ANALIZA LOGICĂ A ALGORITMILOR (I)

DE

E. MUNTEANU
(Cluj)

În această lucrare se formulează o nouă problemă de analiză a algoritmilor şi se propune o metodă de construire a formulei complet ataşate unui algoritm schemă-graf fără cicluri.

§. 1. INTRODUCERE

Succesele obţinute în ultimul timp, în automatizarea programării, prin creerea limbajelor de programare de tip ALGOL şi a translatoarelor din aceste limbaje au dus la uşurarea esenţială a muncii de programare.

Folosirea programelor de programare îl scuteşte pe programator de o muncă laborioasă, care nu este legată de esenţa rezolvării problemei, ci apare datorită particularităţilor maşinii cu ajutorul căreia se rezolvă problema. Totuşi, aşa cum arată experienţa, scrierea algoritmului de rezolvare a problemei într-un limbaj de programare comportă unele greutăţi specifice chiar şi atunci cînd expunerea acestuia în limbajul matematic obişnuit este făcută clar. Acestea sînt greutăţile legate de aducerea algoritmului la o formă comodă pentru programare şi ele sînt legate de specificul maşinilor electronice de calcul în general. Unele dintre aceste greutăţi, cum ar fi de exemplu acelea legate de necesitatea ca reprezentarea informaţiei să aibă o structură asemănătoare cu structura în formă de adrese a memoriei maşinii, pot fi probabil rezolvate în cadrul limbajului ALGOL, printr-o extindere a acestuia.

Pentru a înlătura însă complet greutăţile legate de aducerea algoritmului la o formă comodă pentru programare, considerăm că este necesar să se elaboreze limbaje de programare cu posibilităţi principial noi.

Descrierea metodei de rezolvare a problemei într-un astfel de limbaj, spre deosebire de limbajele existente pînă în prezent nu trebuie să co-

ST. CERC. MAT., TOM. 18, NR. 8, P. 1113–1127, BUCUREŞTI, 1966

incidă cu descrierea precisă a succesiunii de operații, care trebuie realizate pentru a obține rezultatul problemei. Cu alte cuvinte, descrierea rezolvării problemei într-un astfel de limbaj nu trebuie să fie neapărat un algoritm în înțelesul obișnuit al cuvîntului. Munca de elaborare a algoritmului propriu-zis să fie lăsată în seama mașinii.

Se știe că în anumite cazuri, descrierea unui algoritm simplu poate fi înlocuită cu descrierea sistemului de legături logice dintre informația inițială a problemei și rezultatul obținut în urma îndeplinirii algoritmului. Prin urmare este natural ca un astfel de limbaj să conțină posibilitatea de a descrie probleme printr-un sistem de condiții logice. Totuși, posibilitățile unei astfel de descrieri trebie să fie limitate, deoarece, în caz contrar, ar urma să pretindem unui program de programare, dintr-un astfel de limbaj, să elaboreze algoritmul de rezolvare a oricărei probleme matematice. În realitate este vorba de a trece în sarcina mașinii numai cele mai simple elemente de construire a unui algoritm, acelea care au un caracter pur tehnic și nu sînt legate de esența problemei.

Astfel devine actuală formularea următoarei probleme : să se găsească o metodă, care să permită, în cazurile cele mai simple, construirea în mod automat a algoritmilor pornindu-se de la sistemul de legături logice dintre informația inițială și rezultatul final.

Această problemă conduce la abordarea analizei algoritmilor dintr-un punct de vedere cu totul nou. Caracteristica generală a metodelor de analiză a algoritmilor, [1], [2], [3], cunoscute pînă în prezent, constă în determinarea contradicțiilor de ordin formal existente în construcția algoritmului. În cazul cînd algoritmul nu conține contradicții de ordin formal, rămîne totuși deschisă următoarea problemă : algoritmul analizat corespunde problemei pentru care a fost construit?

Este deci natural ca, analiza algoritmilor să se trateze din punctul de vedere al corespondenței acestuia cu sistemul de condiții logice care descrie formularea problemei ce se rezolvă cu algoritmul respectiv. Cu alte cuvinte, problema analizei algoritmilor trebuie formulată în sensul elaborării unor metode de restabilire a sistemului de condiții logice care descrie formularea problemei, pornindu-se numai de la descrierea formală a algoritmului și în a verifica pe urmă dacă acest sistem coincide cu cel dat inițial prin formularea problemei.

O rezolvare a problemei de analiză a algoritmilor în sensul descris mai sus ar putea să constituie un prim pas, în găsirea metodelor de construire automată a algoritmilor. În afară de aceasta ea poate să devină un instrument foarte util în controlul programelor pentru mașinile electronice de calcul.

Într-adevăr, dispunînd de o metodă de stabilire a sistemului de condiții logice, care descrie formularea problemei, pornindu-se numai de la programul de rezolvare a acesteia și comparînd pe urmă acest sistem cu cel dat inițial, se va putea stabili corectitudinea programului relativ la problema propusă.

Greutățile foarte mari, care apar în controlul programelor atît în programarea manuală, cît și în programarea automată, precum și lipsa unor programe universale de control justifică în suficientă măsură importanța acestei idei în rezolvarea problemei de analiză a algoritmilor.

§ 2. FORMULAREA PROBLEMEI DE ANALIZĂ A ALGORITMILOR

Să considerăm problema P și să notăm cu X^0 datele inițiale ale problemei iar cu X^1 rezultatele finale.

Presupunem că pentru rezolvarea problemei P, a fost construit algoritmul $\mathfrak{A}$, descris într-un limbaj formal. În continuare, ca limbaj formal pentru descrierea algoritmilor vom considera limbajul schemelor-graf.

DEFINIȚIA 1. Formula $F(X^0, X^1)$ se numește formulă *atașată algoritmului* $\mathfrak{A}$, dacă ea este adevărată atunci cînd algoritmul $\mathfrak{A}$ aplicat datelor inițiale X^0, le transformă în rezultatul final X^1.

DEFINIȚIA 2. Formula $F(X^0, X^1)$ se numește formulă *complet atașată algoritmului* $\mathfrak{A}$, dacă ea este adevărată atunci și numai atunci cînd aplicînd algoritmul $\mathfrak{A}$ datelor inițiale X^0, acesta le transformă în rezultatul final X^1.

