TUNABLE BINARY PROCEDURES AND PROGRAMS WITH LOOPS*

V. L. Artyukhov, B. P. Kuznetsov, and A. A. Shalyto

UDC 681.3.06

A method is proposed for the evaluation of a system of Boolean formulas using a program with a loop in which some parameters generated by a previously created array are successively passed to a special procedure. The program with a loop is termed a cyclic binary program, and the procedure is termed a tunable binary procedure. Basic design principles and complexity bounds are considered.

INTRODUCTION

There is an ever-growing interest in program realization of systems of Boolean functions (SBF), using compilers [1, 2] and interpreters [3]. Interpretation is the more promising approach in those cases when no special requirements are imposed on the time characteristics of the program. An interpreter is independent of the Boolean functions being realized, and the functions are specified by an array [3]. However, the generation and updating of such an array for the specification of SBF with many variables and formulas is an arduous and time-consuming task. If we can abstract from issues of time and memory optimization, which are the primary concern of [1, 3], and focus on simplicity of the realizing procedure (as in the formula method proposed in [1]), the interpreter may be designed as a cyclic program which includes a general-purpose compiler procedure with parameters. Then the specified SBF may be represented by a table (array) of parameters, and formula evaluation reduces to strictly sequential enumeration of the table rows and substitution of the row parameters into the general-purpose procedure.

In this article we consider some principles of organization and construction of tunable binary procecures, parameter tables, and cyclic binary programs for SBF evaluation. Appropriate complexity bounds are given.

1. Tunable Binary Procedures

Consider a SBF of the form

$$W_1=Z_1\vee Z_2\vee Z_3;\quad W_2=Z_1Z_2\vee Z_3;$$
$$W_3=(Z_1\vee Z_2)Z_3;\quad W_4=Z_1Z_2Z_3. \tag{1}$$

Figure 1 presents graph-schemas (GS) of simple binary programs realizing these formulas. Let us combine the GS of the separate formulas into a generalized GS (Fig. 2a), which constitutes a simple binary program with four inputs and one output (Fig. 2b). To reduce the number of inputs, we introduce the additional variables Z_4 and Z_5, which are called formula-tuning parameters: $Z_4 = Z_5 = 1$ tune to W_1, $Z_4 = 1$ and $Z_5 = 0$ tune to W_2, $Z_4 = 0$ and $Z_5 = 1$ tune to W_3, and $Z_4 = Z_5 = 0$ tune to W_4. Supplementing the GS with three nodes (Fig. 2c), we obtain the GS of a binary program which constitutes a procedure evaluating any of the formulas in (1); this procedure may be utilized as a subprogram with the parameters Z_4 and Z_5. It may be designed as a function subprogram since computations with any of the formulas generate the value of W equal to 1 (0). The resulting function procedure (Fig. 2c) will be termed a tunable binary procedure (TBP), which we denote by $W = W(Z_4, Z_5)$. Its tuning involves assigning the constant values 1 and 0 to the parameters Z_4 and Z_5. The formulas (1) are then evaluated using the following relationships:

$$W_1=W(1,1);\quad W_2=W(1,0);\quad W_3=W(0,1);\quad W_4=W(0,0). \tag{2}$$

The GS in Fig. 2c realizes the function

Leningrad. Translated from Avtomatika i Telemkhanika, No. 11, pp. 112-119, November, 1984. Original article submitted November 10, 1983.

