STUDII ȘI CERCETĂRI MATEMATICE

EXTRAS

2

TOMUL 17
1965

EDITURA ACADEMIEI REPUBLICII POPULARE ROMÎNE

ASUPRA VERIFICĂRII FORMALE A SCHEMELOR LOGICE DE ALGORITMI

DE

L. NEGRESCU și I. PĂVĂLOIU

Institutul de calcul al Academiei R. P. R. — Filiala Cluj

În lucrare se dă un procedeu cu ajutorul căruia se pot determina toate subșirurile de forma :

$$B_i\, B_{i_1} B_{i_2} \ldots\ B_{i_s}\, B_j ,$$

unde B_i este un operator aritmetic, $B_{i_1} B_{i_2} \ldots B_{i_s}$ sînt condiții logice iar B_j este sau operator aritmetic sau condiție logică, care în plus mai satisface anumite proprietăți amintite în lucrare.

Cu ajutorul acestui procedeu s-au stabilit unele proprietăți ale schemelor logice de algoritmi privind verificarea lor formală.

În lucrare este prezentată și schema logică graf a algoritmului de aplicare automată a procedeului dat la o schemă logică de algoritmi arbitrară.

În lucrarea de față ne vom ocupa cu unele probleme legate de analiza formală a schemelor logice de algoritmi. În acest scop vom da un procedeu cu ajutorul căruia se poate ataşa oricărei scheme logice de algoritmi o schemă matricială.

Introducem mai jos cîteva noțiuni deja cunoscute [1] care ne vor servi la expunerea mai clară a rezultatelor obținute.

DEFINIȚIA 1. Se numește memorie o mulțime de obiecte G numite adrese.

DEFINIȚIA 2. Se numește stare a memoriei o corespondență de forma :

$$x = X(z),\ z \in G,\ x \in H,$$

unde H este o mulțime de obiecte pe care le numim stări.

Fie $Y = \{X\}$ mulțimea tuturor stărilor memoriei astfel definite.

DEFINIȚIA 3. Funcția $A(X) = X^*$ care transformă mulțimea Y în ea însăși se numește operator aritmetic.

DEFINIȚIA 4. Se numește predicat peste mulțimea Y orice funcție $p(X)$ care face ca oricărui element al mulțimii Y să-i corespundă unul din numerele 0 sau 1.

ST. CERC. MAT., TOM. 17, NR. 2, P. 271–286, BUCUREȘTI, 1965

DEFINIŢIA 5. Se numeşte funcţie logică peste mulţimea Y o funcţie de forma:

$$\alpha = \alpha(p_1, p_2, \ldots, p_k)$$

care face ca oricărui $X \in Y$ să-i corespundă unul din numerele 0 sau 1, iar $p_1, p_2, \ldots, p_{k-1}, p_k$ sînt predicate peste mulţimea Y.

DEFINIŢIA 6. Se numeşte condiţie logică orice pereche de forma (α, l), unde α este o funcţie logică şi l este un număr natural bine determinat pe care îl numim indice de transfer.

Notăm $\mathcal{A} = \{A\}$, $\mathcal{B} = \{(\alpha, l)\}$ şi fie $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$.

DEFINIŢIA 7. Se numeşte schemă logică de algoritmi un şir finit de elemente aparţinînd mulţimii $\mathcal{C}$ care îndeplineşte următoarele condiţii

1. Elementul cu indicele i este al i-elea termen din şir.

2. Dacă în schemă apar două elemente egale, atunci ele vor avea indici diferiţi potrivit ordinei lor în şirul considerat.

3. Indicele de transfer al oricărei condiţii logice din şirul respectiv nu întrece numărul termenilor şirului.

4. Dacă (α_i, l) este o condiţie logică care apare în şir, atunci $l \neq i+1$ şi $l \neq i$, unde i este indicele de ordine al acestui element în şirul considerat.

Aici, prin egalitatea elementelor mulţimii $\mathcal{C}$ înţelegem următoarele:

Dacă A_1 şi A_2 sînt doi operatori aritmetici egali, atunci oricare ar fi starea memoriei X avem $A_1(X) = A_2(X)$ iar (α_1, l_1) şi (α_2, l_2) sînt două condiţii logice egale dacă $l_1 = l_2$ şi $\alpha_1(X) = \alpha_2(X)$ oricare ar fi starea memoriei X şi invers.

Fie

$$B_1 B_2 \ldots B_{n-1} B_n \tag{1}$$

o schemă logică de algoritmi. Fără a restrînge generalitatea, putem presupune că B_1 este operatorul aritmetic care semnalizează începutul realizării schemei iar B_n este operatorul aritmetic care semnalizează sfîrşitul realizării schemei.

Descriem acum procesul de realizare a schemei (1) prin recurenţă.

Fie X_0 o stare oarecare a memoriei pe care o numim stare iniţială. Începem executarea schemei (1) cu executarea operatorului B_1. Fie $X_1 = B_1(X_0)$ şi fie că în procesul de executare a schemei am făcut l paşi în urma cărora au fost executaţi pe rînd operatorii aritmetici

$$B_1, B_{k_1}, \ldots, B_{k_s} \tag{1'}$$

şi fie X_{s+1} starea memoriei la această etapă. Atunci la pasul $l+1$ se analizează elementul considerat la pasul l şi după cum el este un operator aritmetic sau o condiţie logică efectuăm următoarele operaţii:

a) Dacă elementul executat la pasul l este un operator aritmetic, atunci se analizează elementul care urmează după el şi se execută.

b) Dacă elementul executat la pasul l este o condiţie logică de forma (α, t), atunci în cazul cînd $\alpha(X_{s+1}) = 1$ analizăm elementul imediat următor şi îl executăm. Iar în cazul cînd $\alpha(X_{s+1}) = 0$, analizăm elementul cu indicele t şi îl executăm. Astfel continuînd procesul de execuţie al schemei (1) se pot întîmpla două cazuri:

1° Procesul de executare al schemei (1) se sfîrşeşte cu executarea operatorului B_n.

2° Procesul de executare al schemei (1) continuă indefinit.

Din descrierea procesului de executare a schemei (1) rezultă imediat că după executarea unui oarecare operator aritmetic B_l, se execută cel mult un operator aritmetic şi numai acesta.

În cele ce urmează va fi util să punem în evidenţă toate subşirurile ataşate schemei (1) de forma:

$$B_i B_{i_1} B_{i_2} \ldots B_{i_s} B_j, \qquad (s \geqslant 0,\ i_1 = i+1) \tag{2}$$

cu următoarele proprietăţi:

1) B_i este un operator aritmetic şi dacă $B_j = B_{i+1}$, atunci B_{i+1} este operator aritmetic.