Rezolvarea problemei de analiză a algoritmilor cere, pe lîngă elaborarea unei metode de construire a formulei atașate sau a formulei complet atașate, un algoritm și fixarea unui limbaj, în care urmează să fie scrise aceste formule precum și sistemul de relații logice inițiale, care reprezintă formularea problemei P. În continuare vom descrie toate aceste formule în limbajul logico-matematic.

Presupunem că formularea problemei P este descrisă de formula logică $\mathcal{F}(X^0, X^1)$, care este un predicat definit pe mulțimea datelor inițiale și a rezultatelor finale. De exemplu, pentru găsirea maximului numerelor $x_1, x_2, \ldots, x_n$, predicatul respectiv va fi

$$[(x=x_1) \vee (x=x_2) \vee \ldots \vee (x=x_n)] \& (x \geqslant x_1) \& (x \geqslant x_2) \& \ldots \& (x \geqslant x_n), \tag{1}$$

unde x este maximul căutat. Formula (1) nu este altceva decît definiția maximului numerelor $x_1, x_2, \ldots, x_n$. Ea nu reprezintă algoritmul de găsire a maximului deoarece nu ne dă nici-o informație privitoare la succesiunea de operații, care trebuie realizate pentru a găsi maximul numerelor $x_1, x_2, \ldots, x_n$.

Dacă reușim să construim o formulă $F(X^0, X^1)$, care să satisfacă definiția 1 sau 2, atunci spunem că algoritmul $\mathfrak{A}$ este *corect* relativ la problema P, dacă formula

$$\forall X^0 \forall X^1 [F(X^0, X^1) \supset \mathcal{F}(X^0, X^1)] \tag{2}$$

este adevărată.

Astfel formulată, problema analizei algoritmului $\mathfrak{A}$ se reduce la construirea unei formule, care să satisfacă definiția 1 sau 2 și la verificarea adevărabilității formulei (2).

În această lucrare vom da o metodă de construire a formulei complet atașate unui algoritm schemă-graf fără cicluri.

§ 3. SCHEME — GRAF CU MEMORIE

În cele ce urmează vom ataşa unei scheme-graf Γ definită de A. L. Kalujnin [4] noţiunea de memorie şi de stare de memorie.

O mulţime finită de obiecte $M = \{x_1, x_2, \ldots, x_n\}$ se numeşte *memorie* a schemei-graf Γ. Elementele mulţimii M le vom numi celule ale memoriei.

Să considerăm o mulţime $S = \{x_1^0, x_2^0, \ldots, x_n^0, \ldots, x_1^j, x_2^j, \ldots, x_n^j, \ldots\}$ ale cărei elemente le vom numi *stări*. Starea x_i^j o numim conţinut al celulei x_i.

Imaginea $x_i^j = U(x_i)$ a mulţimii M în mulţimea S se numeşte *stare a memoriei* schemei-graf Γ.

Vectorul $\xi = (x_1^j, x_2^j, \ldots, x_n^j)$ se numeşte *vector complet* al stării memoriei. Vom nota cu $\int = \{\xi\}$ mulţimea vectorilor completi a tuturor stărilor memoriei.

Fie $X = (x_1, x_2, \ldots, x_n)$, un vector complet al memoriei, f un simbol funcţional oarecare iar p un simbol arbitrar de relaţie.

DEFINIŢIA 3. O expresie de forma $y := f(X)$, unde $y \in M$ se numeşte *operator de arbitrare*.

DEFINIŢIA 4. O expresie de forma $p(x)$ se numeşte *operator logic*.

În definiţiile operatorilor de atribuire şi a operatorului logic, funcţiile f şi p sînt definite pe toată memoria schemei-graf Γ, dar de obicei expresiile $f(X)$ şi $p(X)$ nu conţin toate celulele memoriei. Acele celule care nu figurează în aceste expresii nu sînt esenţiale pentru operatorii respectivi.

DEFINIŢIA 5. O expresie de forma

$$\eta = f(\xi),$$

unde $\eta \in S$ şi $\xi \in \int$ se numeşte *formulă elementară ataşată* operatorului de atribuire $y := f(X)$.

DEFINIŢIA 6. O expresie de forma $p(\xi)$, unde $\xi \in \int$ se numeşte *formulă elementară ataşată* operatorului logic $p(X)$.

Cînd nu este esenţial faptul că o formulă elementară este ataşată unui operator de atribuire sau unui operator logic, atunci vom denumi simplu, formulă elementară.

Pentru formulele elementare este valabilă observaţia făcută mai sus în cazul operatorilor.

Prin interpretarea unei scheme-graf Γ cu memorie, definită de mulţimea $T = \{T_1, T_2, \ldots, T_k\}$ a operatorilor de transformare şi de mulţimea $R = \{R_1, R_2, \ldots, R_m\}$ a operatorilor de ramificare, înţelegem următoarele : fiecărui element $T_i \in T$ îi punem în corespondenţă un operator de atribuire f (corespondenţa nu trebuie să fie neapărat biunivocă) şi fiecărui element R_j R îi punem în corespondenţă un operator logic p_j (şi în acest caz corespondenţa nu este obligatorie să fie biunivocă). În ultimul caz se subînţelege descompunerea mulţimii $\int$ în două mulţimi disjuncte $\int_{p_j}^{+}$, ale cărei elemente satisfac proprietatea p_j şi $\int_{\bar{p}_j}$, ale cărei elemente nu satisfac proprietatea p_j. Vom nota o astfel de interpolare prin $\{M, \int; T \to f; R \to p\}$.

Dacă este dată o schemă-graf Γ cu memorie şi o interpretare $\{M, \int; T \to f; R \to p\}$, atunci noi spunem că este dat un algoritm schemă-graf $\mathfrak{A}$, a cărei acţiune este următoarea :

1° Presupunem că vectorul datelor iniţiale $X^\circ \in \int$, parcurgînd nodurile schemei-graf Γ, s-a transformat în vectorul $\xi = (x_1^s, x_2^s, \ldots, \ldots, x_n^s)$ care a ajuns în nodul T_s. Nodului T_i îi corespunde, prin interpretarea dată, operatorul de atribuire $x_h := f_i(X)$. Procesul de calcul se continuă, după unica săgeată care porneşte din acest nod, cu vectorul

$$\xi' = (x_1^{s'}, x_2^{s'}, \ldots, x_h^{s'}, x_{h+1}^{s'}, \ldots, x_n^{s'}) \text{ unde } x_i^{s'} = x_i^s \text{ pentru } i = 1, 2, \ldots, h, h+1, \ldots, n, \text{ iar corespunde } x_h^{s'}$$

se obţine

$$x_h^{s'} = f_i(\xi), \tag{3}$$

care este formula elementară ataşată operatorului de atribuire

$$x_h := f_i(X).$$

2° Presupunem că vectorul ξ, transformatul vectorului X°, a ajuns într-un nod R_j : Nodului R_j îi corespunde, prin interpretarea dată, operatorul logic p_j. În acest caz procesul de calcul se continuă cu acelaşi vector ξ, după săgeata marcată cu semnul plus, dacă formula elementară $p_j(\xi)$ este adevărată sau după săgeata marcată cu semnul minus în caz contrar.

