x_1 x_2 \vee x_1) x_3 \vee x_1 x_2] x_4 = x_1 x_2 x_3 x_4 \vee x_1 x_3 x_4 \vee x_1 x_2 x_4$.

6. TRANSFORMATION OF A LBG INTO A NBF

Before carrying out this transformation, we consider the common properties of NBFs and LBGs.

Since in a NBF it is always possible to replace a conjunction (disjunction) of variables by a single new variable, there is always the analogous possibility of replacing several adjacent conditional vertices in the corresponding LBG by a single vertex which corresponds to the replaced fragment of the NBF. Successively replacing the vertices of the LBG, we can transform it into a graph with a single conditional and two operator vertices. We will call such a transformation of a LBG a reduction. We will call the process of replacing two neighboring conditional vertices by a single vertex a reduction step. We will denote basic arcs by the symbol a_i and additional arcs by $\bar{a}_i$ ($i = \overline{1, n-1}$). Also a_h and $\bar{a}_h$ denote arcs which issue from the output vertex.

We state reduction rules for vertices i and i + 1, whose subgraphs are described by expressions of the form

$$e_i = a_i e_{i+1} \vee \bar{a}_i e_h, \quad e_{i+1} = a_{i+1} e_{i+2} \vee \bar{a}_{i+1} e_h, \tag{3}$$

where i, k, and n are indices of the vertices and $k \geq i + 2$, $n > i + 2$, $k, n \leq h + 2$, $i \leq h$.

As a result of a reduction step, these vertices are replaced by a single one which corresponds to the intermediate variable z_j (j is the index of the intermediate variable). The subgraph of the new vertex is denoted by an expression of the form

$$e_i' = z_j e_{i+2} \vee \bar{z}_j e_n. \tag{4}$$

In (3) and (4), the values of i correspond to the LBG up to the application of the reduction step. The new vertex occupies the position i in the transformed LBG, and the remaining vertices which are to the right of the new one occupy positions with indices which are one smaller than they were before the reduction step. Therefore we must replace (4) by the more precise

$$e_i' = z_j e_{i+1}' \vee \bar{z}_j e_{n-1}, \tag{5}$$

where

$$e_{i+1}' = e_{i+2}, \quad e_{n-1} = e_n.$$

Rule 1. If $K = i + 2$ in the expressions in (3), then the vertices i and i + 1 are replaced by a new one corresponding to the intermediate variable $z_j = a_i \vee a_{i+1}$.

Rule 2. If $K = n > i + 2$ in the expressions in (3), then the vertices i and i + 1 are replaced by a new one corresponding to the intermediate variable $z_j = a_i a_{i+1}$.

Rule 3. If $K \neq i + 2$ and $K \neq n$ in the expressions in (3), then the vertices i and i + 1 are not changed.

Rules 1-3 are used for the vertices of an LBG which correspond both to letters in the original formula and to intermediate variables formed in the preceding reduction steps.

A LBG can be transformed into a NBF making use of the above ideas in the following way. The LBG is represented in the form of a list of expressions for the subgraphs of all of its vertices. Then neighboring vertices are considered pairwise, beginning with the pair of vertices 1 and 2, and reduction steps are applied. If Rule 1 or 2 is applied in the next reduction step, then the following reduction step is applied to the vertex i − 1 and the new vertex i. If Rule 3 is applied in the next reduction step, then the following reduction step is applied to the vertices $i = 1$ and $i + 2$ ($i < h$). The reduction ends when the LBG is transformed into a graph with a single vertex and two outputs. Then a sequence of substitutions of intermediate variables is made, and the result is the required NBF.

Example 6. For the LBG given by the expression (2) with $e_5 = 1$ and $e_6 = 0$ (Fig. 4a), we construct the equivalent NBF. We carry out the reduction steps.

Step 1. We consider the pair of vertices 1 and 2. Then $i = 1$, and from (2) and (3) it follows that $K = 3$ and $n = 4$. Since $K = i + 2$, we apply Rule 1 and replace vertices 1 and 2 by a new vertex 1 corresponding to the intermediate variable $z_1 = x_1 \vee x_2$ (Fig. 4b), and if we take (5) into account, (2) is transformed into the form

$$\begin{aligned} e_1' &= z_1 e_2' \vee \bar{z}_1 e_3', \quad e_3' = \bar{x}_4 e_4' \vee x_4 e_5', \\ e_2' &= x_3 e_3' \vee \bar{x}_3 e_5', \quad e_4' = 1, \quad e_5' = 0, \end{aligned} \tag{6}$$

where $e_i' = e_{i+1}$.

Step 2. We consider the pair of new vertices 1 and 2. From (3) and (6) it follows that $K = 3$ and $n = 5$. Since (Fig. 4c) $K = i + 2$, it follows that $z_2 = z_1 \vee x_3$, and (6) is transformed into the form

$$\begin{aligned} e_1'' &= z_2 e_2'' \vee \bar{z}_2 e_4'', \quad e_5'' = 1, \\ e_2'' &= \bar{x}_4 e_3'' \vee x_4 e_4'', \quad e_4'' = 0. \end{aligned} \tag{7}$$

Step 3. For the new pair of vertices 1 and 2, $K = n = 4$. Therefore, we apply Rule 2, as a result of which the LBG is transformed into a graph with a single conditional vertex (Fig. 4d) corresponding to the variable $z_3 = z_2 x_4$.

Now we make the substitution of variables: $y = z_3 = z_2 \bar{x}_4 = (\bar{z}_1 \vee x_3)\bar{x}_4 = (\overline{(\bar{x}_1 \vee x_2)} \vee x_3)\bar{x}_4 = (x_1 x_2 \vee x_3)\bar{x}_4$.

LITERATURE CITED

1. Yu. N. Butin, M. Ya. Zolotarevskaya, A. P. Kirillov, and V. N. Yung, "On the implementation of control algorithms in dedicated programmable logical devices," Avtomat. Telemekh., No. 6, 131-140 (1983).
2. V. L. Artyukhov, G. A. Kopeikin, and A. A. Shalyto, Tunable Modules for Controlling Logical Devices [in Russian], Énergoizdat, Leningrad (1981).
3. O. P. Kuznetsov, "Program realization of logical functions and automata," Avtomat. Telemekh., No. 7, 163-174; No. 9, 137-149 (1977).
4. S. S. Kamynin, E. Z. Lyubimskii, and M. R. Shura-Bura, "Automation of programming by means of programming programs," in: Problems of Cybernetics [in Russian], Vol. 1, Fizmatgiz, Moscow (1958), pp. 135-171.
5. A. M. Bogomolov and V. A. Tverdokhlebov, Expedient Behavior of Automata [in Russian], Naukova Dumka, Kiev (1975).
6. I. A. Ryabinin and Yu. N. Kireev, Reliability of Marine Electrical Power Systems and Marine Equipment [in Russian], Sudostroenie, Leningrad (1974).
7. V. V. Lipaev, Quality of Program Security [in Russian], Finansy i Statistika, Moscow (1983).