

UN ALGORITM DE DECIDERE ÎN TEORIA ARBORILOR ¹⁾

DE

T. RUS

Institutul de calcul Cluj

În lucrare se definește noțiunea de „arbori echivalenți față de o mulțime de operații de substituții și permutări” (operații care se definesc în lucrare). Se definește de asemenea forma normală a unui arbore ale cărui noduri sînt marcate de simbolurile unei mulțimi date. Algoritmul de decizie asupra echivalenței a doi arbori este format din două părți:

1. Se transformă întii arborii în formele lor normale.
2. Se verifică dacă aceste forme normale coincid. Dacă formele normale coincid, arborii sînt echivalenți.

Algoritmul este în așa fel expus încît se poate folosi (traducîndu-l în limba unei mașini cifrice de calcul) pentru executarea unor operații algebrice cu mașina de calcul, cum ar fi de exemplu ordonarea unui polinom. Să dă de asemenea algoritmul de construire al polinomului derivat al unui polinom, cu ajutorul mașinii de calcul, folosind algoritmul despre care vorbeam mai sus.

1. Noțiunea de arbore a căpătat în ultimul timp o folosire tot mai largă în caracterizarea diferitelor procese de calcul în legătură cu mecanizarea executării acestor procese. Astfel în lucrarea [4] se arată un mod de prezentare cu ajutorul arborilor a simbolismului matematic folosit în rezolvarea cu mașini electronice de calcul a diferitelor procese de calcul aritmetice și nearitmetice. Pentru claritate, vom spune despre un proces de calcul că este nearitmetic dacă datele lui inițiale nu cuprind numai mărimi numerice, ci și de altă natură, iar rezultatul nu apare neapărat sub forma unui număr sau a unui sistem de numere.

Într-o serie de lucrări de lingvistică matematică — mai ales în analiza structurală — teoria arborilor începe să-și găsească un cîmp de aplicație tot mai larg, după cum se vede din lucrările [1], [3]. Teoria arborilor ca

¹⁾ Lucrare prezentată la ședința de comunicări a Institutului de calcul Cluj din 21 ianuarie 1964.

un capitol separat este relativ slab dezvoltată. Rezultatele generale care se folosesc sînt cele din teoria grafelor, arborele fiind un graf particular care nu conține cicluri. În lucrarea [2] se pun cîteva probleme generale asupra structurii arborilor și a operațiilor care se introduc asupra unui arbore. Astfel se demonstrează că față de o mulțime dată de transformări ale unui arbore dat, există un algoritm care permite să decidem dacă doi arbori, a_1 și a_2 , sînt sau nu echivalenți, adică dacă unul din ei poate fi obținut din celălalt prin aplicarea unui număr finit de operații precizate. Practic este important de cunoscut un asemenea algoritm și dacă este cunoscut este important ca să fie executat de o mașină electronică de calcul. Acest lucru ar permite ca o bună parte din munca de formalizare asupra anumitor probleme din domeniile prezentate mai sus să fie efectuată de mașinile electronice de calcul.

În această Notă ne vom referi la lucrările [5] și vom particulariza sistemul formal construit în aceste lucrări dînd cu ajutorul arborilor un algoritm de deciderie asupra echivalenței cuvintelor compatibile în sistemul formal particularizat. Acest algoritm va permite ca anumite calcule algebrice să poată fi executate de o mașină electronică de calcul. Aceasta constituie un prim pas în procesul de folosire efectivă a mașinilor electronice de calcul în rezolvarea unor probleme nearitmetice cum sînt derivarea și integrarea analitică a funcțiilor definite cu ajutorul formulelor.

2. Să considerăm sistemul formal care are ca alfabet mulțimea $\mathfrak{R} = \{\oplus, \otimes, \oplus, *, 0, 1, \dots, 9, x\}$, unde $\oplus, \otimes, \oplus$ sînt simbolurile operațiilor de adunare, înmulțire și ridicare la putere, iar $0, 1, \dots, 9$ sînt constantele date, iar x este o variabilă. Regulile semantice pentru simbolurile introduse sînt cele obișnuite în algebra elementară. Vom presupune de asemenea că orice cuvînt în alfabetul $\{0, 1, \dots, 9\}$ este tot o constantă. Dacă notăm cu p o constantă oarecare în alfabetul $\{0, 1, \dots, 9\}$, atunci ea va fi numită constantă pozitivă. Oricare ar fi constanta pozitivă p , cuvîntul $q = *p$ în alfabetul $\{*, 0, 1, \dots, 9\}$ va fi numit constantă negativă. Asupra simbolurilor $\oplus, \otimes, \oplus$ vom face următoarele ipoteze:

I. Dacă p și q sînt constante, atunci și cuvîntul $(\oplus, p, q)$ este o constantă și anume constantă obținută prin aplicarea operației $\oplus$ asupra constantelor p, q în sens aritmetic obișnuit.

