Limbaje Formale, Automate şi Compilatoare

Curs 4

2014-15

Curs 4

1. Corectitudinea algoritmului de determinare a relaţiei ρ
2. Proprietăţi de închidere pentru clasa limbajelor de tip 3
3. Expresii regulate
4. Automatul echivalent cu o expresie regulată
   - Algoritm
5. Gramatici şi limbaje independente de context

Curs 4

Algoritm pentru determinarea relaţiei ρ

//initializarea tablourilor,
se marchează perechile $F \times (Q - F)$ si $(Q - F) \times F$

```
for (i=0; i<=n-1; i++)
    for (j=i+1,j<=n; j++) {
        lista[qi,qj]=∅;
        if ((qi ∈ F && qj ∉ F) || (qi ∉ F && qj ∈ F))
            separabil[qi,qj]=1;
        else
            separabil[qi,qj]=0;
    }
```

```
for (i=0; i<=n-1; i++)
    for (j=i+1,j<=n; j++) {
    //se selecteaza doar starile inseparabile
        if (separabil[qi,qj]==0) {
            //daca exista a astfel incat ¬δ(qi,a)ρδ(qj,a)
            //inseamna ca qi si qj sunt separabile
            if (∃a ∈ Σ : separabil[δ(qi,a),δ(qj,a)] == 1){
                // qi si qj devin separabile si la fel toate
                // perechile de stari dependente de qi,qj
                update_separabil(qi,qj);
            }
            else {
                for (a ∈ Σ && δ(qi,a) ≠ δ(qj,a))
                    adauga (qi,qj) la lista[δ(qi,a),δ(qj,a)]
            }
        }
    }
```

Algoritm pentru determinarea relaţiei ρ

```
// qi si qj devin separabile si la fel toate
// perechile de stari dependente de qi,qj
update_separabil(qi, qj){
      separabil[qi, qj] = 1;
      for ((q'_i, q'_j) ∈ lista[qi, qj]){
          if (separabil[q'_i, q'_j] == 0)
              update_separabil(q'_i, q'_j);
      }
}
```

Corectitudinea algoritmului

Teorema 1

Algoritmul se termină întotdeauna şi în final se obţine, pentru orice două stări q_i şi q_j, $0 \leq i < j \leq n$: separabil$[q_i, q_j] = 1$ ddacă q_i sep q_j

Corectitudinea algoritmului

Teorema 1

Algoritmul se termină întotdeauna şi în final se obţine, pentru orice două stări q_i şi q_j, $0 \leq i < j \leq n$: $separabil[q_i, q_j] = 1$ ddacă q_i sep q_j

($\Longleftarrow$) Se arată că:

$P(k)$: Pentru orice două stări q_i şi q_j ($0 \leq i < j \leq n$) separabile de către un cuvânt w cu $|w| \leq k$ ($\delta(q_i, w) \in F, \delta(q_j, w) \notin F$), are loc:

$$separabil[q_i, q_j] = 1.$$

Inducţie după $|w|$.

Corectitudinea algoritmului

Teorema 1

Algoritmul se termină întotdeauna şi în final se obţine, pentru orice două stări q_i şi q_j, $0 \leq i < j \leq n$: $separabil[q_i, q_j] = 1$ ddacă q_i $\underline{sep}$ q_j

($\Longrightarrow$) Se arată că:

pentru oricare două stări q_i, q_j ($0 \leq i < j \leq n$) pentru care $separabil[q_i, q_j] = 1$, are loc:

$$q_i \ \underline{sep}\ q_j.$$

Inducţie asupra momentului în care algoritmul face $separabil[q_i, q_j] = 1$.

Curs 4

Închiderea la intersecţie

- Dacă $L_1, L_2 \in \mathcal{L}_3$, atunci $L_1 \cap L_2 \in \mathcal{L}_3$

Fie $A_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, F_1)$ şi $A_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, F_2)$ automate deterministe astfel încât $L_1 = L(A_1)$ şi $L_2 = L(A_2)$.