3° Dacă la o anumită etapă a procesului de calcul elementul X^1 al mulţimii $\int$ ajunge la ieşirea schemei-graf, atunci spunem că X^1 este rezultatul aplicării algoritmului $\mathfrak{A}$ definit de schema-graf Γ prin interpretarea $\{M, \int; T \to f; R \to p\}$. Vom nota acest lucru astfel

$$\mathfrak{A}(X^\circ) = X^1.$$

Dacă în schema-graf a algoritmului $\mathfrak{A}$ este esenţial nu tipul operatorului, ci numărul lui, atunci notăm acest operator cu simbolul O_i. Dacă de la operatorul O_i porneşte o săgeată spre operatorul O_j, atunci vom numi operatorul O_j *succesor* al operatorului O_i, iar pe acesta din urmă îl vom numi *predecesor* al operatorului O_j

3.1. *Distribuirea memoriei unui algoritm schemă-graf.* Fie algoritmul schemă-graf $\mathfrak{A}$ definit de schema-graf Γ prin interpretarea $\{M, \int; T \to f; R \to p\}$.

Vom nota cu $M_r = \{x_{i_1}, x_{i_2}, \ldots, x_{i_r}\}$ acea submulţime a mulţimii M care se caracterizează prin aceea că, în timpul realizării algoritmului $\mathfrak{A}$, conţinutul celulelor x_{i_k} $(k = 1, 2, \ldots, r)$ se schimbă. Mulţimea M_r este formată din acele celule ale memoriei care figurează în membrul drept al operatorilor de atribuire. Mulţimea celorlalte celule ale memoriei o vom nota cu $M_c = \{c_1, c_2, \ldots, c_s\}$ $(s + r = n$, unde n este numărul celulelor memoriei). Notăm cu c_i^0 starea iniţială a celulei c_i şi cu $M_c^0 = \{c_1^0, c_2^0, \ldots, c_s^0\}$ mulţimea stării iniţiale a celulelor mulţimii M_c. Evident că $M_c \in S$.

Fie $\{x_{j_1}, x_{j_2}, \ldots, x_{j_m}\}$ acele celule ale mulţimii M_r ale căror stări iniţiale sînt date iniţiale pentru algoritmul $\mathfrak{A}$. Pentru simplitate presupunem

că acestea sînt primele m celule ale memoriei și mulțimea lor o notăm în continuare cu $N = \{x_1, x_2, \ldots, x_m\}$. Mulțimea N poate să fie vidă. Notăm cu $N^0 = \{x_1^0, x_2^0, \ldots, x_m^0\}$ mulțimea stării inițiale a celulelor mulțimii N. Atunci elementele mulțimii $I^0 = N^0 \cup M_c^0$ sînt date inițiale ale algoritmului $\mathfrak{A}$.

Dacă mulțimea N nu coincide cu mulțimea M_V, atunci înseamnă că în timpul acționării algoritmului $\mathfrak{A}$, anumite celule ale mulțimii M_V, se folosesc pentru păstrarea unor rezultate intermediare sau a unor rezultate finale. Acele celule care se folosesc numai pentru păstrarea rezultatelor intermediare, se numesc *celule lucrătoare*. Este evident că algoritmul $\mathfrak{A}$ trebuie în așa fel construit, încît rezultatul final obținut să nu depindă de starea inițială a celulelor lucrătoare.

După terminarea procesului de calcul, anumite celule ale mulțimii M_V (e posibil ca toate) conțin rezultatul aplicării algoritmului $\mathfrak{A}$.

Notăm mulțimea acestor celule cu $R = \{x_{t_1}, x_{t_2}, \ldots, x_{t_p}\}$. Dacă notăm cu x^1 starea finală a celulei x, atunci mulțimea $R^1 = \{x_{t_1}^1, x_{t_2}^1, \ldots, x_{t_p}^1\}$ este mulțimea rezultatelor aplicării algoritmului $\mathfrak{A}$.

Pentru descrierea unei stări intermediare a unei celule oarecare introducem noțiunea de *rang*.

Simbolurile x_i^j care intră într-o formulă elementară se numesc *variabile* ale formulei elementare respective. Numărul j se numește rangul variabilei x_i^j. Rangul fiecărei variabile este un număr întreg pozitiv care se referă la celula a cărei stare este variabila respectivă.

Din notația introdusă pentru starea finală a unei celule rezultă că toate variabilele cu rangul 1 reprezintă rezultate finale ale algoritmului respectiv.

Dacă algoritmul schemă-graf $\mathfrak{A}$ nu conține cicluri și dacă analizăm schema-graf corespunzătoare de la ieșire spre intrare, atunci de fiecare dată cînd conținutul unei celule se schimbă, rangul variabilei corespunzătoare se mărește (de obicei cu o unitate). Dacă x_i^n este starea inițială a celulei x_i, atunci convenim să numim numărul n, *rang maximal* al variabilei x_i^n. Dacă x_i^n este o variabilă cu rang maximal, atunci notăm această variabilă cu x_i^0.

§ 4. CONSTRUIREA FORMULEI COMPLET ATAȘATE UNUI ALGORITM SCHEMĂ-GRAF FĂRĂ CICLURI

În continuare printr-un algoritm schemă-graf $\mathfrak{A}$ vom înțelege întotdeauna un algoritm fără cicluri, adică un algoritm a cărui schemă-graf nu conține cicluri [5].

Presupunem că schema-graf a algoritmului $\mathfrak{A}$ conține $n-1$ operatori în afară de intrare și de ieșire. Introducem următoarea numerotare a operatorilor algoritmului $\mathfrak{A}$:

1°. Notăm cu O_0 intrarea schemei-graf a algoritmului $\mathfrak{A}$.