II. Dacă p și q sînt constante, atunci și cuvîntul $(\otimes, p, q)$ este constantă și anume constantă obținută prin aplicarea operației $\otimes$ asupra constantelor p și q în sens aritmetic obișnuit.

III. Dacă p este o constantă pozitivă și q o constantă oarecare, atunci și cuvîntul $(\oplus, p, q)$ este o constantă și anume constantă obținută prin aplicarea operației de ridicare la puterea p a constantei q din aritmetica obișnuită.

De asemenea presupunem că avem un algoritm de obținere a constantei C_p^q cu ajutorul operațiilor introduse. Constanta C_p^q are sensul combinărilor de p elemente luate cîte q .

În cele ce urmează asupra simbolurilor mulțimii $\mathfrak{R}$ și asupra cuvintelor în alfabetul $\{*, 0, 1, \dots, 9\}$ definim o funcție în felul următor. Dacă $s \in \mathfrak{R}$, atunci

$$\delta(s) = \begin{cases} p, & p \geq 2 \text{ dacă } s = \oplus \text{ sau } \otimes \\ q, & q = 2 \text{ dacă } s = \pi \\ 0 & \text{dacă } s \neq \oplus, \otimes, \pi, \end{cases}$$

p și q fiind numere întregi.

Această funcție o vom numi funcție grad în sensul introdus în [4], deoarece are ca valoare numărul operandilor pe care poate să-i lege simbolul unei operații. Pentru consecvență ea a fost extinsă și asupra simbolurilor care nu sînt operații. În cele ce urmează asupra variabilei x nu facem ipoteze, considerînd că ea a fost precizată în lucrarea [5].

Constantele și variabilele definite mai sus le vom numi în continuare elemente de bază. Vom atașa fiecărui element de bază un arbore elementar format dintr-o singură rădăcină [4] marcată de simbolul elementului de bază respectiv.

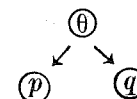
Noțiunea de arbore o vom introduce recursiv cu ajutorul următoarei definiții:

DEFINIȚIA 1. 1) Oricare arbore elementar este un arbore.

2) Dacă a_1 și a_2 sînt arbori, atunci și $a = (\oplus, a_1, a_2)$ și $a = (\otimes, a_1, a_2)$ sînt arbori care au ca rădăcină simbolul $\oplus$ respectiv $\otimes$ de la care pleacă două ramuri, una care unește rădăcina arborelui a cu rădăcina arborelui a_1 , iar cealaltă care unește rădăcina arborelui a cu rădăcina arborelui a_2 .

3) Dacă p este un arbore elementar corespunzător unei constante pozitive și a_1 un arbore, atunci și $a = (\oplus, p, a_1)$ este un arbore care are ca rădăcină simbolul $\oplus$ de la care pleacă două ramuri, una care unește rădăcina arborelui a_1 cu rădăcina arborelui p , iar alta care unește rădăcina arborelui a cu rădăcina arborelui p .

Dacă notăm cu $\oplus$ și $\otimes$ arborii elementari atașați constantelor p și q , atunci un arbore obținut din cei doi arbori elementari după procedeul descris mai sus poate fi reprezentat sub forma



unde θ este unul din simbolurile operatorilor. În continuare însă vom folosi notația liniară a arborilor, pentru că în felul acesta se face economie de spațiu. Astfel arborele de mai sus îl vom nota cu (θ, p, q) .