Automation and Remote Control. Vol. 45, No 11, Part 2, November, 1984

0005-1179/84/4511-1481$08.50

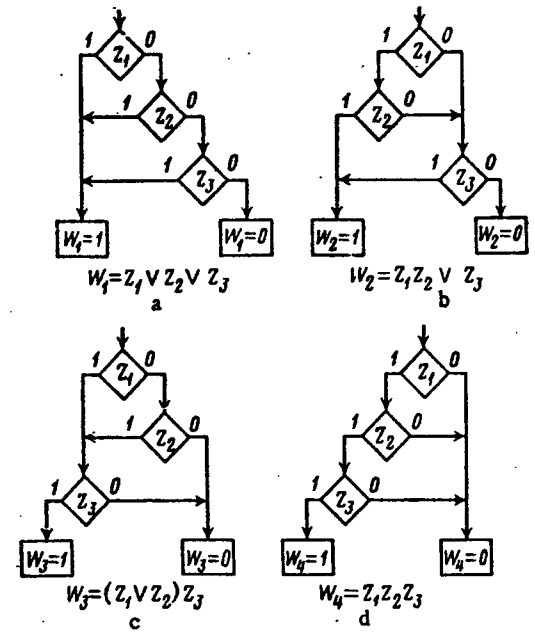

Fig. 1

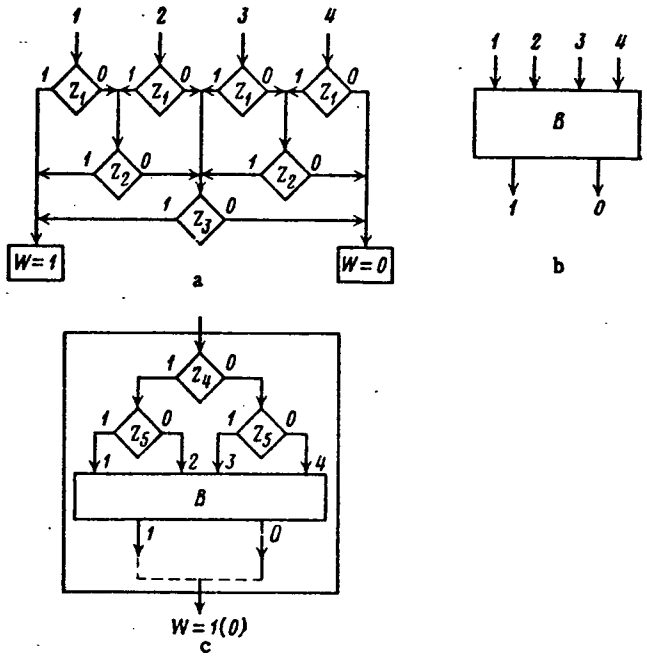

Fig. 2

$$W=Z_1Z_3(Z_1 \vee Z_2 \vee Z_3) \vee Z_1\bar{Z}_3(Z_1Z_2 \vee Z_3) \vee \vee \bar{Z}_1Z_3Z_3(Z_1 \vee Z_2) \vee \bar{Z}_1\bar{Z}_3Z_1Z_2Z_3, \quad (3)$$

which, by analogy with [4, 5], will be called the generating function (GF). Since (1) is a system of representatives for all the PN-types of repetition-free three-letter normal formulas in the basis {&, $\vee$, $^-$} [4], the binary procedure may be used to evaluate arbitrary three-leter normal Boolean formulas in this basis [4] by substituting the relevant letters from the target formula for the variables Z_1, Z_2, Z_3 in the function (3) and the constants 1 and 0 for the variables Z_4 and Z_5. For example, to evaluate the formula $Y=\bar{X}_1X_3 \vee X_2$ we may use the relationship W_2 from (2), taking $Z_4 = 1$, $Z_5 = 0$. We then substitute $Z_1 = X_1$, $Z_2 = X_3$, $Z_3 = \bar{X}_2$. The function (3) is thus a TBP with the parameters $Z_1, \ldots, Z_5$, i.e., in general $W = W(Z_1, \ldots, Z_5)$. By analogy with [4], the parameters Z_1, Z_2, Z_3 are called informational parameters, and Z_4, Z_5 tuning parameters. The TBP GF also may be represented by a function of the form

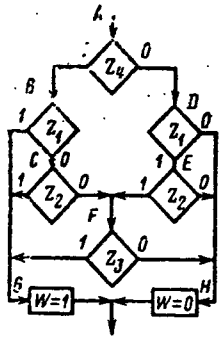

Fig. 3

$$W=W(Q, P, R), \tag{4}$$

where Q is the set of informational parameters, P is the set of tuning parameters, R is the set of informational-tuning parameters [4].

A K-universal TBP (TBP-K) is a TBP realizing a GF whose component formulas constitute a collection of representatives for all the PN-types of repetition-free normal formulas in the basis {&, $\vee$, $^-$} with K or fewer letters (informational parameters of the TBP).

The function (3) is the GF of TBP-3 with three informational and two tuning parameters. The GF of TBP-3 with partially overlapping informational and tuning parameters may be represented, using the method of [4], in the form

$$W=(Z_1 \vee Z_2 \vee Z_3)Z_4 \vee Z_1 Z_2 Z_3. \tag{5}$$

The PN-type representative of the formula W_1 from (1) is realized in this case taking $Z_4 = 1$, the representative of W_2 is realized for $Z_3 = 1$, that of W_3 for $Z_3 = 0$, and of W_4 for $Z_4 = 0$. The variables Z_1 and Z_2 are informational variables, and the variables Z_3 and Z_4 alternate as informational and tuning parameters. The GS of the corresponding TBP-3 is shown in Fig. 3. Note that it also may be realized as a linear GS with seven condition nodes and two operator nodes, using the formula method.