Automatul A (determinist) care recunoaşte $L_1 \cap L_2$:

$$A = (Q_1 \times Q_2, \Sigma_1 \cap \Sigma_2, \delta, (q_{01}, q_{02}), F_1 \times F_2)$$

$\delta((q_1, q_2), a)) = (q_1', q_2')$ ddacă

- $\delta_1(q_1, a) = q_1'$
- $\delta_2(q_2, a) = q_2'$

Închiderea la diferenţă

- Dacă $L \in \mathcal{L}_3$ atunci $\overline{L} = (\Sigma^* \setminus L) \in \mathcal{L}_3$

Fie $A = (Q, \Sigma, \delta, q_0, F)$ automat cu $L(A) = L$.

Automatul A' care recunoaşte $\overline{L} = \overline{L(A)}$:

$$A' = (Q, \Sigma, \delta, q_0, Q \setminus F)$$

Închiderea la diferenţă

- Dacă $L \in \mathcal{L}_3$ atunci $\overline{L} = (\Sigma^* \setminus L) \in \mathcal{L}_3$

Fie $A = (Q, \Sigma, \delta, q_0, F)$ automat cu $L(A) = L$.

Automatul A' care recunoaşte $\overline{L} = \overline{L(A)}$:

$$A' = (Q, \Sigma, \delta, q_0, Q \setminus F)$$

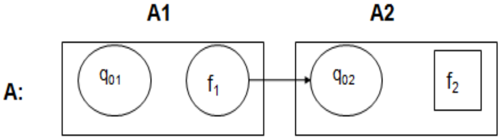
- Dacă $L_1, L_2 \in \mathcal{L}_3$ atunci $L_1 \setminus L_2 \in \mathcal{L}_3$: $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$

Închiderea la produs

- Fie $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, \{f_1\})$ şi $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, \{f_2\})$ automate cu o singură stare finală astfel încât $L_1 = L(A_1)$ şi $L_2 = L(A_2)$.

  Automatul A (cu ϵ-tranziţii) care recunoaşte $L_1 \cdot L_2$:

$$A = (Q_1 \cup Q_2, \Sigma, \delta, q_{01}, \{f_2\})$$

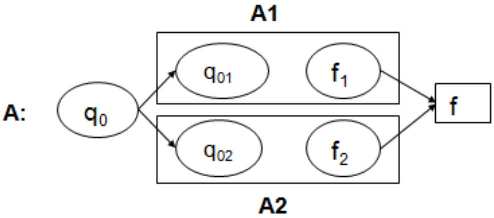


Închiderea la reuniune

- Fie $A_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, \{f_1\})$ şi $A_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, \{f_2\})$ automate cu o singură stare finală astfel încât $L_1 = L(A_1)$ şi $L_2 = L(A_2)$.

  Automatul A (cu ϵ-tranziţii) care recunoaşte $L_1 \cup L_2$:

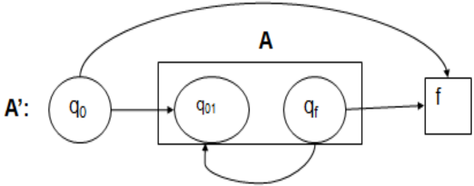
$$A = (Q_1 \cup Q_2 \cup \{q_0, f\}, \Sigma_1 \cup \Sigma_2, \delta, q_0, \{f\})$$