2°. Dacă succesorul operatorului de atribuire O_i sau a intrării este operatorul O_j, atunci convenim ca $j > i$; dacă succesorii operatorului

logic O_k sînt operatorii O_{k_1} și O_{k_2}, atunci convenim ca $k_1 > k$ și $k_2 > k$.

3°. Notăm cu O_n ieșirea schemei-graf a algoritmului $\mathfrak{A}$.

Operatorii oricărui algoritm fără cicluri pot fi întotdeauna numerotați în așa fel încît să fie satisfăcute condițiile 1°, 2° și 3°. Pentru a demonstra această afirmație, indicăm următorul procedeu de realizare a unei astfel de numerotări:

a) Notăm intrarea schemei-graf cu O_0

b) Presupunem că operatorii $O_0, O_1, \ldots, O_i$ au fost astfel numerotați încît sînt satisfăcute condițiile 1°, 2° și 3° și în nici unul din acești operatori nu intră săgeți care pornesc din operatori nenumerotați. Dacă algoritmul nu conține operatori nenumerotați, atunci numerotarea este terminată și ea satisface condițiile 1°, 2°, și 3°.

c) Dacă algoritmul conține operatori nenumerotați, atunci căutăm printre aceștia unul în care nu intră nici-o săgeată ce pornește din operatori nenumerotați.

Din ipoteza că algoritmul $\mathfrak{A}$ nu conține cicluri, rezultă că întotdeauna găsim un astfel de operator. Într-adevăr, să presupunem că un astfel de operator nu există. Atunci fie O un operator nenumerotat. Operatorul O are cel puțin un predecesor nenumerotat, care la rîndul său are de asemenea un predecesor nenumerotat. Întrucît numărul operatorilor nenumerotați este finit, rezultă că operatorul O aparține unui ciclu. Această concluzie contrazice însă ipoteza că algoritmul $\mathfrak{A}$ nu conține cicluri.

Notăm cu O_{i+1} operatorul în care nu intră nici-o săgeată ce pornește din operatori nenumerotați. După un număr finit de pași, numerotarea operatorilor algoritmului $\mathfrak{A}$ se termină și ea satisface condițiilor 1°, 2° și 3°.

4.1. *Construirea formulei elementare atașate operatorilor de atribuire.* Formula elementară atașată unui operator de atribuire așa cum a fost definită (definiția 5) nu este unică. Vom arăta că dacă este dată o formulă logică φ, ce satisface o anumită condiție și un operator de atribuire $z := g(X)$, atunci relativ la φ întotdeauna se poate construi în mod unic formula elementară atașată operatorului $z := g(X)$.

Vom nota cu $A_\varphi[z := g(X)]$ expresia din membrul drept a formulei elementare atașate operatorului $z := g(X)$, relativ la φ. Prin notația $[\psi]_\omega^x$ înțelegem rezultatul operației de înlocuire a fiecărei intrări libere a variabilei x în ψ, cu expresia ω.

Condiția α. Dacă variabila x_i^s intră liberă în formula ψ și $s \geqslant 2$, atunci ea este unica variabilă, de rang mai mare sau egal cu doi, relativă la celula x_i care intră liberă în ψ.

Fie dat operatorul $z := g(X)$ și formula φ, care satisface condiția α. Expresia $A_\varphi[z := g(X)]$ se construiește în felul următor:

C 1.1. Dacă z este o componentă a vectorului X și dacă există variabila z^s, unde $s \geqslant 2$, care intră liberă în φ, atunci definim

$$A_\varphi^1\lfloor z := g(X)] :: = [g(X)]_{z^{s+1}}^z \quad (4)$$

[1]) Simbolul folosit $::=$ în această formulă precum și în continuare însemnează „este prin definiție''.

C 1.2 Dacă z este o componentă a vectorului X și z^1 este unica variabilă relativ la z care intră liberă în φ sau φ nu conține variabile relativ la z, atunci definim

$$A^1_\varphi [z : = g(X)] : : = [g(X)]^z_{z^1}. \tag{5}$$

C 1.3. Dacă z nu este o componentă a vectorului X, atunci definim

$$A^1_\varphi [z : = g(X)] : : = g(X). \tag{6}$$

C 2. Pentru toate celulele C_i, care intră în expresia $A^1_\varphi [z : = g(X)]$, definim

$$A^2_\varphi [z : = g(X)] : : = [A^1_\varphi [z : = g(X)]]^{c_i}_{c_0} \tag{7}$$

C 3. Pentru toate celulele x_i, care intră în expresia $A^2_\varphi [z : = g(X)]$ și relativ la care nu există nici-o variabilă ce intră în φ, definim

$$A^3_\varphi [z : = g(X)] : : = [A^2_\varphi [z : = g(X)]]^{x_i}_{x^1_i}. \tag{8}$$

C 4. Pentru toate celulele x_i, care intră în expresia $A^3_\varphi [z : = g(X)]$ și relativ la care există variabila $x^{s_i}_i$, unde $s_i \geqslant 2$, care intră liberă în φ, definim

$$A^3_\varphi [z : = g(X)] : : = [A^3_\varphi [z : = g(X)]]^{x_i}_{x^{s_i}_i}. \tag{9}$$

C 5. Pentru toate celulele x_i, care intră în expresia $A^4_\varphi [z : = g(X)]$ și relativ la care există variabilele x^1_i, care intră în membrul drept al unei formule elementare atașate unui operator de atribuire sau într-o formulă elementară atașată unui operator logic, conținute în φ, definim

$$A^5_\varphi [z : = g(X)] : : = [A^4_\varphi [z : = g(X)]]^{x_i}_{x^1_i} \tag{10}$$

C 6. Pentru toate celulele x_i, care intră în expresia $A^5_\varphi [z : = g(X)]$ și relativ la care formula φ conține formule elementare de forma $x^1_i = f(\xi)$, unde x^1_i nu intră în expresia $f(\xi)$, definim

$$A^6_\varphi [z : = g(X)] : : = [A^5_\varphi [z : = g(X)]]^{x_i}_{x^2_i} \tag{11}$$

Expresia $A^6_\varphi [z := g(X)]$ este membrul drept al formulei elementare, relativ la φ, atașate operatorului de atribuire $z : = g(X)$ adică $A_\varphi [z : = g(X)]$. În procesul de construire a expresiei $A_\varphi [z : = g(X)]$ cele 6 condiții descrise mai sus se verifică întotdeauna în ordinea C1, C2, C3, C4, C5, C6.