2. Generating the TBP

The starting data for TBP generation are the formulas of the given SBF, and the output is a procedure code. Let us consider the following steps leading to the generation of a TBP: choosing the base SBF, constructing the TBP GF, constructing the TBP GS, coding the TBP.

2.1. Choosing the Base SBF. The base SBF is the SBF realized by the TBP whose arguments are the informational (informational-tuning) parameters of the TBF. If a concrete unit (logical machine) is intended for program realization of a single SBF, this SBF is the base SBF for a special-purpose TBP. If the machine is intended for program realization of various formulas, which are not prespecified in advance, it is advisable to choose the base SBF as a system consisting of all the PN-type representatives of repetition-free normal formulas in the basis {&, $\vee$, $^-$} with K or fewer letters.

2.2. Constructing the TBP GF. The TBP GF is constructed following [4]. GF construction involves filling the TBP tuning table. The tuning table of the TBP realizing the GF (5) is shown in Table 1.

2.3. Constructing the TBP GS. The construction of the TBP GS from the GF is a combinatorial problem, usually solved by heuristic methods [2, 6-8]. Once the GS is constructed, its nodes are labeled.

2.4. Coding the TBP. The TBP is implemented in a language whose operators mostly coincide with those used for the description of binary programs [1]. It contains the following instruction set:

1. Operators for two- and one-address conditional jumps:

M: if X = 1(0) then M_1, else M_2;

M: if X = 1(0) then M_1;

TABLE 1. Tuning of TBP-3

Formula number	SBF formula (5)	Arithmetic polynomial	Tuning	
			Z_a	Z_b
1	$Z_1 \vee Z_2 \vee Z_3$	1+1+1	Z_3	1
2	$Z_1 Z_2 \vee Z_4$	2+1	1	Z_4
3	$(Z_1 \vee Z_2) Z_4$	(1+1)1	0	Z_4
4	$Z_1 Z_2 Z_3$	3	Z_3	0

2. Unconditional jump operator:

M: go to M_1;

3. Function header operator:

name: begin (parameter-1, ..., parameter-m), where "name" is the TBP name, "parameter-i" is a TBP parameter, m is the number of variables in the TBP GF.

4. Assignment operators for functions:

M: name = 1(0), end;

These operators terminate the procedure and its value is set equal to 1 or 0 (since it is a function procedure).

5. Procedure end operator:

End name;

The operators M, M_1, and M_2 specify operator or jump labels. When two-address conditional jump operators are used, the TBP is coded starting with the GS root node, which is followed in arbitrary order by two-address conditional jump operators with labels corresponding to GS labels. If only single-address jumps are allowed, the transition from the GS to TBP code is essentially more difficult and it should be considered as a problem of minimizing the number of unconditional jumps [7]. For TBP-3 with the GS shown in Fig. 3, a procedure using only single-address conditional jumps has the following form:

```
TBP-3:  begin (Z1, Z2, Z3, Z4);
A:  if Z4 = 0, then D; G:  TBP-3 = 1, end;
B:  if Z1 = 1, then G; D:  if Z1 = 0, then H;
C:  if Z2 = 1, then G; E:  if Z2 = 1, then F;
F:  if Z3 = 0, then H; H:  TBP-3 = 0, end;
                           End TBP-3;
```
(6)

3. Decomposition of a Given SBF

TBP-K can be used to realize arbitrary formulas in the basis $\{\&, \vee, ^-\}$. To this end we have to decompose each formula in the basis of TBP-K formulas. The decomposition process is analogous to the process of formula realization on K-universal modules, described in [4, 9]. For example, given a SBF of the form

$$\begin{aligned} Y_1 &= X_1(X_3 \vee X_4) \vee \overline{X}_1(X_2 X_2 \vee \overline{X}_2(\overline{X}_3 \vee \overline{X}_4)); \\ Y_2 &= X_1 \vee X_2; \\ Y_3 &= X_2(\overline{X}_1 X_3 X_4 \vee \overline{X}_1 X_3 X_5 \vee X_1 X_4 X_5) \vee \overline{X}_2 \overline{X}_3 X_5 \vee \overline{X}_2 X_4, \end{aligned} \tag{7}$$