2) $B_{i_1}, \ldots, B_{i_s}$, $(s > 0)$ sînt condiţii logice astfel încît dacă $B_{i_{l+s}}$ este condiţia logică care urmează după B_{i_l} în subşirul (2), atunci sau $i_{l+1} = i_l + 1$ sau i_{l+4} este indicele de transfer al elementului B_{i_l} după cum în acest element funcţia logică apare ca şi în schema (1) sau apare negaţia ei.

3) B_j este un operator aritmetic dacă elementele subşirului B_{i_1}, $B_{i_2}, \ldots, B_{i_s}$, $(s > 0)$ au proprietatea că oricare ar fi un element din acest subşir B_{j_ν} cu funcţia sa logică α_{j_ν}, atunci atît această funcţie cît şi negaţia ei nu mai apar în elementele acestui subşir.

4) Dacă B_j este o condiţie logică cu funcţia logică α_j, atunci neapărat există în subşirul $B_{i_1}, B_{i_2}, \ldots, B_{i_s}$, $(s > 0)$ un indice i_μ astfel ca $i_\mu = j$. Evident, dacă notăm cu β_{i_μ} funcţia logică din elementul B_{i_μ} a subşirului (2), atunci între β_{i_μ} şi α_j avem relaţiile $\beta_{j_\mu} = \alpha_j$ sau $\beta_{i_\mu} = \bar{\alpha}_j$. Dăm mai jos un procedeu de obţinere al tuturor subşirurilor de forma (2).

Fie B_i, $0 < i < n$ un oarecare operator aritmetic din şirul (1). Începem analiza primului element care urmează după B_i, adică analizăm elementul B_{i+1}. Se pot întîmpla două cazuri.

1) B_{i+1} este un operator aritmetic; atunci subşirul (2) are forma:

$$B_i B_{i+1}, \quad (s = 0,\ j = i+1). \tag{3}$$

2) B_{i+1} este o condiţie logică; atunci ea se scrie la dreapta lui B_i împreună cu toate condiţiile logice care urmează la dreapta ei în şirul (1) pînă cînd întîlnim primul operator aritmetic în acest şir. Obţinem astfel subşirul:

$$B_i B_{i+1} B_{i+2} \ldots B_{i+k} B_t \qquad (t = i+k+1), \tag{4}$$

unde B_t este primul operator aritmetic care se află la dreapta lui B_t în şirul (1).

Din subşirul (4) se formează un şir de subşiruri în felul următor:

Se scrie operatorul aritmetic B_i şi la dreapta lui condiţia logică B_{i+1} înlocuind funcţia logică α_{i+1} cu $\bar{\alpha}_{i+1}$. După acest element scriem elementul din şirul (1) cu indicele egal cu indicele de transfer al lui B_{i+1}.

Dacă acest element este un operator aritmetic, atunci se obţine al doilea termen al şirului căutat. Dacă însă acest element este o condiţie logică, atunci scriem la dreapta lui toate elementele care urmează după el în şirul (1), pînă cînd întîlnim un operator aritmetic şi astfel obţinem al doilea termen al şirului căutat. Pentru a obţine al treilea termen procedăm analog cu elementul B_{i+2} al subşirului (4). Procedăm în acest fel pînă cînd epuizăm toate condiţiile logice din subşirul (4). După aceasta considerăm al doilea termen al şirului construit pînă aici şi procedăm cu el ca şi cu subşirul (4) începînd cu prima condiţie logică a cărei funcţie nu a fost încă barată, dacă există o astfel de condiţie. În cazul cînd nu există o astfel de condiţie, se trece la termenul următor din şir. În general dacă presupunem că termenul care trebuie prelucrat din şir este :

$$B_i B_{i+1} B_{k_2} \dots B_{k_p} (\bar{\alpha}_{k_p+1,\, i_p+1}) (\alpha_{i_p+1,\, s_1}) \dots (\alpha_{i_p+q,\, s_q}) B_{i_p+q+1},$$

unde $B_{i+1} B_{k_2} \dots B_{k_p}$ sînt condiţii logice din şirul (1), care eventual pot să conţină şi negaţiile funcţiilor logice, atunci din el se obţin următorii termeni ai şirului :

$$B_i B_{i+1} B_{k_2} \dots B_{k_p} (\bar{\alpha}_{k_p+1,\, i_p+1}) (\bar{\alpha}_{i_p+1,\, s_1}) (\alpha_{s_1}, u_1) \dots (\alpha_{s_1+v}, u_{v+1}) B_{s_1+v+1}$$

$$B_i B_{i+1} B_{k_2} \dots B_{k_p} (\bar{\alpha}_{k_p+1,\, i_p+1}) (\alpha_{i_p+1,\, s_1}) (\bar{\alpha}_{i_p+2,\, s_2}) (\alpha_{s_2,\, t_2}) \dots$$

$$\dots (\alpha_{s_2+\mu,\, t_\mu+2}) B_{s_2+\mu+1} \qquad (5)$$

$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$

$$B_i B_{i+1} B_{k_2} \dots B_{k_p} (\bar{\alpha}_{k_p+1, i_p+1}) (\alpha_{i_p+1,\, s_1}) \dots (\bar{\alpha}_{i_p+q\ s_q}) (\alpha_{s_q,\, l_q}) \dots$$

$$(\alpha_{s_q+v,\, l_q+v}) B_{s_q+v+1}$$

Fiecare dintre $B_{s_1+1}, B_{s_2+\mu+1}, \dots, B_{s_q+v+1}$ este sau un operator aritmetic, în cazul cînd toţi indicii care apar în subşirul corespunzător lui din (5) sînt diferiţi între ei, exceptînd indicele primului element din acest şir şi indicele ultimului element din acelaşi subşir, care pot fi şi egali, sau este o condiţie logică dacă şi numai dacă în subşirul corespunzător din (5) există un indice diferit de i şi egal cu indicele lui. Acest procedeu continuă pînă cînd el nu se mai poate aplica nici unui termen din şir, considerînd că dacă un termen din şir s-a prelucrat complet, atunci el nu se mai prelucrează. Aici trebuie să subliniem că la prelucrarea unui subşir în care ultimul element este o condiţie logică, funcţia sa logică nu se barează. Pentru a uşura înţelegerea procedeului descris aici, dăm mai jos un exemplu.