Din aceste condiții rezultă că, dacă expresia $A_\varphi [z : = g(X)]$ conține o variabilă relativ la celula x, atunci conține numai una la această celulă.

4.2. *Construirea formulei elementare atașate operatorului logic.*

Dacă este dată o formulă φ și operatorul logic $p(X)$, atunci, ca și în cazul operatorului de atribuire, se poate întotdeauna construi în mod unic o formulă elementară, relativ la φ, atașată operatorului $p(X)$. Notăm această formulă cu $L_\varphi [p(X)]$.

Construirea formulei $L_\varphi [p(X)]$ se face în mod analog cu construirea formulei $A_\varphi [z : = f(X)]$, exceptînd doar condiția C4, care se înlocuiește cu următoarea :

C'4. Pentru toate celulele x_i, care intră în expresia $L^3_\varphi [p(x)]$ și relativ la care există variabilele $x^{s_1}_i$, $x^{s_2}_i$, ..., $x^{s_r}_i$, care intră libere în φ și pentru care $s_k \geqslant 2$ $(k = 1, 2, \ldots, r)$, definim

$$L^4_\varphi [p(x)] : : = [L^3_\varphi [p(x)]]^{x_i}_{x^s_i}, \tag{12}$$

unde $s = \max(s_1, s_2, \ldots, s_r)$.

Evident că pentru construirea formulei $L_\varphi [p(x)]$ nu are sens să fie folosită condiția C 1. Și pentru $L_\varphi [p(x)]$ este adevărată proprietatea : dacă $L_\varphi [p(x)]$ conține o variabilă relativă la celula x, atunci conține numai una relativ la această celulă.

4.3. *Operația de egalare a rangurilor* (ER). Operația ER se aplică la numai două formule φ_1 și φ_2, care satisfac condiția α. Ca rezultat al aplicării acestei operații, vom obține două formule $E(\varphi_1, \varphi_2)$ și $E(\varphi_2, \varphi_1)$, în felul următor.

Dacă variabilele x^s și $x^{s'}$ intră libere în formula φ_1 și respectiv φ_2 iar $s \geqslant 2$ și $s > s'$, atunci înlocuim fiecare intrare a variabilei $x^{s'}$ în formula φ_2 cu variabila x^s. Dacă $s' = 1$, înlocuirea se face numai atunci cînd x^1 intră în expresia din membrul drept a unei formule elementare atașate unui operator de atribuire sau într-o formulă elementară atașată unui operator logic, conținute în φ.

Efectuînd toate înlocuirile posibile, notăm formula obținută cu $E(\varphi_2, \varphi_1)$.

Schimbînd între ele formulele φ_1 și φ_2 și făcînd toate înlocuirile posibile în aceleași condiții, obținem al doilea rezultat al operației, adică formula pe care o notăm cu $E(\varphi_1, \varphi_2)$.

După aplicarea operației ER, toate variabilele care intră în formulele $E(\varphi_1, \varphi_2)$ și $E(\varphi_2, \varphi_1)$ și care satisface condițiile aplicării acestei operații, au același rang.

4.4. *Construirea formulei complet atașate algoritmului* $\mathfrak{A}$. Fie dat algoritmul $\mathfrak{A}$, precum și mulțimile N, R și Mc. Construirea formulei complet atașate algoritmului $\mathfrak{A}$ se face inductiv în felul următor :

1°. Atașăm operatorului O_n, adică ieșirii, formula

$$F_n : : = (0 = 0).$$

Este evident că formula F_n satisface condiția α.

2°. Presupunem că fiecărui operator O_j $(j = i + 1, \ldots, n)$ i-am atașat formula F_j, care satisface condiția α. Pentru a construi formula F_i, atașată operatorului O_i, deosebim două cazuri

2.1. Operatorul O_i este operatorul de atribuire

$$y : = f(X)$$

În acest caz presupunem că operatorul O_t este succesorul operatorului O_i. Din ipoteza de inductivitate rezultă că formula F_t este construită, și ea satisface condiția α.

2.1.1. Dacă formula F_t nu conține variabile relativ la celula y și $y \in R$, atunci

(I) $$F_i :: = [y' = A_{F_t}[y := f(X)] \& F_t],$$

iar dacă $y \overline{\in} R$, atunci

(II) $$F_i :: = F_t.$$

Într-adevăr, dacă formula F_t nu conține variabile relativ la celula y, atunci înseamnă că în timpul realizării procesului de calcul, începînd cu operatorul O_t și pînă la sfîrșit, conținutul celulei y rămîne neschimbat. Prin urmare, dacă $y \in R$, atunci starea celulei y, după realizarea operatorului O_i, este o stare finală, ceea ce exprimă formula (I). Dacă $y \overline{\in} R$, înseamnă că rezultatul intermediar obținut după realizarea operatorului O_i este un rezultat de prisos. În acest caz operatorul O_i este un operator de prisos ceea ce exprimă formula (II).

2.1.2. Dacă formula F_t conține variabila y^s, unde $s \geqslant 2$, atunci

(III) $$F_i :: = [F_t]^{y^s}_{A_{F_t}[y := f(X)]}.$$

În acest caz conținutul celulei y obținut după realizarea operatorului O_i se folosește în continuare ca rezultat intermediar.

2.1.3. Dacă formula F_t conține o formulă elementară atașată unui operator logic, în care intră liberă variabila y^1 sau o formulă elementară atașată unui operator de atribuire, în a cărei expresie din membrul drept intră de asemenea variabila y^1 și $y \in R$, atunci

(IV) $$F_i :: = [y^1 = A_{F_t}[y := f(X)]] \& [F_t]^{y^1}_{A_{F_t}[y := f(X)]},$$

iar dacă $y \overline{\in} R$, atunci

(V) $$F_i :: = [F_t]^{y^1}_{A_{F_t}[y := f(X)]}.$$

Din condiția 2.1.3 rezultă că starea celulei y, obținută după executarea operatorului O_i, se folosește în continuare ca un rezultat intermediar. Prin urmare, dacă $y \in R$, atunci starea respectivă este în același timp și un rezultat final, ceea ce exprimă formula (IV). Dacă $y \overline{\in} R$, atunci starea corespunzătoare a celulei y reprezintă numai un rezultat intermediar (formula (V)).