we can represent Y_1 from (7) by the following decomposition into TBP-3 basic formulas:

$$Y_1 = X_1(X_3 \vee X_4) \vee \overline{X}_1(X_2 X_3 \vee \overline{X}_2(\overline{X}_3 \vee \overline{X}_4)) = V_1 \vee \overline{X}_1(X_2 X_3 \vee V_2) = V_1 \vee \overline{X}_1 V_3 = V_4, \tag{8}$$

where

$$\begin{aligned} V_1 &= X_1(X_3 \vee X_4), \quad V_2 = \overline{X}_2(\overline{X}_3 \vee \overline{X}_4), \\ V_3 &= X_2 X_3 \vee V_2, \quad V_4 = V_1 \vee \overline{X}_1 V_3. \end{aligned}$$

Here V_1 and V_2 are the fragments of the original formula realized by the formula W_3 from (1), and V_3 and V_4 are the fragments realized by the formula W_2 from (1) by appropriate substitution. The decomposition process is accompanied by filling a decomposition table.

TABLE 2. SBF Decomposition Table

Decomposition step index	Formula fragment	Type of SBF basic formula	Substitution of TBP parameters (inputs)				The realized fragment (output)
			z_1	z_2	z_3	z_4	
1	2	3	4	5	6	7	8
1	$X_1(X_2 \vee X_4)$	$(1+1)1$	X_2	X_4	0	X_1	V_1
2	$(\overline{X}_2 \vee \overline{X}_4)\overline{X}_2$	$(1+1)1$	$\overline{X}_2$	$\overline{X}_4$	0	$\overline{X}_2$	V_2
3	$X_2X_3 \vee V_2$	$2+1$	X_2	X_3	1	V_2	V_3
4	$\overline{X}_1V_3 \vee V_1$	$2+1$	$\overline{X}_1$	V_3	1	V_1	Y_1
5	$(X_1 \vee X_2)1$	$(1+1)1$	X_1	X_2	0	1	Y_2
6	$\overline{X}_2X_3X_4$	3	$\overline{X}_1$	X_3	X_4	0	V_1
7	$\overline{X}_1\overline{X}_3X_5$	3	$\overline{X}_1$	$\overline{X}_3$	X_5	0	V_2
8	$X_1X_4X_5$	3	X_1	X_4	X_5	0	V_3
9	$V_1 \vee V_2 \vee V_3$	$1+1+1$	V_1	V_2	V_3	1	V_4
10	$\overline{X}_2\overline{X}_3X_5$	3	$\overline{X}_2$	$\overline{X}_3$	X_5	0	V_5
11	$\overline{X}_3X_4 \vee V_5$	$2+1$	$\overline{X}_3$	X_4	1	V_5	V_4
12	$X_2V_4 \vee V_6$	$2+1$	X_2	V_4	1	V_6	Y_3

TABLE 3. The Array S for SBF

Array row index	TBP inputs				Output (S_{i5})	Array row index	TBP inputs				Output (S_{i5})
	z_1 (S_{i1})	z_2 (S_{i2})	z_3 (S_{i3})	z_4 (S_{i4})			z_1 (S_{i1})	z_2 (S_{i2})	z_3 (S_{i3})	z_4 (S_{i4})	
1	X_3	X_4	0	X_1	V_1	7	$\overline{X}_1$	$\overline{X}_3$	X_5	0	V_2
2	$\overline{X}_3$	$\overline{X}_4$	0	$\overline{X}_2$	V_2	8	X_1	X_4	X_5	0	V_3
3	X_2	X_3	1	V_3	V_3	9	V_1	V_2	V_3	1	V_4
4	$\overline{X}_1$	V_3	1	V_1	Y_1	10	$\overline{X}_2$	$\overline{X}_3$	X_5	0	V_5
5	X_1	X_2	0	1	Y_2	11	$\overline{X}_3$	X_4	1	V_5	V_4
6	$\overline{X}_1$	X_3	X_4	0	V_1	12	X_2	V_4	1	V_6	Y_3

Table filling can be approximately described in the following form. Identify a separable fragment in the original formula (or in the formula obtained in the preceding decomposition step) [4]. Using the TBP-K tuning table identify the basic formula. In the next row of the decomposition table enter the step index, the realized fragment of the original or the previously transformed formula, the arithmetic polynomial of the TBP basic formula, the variables and the constants determined by the corresponding row in the TBP tuning table (Table 1), and also the symbol of the fragment or the output variable realized in the particular step. The fragment is then considered as a new letter (intermediate variable) which is saved in the main memory. Table 2 is a decomposition table for the SBF (7).