Fie dată schema logică de algoritmi :

$$A_1(\alpha_2, 11)\ (\alpha_3, 8)\ (\alpha_4, 11)\ A_5(\alpha_6, 11)\ (\alpha_7, 11)\ (\alpha_8, 11)\ (\alpha_9, 6)\ (\alpha_{10}, 5)\ A_{11} A_{12}.$$

Începem procedeul cu operatorul A_1. Primul termen al şirului este :

$$A_1(\alpha_2, 11)(\alpha_3, 8)\ (\alpha_4, 11)\ A_5 \qquad (1')$$

Din acest termen se obţin următorii termeni ai şirului :

$$A_1(\bar{\alpha}_2, 11)\ A_{11}, \qquad (2')$$

$$A_1(\alpha_2, 11)(\bar{\alpha}_3, 8)(\alpha_8, 11)(\alpha_9, 6)(\alpha_{10}, 5) A_{11} \qquad (3')$$

$$A_1(\alpha_2, 11)(\alpha_3, 8)(\bar{\alpha}_4, 11)\ A_{11} \qquad (4')$$

Astfel primul termen al şirului a fost prelucrat. Se observă că celui de-al doilea termen nu i se mai poate aplica procedeul. Prelucrăm atunci termenul al treilea şi obţinem următorii termeni din şir :

$$A_1(\alpha_2, 11)(\bar{\alpha}_3, 8)\ (\bar{\alpha}_8, 11) A_{11}, \qquad (5')$$

$$A_1(\alpha_2, 11)(\bar{\alpha}_3, 8)\ (\alpha_8, 11)(\bar{\alpha}_9, 6)(\alpha_6, 11),\ (\alpha_7, 11)(\alpha_8, 11) \qquad (6')$$

$$A_1(\alpha_2, 11)(\bar{\alpha}_3, 8)(\alpha_8, 11)(\alpha_9, 6)(\bar{\alpha}_{10}, 5)\ A_5. \qquad (7')$$

Astfel a fost prelucrat şi termenul al treilea al şirului căutat. Evident, termenilor (4') şi (5') nu li se mai poate aplica procedeul.

Prelucrăm acum termenul (6') şi obţinem pe rînd termenii

$$A_1(\alpha_2, 11)(\bar{\alpha}_3, 8)(\alpha_8, 11)(\bar{\alpha}_9, 6)(\bar{\alpha}_6, 11) A_{11}, \qquad (8')$$

$$A_1(\alpha_2, 11)(\bar{\alpha}_3, 8)(\alpha_8, 11)(\bar{\alpha}_9, 6)(\alpha_6, 11)(\bar{\alpha}_7, 11)\ A_{11}. \qquad (9')$$

Se observă că procedeul nu se mai aplică la nici unul din termenii (7'), (8') şi (9') care au mai rămas. Cu aceasta au fost puse în evidenţă toate şirurile de forma (2) relativ la schema logică dată pentru operatorul aritmetic A_1.

Este evident că în subşirurile şirului obţinut cu acest procedeu dacă o condiţie logică apare o dată într-un anumit subşir, atunci există cel puţin un termen din acest şir care conţine această condiţie cu negaţia funcţiei logice respective. În cele ce urmează ne vom referi la acest procedeu sub denumirea de procedeul P.

TEOREMA 1. *Procedeul P se termină într-un număr finit de paşi.*

Demonstraţie. Fie

$$B_{k_1}, B_{k_2}, \dots B_{k_l} \qquad (6)$$

toate condiţiile logice ale schemei (1) şi fie

$$C_{k_1}, C_{k_2} \dots C_{k_l}, \qquad (7)$$

condiţiile logice obţinute din şirul (6) prin negarea funcţiilor logice conţinute în elementele sale.

Reunind şirurile (6) şi (7) şi renumerotînd obţinem următorul şir :

$$L_1, L_2, \dots, L_{2l}. \qquad (8)$$

Este evident că termenii şirului obţinut în cursul aplicării procedeului P care au ca ultim element un operator aritmetic sînt astfel încît ei conţin o combinaţie de elemente din şirul (8) în care fiecare element apare o singură dată. Dacă un termen al şirului obţinut prin procedeul P are ca ultim element o condiţie logică, atunci el se înlocuieşte cu subşirul obţinut din acest termen dacă se omite ultima condiţie logică. Aceasta nu modifică cu nimic numărul de paşi în procedeul P. Prin aceasta se ajunge ca şi în

acest termen toate elementele să fie distincte. Cu un număr maxim de combinaţii pe care le putem face cu elementele şirului (8), ţinînd cont de observaţiile de mai sus, nu întrece pe 2^{2t}, rezultă că procesul trebuie să se termine într-un număr finit de paşi $m \leqslant 2^{2t}$ relativ la fiecare operator aritmetic.

TEOREMA 2. *Orice subşir pus în evidenţă de procedeul P este de forma (2) şi invers.*

Demonstraţie. Fie

$$B_i B_{i_1} B_{i_2} \ldots B_{i_{l-1}} B_{i_l} \tag{9}$$

un subşir oarecare pus în evidenţă de procedeul P, unde B_i este un operator aritmetic. Dacă $i_t = i + 1$, atunci B_{i_t} este un operator aritmetic şi $B_i B_{i+t}$ are forma (2) pentru $s = 0$. Fie $i_t \neq i + 1$ şi avem două cazuri după cum B_{i_t} este operator aritmetic sau condiţie logică. Presupunem că B_{i_t} este o condiţie logică. Atunci între elementele $B_{i_1}, B_{i_2} \ldots B_{i_{t-1}}$ există un oarecare element B_{i_v} pentru care $i_v = i_t$.

Rămîne acum să arătăm că subşirul (9) satisface şi proprietatea 2) a subşirului (2). Într-adevăr, subşirul (9) fiind pus în evidenţă de procedeul P, are proprietatea că după o oarecare condiţie logică B_{i_q} care conţine funcţia logică aşa cum apare în schema (1), urmează elementul $B_{i_{q+1}}$, adică $i_{q+1} = i_q + 1$, iar dacă B_{i_q} conţine negaţia acestei funcţii logice, atunci după B_{i_q} urmează elementul cu indicele egal cu indicele de transfer al lui B_{i_q}. Rezultă că în acest caz şirul (9) este de forma (2). În mod analog se arată că subşirul (9) în care B_{i_t} este operator aritmetic, este de forma (2). Invers, fie

$$B_i B_{k_1} B_{k_2} \ldots B_{k_l}, \ (l > 1) \tag{10}$$

un subşir de forma (2), unde $B_i B_{k_1} \ldots B_{k_l}$ aparţin schemei (1) şi considerăm cazul cînd imediat în dreapta lui B_i urmează condiţia logică B_{i+1}. Deoarece şirul (10) este de forma (2), rezultă că $k_1 = i + 1$.

Considerăm mulţimea tuturor subşirurilor puse în evidenţă de procedeul P care încep cu B_i. Notăm această mulţime cu $\mathfrak{L}_i^0$. Vom arăta că şirul (10) aparţine lui $\mathfrak{L}_i^0$.