2.2. Operatorul O_i este operatorul logic $p(X)$. Presupunem că operatorul O_{t_1} este succesorul O_i, urmînd săgeata care pleacă din acesta și este marcată cu „+", iar operatorul O_{t_2} este succesorul operatorului O_i, urmînd săgeata care pleacă din aceasta și este marcată cu „—". Din ipoteza de inductivitate rezultă că formulele F_{t_1} și F_{t_2} atașate operatorilor O_t, și relativ la O_{t_2}, sînt construite și ele satisfac condiția α. Notăm cu φ formula $F_{t_1} \& F_{t_2}$ și definim

(VI) $$F_i :: = [L_\varphi[p(X)] \supset E(F_{t_1}, F_{t_2})] \& [\neg L_\varphi[p(X)] \supset E(F_{t_2}, F_{t_2})].$$

Vom demonstra că formula F_i, construită prin metoda indicată, satisface condiția α.

Presupunem că formula F_i are forma (I). În acest caz expresia $A_{F_t}[y := f(X)]$ poate fi obținută cu ajutorul formulelor (9) sau (11). Presupunem că F_i nu satisface condiția α. Atunci există variabilele x^s și $x^{s'}$ care intră în F_i și pentru care avem $s \geqslant 2$, $s' \geqslant 2$ și $s \neq s'$. Întrucît prin ipoteza inductivă am presupus că F_t satisface condiția α, rezultă că în formula F_t intră numai una din aceste formule. Presupunem că $x^{s'}$ intră în F_t și x^s intră în expresia $A_{F_t}[y := f(X)]$. Însă, dacă expresia $A_{F_t}[y := f(X)]$ a fost construită cu ajutorul formulei (9), atunci $s = s'$, iar dacă această expresie a fost construită cu ajutorul formulei (11), atunci $s = 2$ și $s' = 1$. Concluziile la care am ajuns contrazic ipoteza că $s' = s$ și $s' \geqslant 2$. Prin urmare, formula F_i de forma (I) satisface condiția α.

Presupunem că formula F_i are forma (III). În acest caz expresia $A_{F_t}[y := f(X)]$ se obține aplicînd formulele de construire a acestei expresii, începînd cu formula (4). Din aceasta rezultă că unica variabilă relativ la celula y și care intră în expresia $A_{F_t}[y := f(X)]$ este y^{s+1}, unde $s \geqslant 2$. Atunci singura variabilă de rang mai mare sau egal cu doi, care intră în formula F_t, relativ la celula y, este y^{s+1}. Prin urmare, relativ la stările celulei y, formula F_i de forma (III) satisface condiția α. Demonstrația că formula F_i de forma (III) satisface condiția α și relativ la restul variabilelor se face la fel ca și în cazul cînd formula F_i este de forma (I).

Presupunem că formula F_i are forma (IV) sau (V). În acest caz expresia $A_{F_t}[y := f(X)]$ se obține aplicînd formulele corespunzătoare, începînd cu formula (5). Din aceasta rezultă că unica variabilă relativă la celula y și care intră în expresia $A_{F_t}[y := f(X)]$ este y^2. Prin urmare, formula F_i nu conține alte variabile relative la celula y, decît pe y^2 și y^1. Înseamnă deci că relativ la stările celulei y, formula F_i satisface condiția α. Demonstrația că formula F_i satisface condiția α și relativ la restul variabilelor se face la fel ca și în cazul cînd F_i este de forma (I).

Presupunem că F_i este de forma (VI) și ea nu satisface condiția α. Atunci înseamnă că există variabilele x^s și $x^{s'}$, care intră în formula F_i și pentru care $s \neq s'$, $s \geqslant s$ și $s' \geqslant 2$. Din definiția operației ER rezultă că fiecare dintre formulele $E(F_{t_2}, F_{t_1})$ și $E(F_{t_1}, F_{t_2})$ poate conține numai una din aceste variabile. Fie, de exemplu, variabila x^s conținută în formula elementară $L_\varphi[p(X)]$ și variabila $x^{s'}$ conținută în formula $E(F_{t_1}, F_{t_2})$. Din definiția operației ER rezultă că variabila $x^{s'}$ intră numai în formula F_{t_1} și că nu există o altă variabilă $x^{s''}$, unde $s'' > s'$, care să intre în formula F_{t_2}. Din aplicarea formulei (12) pentru construirea expresiei $L_\varphi[p(X)]$ rezultă că $s = s'$, sau dacă $s = 2$,

atunci formula $L_\varphi [p(X)]$ poate conține variabila x^s numai ca rezultat al aplicării formulei (11), și în acest caz $s' = 1$. Ambele aceste cazuri contrazic ipoteza că formula F_i de forma (VI) nu satisface condiția α.

3°. După $n+1$ pași obținem formula F_0 care satisface condiția α.

Din faptul că formula F_0 satisface condiția α, rezultă că dacă ea conține două variabile relativ la aceiași celulă, atunci una dintre ele are rangul egal cu unitatea. Prin urmare, dacă $x_{i_1}^{\rho_1}, x_{i_2}^{\rho_2}, \ldots, x_{i_p}^{\rho_p}$, unde $\rho_j \geqslant 2$ $(j = 1, 2, \ldots, p)$ sînt toate variabile care intră în F_0, atunci ρ_i este rangul maximal al variabilei $x_{i_j}^{\rho_i}$. Conform convenției făcute, $x_{i_j}^{\rho_j}$ reprezintă conținutul inițial al celulei x_{i_j} și acesta se notează cu $x_{i_j}^0$.

Înlocuim în formula F_0 toate variabilele $x_{i_k}^{\rho_k}$, unde $\rho_k \geqslant 2$ cu variabilele $x_{i_k}^0$. După această înlocuire, formula obținută conține numai variabile cu rang zero sau unu. Notăm această formulă cu $F(X^0, X^1)$. Toate variabilele de rang egal cu unitatea care intră în $F(X^0, X^1)$ reprezintă rezultate finale ale algoritmului $\mathfrak{A}$. Într-adevăr, dacă variabila x^1 ar figura în expresia din membrul drept al unei formule elementare atașate unui operator de atribuire sau într-o formulă elementară atașată unui operator logic, atunci $x \in M_c$. Această concluzie nu poate fi adevărată deoarece din regulile de construire a expresiilor $A_\varphi [z := g(X)]$ și $L_\varphi [p(x)]$ rezultă că în acest caz formula F_0 trebuie să conțină pe x^0 și nu pe x^1, așa cum am presupus noi. Prin urmare intrarea variabilei x^1 în formula $F(X^0, X^1)$ este membrul stîng al unei formule elementare atașate unui operator de atribuire. Componentele vectorului X^1 sînt deci elemente ale mulțimii R^1, iar componentele vectorului X^0 sînt elemente ale mulțimii I^0.