Închiderea la iteraţie

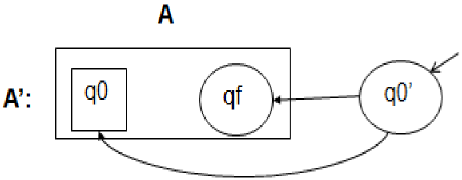
- Fie $A = (Q, \Sigma, \delta, q_{01}, \{f\})$ automat cu o singură stare finală astfel încât $L(A) = L$.
  Automatul A (cu ϵ-tranziţii) care recunoaşte L^* $(= L(A)^*)$:

$$A = (Q \cup \{q_0, f\}, \Sigma, \delta', q_0, \{f\})$$



Închiderea la operaţia de oglindire

- Fie $A = (Q, \Sigma, \delta, q_0, \{q_f\})$ automat cu o singură stare finală. Automatul A (cu ϵ-tranziţii) care recunoaşte $L(A)^R$:
  $A = (Q \cup \{q_0'\}, \Sigma, \delta', q_0', \{q_0\})$:
  - $\delta'(q_1, a) = q_2$ ddacă $\delta(q_2, a) = q_1$ (inversarea arcelor în graful de tranziţie)
  - $\delta'(q_0', \epsilon) = q_f$
  - dacă $\epsilon \in L(A)$, atunci $\delta'(q_0', \epsilon) = q_0$



Curs 4

Expresii regulate - definiţie

- Reprezentarea limbajelor de tip 3 prin expresii algebrice

Definiţie 1

Dacă Σ este un alfabet atunci o expresie regulată peste Σ se defineşte inductiv astfel:

- $\emptyset$, ϵ, *a* (*a* $\in \Sigma$) *sunt expresii regulate ce descriu respectiv limbajele* $\emptyset$, $\{\epsilon\}$, $\{a\}$.
- *Dacă* E, E_1, E_2 *sunt expresii regulate atunci:*
  - $(E_1|E_2)$ *este expresie regulată ce descrie limbajul* $L(E_1) \cup L(E2)$
  - $(E_1 \cdot E_2)$ *este expresie regulată ce descrie limbajul* $L(E_1)L(E_2)$
  - (E^*) *este expresie regulată ce descrie limbajul* $L(E)^*$

Expresii regulate - definiţie

- Reprezentarea limbajelor de tip 3 prin expresii algebrice

Definiţie 1

Dacă Σ este un alfabet atunci o expresie regulată peste Σ se defineşte inductiv astfel:

- $\emptyset$, ϵ, a ($a \in \Sigma$) *sunt expresii regulate ce descriu respectiv limbajele* $\emptyset$, $\{\epsilon\}$, $\{a\}$.
- *Dacă* E, E_1, E_2 *sunt expresii regulate atunci:*
    - $(E_1|E_2)$ *este expresie regulată ce descrie limbajul* $L(E_1) \cup L(E2)$
    - $(E_1 \cdot E_2)$ *este expresie regulată ce descrie limbajul* $L(E_1)L(E_2)$
    - (E^*) *este expresie regulată ce descrie limbajul* $L(E)^*$

- Ordinea de prioritate a operatorilor este $*$, $\cdot$, $|$

Exemple

- $(a|b)|(c|d) \longrightarrow \{a, b, c, d\}$
- $(0|1) \cdot (0|1) \longrightarrow \{00, 01, 10, 11\}$
- $a^*b^* \longrightarrow \{a^n b^k | n, k \geq 0\}$
- $(a|b)^* \longrightarrow \{a, b\}^*$
- $(0|1|2|...|9)(0|1|2...|9)^*$ descrie mulţimea întregilor fără semn
- $(a|b|c|...|z)(a|b|c|...|z|0|1|2...|9)^*$ descrie mulţimea identificatorilor

Două expresii regulate E_1, E_2 sunt echivalente, şi scriem $E_1 = E_2$ dacă $L(E_1) = L(E_2)$

Proprietăţi

- $(p|q)|r = p|(q|r)$
- $(pq)r = p(qr)$
- $p|q = q|p$
- $p \cdot \epsilon = \epsilon \cdot p = p$
- $p|\emptyset = p|p = p$
- $\emptyset \cdot p = p \cdot \emptyset = \emptyset$
- $p(q|r) = pq|pr$
- $(p|q)r = pr|qr$
- $\epsilon|pp^* = p^*$
- $\epsilon|p^*p = p^*$

De la o expresie regulată la automatul finit

Teorema 2

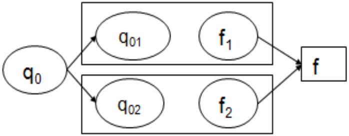
Pentru orice expresie regulată E peste Σ există un automat finit (cu ϵ -tranziţii) A, astfel încât $L(A) = L(E)$.