4. Cyclic Binary Programs

The simplest technique for designing a program that will realize a given SBF by using a TBP involves writing out a sequence of assignment operators in which the left-hand parts are an intermediate or an output variable and the right-hand part is the TBP name with the parameters listed in the corresponding row of the decomposition table. For example, a program realizing the SBF (7) using TBP-3 with Table 2 has the form

1. $V_1 = \text{TBP-}3(X_3, X_4, 0, X_1)$;
.
.
.
4. $Y_1 = \text{TBP-}3(\overline{X}_1, V_3, 1, V_1)$;
.
.
.
12. $Y_3 = \text{TBP-}3(X_2, V_4, 1, V_6)$.

A sequence of identical assignment operators may be replaced with a loop, in which the loop parameter (i) is the row index in the decomposition table. The input and the output

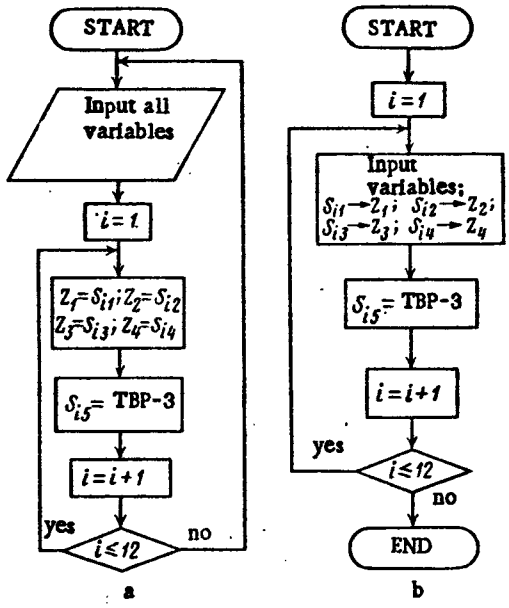

Fig. 4

(intermediate) variables are then moved outside the program and are represented by an array (Table 3), which is henceforth denoted by S. This array is generated from Table 2 by deleting columns 2 and 3.

Figure 4 shows flowcharts of SBF-realization programs using TBP. In Fig. 4a, the SBF is evaluated infinitely many times, which is usually required under program control, and all the input values are entered simultaneously for each SBF realization before the first row of the array S is processed. In Fig. 4b, the SBF is realized only once, and for each loop execution we only enter the variables corresponding to the row i in the array S.

As an example let us evaluate the formula Y_1 from the SBF (7) using the program from Fig. 4a. Let $X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 0$, and $X_5 = 0$; then for i = 1 from the first row in Table 3 we get $Z_1 = 1$, $Z_2 = 0$, $Z_3 = 0$, $Z_4 = 1$. Then, following the code of the procedure TBP-3 we obtain the sequence of jumps A, B, G (identified by the labels in the procedure (6)) which generate $V_1 = 1$. For i = 2: $Z_1 = 0$, $Z_2 = 1$, $Z_3 = 0$, $Z_4 = 0$; the jumps are A, D, H and $V_2 = 0$. For i = 3: $Z_1 = 1$, $Z_2 = 1$, $Z_3 = 1$, $Z_4 = 0$; the jumps are A, D, E, F, G and $V_3 = 1$. For i = 4: $Z_1 = 0$, $Z_2 = 1$, $Z_3 = 1$, $Z_4 = 1$; the jumps are A, B, C, G and $Y_1 = 1$.

The other formulas in the SBF (7) are similarly evaluated.

Programs with loops using TBP to process arrays of this type will be called cyclic binary programs (CBP). The CBP using TBP-K is independent of the particular SBF being realized and in this sense is general-purpose.

CBP may be used with array S of infinite length, and yet it is always easy to recover the specified SBF and to modify it as necessary.

5. Complexity Bounds

The CBP occupies a fixed space in memory and is characterized by a virtually constant loop execution time t_c. The memory required to store the array S is given by

$$L_s=(m+1)L, \tag{9}$$

where m is the number of variables in the TBP GF; L is the number of rows in S (the number of decomposition steps). From [5] we thus have

$$m=K+\left\lceil \log_2\left(2+\sum_{i=\lceil K/2\rceil}^{K-1} Q(j)\right)-1\right\rceil, \tag{10}$$

where Q(j) is the number of PN-types of repetition-free j-letter normal formulas in the basis $\{\&, \vee, \neg\}$ [4].