Fie $\mathfrak{L}_i^0 = \mathfrak{L}_i^{01} \cup \mathfrak{L}_i^{02}$, unde $\mathfrak{L}_i^{01}$ este mulţimea tuturor elementelor din $\mathfrak{L}_i^0$, în care condiţia logică B_{k_1} conţine funcţia logică la fel ca în schema (1) iar $\mathfrak{L}_i^{02}$ este mulţimea tuturor elementelor din $\mathfrak{L}_i^0$ în care elementul B_{k_1} conţine negaţia funcţiei logice respective. Deoarece $k_1 = i + 1$, urmează că primele două elemente din şirul (10) coincid cu primele două elemente ale tuturor subşirurilor uneia din clasele $\mathfrak{L}_i^{01}$ sau $\mathfrak{L}_i^{02}$.

Continuînd astfel să presupunem că am construit clasa $\mathfrak{L}_i^{0,\varepsilon_{k_1},\varepsilon_{k_2},\ldots,\varepsilon_{k_p}}$, unde :

$$\varepsilon_{k_i} = \begin{cases} 1) \text{ dacă funcţia logică din condiţia } B_{k_i} \text{ apare în termenii acestei clase la fel ca şi în schema (1).} \\ 2) \text{ dacă funcţia logică din condiţia } B_{k_i} \text{ care apare în termenii acestei clase este negaţia funcţiei logice care apare în elementul respectiv din schema (1).} \end{cases}$$

$$i = 1, \ldots, p, \ 1 \leqslant p \leqslant l - 1,$$

şi că primele p+1 elemente din subşirurile acestei clase coincid cu primele $p + 1$ elemente din subşirul (10). Din această clasă vom construi două clase disjuncte astfel : $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_p},1}$ este o submulţime a acestei mulţimi în care elementul al $p = 2$ -lea al fiecărui subşir din ea conţine funcţia logică aşa cum apare în schema (1) în acel element, iar $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_p},2}$ este complimentara ei faţă de $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_p}}$.

Să arătăm că mulţimile $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_p},1}$ şi $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_p},2}$ nu sînt vide în ipoteza că $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_p}}$ nu este vidă şi $1 \leqslant p \leqslant l - 2$. În clasa $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_p}}$ există un subşir ale cărui condiţii logice începînd cu condiţia logică B_{p+1} conţin toate funcţiile logice aşa cum apar ele în schema (1). Aceasta urmează din descrierea procedeului P. Acest subşir aparţine atunci clasei $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_p},1}$. Este evident că aplicînd procedeul P acestui subşir prin negarea funcţiei logice din B_{p+1} se obţine un element din $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_p},2}$. Rezultă că cele două mulţimi nu sînt vide.

Este evident că într-una din cele două clase toate subşirurile au proprietatea că primele $p + 2$ elemente ale lor coincid cu primele $p + 2$ elemente ale subşirului (10). De aici rezultă că clasa $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_l-1}}$ nu este vidă şi deci primele l elemente ale subşirurilor ei coincid cu primele l elemente din subşirul (10).

Dacă condiţia logică B_{k_l-1} din subşirul (10) conţine funcţia logică, aşa cum apare ea în schema (1), atunci $k_l = k_{l-1} + 1$ deoarece şirul (10) satisface proprietăţile şirului (2). Dar atunci $\varepsilon_{k_l-1} = 1$ şi deci dacă T este un subşir care aparţine clasei $\mathfrak{L}_j^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_l-1}}$, atunci conform procedeului P rezultă că după elementul B_{k_l-1} urmează elementul $B_{k_l-1} + 1$ adică chiar B_{k_l}. Dacă condiţia logică B_{k_l-1} din subşirul (10) conţine negaţia funcţiei logice, atunci K_l este indicele de transfer al acestei condiţii logice. În acest caz $\varepsilon_{k_l-1} = 2$ şi deci în subşirurile clasei $\mathfrak{L}^{0,\varepsilon_{k_1},\varepsilon_{k_2},\ldots,\varepsilon_{k_l-1}}$ după elementul B_{k_l-1} urmează elementul B_{k_l}, unde k_l este indicele de transfer al lui B_{k_l-1}. Deoarece elementul B_{k_l} din subşirul (10) poate fi operator aritmetic sau condiţie logică după cum $k_l \neq k_s$, $s = 1, 2, \ldots, l-1$ sau nu, rezultă că procedeul P aplicat la subşirurile mulţimii $\mathfrak{L}_i^{0,\varepsilon_{k_1},\ldots,\varepsilon_{k_l-1}}$ ne dă cel puţin un subşir care coincide cu subşirul (10).

În cazul cînd în (10) $l = 1$ şi B_{k_1} este operator aritmetic, atunci evident procedeul P ne dă subşirul $B_i B_{i+1}$ care coincide cu subşirul (10). Astfel teorema este complet demonstrată.

Aplicînd procedeul P fiecărui operator aritmetic din schema (1), se pot obţine toate subşirurile posibile de forma (2) ataşate unei scheme logice de algoritmi.

Vom ataşa acestui procedeu o schemă logică graf care ne permite obţinerea automată a subşirurilor de forma (2). Pentru aceasta introducem următoarele notaţii :

1) p_k operatorul logic care recunoaşte dacă elementul B_k al schemei (1) este operator aritmetic sau condiţie logică.

2) $S(B_k)$ operatorul care scrie elementul B_k la dreapta subşirului obţinut anterior din aplicarea anterioară a acestui operator, iar S aplicat

primului element cu care începe subşirul respectiv este acelaşi cu scrierea lui.

3) S_j un şir de celule în care se introduce un subşir de forma (2), $j = 0,1, \ldots$

4) φ un operator logic la început identic fals.

5) U un şir de celule de lucru notate cu C_j.

6) K_s operatorul care barează funcţia logică α_s

7) $\Phi(U)$ condiţia logică falsă în cazul cînd nu există nici o funcţie logică barată în şirul U iar în caz contrar adevărată.

8) $O(U)$ operatorul care determină indicele de ordine a ultimei funcţii logice barate din şirul U.

9) A_s operatorul care determină indicele de transfer al condiţiei logice B_s.

10) ψ operatorul logic care este adevărat dacă într-un şir de forma (2) se repetă o funcţie logică sau apare o condiţie logică care conţine negaţia funcţiei logice care a mai figurat în acelaşi subşir şi este fals în caz contrar.

11) T_q un şir de celule în care se înscrie subşirul determinat de valoarea adevărată a lui ψ.

12) $O(k)$ operatorul de restabilire a parametrului k.

13) θ numărul operatorilor aritmetici din şirul (10).

14) ω un semnal oarecare.

Cu aceste notaţii, schema logică graf a procedeului P este dată în anexă (fig 1 şi 2).

În cele ce urmează, bazîndu-se pe procedeul P introdus mai sus vom demonstra cîteva afirmaţii care se referă la schemele logice de algoritmi. În acest scop vom introduce mai întîi cîteva noţiuni.

DEFINIŢIA 8. Spunem despre şirul (2) că este un drum admis dacă B_j este operator aritmetic.

Dacă B_j este o condiţie logică, atunci spunem că şirul (2) este un drum neadmis.

Aplicînd procedeul P tuturor operatorilor aritmetici ai schemei (1) să presupunem că am obţinut în şirul $S_0, S_1, \ldots, S_\lambda$ toate drumurile admise.

Notăm cu S_{ij} mulţimea tuturor drumurilor admise care încep cu operatorul B_i şi se termină cu B_j. Obţinem astfel un subşir S_{ij} de forma :

$$\begin{aligned} &B_i B_{k_1^1} B_{k_1^2} \ldots B_{k_1^{s_1}} B_j, \\ &B_i B_{k_2^1} B_{k_2^2} \ldots B_{k_2^{s_2}} B_j, \\ &\cdots\cdots\cdots\cdots \\ &B_i B_{k_m^1} B_{k_m^2} \ldots B_{k_m^{s_m}} B_j. \end{aligned} \tag{11}$$

Considerăm funcţia logică α_{ij} de forma :

$$\alpha_{ij} = \bigvee_{i=1}^{m} \ \& \ \beta_{k_i^j}, \tag{12}$$

unde $\beta_{k_i^j}$ este funcţia logică conţinută în condiţia logică $B_{k_i^j}$. În cazul cînd mulţimea S_{ij} este vidă, atunci $\alpha_{ij} \equiv 0$. Dacă $j = i + 1$, atunci $\alpha_{ij} \equiv 1$.

Cu funcţiile $\alpha_{i,j}$ astfel definite fiecărei scheme logice de algoritmi i se poate ataşa o schemă matricială în felul următor : Fie $A_0, A_1, \ldots, A_k, A_{k+1}$ toţi operatorii aritmetici din schema (1), $A_0 = B_1$ şi $A_{k+1} = B_n$. Fiecărei perechi de operatori aritmetici (A_i, A_j) îi corespunde o funcţie logică α_{ij} definită ca mai sus. Matricea ataşată are forma următoare :

	A_1	A_2	. . .	A_j	. . .	A_{k+1}
A_0	$\alpha_{0,1}$	$\alpha_{0,2}$	. . .	$\alpha_{0,j}$	. . .	$\alpha_{0,k+1}$
A_1	$\alpha_{1,1}$	$\alpha_{1,2}$	. . .	$\alpha_{1,j}$	. . .	$\alpha_{1,k+1}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
A_i	$\alpha_{i,1}$	$\alpha_{i,2}$	. . .	$\alpha_{i,j}$	. . .	$\alpha_{i,k+1}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
A_k	$\alpha_{k,1}$	$\alpha_{k,2}$	. . .	$\alpha_{k,j}$	. . .	$\alpha_{k,k+1}$

(13)

Exemplu. Fie schema logică de algoritmi

$$A_0 A_1(\alpha_2, 5)\ (\alpha_3, 6)\ (\alpha_4, 8)\ A_5(\alpha_6, 10)\ A_7\ (\alpha_8, 3)\ A_9 A_{10}.$$

Construim mulţimile $S_{i,j}$:

$$S_{01} = A_0 A_1,$$

mulţimile S_{0i} pentru $i \neq 1$ sînt vide.

Mulţimea $S_{1,5}$ conţine următoarele subşiruri

$$A_1(\alpha_2, 5)(\alpha_3, 6)(\alpha_4, 8) A_5,$$

$$A_1(\bar{\alpha}_2, 5)\ A_5.$$

Mulţimea $S_{1,7}$ conţine următorul subşir

$$A_1(\alpha_2, 5)(\bar{\alpha}_3, 6)(\alpha_6, 10)\ A_7$$

Mulţimea $S_{1,9}$ conţine următorul subşir

$$A_1(\alpha_2, 5)(\alpha_3, 6)(\bar{\alpha}_4, 8)(\alpha_8, 3)\ A_9$$

Mulţimea $S_{1,10}$ conţine următorul subşir

$$A_1(\alpha_2, 5)(\bar{\alpha}_3, 6)(\bar{\alpha}_6, 10)\ A_{10}$$

Mulţimile $S_{1,1}$ şi $S_{1,0}$ sînt vide.

Mulţimile $S_{5,0}$; $S_{5,1}$; $S_{5,5}$ şi $S_{5,9}$ sînt vide.

Mulţimile $S_{5,7}$ şi $S_{5,10}$ conţin respectiv subşirurile

$$A_5(\alpha_6, 10) A_7$$

$$A_5(\bar{\alpha}_6, 10) A_{10}$$

Mulţimile $S_{7,0}$ şi $S_{7,1}$ sînt vide iar mulţimile $S_{7,5}$; $S_{7,7}$; $S_{7,9}$ şi $S_{7,10}$ conţin respectiv subşirurile

$$A_7(\overline{\alpha_8}, 3)\,(\alpha_3, 6)\,(\alpha_4, 8)\,A_5,$$
$$A_7(\overline{\alpha_8}, 3)\,(\overline{\alpha_3}, 6)\,(\alpha_6, 10)\,A_7,$$
$$A_7(\alpha_8, 3)\,A_9$$
$$A_7(\overline{\alpha_8}, 3)\,(\overline{\alpha_3}, 6)\,(\overline{\alpha_6}, 10)\,A_{10}.$$

Mulţimile $S_{9,1}$; $S_{9,5}$; $S_{9,7}$ sînt vide iar mulţimea $S_{9,10}$ are forma :

$$A_9\,A_{10}.$$

Construim acum funcţiile $\alpha_{i,j}$:

$$\alpha_{0,1} = 1, \quad \alpha_{0,5} = 0, \quad \alpha_{0,7} = 0, \quad \alpha_{0,9} = 0, \quad \alpha_{0,10} = 0,$$
$$\alpha_{1,1} = 0, \quad \alpha_{1,5} = (\alpha_2 \,\&\, \alpha_3 \,\&\, \alpha_4) \vee \overline{\alpha_2}, \quad \alpha_{1,7} = \alpha_2 \,\&\, \overline{\alpha_3} \,\&\, \alpha_6,$$
$$\alpha_{1,9} = \alpha_2 \,\&\, \alpha_3 \,\&\, \overline{\alpha_4} \,\&\, \alpha_8, \quad \alpha_{1,10} = \alpha_2 \,\&\, \overline{\alpha_3} \,\&\, \overline{\alpha_6}, \quad \alpha_{5,1} = 0$$
$$\alpha_{5,5} = 0, \quad \alpha_{5,7} = \alpha_6, \quad \alpha_{5,9} = 0, \quad \alpha_{5,10} = \overline{\alpha_6}, \quad \alpha_{7,1} = 0,$$
$$\alpha_{7,5} = \overline{\alpha_8} \,\&\, \alpha_3 \,\&\, \alpha_4, \quad \alpha_{7,7} = \overline{\alpha_8} \,\&\, \overline{\alpha_3} \,\&\, \alpha_6, \quad \alpha_{7,9} = \alpha_8,$$
$$\alpha_{7,10} = \overline{\alpha_8} \,\&\, \overline{\alpha_3} \,\&\, \overline{\alpha_6}, \quad \alpha_{9,1} = 0, \quad \alpha_{9,5} = 0,$$
$$\alpha_{9,7} = 0, \quad \alpha_{9,9} = 0, \quad \alpha_{9,10} = 1.$$