Asupra arborilor definiți mai sus facem următoarele ipoteze:

I₁. Oricare ar fi arborii a_1, a_2 are loc relația

$$(\oplus, a_1, a_2) = (\oplus, a_2, a_1).$$

I₂. Dacă a_1, a_2, a_3 sînt arbori, atunci

$$(\oplus, a_1, (\oplus, a_2, a_3)) = (\oplus, (\oplus, a_1, a_2), a_3) = (\oplus, a_1, a_2, a_3).$$

I₃. Dacă a este un arbore, atunci există în mulțimea arborilor elementari un arbore Λ_+ cu proprietatea că $(\oplus, a, \Lambda_+) = (\oplus, \Lambda_+, a) = a$. Arborele Λ_+ îl vom numi arbore neutru relativ la simbolul $\oplus$.

I₄. Dacă a este un arbore, atunci există un arbore a_1 cu proprietatea că $(\oplus, a, a_1) = (\oplus, a_1, a) = \Lambda_+$.

II₁. Oricare ar fi arborii a_1, a_2 are loc relația

$$(\otimes, a_1, a_2) = (\otimes, a_2, a_1).$$

II₂. Dacă a_1, a_2, a_3 sînt arbori, atunci

$$(\otimes, a_1, (\otimes, a_2, a_3)) = (\otimes, (\otimes, a_1, a_2), a_3) = (\otimes, a_1, a_2, a_3).$$

II₃. Dacă a este un arbore, atunci există în mulțimea arborilor elementari un arbore $\Lambda_\times$ cu proprietatea că

$$(\otimes, a, \Lambda_\times) = (\otimes, \Lambda_\times, a) = a.$$

III₁. Dacă a_1, a_2, a_3 sînt arbori, atunci are loc relația

$$(\otimes, a_1, (\oplus, a_2, a_3)) = (\oplus, (\oplus, a_1, a_2), (\otimes, a_1, a_3)).$$

IV₁. Dacă p_1 și p_2 sînt arbori corespunzători a două constante pozitive iar a este un arbore, atunci are loc relația

$$(\otimes, (\oplus, p_1, a), (\pi, p_2, a)) = (\oplus, (\oplus, p_1, p_2), a).$$

Dacă constantele corespunzătoare lui p_1 și p_2 sînt egale cu p în sens algebric, atunci avem și

$$(\oplus, (\pi, p, a), (\pi, p, a)) = (\oplus, 2, (\pi, p, a)).$$

IV₁. Există în mulțimea arborilor elementari arborii Λ_π și $\Lambda_{\pi'}$ cu proprietățile că oricare ar fi arborele a au loc relațiile

$$(\oplus, \Lambda_\pi, a) = a; (\oplus, \Lambda_{\pi'}, a) = \Lambda_\pi.$$

Din condițiile de mai sus rezultă că dacă p este un arbore elementar corespunzător unei constante pozitive p și dacă a_1 și a_2 sînt arbori oarecare, atunci are loc relația

$$(\oplus, p_1, (\oplus, a_1, a_2)) = (\oplus, \bar{a}_0, \bar{a}_1, \dots, \bar{a}_p),$$

unde

$$\bar{a}_i = (\otimes, c_p^i, (\oplus, (\oplus, p, *i, a_1), (\oplus, i, a_2))); i = 0, 1, \dots, p,$$

unde prin i am notat arborele elementar corespunzător constantei i iar prin C_p^i arborele elementar corespunzător constantei combinațiilor.

Simbolul egalității introdus mai sus trebuie considerat ca egalitate în sens grafic, el nefăcînd parte din mulțimea $\mathfrak{R}$. În mulțimea arborilor definiți mai sus cu proprietățile I—IV vom introduce două operații. Prima este operația de permutare pe care o definim în felul următor fie arborele $a = (\theta, a_1, a_2, \dots, a_k)$, $k \geq 2$. Deoarece $\delta(\pi) = 2$ și nu există

pentru π o proprietate analogă proprietăților I₁ și II₂, operația de permutare se va defini numai pentru cazul cînd arborele a pus în evidență mai sus are rădăcina θ , simbolul $\oplus$ sau $\otimes$. Arborii $a_1, \dots, a_k$ vor fi numiți subarbori principali ai arborelui a . Fie atunci $\Phi = (\Phi_1, \dots, \Phi_k)$ o permutare a șirului $\{1, 2, \dots, k\}$. Arborele a se transformă atunci prin aplicarea operației de permutare P_Φ în arborele echivalent cu el, a' , astfel $a' = P_\Phi(a) = (\theta, a_{\Phi_1}, a_{\Phi_2}, \dots, a_{\Phi_k})$.