By analogy with [4, 9] we have the bounds

$$\left]\frac{h-1}{K-1}\right[\leqslant L\leqslant\left]\frac{2(h-1)}{K}\right[, \tag{11}$$

$$\left]\frac{H-N}{K-1}\right[\leqslant L\leqslant\left]\frac{2(H-N)}{K}\right[+N-1, \tag{12}$$

where L is the number of SBF decomposition steps, h is the number of letters in the right-hand part of a formula, N is the number of formulas in the SBF, H is the total number of letters in the right-hand parts of the SBF,

$$H=\sum_{i=1}^{N}h_i.$$

L is thus linearly dependent on the number of letters in the given SBF. Here

$$T=t_c L, \tag{13}$$

where T is the time to process the array S once.

Therefore T is also linearly dependent on the number of letters in the SBF being realized.

CONCLUSIONS

1. A new class of programs is proposed for the realization of SBF. These are so-called cyclic binary programs (CBP) in which the body of a loop includes a tunable binary procedure (TBP) capable of realizing any of the formulas in the base SBF when the values of the tuning parameters are specified.

2. The SBF is prepared for realization by stepwise decomposition into the formulas of a K-universal TBP, which involves constructing a decomposition table and generating from it a parameter array whose rows are then processed in strict sequential order by the CBP. This ensures SBF realization with linear complexity bounds, while preserving readability and ease of SBF modification.

3. CBP of this type may be used to realize arbitrary SBF specified in the basis $\{\&, \vee, \bar{}\}$ in accordance with the above-given bounds. This approach, however, is more effective for the realization of weakly connected formulas with infrequent repetition of variables which is the common case in industrial automation systems [4].

4. The proposed approach may be used to construct CBP with TBP which will act as general-purpose programs for some classes of formulas and functions. If we use a general-purpose TBP on the class of DNF, then we obtain the bounds derived in [4]. If we use a general-purpose TBP on the class of K-letter formulas in the basis $\{\&, \vee, \bar{}, \oplus\}$ then the corresponding CBP will realize with the above bounds arbitrary formulas and systems of formulas in the basis $\{\&, \vee, \bar{}, \oplus\}$. The upper bounds from (11) and (12) are also valid for CBP which use general-purpose TBP on the class of arbitrary functions of K variables.

LITERATURE CITED

1. O. P. Kuznetsov, "Program realization of logical functions and automata," Avtomat. Telemekh., No. 7, 163-174 (1977).
2. V. L. Artyukhov, V. I. Rubinov, and A. A. Shalyto, "Constructing generalized binary programs for systems of Boolean functions," in: Issues of Ship Building, series "Naval Automation" [in Russian], No. 27, Leningrad (1982), pp. 38-46.
3. E. I. Pupyrev, "Interpreters realizing Boolean functions and automata," Avtomat. Telemekh., No. 1, 132-140 (1982).
4. V. L. Artyukhov, G. A. Kopeikin, and A. A. Shalyto, Tunable Modules for Logical Circuit Control [in Russian], Energoizdat, Leningrad (1981).
5. V. L. Artyukhov, G. A. Kopeikin, and A. A. Shalyto, Naval Logical Control Systems. Unified Logical Circuits [in Russian], Izd. Inst. Povysheniya Kvalifikatsii Rukovodyashchikh Rabotnikov i Spetsialistov Sudostroitel'noi Promyshlennosti, Leningrad (1981).
6. E. Humby, Programs from Decision Teachers, American Elsevier (1973).

7. Yu. N. Butin, M. Ya. Zolotarevskaya, A. P. Kirillov, and V. N. Yung, "Realization of logical-control algorithms in special-purpose programmable logical units," Avtomat. Telemekh., No. 6, 131-141 (1983).
8. B. P. Kuznetsov, "Structured binary programs," in: Issues of Ship Building, series "Naval Automation" [in Russian], No. 29, Leningrad (1983), pp. 27-35.
9. V. A. Artyukhov, G. A. Kopeikin, and A. A. Shalyto, "Complexity bounds for the realization of Boolean formulas by tree networks of tunable modules," Avtomat. Telemekh., No. 11, 124-130 (1981).