Vom clasifica schemele logice de algoritmi după cum urmează. Considerăm mulţimea tuturor schemelor de forma (1) şi o împărţim în trei clase S', S'' şi S'''.

1) Din mulţimea S' fac parte acele scheme de forma (1) pentru care oricare ar fi starea memoriei X_0 executarea schemei cu această stare se sfîrşeşte cu executarea elementului final B_n. Elementele acestei mulţimi le vom numi scheme complet admise.

2) Din mulţimea S'' fac parte acele scheme de forma (1) pentru care există cel puţin o stare a memoriei X_0 astfel încît executînd schema cu această stare, ea se sfîrşeşte cu elementul final B_n. Elementele acestei mulţimi le vom numi scheme parţial admise. Evident, avem $S' \in S''$.

3) Din mulţimea S''' fac parte acele scheme pentru care oricare ar fi starea memoriei X_0 executarea schemei cu această stare nu ne conduce niciodată la elementul final B_n. Elementele acestei mulţimi le vom numi scheme neadmise.

Cu ajutorul schemelor matriciale ataşate schemelor logice de algoritmi, folosind procedeul P, se pot caracteriza clasele S', S'' şi S''' definite mai sus.

În acest scop va trebui mai întîi să descriem modul de realizare al schemei matriciale (13). Mai întîi observăm că din modul de realizare a schemei (1) rezultă că după executarea unui oarecare operator aritmetic A_s nu se pot executa în acelaşi timp decît cel mult un operator aritmetic A_j. De aici rezultă că elementele matricei (13) verifică următoarea egalitate

$$\alpha_{i,s}(X) \,\&\, \alpha_{i,j}(X) \equiv 0 \quad (j \neq s,\ i = \overline{0,k}) \tag{14}$$

oricare ar fi starea memoriei X.

Executarea schemei matriciale (13) începe cu operatorul A_0. Fie X_0 o stare iniţială a memoriei şi $X_1 = A_0(X_0)$ starea memoriei obţinute după executarea operatorului A_0. Executăm acum pe rînd elementele primei linii din matricea (13). Presupunem că $\alpha_{0s}(X_1) = 1$. Urmează acum să se execute operatorul A_s. După executarea lui, cu starea memoriei obţinută se execută elementele liniei $s + 1$ şi fie $\alpha_{s+1,r}(X_{s+1}) = 1$; atunci se execută operatorul A_r. Procedeul continuă apoi în mod analog.

Ca şi în cazul schemelor logice de algoritmi se pot întîmpla şi aici următoarele cazuri :

1) Procesul de executare a schemei matriciale (13) se termină cu executarea operatorului aritmetic de oprire A_{k+1}.

2) Procesul de executare a schemei matriciale (13) continuă indefinit.

Schemele matriciale se pot clasifica în mod analog ca şi schemele logice de algoritmi. Putem enunţa atunci următoarea teoremă :

TEOREMA 3. *Condiţia necesară şi suficientă ca o schemă logică de algoritmi să fie complet admisă (parţial admisă, neadmisă) este ca schema matricială ataşată ei prin procedeul P să fie complet admisă (parţial admisă, neadmisă).*

Vom demonstra teorema numai în cazul schemelor logice de algoritmi complet admise întrucît celelalte cazuri rezultă imediat din acestea.

Necesitatea condiţiei. Presupunem că schema logică de algoritmi (1) este complet admisă. Aceasta înseamnă că X_0 fiind o stare oarecare a memoriei şi începînd executarea schemei cu această sare, ultimul operator aritmetic care se execută este operatorul de oprire B_n.

Fie

$$B_1 B_{i_1} B_{i_2} \ldots B_{i_k} B_n, \qquad (0 < i_s \leqslant n) \tag{15}$$

şirul de operatori aritmetici care se execută în ordinea arătată începînd cu operatorul B_1 şi sfîrşind cu operatorul B_n, unde indicii i_s pot să se şi repete. Începem executarea cu starea memoriei X_0. După executarea operatorului B_1 se obţine starea memoriei X_{i_1}.

Fie

$$B_1 D_1 B_{i_1}$$
$$B_1 D_2 B_{i_1}$$
$$\ldots\ldots$$
$$B_1 D_t B_{i_1}$$

toate elementele puse în evidenţă de procedeul P care încep cu B_1 şi se termină cu B_{i_1}. Deoarece după operatorul B_1 urmează să se execute operatorul B_{i_1}, rezultă că cel puţin una din relaţiile $D_j(X_{i_1}) = 1$, $1 \leqslant j \leqslant t$ este adevărată. Dar α_{0i_1} este de forma

$$\alpha_{0,i_1} = \bigvee_{i=1}^{t} D_i(X_{i_1}) = 1$$

ceea ce ne dovedeşte că şi în schema matricială ataşată schemei respective după executarea operatorului B_1 se execută operatorul B_{i_1}. Vom arăta

acum că $\alpha_{0j}(X_{i_1})=0$ pentru $j \neq i_1$. Presupunem că $\alpha_{0,i_1}(X_{i_1})=1$. De aici rezultă că după executarea operatorului B_1 se execută în acelaşi timp atît operatorul B_{i_1} cît şi operatorul B_{j_1} ($B_{i_1} \neq B_{j_1}$) ceea ce este absurd. Presupunem acum că în schema logică de algoritmi am executat un oarecare operator B_{i_ν} după care trebuie să executăm operatorul $B_{i_{\nu+1}}$.