<u>*Demonstratie*</u>: inducţie structurală.

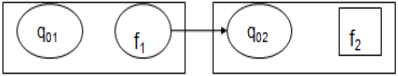
- Dacă $E \in \{\emptyset, \epsilon, a\}$ ($a \in \Sigma$) atunci automatul corespunzător este respectiv:

Demonstrație

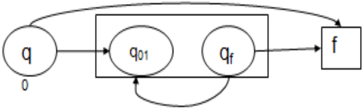
- $E = E_1 | E_2$



- $E = E_1 E_2$



- $E = E_1^*$



Reprezentarea expresiilor regulate sub formă de arbore

- **Intrare**: Expresia regulată $E = e_0 e_1 \ldots e_{n-1}$
  Precedenţa operatorilor:
  prec($|$) = 1, prec($\cdot$) = 2, prec($*$) = 3 (prec(()= 0).
- **Ieşire**: Arborele asociat: t.
- **Metoda**: Se consideră două stive:
  - STIVA1 stiva operatorilor
  - STIVA2 stiva arborilor (care va conţine arborii parţiali construiţi)
  - Metoda $tree(op, tS, tD)$

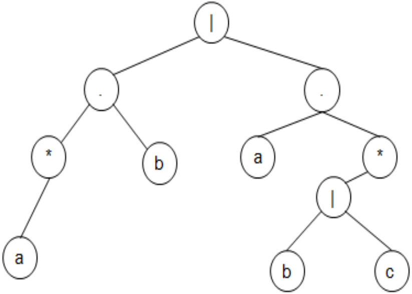
Algoritm

```
i = 0;
while(i < n) {
     c = e_i;
     switch(c) {
         case '(':  { STIVA1.push(c); break; }
         case simbol (din alfabet):  { STIVA2.push(tree(c,NULL,NULL)); break; }
         case operator:  {
              while (prec(STIVA1.top())>=prec(c))
                    build_tree();
              STIVA1.push(c); break;
         }
         case ')':  {
              do { build_tree();} while(STIVA1.top()!= '(');
              STIVA1.pop(); break;
         }
     }
     i++;
}
while(STIVA1.not_empty()) build_tree();
t = STIVA2.pop();
```

Algoritm

```
build_tree()
     op = STIVA1.pop();
     tD = STIVA2.pop();
     switch (op) {
            case '*':  {
                 t = tree(op, tD, NULL);
                 STIVA2.push(t); break;
            }
            case'|', '·':  {
                tS = STIVA2.pop();
                t = tree(op, tS, tD);
                STIVA2.push(t); break;
            }
}
```

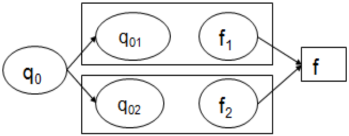
Exemplu



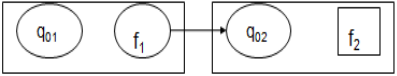
$a^* \cdot b | a \cdot (b|c)^*$

Curs 4

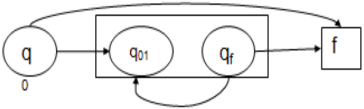
Automatul echivalent cu o expresie regulată



- $E = E_1|E_2$



- $E = E_1E_2$



- $E = E_1^*$

Observaţii

- pentru orice apariţie a unui simbol din Σ, cât şi pentru ϵ, dacă acesta apare explicit în E, este nevoie de 2 stări în automatul construit.
- fiecare din apariţiile operatorilor $|$ şi $*$ dintr-o expresie regulată E introduce două noi stări în automatul construit
- operatorul $\cdot$ nu introduce alte stări
- dacă n este numărul de simboluri din E iar m este numărul de paranteze împreună cu apariţiile simbolului $\cdot$, atunci numărul stărilor automatului echivalent cu E este $p = 2(n - m)$.