TEOREMĂ. *Formula $F(X^0, X^1)$ este o formulă complet atașată algoritmului $\mathfrak{A}$.*

Pentru a demonstra această teoremă introducem următoarele notații.

Notăm cu $\mathfrak{A}_i$ algoritmul schemă-graf dat de schema-graf prin interpretarea $\{M_i, \int; T \to f; R \to p\}$, unde M_i este acea parte a memoriei (poate fi chiar memoria M) care se folosește la realizarea algoritmului $\mathfrak{A}$, începînd cu îndeplinirea operatorului O_i. Schema-graf Γ_i constă din operatorii $O_i, O_{i_1}, \ldots, O_{i_r}$, unde O_{i_r} este ieșirea algoritmului $\mathfrak{A}$, și fiecare operator O_{i_k} $(k = 1, 2, \ldots, r)$ este succesorul cel puțin unuia dintre operatorii $O_i, O_{i_1}, \ldots, O_{i_{k-1}}$. Dacă este dată schema-graf a algoritmului $\mathfrak{A}$, atunci întotdeauna se poate construi schema-graf Γ_i prin înlăturarea tuturor operatorilor algoritmului $\mathfrak{A}$ în afară de operatorii $O_i, O_{i_1}, \ldots, O_{i_r}$, și a săgeților care pornesc din operatorii înlăturați.

Vom nota cu ξ_i^0 și cu ξ_i^1, datele inițiale și respectiv rezultatele finale ale algoritmului $\mathfrak{A}_i$. Ca date inițiale pentru algoritmul $\mathfrak{A}_i$ vom considera acele date inițiale ale algoritmului $\mathfrak{A}$ și acele rezultate intermediare care au fost obținute prin realizarea procesului de calcul pînă la îndeplinirea operatorului O_i, care se folosesc la obținerea rezultatelor finale începînd cu îndeplinirea operatorului O_i. Ca rezultate finale ale algoritmului $\mathfrak{A}_i$ vom considera acele rezultate finale ale algoritmului $\mathfrak{A}$ care se obțin prin realizarea acestui algoritm începînd cu îndeplinirea operatorului O_i.

Notăm cu $F_i^*(\xi_i^0, \xi_i^1)$ formula obținută după înlocuirea în F_i a datelor inițiale ale algoritmului $\mathfrak{A}_i$ cu vectorul ξ_i^0 și a rezultatelor finale cu vectorul ξ_i^1.

Vom demonstra că formula $F_i^*(\xi_i^0, \xi_i^1)$ este o formulă complet atașată algoritmului $\mathfrak{A}_i$.

a) Formula F_n^* este o formulă complet atașată algoritmului $\mathfrak{A}_n$. Într-adevăr, algoritmul $\mathfrak{A}_n$ este algoritmul vid care constă din realizarea operatorului O_n, adică a ieșirii algoritmului $\mathfrak{A}$. Formula complet atașată acestui algoritm este $\xi_n^0 = \xi_n^1$, unde $\xi_n^0 := \xi_n^1 := 0$ și, prin urmare, formula $F_n^* := (0 = 0)$ este complet atașată algoritmului $\mathfrak{A}_n$.

b) Presupunem că formula F_n^* $(j = i+1, \ldots, n)$ este complet atașată algoritmului $\mathfrak{A}_n$.

1°. Presupunem că F_i are forma (I), unde F_t nu conține variabile relativ la celula y. Atunci y^1 este un rezultat final al algoritmului $\mathfrak{A}$. Rezultatele finale ale algoritmului $\mathfrak{A}_i$ vor fi y^1 și ξ_t^1, unde ξ_t^1 sînt rezultatele finale ale algoritmului $\mathfrak{A}_t$. Notăm cu $\overline{\xi}_i^0$ acele date inițiale ale algoritmului $\mathfrak{A}_i$ care sînt folosite pentru obținerea rezultatului y^1. Formula $F_i^*(\xi_i^0, \xi_i^1)$ va avea următoarea formă

$$(y^1 = f(\overline{\xi}_i^0)) \,\&\, F_t^*(\xi_{it}^0\ \xi_t^1). \tag{13}$$

Presupunem că formula (13) este adevărată. Atunci $y^1 = f(\overline{\xi}_i^0)$ și $F_i(\xi_t^0, \xi_t')$ sînt formule adevărate și prin urmare $\mathfrak{A}_t(\xi_t^0) = \xi_t'$ iar prin realizarea operatorului O_i, datele inițiale $\overline{\xi}_i^0$ se transformă în rezultatul y^1. Prin urmare

$$\mathfrak{A}_i(\xi)_i^0 = \xi_t^1.$$

Presupunem că $\mathfrak{A}_i(\xi_i^0) = \xi_t^1$. Atunci, prin realizarea operatorului O_i, datele inițiale $\overline{\xi}_i^0$ sînt transformate în rezultatul y^1 și avem de asemenea $\mathfrak{A}_t(\xi_t^0) = \xi_t'$. Prin urmare, formulele $y' = f(\overline{\xi}_i^0)$ și $F_t^*(\xi_t^0, \xi_t^1)$ sînt adevărate, deci și formula (13) este adevărată.

2°. Presupunem că formula F_i este de forma (III), unde F_t conține variabila y^s, $s \geqslant 2$. În acest caz y^s este un rezultat intermediar al algoritmului $\mathfrak{A}_i$.

Presupunem că formula F_i^* este adevărată. Atunci formulele $y^s = f(\overline{\xi}_i^0)$ și F_t^* sînt adevărate. Prin urmare, $\mathfrak{A}_t(\xi_t^0) = \xi_t^1$, unde componenta y^s a vectorului ξ_t^0 a fost înlocuită cu expresia $f(\overline{\xi}_i^0)$. Din aceasta rezultă că $\mathfrak{A}_i(\xi_i^0) = \xi_i^1$, unde $\xi_i^1 = \xi_t^1$ iar datele inițiale ale algoritmului $\mathfrak{A}$ sînt $\overline{\xi}_i^0$ și ξ_t^0.

Presupunem că $\mathfrak{A}_i(\xi_i^0) = \xi_i^1$. Atunci $\mathfrak{A}_t(\xi_t^0) = \xi_t'$ și după îndeplinirea operatorului O_i, $\overline{\xi}_i^0$ se transformă în y^s. Prin urmare formulele $F_t^*(\xi_t^0, \xi_t^1)$ și $y^s = f(\overline{\xi}_i^0)$ sînt adevărate. Din aceasta rezultă că și formula F_i^* este adevărată.