De asemenea presupunem că în schema matricială am executat operatorul B_{i_ν} .Vom arăta că şi aici după executarea lui se va executa operatorul $B_{i_{\nu+1}}$. Într-adevăr, fie

$$B_{i_\nu}\, C_1\, B_{i_{\nu+1}}$$
$$\cdots\cdots$$
$$B_{i_\nu}\, C_l\, B_{i_{\nu+1}}$$

subşirurile puse în evidenţă de procedeul P care încep cu elementul B_{i_ν} şi se termină cu $B_{i_{\nu+1}}$. Deoarece după executarea operatorului $B_{i_{\nu+1}}$ se execută operatorul $B_{i_{\nu+1}}$ rezultă că există cel puţin un C_j pentru care $C_j(X_{i_{\nu+1}}) = 1$, $1 \leqslant j \leqslant l$. Dar cum $\alpha_{i_\nu, i_{\nu+1}}$ are forma

$$\alpha_{i_\nu, i_{\nu+1}} = \bigvee_{i=1}^{l} C_i$$

rezultă că $\alpha_{i_\nu, i_{\nu+1}}(X_{i_{\nu+1}}) = 1$. De asemenea se arată uşor că $\alpha_{i_\nu, j}(X_{i_{\nu+1}})=0$ pentru $j \neq i_\nu + 1$, ceea ce ne demonstrează că după executarea operatorului B_{i_ν} se execută operatorul $B_{i_{\nu+1}}$. În baza ipotezelor inducţiei necesitatea condiţiei este demonstrată.

Suficienţa condiţiei. Presupunem că schema matricială este complet admisă şi repetînd raţionamentul de la demonstraţia necesităţii se va găsi că şi schema logică de algoritmi este complet admisă.

TEOREMA 4. *Condiţia necesară ca o schemă logică de algoritmi să fie complet admisă este ca toate subşirurile puse în evidenţă de procedeul P să aibă ca ultim element un operator aritmetic.*

Demonstraţia o facem prin reducere la absurd. Presupunem că schema logică de algortimi (1) este complet admisă şi că prin procedeul P s-a pus în evidenţă un subşir de forma (2) în care ultimul element este o condiţie logică. Fie acesta subşirul

$$B_i\, B_{i_1}\, B_{i_2}\, \ldots\, B_{i_{k-1}}\, B_{i_k}$$

astfel că $i_k = i_s$, $(1 \leqslant s \leqslant k-1)$ şi B_i este primul operator aritmetic din schema (1) cu această proprietate. Vom arăta că în acest caz există o stare a memoriei X_0 pentru care executarea schemei (1) începînd cu această stare nu se sfîrşeşte cu executarea operatorului B_n .

Într-adevăr, să presupunem că executînd schema (1), începînd cu starea X_o, se ajunge la operatorul B_i. O astfel de stare se poate construi uşor. După executarea acestui operator presupunem că se obţine starea

X_{i+1} pentru care $B_{i_t}(X_{i+1}) = 1$ pentru $t = 1,2, \ldots, k-1$, unde β_{i_t} este funcţia logică conţinută în condiţia logică B_{i_t} aşa cum apare în subşirul de mai sus. Întrucît $i_k = i_s$, $(1 \leqslant s \leqslant k-1)$, rezultă că funcţia logică din elementul B_{i_k} coincide cu α_{i_s} adică cu funcţia logică din condiţia logică B_{i_s} aşa cum apare ea în schema (1). Dacă $\beta_{i_s} = \alpha_{i_s}$, atunci $\alpha_{i_k} = \alpha_{i_s} = 1$ şi rezultă că după verificarea condiţiei B_{i_k} urmează din nou verificarea condiţiei $B_{i_{s+1}}$ pentru aceeaşi stare X_{i+1}. Deoarece $\beta_{i_t}(X_{i+1}) = 1$ pentru $t=1, 2, \ldots, k-1$, rezultă că se ajunge din nou la verificarea condiţiei logice B_{i_k} pentru aceeaşi stare X_{i+1} şi acest proces continuă indefinit.

Dacă $\beta_{i_s}=\overline{\alpha_{i_s}}$, rezultă că $\alpha_{i_k}(X_{i+1})=0$ şi deci $\beta_{i_1}(X_{i+1}) \,\&\, \beta_{i_2}(X_{i+1}) \,\&\, \ldots$ $\ldots \,\&\, \beta_{i_k}(X_{i+1}) = 0$.

Considerînd acum toate subşirurile de forma (2) puse în evidenţă de procedeul P, care încep cu B_i şi observînd că toate aceste subşiruri conţin cel puţin un indice i_r $(1 \leqslant r \leqslant k-1)$ pentru care condiţia logică cu acest indice conţine negaţia funcţiei logice B_{i_r}, rezultă că pentru starea X_{i+1} executarea schemei logice se întrerupe la elementul B_i ceea ce contrazice ipoteza că schema este complet admisă. Cu aceasta teorema este demonstrată.

TEOREMA 5. *Condiţia suficientă pentru ca o schemă matricială să fie complet admisă este ca oricare ar fi două stări ale memoriei X_1 şi X_2 să avem relaţiile*

$$\alpha_{i,s}(X_1) \,\&\, \alpha_{j,j}(X_2) \equiv 0, \quad i \neq j,\ i,\ j,\ s = 1,\ 2,\ \ldots,\ k+2$$

şi (16)

$$\bigvee_{s=1}^{k+1} \alpha_{i,s}(X) = 1, \quad \alpha_{i,i}(X) = 0,\ i = 0,1,\ldots,\ k$$

oricare ar fi X o stare a memoriei.

Fie X_0 o stare oarecare a memoriei pe care o considerăm ca stare iniţială. După executarea operatorului A_0 se obţine starea X_1. Fie $\alpha_{0i}(X_1) = 1$. Rezultă că urmează să se execute operatorul aritmetic A_{i_1}. Din condiţiile (16) rezultă că acest operator odată executat nu se va mai executa niciodată. Tot din condiţiile (16) rezultă că procesul nu se poate termina cu executarea unui oarecare operator aritmetic A_t $(t \neq k+1)$. Deci după un număr finit de paşi, executarea schemei se termină cu operatorul de oprire A_{k+1}. Astfel teorema este demonstrată.

Fie $X \in Y$ o stare oarecare a memoriei. Să presupunem că executînd schema matricială începînd cu această stare una şi numai una din funcţiile $\alpha_{0,i_1} = 1$. Atunci presupunem că pentru toate stările care rezultă din X pentru executarea tuturor operatorilor diferiţi de A_0 avem $\alpha_{j,i1} = 0, j \neq 0$. În general dacă se execută operatorul A_t şi avem $\alpha_{t,i_t} = 1$, atunci $\alpha_{j,i_t} = 0$, $j = t$, pentru toate celelalte stări obţinute din executarea celorlalţi operatori aritmetici diferiţi de A_t din schema matricială, cu starea iniţială a memoriei X_0. Dacă o schemă matricială verifică această proprietate pentru orice stare iniţială a memoriei X atunci spunem că schema matricială satisface proprietatea S faţă de starea iniţială X.