- din orice stare *i* a automatului, se fac cel mult două tranziţii:
  - $\delta(i, a) = j$ $(\delta(i, \epsilon) = \emptyset)$
  - $\delta(i, \epsilon) = \{j\}$ $(\delta(i, a) = \emptyset)$
  - $\delta(i, \epsilon) = \{j, k\}$ $(\delta(i, a) = \emptyset)$

- din orice stare i a automatului, se fac cel mult două tranziţii:
  - $\delta(i, a) = j$ $(\delta(i, \epsilon) = \emptyset)$
  - $\delta(i, \epsilon) = \{j\}$ $(\delta(i, a) = \emptyset)$
  - $\delta(i, \epsilon) = \{j, k\}$ $(\delta(i, a) = \emptyset)$

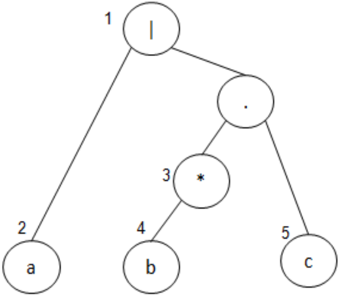
- reprezentarea automatului echivalent cu o expresie regulată se poate face cu 3 tablouri de dimensiune p, unde p este numărul stărilor (acestea sunt numerotate de la 1 la p):
  - $simbol[i] = a$: din starea i se face o tranziţie cu simbolul a (dacă $simbol[i] =$' ', $\delta(i, a) = \emptyset$)
  - $next1[i] = j$: din starea i se face o tranziţie către starea j $(\delta(i, a) = j$, dacă $simbol[i] = a$, altfel $j \in \delta(i, \epsilon))$
  - $next2[i] = k$: din starea i se face o ϵ - tranziţie către k $(\delta(i, \epsilon) = \{j, k\}$ şi $next1[i] = j)$.

Algoritm

- **Intrare**: Expresia regulată E cu n simboluri dintre care m sunt paranteze şi apariţii ale operatorului produs;
- **Ieşire**: Vectorii *simbol*, *next*1, *next*2 de dimensiune $p = 2(n - m)$ ce descriu automatul cu ϵ - tranziţii echivalent cu E, starea finală f;
- **Metoda**:

1. Se construieşte arborele ataşat expresiei E;
2. Se parcurge arborele în preordine şi se ataşează nodurilor vizitate, exceptând pe cele etichetate cu produs , respectiv numerele $1, 2, \ldots, n - m$;

Exemplu



$E = a|b^* \cdot c$

3. Se parcurge arborele în postordine şi se ataşează fiecărui nod N o pereche de numere $(N.i, N.f)$ care reprezintă starea iniţială respectiv finală a automatului echivalent cu expresia corespunzătoare subarborelui cu rădăcina N, astfel:

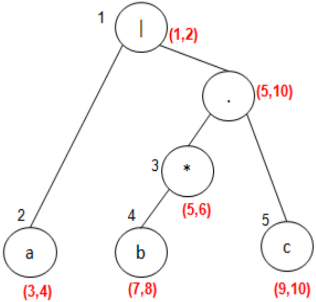
   - Dacă nodul are numărul k (de la pasul 2) atunci:

     $$N.i = 2k - 1, N.f = 2k;$$

   - Dacă nodul este etichetat cu produs şi S este fiul stâng al lui N, iar D fiul drept, atunci:

     $$N.i = S.i \text{ iar } N.f = D.f$$

Exemplu



$E = a|b^* \cdot c$

4. for($j = 1..2(n - m)$) $\{simbol[j] =' ', next1[j] = next2[j] = 0\}$

4. for($j = 1..2(n-m)$) $\{simbol[j] =' \ ', next1[j] = next2[j] = 0\}$
5. Se parcurge din nou arborele obţinut în postordine.