3°. Dacă formula F_i este de forma (IV) sau (V), atunci demonstrarea teoremei se face în mod analog ca și în cazurile 1° și 2°.

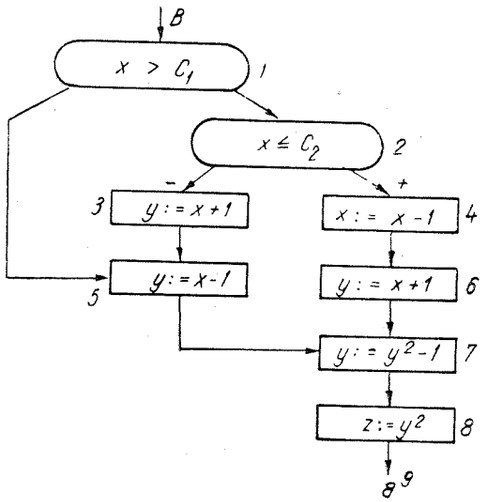
4°. Presupunem că F_i este de forma (VI), unde F_t și F_{t_2} sînt formule complet atașate algoritmilor $\mathfrak{A}_{t_1}$ și respectiv $\mathfrak{A}_{t_2}$.

Presupunem că formula F_i^*, corespunzătoare lui F_i, este adevărată. Atunci sînt adevărate formulele $p(\overline{\xi}_i^0) \supset E\,[F_{t_1}, F_{t_2}]$ și $\neg\, p(\overline{\xi}_i^0) \supset E(F_{t_2}, F_{t_1})$. Fie $p(\overline{\xi}_i^0)$ o formulă adevărată. Atunci $E(F_{t_1}, F_{t_2})$ este adevărată și prin urmare F_{t_1} este adevărată. Rezultă de aici că $\mathfrak{A}_{t_1}(\xi_{t_1}^0) = \xi_{t_1}^1$. Atunci $\mathfrak{A}_i(\xi_i^0) = \xi_i^1$, unde $\xi_i^1 = \xi_{t_1}^1$ și datele inițiale ale algoritmului $\mathfrak{A}_i$ sînt $\overline{\xi}_i^0$ și $\xi_{t_1}^0$. Dacă presupunem că $\neg\, p(\overline{\xi}_i^0)$ este adevărată, atunci ajungem la concluzia că $\mathfrak{A}_i(\xi_i^0) = \xi_i^1$, unde $\xi_i^1 = \xi_{t_2}^1$ iar datele inițiale ale algoritmului $\mathfrak{A}_i$ sînt $\overline{\xi}_i^0$ și $\xi_{t_2}^0$.

Presupunem că $\mathfrak{A}_i(\xi_i) = \xi_i^1$. În acest caz sau formula $p(\overline{\xi}_i^0)$ este adevărată și $\mathfrak{A}_{t_1}(\xi_{t_1}^0) = \xi_{t_1}^1$ sau formula $\neg\, p(\overline{\xi}_i^0)$ este adevărată și $\mathfrak{A}_{t_2}(\xi_{t_2}^0) = \xi_{t_2}^1$. În ambele cazuri rezultă imediat că formula F_i^* este adevărată.

Am demonstrat în felul acesta teorema enunțată pentru formula $F_i^*(\xi_i^0, \xi_i^1)$ și algoritmul $\mathfrak{A}_i$. Teorema este deci adevărată și pentru formula F_0 și algoritmul $\mathfrak{A}_0$, care este algoritmul $\mathfrak{A}$.

Exemplu. Să considerăm următorul algoritm schemă-graf (figură). În acest caz $N = \{x\}$, $R = \{x\}$, $M_c = \{c_1, c_2\}$.



Aplicînd metoda de construire a formulei complet atașate acestui algoritm obținem

$$F_9 :: = (0 = 0)$$
$$F_8 :: = (z^1 = (y^1)^2)$$
$$F_7 :: = (z^1 = ((y^2)^2 - 1)^2)$$
$$F_6 :: = (z^1 = ((x^1 + 1)^2 - 1)^2)$$
$$F_5 :: = (z^1 = ((x^1 - 1)^2 - 1)^2)$$
$$F_4 :: = (z^1 = ((x^2)^2 - 1)^2)$$

$$F_3 :: = (z^1 = ((x^1 - 1)^2 - 1)^2)$$
$$F_2 :: = [(x^2 \leqslant c_2^0) \supset F_4] \,\&\, [\neg\, (x^2 \leqslant c_2^0) \supset [F_3]_{x^2}^{x^1}]$$
$$F_1 :: = [(x_1^2 > c_1^0) \supset F_2] \,\&\, [\neg\, (x_1^2 > c_1^0) \supset [F_5]_{x^2}^{x^1}]$$
$$F_0 :: = F_1.$$

După înlocuirea variabilei x^2 cu variabila x^0 obținem formula

$$\{(x^0 > c_1^0) \supset [[(x^0 \leqslant c_2^0) \supset (z^1 = ((x^0)^2 - 1)^2)] \,\&\, [\neg\, (x^0 \leqslant c_2) \supset$$
$$\supset (z^1 = ((x^0 - 1)^2 - 1)^2)]]\} \,\&\, \{\neg\, (x^0 > c_1^0) \supset (z' = ((x^0)^2 - 1)^2)\},$$

care este formula complet atașată algoritmului considerat.

Primită la redacție la 9 decembrie 1965

Institutul de calcul al Academiei
Republicii Socialiste România — Filiala Cluj

BIBLIOGRAFIE

1. M. R. Karp, *A note on the application of graph theory to digital computer programmin*, Inf. and Control, **3**, 2 170—190 (1960).
2. T. R. Prosser, *Application of Boolean matrices to the analysis of flow diagrams*. Proc. Eastern Joint Computer Conf., 1959.
3. В. В. Мартинюк, *О некоторых методах анализа операторных схем*, Автореферат, МГУ, 1963.
4. Л. А. Калугин, *Об алгоритмизации математических задач*. Проблемы кибернетики, вып. **3** (1959).
5. C. Berge, *Theorie des Graphes et ses applications*. Dunod, Paris, 1958.
6. Е. Мунтяну, *Метод анализа граф-схемных алгорифтов*, Matematica, **5/28**/, 2 (1963).