Să considerăm acum grupul G al permutărilor Φ_k , $k \geq 2$ al elementelor $\{1, 2, \dots, k\}$. O P -transformare a arborelui a este o aplicație a transformării P_{Φ_k} a permutării Φ_k a grupului G asupra arborelui a avînd k subarbori principali. Evident că o P -transformare se poate aplica și asupra unui subarbore oarecare al arborelui a .

A doua operație care se introduce asupra unui arbore este operația de substituție notată cu S . Fie a un arbore oarecare cu rădăcina θ iar a_1 un subarbore principal al său. Prin substituția arborelui a_1 cu arborele a'_1 vom înțelege înlocuirea subarborelui a_1 din arborele a cu arborele a'_1 , identificînd rădăcina lui a_1 cu rădăcina lui a'_1 . Dacă notăm cu $S_1 = (a_1 \rightarrow a'_1)$ operația de substituție, atunci dacă $a = (\theta, a_1, a_2, \dots, a_k)$, $k \geq 2$, $T_{S_1}(a) = (\theta, a'_1, a_2, \dots, a_k)$. Dacă a_j se consideră ca arbore, atunci operația poate fi aplicată și pentru a_j și în general pentru oricare subarbore al arborelui a .

Fie acum S o mulțime de perechi de arbori $\{a_i, a'_i\}$. O S -transformare a arborelui a este o aplicare a substituției $(a_i \rightarrow a'_i)$ sau $(a'_i \rightarrow a_i)$ într-un nod oarecare al arborelui a .

DEFINIȚIA 2. Doi arbori se numesc S -echivalenți dacă unul poate fi obținut din celălalt prin aplicarea unui număr finit de S -transformări.

DEFINIȚIA 3. Doi arbori se numesc P -echivalenți dacă unul poate fi obținut din celălalt prin aplicarea unui număr finit de P -transformări.

DEFINIȚIA 4. Doi arbori se numesc echivalenți relativ la o mulțime dată de substituții și permutări dacă unul se poate obține din celălalt prin aplicarea unui număr finit de S -transformări și P -transformări.

Este evident că atît definiția dată mai sus arborilor, cît și operațiile asupra lor precum și definițiile echivalenței sînt valabile și în cazul general [2] cînd se consideră ca arbori elementari rădăcinile corespunzătoare, în general, ale elementelor de bază ale unui sistem formal [5] iar ca noduri [4] oricare din elementele operatorilor sau a relațiilor. Precizarea rădăcinilor și nodurilor prin considerarea mulțimii $\mathfrak{R}$ ne este utilă în acest caz numai pentru a construi efectiv un algoritm de decizie care să permită, să spunem, dacă doi arbori sînt sau nu echivalenți față de o mulțime dată de substituții și permutări.

În continuare va fi comod să considerăm arborii elementari a ca arbori de structura $(\otimes, (\oplus, \Lambda, a), a_1)$, unde $\Lambda = \Lambda_\pi$ dacă a este arborele elementar corespunzător unei variabile și $\Lambda = \Lambda_{\pi'}$ dacă a este arborele elementar corespunzător unei constante. Dacă a este arborele elementar corespunzător unei constante, atunci a_1 este chiar constanta respectivă. În caz contrar $a_1 = \Lambda_\pi$. În continuare vom nota cu 0 pe $\Lambda_{\pi'}$ și cu 1 pe Λ_π .

Vom construi acum un algoritm $\mathfrak{A}$ care ne va permite să decidem dacă doi arbori oarecare a_1 și a_2 sînt sau nu echivalenți în sensul introdus mai sus. Pentru aceasta vom defini forma normală a unui arbore a și vom construi algoritmul de transformare al arborelui a în arborele corespunzător lui pus sub formă normală. Doi arbori vor fi echivalenți dacă formele lor normale sînt egale în sens grafic.

DEFINIȚIA 5. Arborele a se zice pus sub formă normală dacă are forma $a = (\oplus, a_0, \dots, a_k)$, unde $a_i = (\otimes, (\overline{\otimes}, i, x), p_i)$, $i = 0, 1, \dots, k$, x fiind arborele elementar corespunzător variabilei x , p_i fiind arborele elementar corespunzător constantei p_i , iar i este arborele elementar corespunzător constantei i .

În cele ce urmează vom numi nucleu orice arbore de structura $(\otimes, (\overline{\otimes}, p_1, x) q_1)$, unde p_1, q_1 sînt arbori elementari corespunzători constantelor p_1, q_1 ; p_1 numindu-se gradul nucleului iar q_1 coeficientul său.

Vom aplica acum arborelui a un proces de omogenizare care va consta în transformarea lui în așa fel încît fiecare vîrf al său să fie un vîrf al unui oarecare nucleu. Această transformare este posibilă deoarece se observă că dacă $\overline{\otimes}$ este arborele elementar corespunzător constantei p , atunci acesta este echivalent cu nucleul $(\otimes, (\overline{\otimes}, 0, x), p)$, iar dacă p este arborele elementar corespunzător variabilei x , atunci acesta este echivalent cu nucleul $(\otimes, (\overline{\otimes}, 1, x), 1)$. Echivalența în acest caz trebuie considerată din punct de vedere al echivalenței cuvintelor în sistemul nostru, reprezentate de cei doi arbori, în ipoteza interpretării simbolurilor π și $\otimes$ ca simbolurile operațiilor obișnuite de ridicare la putere și înmulțire. Procesul de omogenizare despre care a fost vorba mai sus constă în următoarele:

Se examinează arborele a pe nivele. Fie, de exemplu, nivelul i . Examinarea acestui nivel constă în următoarele: se consideră toate nodurile de pe acest nivel și anume $(\theta_1), (\theta_2), \dots, (\theta_j)$ care se pot presupune marcate de simboluri de operatori. Fie (θ_k) unul dintre aceste noduri. Dacă pe nivelul $i + 1$ subarborele care are pe (θ_k) ca rădăcină conține cel puțin

un vîrf, atunci se consideră acest subarbore și se procedează cu el în felul următor: dacă vîrfurile respectiv aparțin unui nucleu, atunci nu se produce nici o schimbare. Dacă însă acest vîrf nu aparține unui nucleu, atunci el se înlocuiește cu nucleul echivalent cu el, corespunzător. Aceasta este valabilă pentru toate vîrfurile subarborelui cu rădăcină (θ_k) de pe

nivelul $i + 1$. În mod analog se va proceda pentru $k = 1, 2, \dots, j$. O dată epuizați toți subarborii cu rădăcinile pe nivelul i se trece la subarborii cu rădăcinile pe nivelul $i + 1$ și procesul se continuă pînă cînd se epuizează toate nivelele arborelui a . Deoarece avem de-a face cu un număr finit de nivele, după un număr finit de pași vom ajunge la arborele a transformat în așa fel ca fiecare vîrf al său să aparțină unui nucleu. Procesul începe cu $i = 1$.

Pentru a construi algoritmul care transformă arborele în forma lui normală vom proceda în felul următor: fie S_1 mulțimea următoarelor substituții

$$\left. \begin{aligned} (\oplus, (\otimes, (\overline{\otimes}, p_1, x), q_1), (\otimes, (\overline{\otimes}, p_1, x), q_2)) &\rightarrow (\otimes, (\overline{\otimes}, p_1, x), (\oplus, q_1, q_2)) \\ (\otimes, (\otimes, (\overline{\otimes}, p_1, x), q_1), (\otimes, (\overline{\otimes}, p_2, x), q_2)) &\rightarrow (\otimes, (\overline{\otimes}, (\oplus, p_1, p_2), x), (\otimes, q_1, q_2)) \\ (\overline{\otimes}, p, (\otimes, (\overline{\otimes}, p_1, x), q_2)) &\rightarrow (\otimes, (\overline{\otimes}, (\otimes, p, p_1), x), (\pi, p, q_1)) \end{aligned} \right\} (S_1)$$

În această listă de substituții s-au folosit următoarele notații: $a_1 \rightarrow a_2$ înseamnă că arborele a_1 se înlocuiește cu arborele a_2 . Prin (θ, p_1, p_2) înțelegem arborele elementar atașat constantei obținută în urma aplicării operației θ asupra constantelor p_1 și p_2 oricare ar fi aceste constante. De asemenea se presupune că oricare arbore de forma $(\otimes, (\pi, p, x), q)$ este un nucleu.