   Dacă N este nodul curent iar S si D sunt fii sai, atunci, în funcţie de eticheta lui N, se execută următoarele:

4. for($j = 1..2(n - m)$) $\{simbol[j] =' '\ , next1[j] = next2[j] = 0\}$

5. Se parcurge din nou arborele obţinut în postordine.
   Dacă N este nodul curent iar S si D sunt fii sai, atunci, în funcţie de eticheta lui N, se execută următoarele:

   - Dacă N este etichetat cu a (deci este frunza): $\delta(N.i, a) = N.f$

$$simbol[N.i] = a, next1[N.i] = N.f$$

4. for($j = 1..2(n-m)$) $\{simbol[j] =' ', next1[j] = next2[j] = 0\}$

5. Se parcurge din nou arborele obţinut în postordine.
   Dacă N este nodul curent iar S si D sunt fii sai, atunci, în funcţie de eticheta lui N, se execută următoarele:

   - Dacă N este etichetat cu a (deci este frunza): $\delta(N.i, a) = N.f$

     $$simbol[N.i] = a, next1[N.i] = N.f$$

   - Dacă N este etichetat cu $|$: $\delta(N.i, \epsilon) = \{S.i, D.i\}$, $\delta(S.f, \epsilon) = N.f$, $\delta(D.f, \epsilon) = N.f$

     $$next1[N.i] = S.i, next2[N.i] = D.i,$$

     $$next1[S.f] = N.f, next1[D.f] = N.f$$

4. for($j = 1..2(n-m)$) $\{simbol[j] =' '\ , next1[j] = next2[j] = 0\}$

5. Se parcurge din nou arborele obţinut în postordine.
   Dacă N este nodul curent iar S si D sunt fii sai, atunci, în funcţie de eticheta lui N, se execută următoarele:

   - Dacă N este etichetat cu a (deci este frunza): $\delta(N.i, a) = N.f$
   $$simbol[N.i] = a, next1[N.i] = N.f$$
   - Dacă N este etichetat cu $|$: $\delta(N.i, \epsilon) = \{S.i, D.i\}$, $\delta(S.f, \epsilon) = N.f$, $\delta(D.f, \epsilon) = N.f$
   $$next1[N.i] = S.i, next2[N.i] = D.i,$$
   $$next1[S.f] = N.f, next1[D.f] = N.f$$
   - Dacă N este etichetat cu $\cdot$: $\delta(S.f, \epsilon) = D.i$
   $$next1[S.f] = D.i$$

4. for($j = 1..2(n-m)$) $\{simbol[j] =' '\ , next1[j] = next2[j] = 0\}$
5. Se parcurge din nou arborele obţinut în postordine.
   Dacă N este nodul curent iar S si D sunt fii sai, atunci, în funcţie de eticheta lui N, se execută următoarele:
   - Dacă N este etichetat cu a (deci este frunza): $\delta(N.i, a) = N.f$
     $$simbol[N.i] = a, next1[N.i] = N.f$$
   - Dacă N este etichetat cu $|$: $\delta(N.i, \epsilon) = \{S.i, D.i\}$, $\delta(S.f, \epsilon) = N.f$, $\delta(D.f, \epsilon) = N.f$
     $$next1[N.i] = S.i, next2[N.i] = D.i,$$
     $$next1[S.f] = N.f, next1[D.f] = N.f$$
   - Dacă N este etichetat cu $\cdot$: $\delta(S.f, \epsilon) = D.i$
     $$next1[S.f] = D.i$$
   - Dacă N este etichetat cu $*$ (D nu există în acest caz): $\delta(N.i, \epsilon) = \{S.i, N.f\}$, $\delta(S.f, \epsilon) = \{S.i, N.f\}$
     $$next1[N.i] = S.i, next2[N.i] = N.f,$$
     $$next1[S.f] = S.i, next2[S.f] = N.f$$