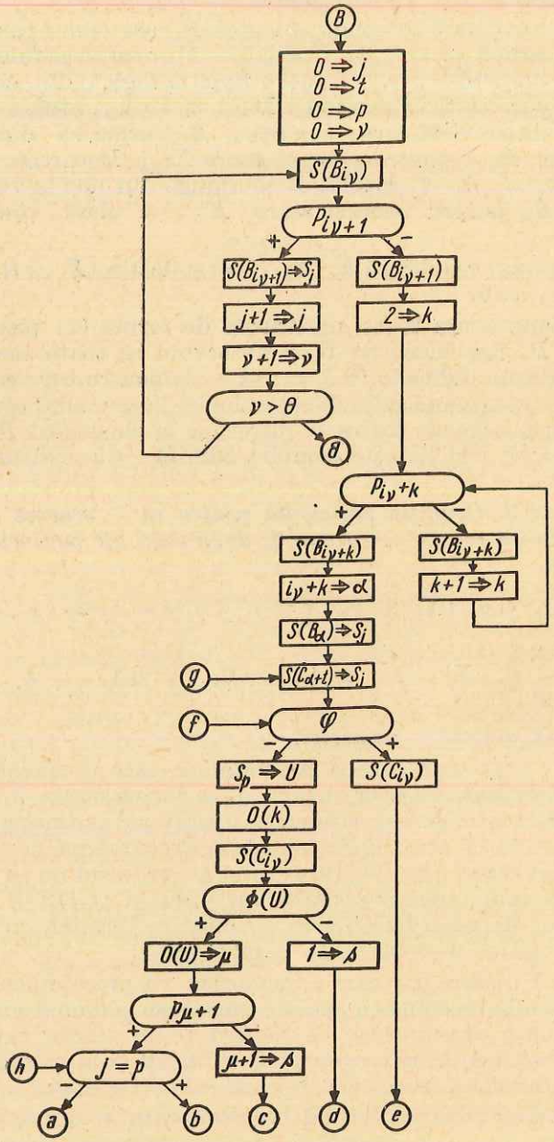
B
0 ⇒ j
0 ⇒ t
0 ⇒ p
0 ⇒ γ
S(Bi,γ)
Pi,γ+1
S(Bi,γ+1) ⇒ Sj
S(Bi,γ+1)
j+1 ⇒ j
2 ⇒ k
γ+1 ⇒ γ
γ > θ
θ
Pi,γ+k
S(Bi,γ+k)
S(Bi,γ+k)
i,γ+k ⇒ α
k+1 ⇒ k
S(Bα) ⇒ Sj
g
S(Cα+t) ⇒ Sj
f
φ
Sp ⇒ U
S(Ci,γ)
0(k)
S(Ci,γ)
φ(U)
0(U) ⇒ μ
1 ⇒ δ
Pμ+1
h
j = p
μ+1 ⇒ δ
a
b
c
d
e

Fig. 1

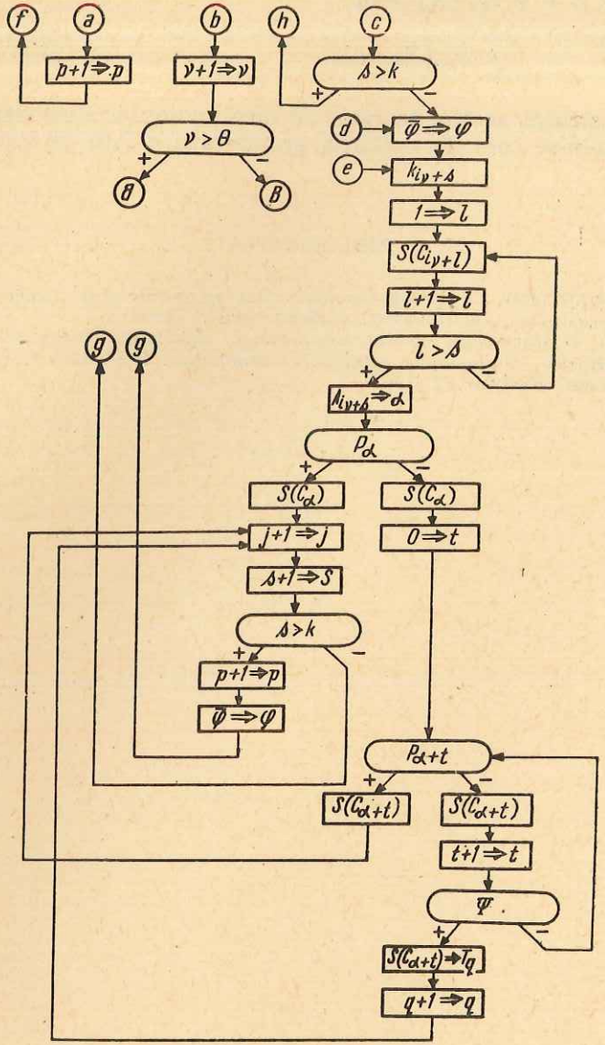
f
a
b
h
c
p+1 ⇒ p
γ+1 ⇒ γ
δ > k
γ > θ
d
φ̄ ⇒ φ
θ
B
e
ki,γ+δ
1 ⇒ l
S(Ci,γ+l)
l+1 ⇒ l
l > δ
g
g
ki,γ+δ ⇒ α
Pα
S(Cα)
S(Cα)
j+1 ⇒ j
0 ⇒ t
δ+1 ⇒ S
δ > k
p+1 ⇒ p
φ̄ ⇒ φ
Pα+t
S(Cα+t)
S(Cα+t)
t+1 ⇒ t
φ
S(Cα+t) ⇒ Tq
q+1 ⇒ q

Fig. 2

TEOREMA 6. *Condiţia suficientă pentru ca o schemă matricială să fie complet admisă este ca ea să satisfacă proprietatea S faţă de orice stare a memoriei* $X \in Y$ şi $\alpha_{i,i}(X) = 0$,

$$\bigvee_{s=1}^{k+1} \alpha_{i,s}(X) = 1, \qquad (i = 0,1,\ldots,k).$$

Demonstraţia acestei teoreme se face analog cu demonstraţia teoremei 5, ţinîndu-se cont de definiţia proprietăţii S faţă de starea memoriei X.

BIBLIOGRAFIE

1. Р.И. ПОДЛОВЧЕНКО, *О преобразованиях схем программ и их применении в программировании*. Проблемы кибернетики 7, (1962).
2. И.И. ЯНОВ, *О логических схемах алгоритмов*. Проблемы кибернетики, I (1958).
3. В.Ф. ДЬЯЧЕНКО, *Построение граф-схем алгоритмов*. Проблемы передачи информации, Выпуск 12 (1963).