Primul pas al algoritmului constă atunci în aplicarea substituțiilor (S_1) în ordinea în care apar în listă, de atîtea ori de cîte ori este posibil. Aceasta trebuie înțeleasă în felul următor: se consideră prima formulă de substituție din lista (S_1) care se poate aplica și se aplică arborelui de atîtea ori de cîte ori este posibil. După aceea se trece la următoarea formulă și se procedează la fel. Dacă o oarecare formulă nu se mai poate aplica, atunci algoritmul se ia de la început. Cînd nici o formulă nu se mai poate aplica, atunci se trece la următorul pas al algoritmului.

Se poate observa ușor că primul pas al algoritmului după un număr finit de aplicări transformă arborele a într-un arbore a' care posedă următoarele proprietăți:

- 1) Arborele a' nu conține nici un subarbore de structura arborilor din părțile stîngi ale formulelor (S_1) .
- 2) Oricare subarbore al arborelui a' format din două nuclee are una din formele

$$(\oplus, a_1^1, a_2^1), (\otimes, a_1^2, a_2^2), (\overline{\otimes}, a_1^3, a_2^3)$$

unde arborii a_i^j au proprietățile următoare:

- a) a_1^1, a_2^1 sînt nuclee de grade diferite;
- b) a_1^2, a_2^2 sînt subarbori de tipul $(\oplus, a_1^1, a_2^1)$;
- c) a_1^3 este subarbore de tipul $(\oplus, a_1', a_2')$ iar a_2^3 este arborele elementar corespunzător unei constante.

Pasul al doilea al algoritmului $\mathfrak{A}$ constă în aplicarea substituțiilor (S_2) de atîtea ori de cîte ori este posibil în mod analog cu primul pas $(\otimes, (\oplus, a_1^1, a_2^1), (\oplus, a_1^2, a_2^2)) \rightarrow (\oplus, (\otimes, a_1^1, a_1^2), (\otimes, a_1^1, a_2^2), (\otimes, a_2^1, a_1^2), (\otimes, a_2^1, a_2^2))$

$$(\overline{\otimes}, p, (\oplus, a_1^1, a_2^1)) \rightarrow (\oplus, a_0^1, a_1^1, \dots, a_p^1)$$

unde

$$a_i^1 = (\otimes, (\otimes, (\pi, \Lambda_\pi, x), C_p^i), (\otimes, (\otimes, (\oplus, p, * i), a_i^1), (\otimes, (\otimes, i, a_i^1), 1))), i = 0, 1, \dots, p$$

p fiind constanta pozitivă corespunzătoare arborelui elementar $\mathcal{P}$. Dacă

pasul al doilea al algoritmului $\mathfrak{A}$ nu se mai poate aplica, atunci se trece la primul pas. Acest proces se repetă pînă cînd nu se mai poate aplica nici primul pas al algoritmului nici al doilea. La sfîrșitul acestui proces, arborele a^1 se va transforma într-un arbore a^2 de forma

$$a^2 = (\oplus, a_1^1, a_2^1, \dots, a_k^1),$$

unde fiecare din arborii a_i^1 sînt nuclee de grade diferite sau subarbori ai arborelui a^2 de forma $(\oplus, a_1^1, a_2^1, \dots, a_r^1)$. În acest ultim caz, arborele a^2 se va transforma în așa fel ca să devină de forma $(\oplus, a_1^1, a_2^1, \dots, a_s^1)$, unde fiecare din a_i^1 sînt nuclee de grade diferite. Aceasta se realizează cu ajutorul substituției:

$$(\oplus, (\oplus, a_1^1, \dots, a_{r_1}^1), (\oplus, a_1^2, \dots, a_{r_2}^2)) \rightarrow (\oplus, a_1^1, \dots, a_{r_1}^1, a_1^2, \dots, a_{r_2}^2)$$

și cu ajutorul aplicării primului pas al algoritmului $\mathfrak{A}$.

Se trece apoi la aplicarea celui de-al treilea pas al algoritmului $\mathfrak{A}$ care constă în următoarele: se construiește permutarea $\Phi = (\Phi_1, \Phi_2, \dots, \Phi_k)$ a șirului $\{1, 2, \dots, k\}$ în felul următor: Φ_1 este indicele i al șirului $\{1, 2, \dots, k\}$ corespunzător celui mai mic grad al nucleelor $a_1^1, \dots, a_k^1$. Dacă Φ_i este construit, atunci Φ_{i+1} se construiește în același mod lăsînd la o parte indicele i folosit și nucleul corespunzător lui. La pasul al treilea al algoritmului $\mathfrak{A}$ se va transforma arborele a^2 prin aplicarea permutării Φ . În acest fel obținem $P_\Phi(a^2) = (\oplus, a_{\Phi_1}, a_{\Phi_2}, \dots, a_{\Phi_k})$, unde gradele nucleelor $a_{\Phi_1}, \dots, a_{\Phi_k}$ sînt astfel ca $p_{\Phi_1} < p_{\Phi_2} < \dots < p_{\Phi_k}$. În felul acesta se vede că arborele $\tilde{a} = P_\Phi(a^2)$ este forma normală a arborelui a .

Prin urmare oricare ar fi arborii a_1 și a_2 , algoritmul $\mathfrak{A}$ aplicat arborelui a_1 îl transformă în arborele $\tilde{a}_1$ iar aplicat arborelui a_2 îl transformă în $\tilde{a}_2$. Dacă $\tilde{a}_1 = \tilde{a}_2$, în sens grafic spunem că arborii a_1 și a_2 sînt echivalenți. Se vede însă simplu că arborelui a_1 îi corespunde în sistemul formal cu alfabetul $\mathfrak{A}$ un cuvînt compatibil P_1 , iar arborelui a_2 un cuvînt compatibil P_2 . Dacă $\tilde{P}_1 = \tilde{P}_2$, atunci cuvintele P_1 și P_2 sînt echivalente față de sistemul de operații corespunzător S -transformărilor și P -transformărilor definite mai sus.

DEFINIȚIA 6. Arborele a_2 se numește derivatul arborelui a_1 dacă a_2 poate fi obținut din a_1 prin aplicarea substituțiilor

$$\begin{aligned} (\otimes, (\otimes, p, x), q) &\rightarrow (\otimes, (\otimes, (\oplus, p, * 1), x), (\otimes, p, q)) \text{ dacă } p \geq 1 \\ (\otimes, (\otimes, 0, x), q) &\rightarrow (\otimes, (\otimes, 0, x), 0) \text{ dacă } p = 0 \end{aligned}$$

Se poate observa ușor că oricare ar fi arborele a în sistemul formal al alfabetului $\mathfrak{A}$, el admite un arbore derivat a' care se obține din arborele a prin aplicarea asupra acestuia din urmă a următorului algoritm:

Primul pas al algoritmului constă în transformarea arborelui în forma lui normală $\tilde{a}$.

Al doilea pas al algoritmului constă în aplicarea asupra arborelui $\tilde{a}$ a substituțiilor

$$\begin{aligned} (\otimes, (\otimes, i, x), p_i) &\rightarrow (\otimes, (\otimes, (\oplus, i, * 1), x), (\otimes, p, i)) \text{ pentru } i \geq 1 \\ (\otimes, (\otimes, 0, x), p_0) &\rightarrow (\otimes, (\otimes, 0, x), 0) \text{ pentru } i = 0 \end{aligned}$$

cîte o singură dată asupra fiecărui nucleu al arborelui $\tilde{a}$. Arborele obținut este arborele derivat al arborelui a .

Într-o notă ulterioară vom arăta cum se pot preda unei mașini electronice de calcul algoritmul descriși mai sus, care va executa calculele algebrice corespunzătoare.

Primită la redacție la 18 august 1964

BIBLIOGRAFIE

1. NOAM CHOMSKY, *On the notion „Rule of Grammar”, Structure of Language and its Mathematical aspects*, Providence, 1–24 (1961).
2. TREVOR EVANS, *A decision problem for transformation of trees*. Canadian Journal of Mathematics, 15, 4, 584–590 (1963).
3. NELSON GOODMAN, *Graphs for Linguistics, Structure of Language and its Mathematical aspects*, Providence, 51–55 (1961).
4. KENNETH E. IVERSON, *A programming Language*. John Wiley and Sons, Inc. New York, London, 1962.
5. T. RUS, *Asupra unui sistem formal*. I. Studii și cercetări matematice, 15, 4 (1964).
6. T. RUS, *Asupra unui sistem formal*. II. Studii și cercetări matematice, 15, 5 (1964).