4. for($j = 1..2(n - m)$) $\{simbol[j] =$' ' , $next1[j] = next2[j] = 0\}$

5. Se parcurge din nou arborele obţinut în postordine.
   Dacă N este nodul curent iar S si D sunt fii sai, atunci, în funcţie de eticheta lui N, se execută următoarele:

   - Dacă N este etichetat cu a (deci este frunza): $\delta(N.i, a) = N.f$

     $$simbol[N.i] = a, next1[N.i] = N.f$$

   - Dacă N este etichetat cu $|$: $\delta(N.i, \epsilon) = \{S.i, D.i\}$, $\delta(S.f, \epsilon) = N.f$, $\delta(D.f, \epsilon) = N.f$

     $$next1[N.i] = S.i, next2[N.i] = D.i,$$

     $$next1[S.f] = N.f, next1[D.f] = N.f$$

   - Dacă N este etichetat cu $\cdot$: $\delta(S.f, \epsilon) = D.i$

     $$next1[S.f] = D.i$$

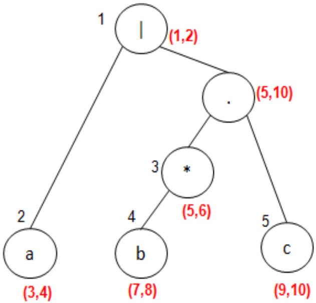
   - Dacă N este etichetat cu $*$ (D nu există în acest caz): $\delta(N.i, \epsilon) = \{S.i, N.f\}$, $\delta(S.f, \epsilon) = \{S.i, N.f\}$

     $$next1[N.i] = S.i, next2[N.i] = N.f,$$

     $$next1[S.f] = S.i, next2[S.f] = N.f$$

6. f este starea pentru care next1[f] = next2[f] = 0

Exemplu



$$E = a|b^* \cdot c$$

Exemplu

p	simbol[p]	next1[p]	next2[p]
1		3	5
2		0	0
3	a	4	0
4		2	0
5		7	6
6		9	0
7	b	8	0
8		7	6
9	c	10	0
10		2	0

Exemplu

δ	a	b	c	ϵ
1	$\emptyset$	$\emptyset$	$\emptyset$	$\{3,5\}$
2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
3	4	$\emptyset$	$\emptyset$	$\emptyset$
4	$\emptyset$	$\emptyset$	$\emptyset$	$\{2\}$
5	$\emptyset$	$\emptyset$	$\emptyset$	$\{6,7\}$
6	$\emptyset$	$\emptyset$	$\emptyset$	$\{9\}$
7	$\emptyset$	8	$\emptyset$	$\emptyset$
8	$\emptyset$	$\emptyset$	$\emptyset$	$\{6,7\}$
9	$\emptyset$	$\emptyset$	10	$\emptyset$
10	$\emptyset$	$\emptyset$	$\emptyset$	$\{2\}$

Corectitudinea algoritmului

Teorema 3

Algoritmul descris este corect: automatul cu ϵ - tranziţii obţinut este echivalent cu expresia regulată E.

Demonstraţie:

- Modul în care au fost alese perechile (i, f) de stări pentru fiecare nod al arborelui construit corespunde construcţiilor din teorema 1.
- Deasemenea, tranziţiile care se definesc în pasul 5 al algoritmului urmăresc construcţia din teorema 1.

Automatul obţinut este echivalent cu expresia dată la intrare.

Curs 4

1. Corectitudinea algoritmului de determinare a relaţiei ρ
2. Proprietăţi de închidere pentru clasa limbajelor de tip 3
3. Expresii regulate
4. Automatul echivalent cu o expresie regulată
   - Algoritm
5. Gramatici şi limbaje independente de context

Gramatici independente de context

- Gramatici de tip 2 (independente de context): $G = (N, T, S, P)$
  - N şi T sunt mulţimi nevide, finite, disjuncte de neterminali (variabile), respectiv terminali
  - $S \in N$ este simbolul de start
  - $P = \{x \to u | x \in N, u \in (N \cup T)^*\}$ este mulţimea regulilor (producţiilor).

- Un limbaj L este de tip 2 (independent de context: $L \in \mathcal{L}_2$) dacă există o gramatică G de tip 2 astfel încât $L(G) = L$

Derivări extrem stângi/drepte

Fie $G = (N, T, S, P)$ si $w \in L(G)$

- derivare extrem stângă pentru w: derivarea în care, la orice pas se înlocuieşte cel mai din stânga neterminal din cuvântul obţinut
- derivare extrem dreaptă pentru w: derivarea în care, la orice pas se înlocuieşte cel mai din dreapta neterminal din cuvântul obţinut

Exemplu

$G = (\{E\}, \{a, b, +, *), (\}, E, P)$ unde:

$P : E \rightarrow E + E|E * E|(E)|a|b$

Fie $a + (b * a)$

- Derivare extrem stângă:

  $E \Rightarrow E{+}E \Rightarrow a{+}E \Rightarrow a{+}(E) \Rightarrow a{+}(E{*}E) \Rightarrow a{+}(b{*}E) \Rightarrow a{+}(b{*}a)$

- Derivare extrem dreaptă:

  $E \Rightarrow E + E \Rightarrow E + (E) \Rightarrow E + (E * E) \Rightarrow E + (E * a) \Rightarrow$
  $E + (b * a) \Rightarrow a + (b * a)$

- Există derivări care nu sunt nici extrem drepte nici extrem stângi!

Arbori sintactici

Definiţie 2

Un *arbore sintactic (arbore de derivare, arbore de parsare)* *în gramatica G este un arbore ordonat, etichetat, cu următoarele proprietăţi:*

- *rădăcina arborelui este etichetată cu S ;*
- *fiecare frunză este etichetată cu un simbol din T sau cu ϵ ;*
- *fiecare nod interior este etichetat cu un neterminal;*
- *dacă A etichetează un nod interior care are n succesori etichetaţi de la stânga la dreapta respectiv cu X_1, X_2,..., X_n, atunci $A \to X_1X_2\ldots X_n$ este o regulă.*
  Cazul în care regula este $A \to \epsilon$ reprezintă un caz special: nodul etichetat cu A are un singur descendent etichetat cu ϵ.

Arbori sintactici

Definiţie 3

- *Frontiera unui arbore de derivare este cuvântul* $w = a_1 a_2 \ldots a_n$ *unde* a_i, $1 \leq i \leq n$ *sunt etichetele nodurilor frunză în ordinea de la stânga la dreapta.*

- *Arbore de derivare pentru un cuvânt* w*: arbore de derivare cu frontiera* w.

- *Un X-arbore de derivare este un subarbore al unui arbore de derivare care are eticheta rădăcinii* X. *Un arbore de derivare este un S-arbore de derivare.*

Exemplu

$G = (\{E\}, \{a, b, +, *), (\}, E, P)$ unde:
$P : E \to E + E | E * E | (E) | a | b$

$a + (b * a)$

- Derivare extrem stângă:
  $E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + (E) \Rightarrow$
  $a + (E * E) \Rightarrow a + (b * E) \Rightarrow a + (b * a)$
- Derivare extrem dreaptă:
  $E \Rightarrow E + E \Rightarrow E + (E) \Rightarrow E + (E * E) \Rightarrow$
  $E + (E * a) \Rightarrow E + (b * a) \Rightarrow a + (b * a)$

- Arbore de derivare pentru $a + (b * a)$: