Limbaje Formale, Automate şi Compilatoare

Curs 5

2018-19

Curs 5

1. Expresii regulate
2. Automatul echivalent cu o expresie regulată
   - Algoritm
3. Lema Bar-Hillel
4. Gramatici şi limbaje independente de context
5. Eliminarea regulilor de ştergere şi a redenumirilor

Curs 5

De la o expresie regulată la automatul finit

Teorema 1

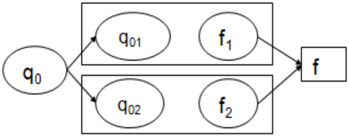
Pentru orice expresie regulată E peste Σ există un automat finit (cu ϵ -tranziţii) A, astfel încât $L(A) = L(E)$.

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

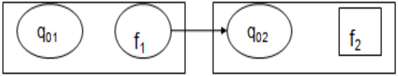
- Dacă $E \in \{\emptyset, \epsilon, a\}$ ($a \in \Sigma$) atunci automatul corespunzător este respectiv:

Demonstrație

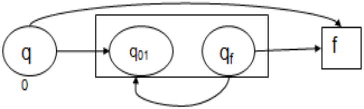
- $E = E_1 | E_2$



- $E = E_1 E_2$



- $E = E_1^*$



Reprezentarea expresiilor regulate sub formă de arbore

- **Intrare**: Expresia regulată $E = e_0 e_1 \dots e_{n-1}$
  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 $tree(r, tS, tD)$

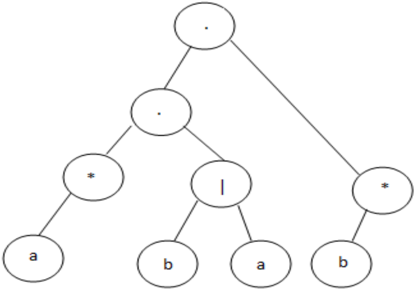
Algoritm

```
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();
     t1 = STIVA2.pop();
     switch (op) {
            case '*':  {
                 t = tree(op, t1, NULL);
                 STIVA2.push(t); break;
            }
            case'|', '·':  {
                t2 = STIVA2.pop();
                t = tree(op, t2, t1);
                STIVA2.push(t); break;
            }
}
```

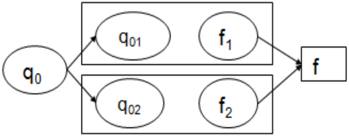
Exemplu



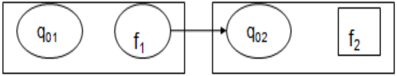
$a^* \cdot (b|a) \cdot b^*$

Curs 5

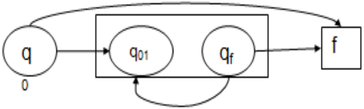
Automatul echivalent cu o expresie regulată



- $E = E_1 | E_2$



- $E = E_1 E_2$



- $E = E_1^*$

Observaţii

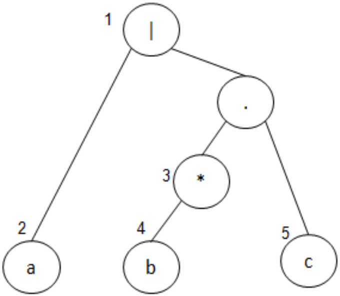
- pentru orice apariţie a unui simbol din Σ, cât şi pentru ϵ, dacă acesta apare explicit în E, este nevoie de 2 stări în automatul construit.
- fiecare din apariţiile operatorilor $|$ şi $*$ dintr-o expresie regulată E introduce două noi stări în automatul construit
- operatorul $\cdot$ nu introduce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

Algoritm

- **Intrare**: Expresia regulată E cu n simboluri dintre care m sunt paranteze şi apariţii ale operatorului produs;
- **Ieşire**:Automatul (cu $p = 2(n - m)$ stări) cu ϵ - tranziţii echivalent cu E
- **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:

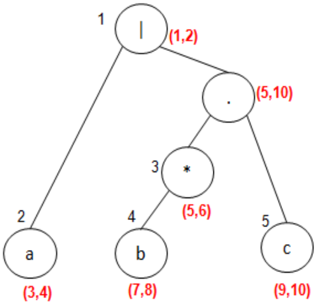
   - Dacă nodul are numărul k (de la pasul 2) atunci:

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

   - 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. 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. 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):

$$\delta(N.i, a) = N.f$$

4. 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):

$$\delta(N.i, a) = N.f$$

   - Dacă N este etichetat cu $|$:

$$\delta(N.i, \epsilon) = \{S.i, D.i\},$$

$$\delta(S.f, \epsilon) = N.f, \ \delta(D.f, \epsilon) = N.f$$

4. 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):

$$\delta(N.i, a) = N.f$$

   - Dacă N este etichetat cu $|$:

$$\delta(N.i, \epsilon) = \{S.i, D.i\},$$

$$\delta(S.f, \epsilon) = N.f, \ \delta(D.f, \epsilon) = N.f$$

   - Dacă N este etichetat cu $\cdot$:

$$\delta(S.f, \epsilon) = D.i$$

4. 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):

   $$\delta(N.i, a) = N.f$$

   - Dacă N este etichetat cu $|$:

   $$\delta(N.i, \epsilon) = \{S.i, D.i\},$$

   $$\delta(S.f, \epsilon) = N.f, \ \delta(D.f, \epsilon) = N.f$$

   - Dacă N este etichetat cu $\cdot$:

   $$\delta(S.f, \epsilon) = D.i$$

   - Dacă N este etichetat cu $*$ (D nu există în acest caz):

   $$\delta(N.i, \epsilon) = \{S.i, N.f\},$$

   $$\delta(S.f, \epsilon) = \{S.i, N.f\}$$

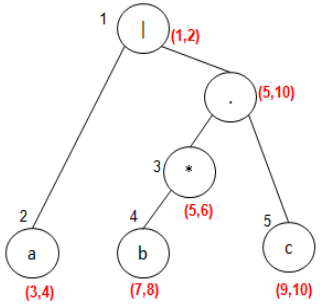
4. Se parcurge din nou arborele obţinut în postordine.

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

5. Starea iniţială a automatului este $N.i$, starea finală $N.f$, unde N este nodul rădăcină;

Exemplu



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

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 2

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 2.
- 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 5

Lema Bar-Hillel (lema de pompare)

Lema 3.1

Fie L un limbaj de tip 3. Există un număr k astfel încât oricare ar fi cuvântul $w \in L$ cu $|w| \geq k$, acesta are o descompunere de forma $w = xyz$, unde $0 < |y| \leq k$, şi $xy^iz \in L$ oricare ar fi $i \geq 0$.

Fie $A = (Q, \Sigma, \delta, q_0, F)$ astfel ca $L(A) = L$. Dacă $|Q| = n$ este numărul stărilor din N , fie $k = |Q| = n$, se arată că are loc proprietatea enunţată:
Fie $w = a_1a_2 \ldots a_m,\ m \geq k = n$
Fie $q_0 = \delta(q_0, \epsilon)$, $q_1 = \delta(q_0, a_1)$, $\ldots$ $q_n = \delta(q_0, a_1 \ldots a_n)$
Există două stări egale: $q_l = \delta(q_0, a_1 \ldots a_l)$, $q_j = \delta(q_0, a_1 \ldots a_j)$

$0 \leq l < j \leq n$.

Demonstrație

$w = a_1 a_2 \ldots a_m$, $m \geq k = n$

Fie $x = a_1 a_2 \ldots a_l$, $y = a_{l+1} \ldots a_j$ şi $z = a_{j+1} \ldots a_{m-1} a_m$
$w = xyz$, cu $0 < |y| \leq k$.

- $q_l = \delta(q_0, a_1 \ldots a_l) = q_j = \delta(q_0, a_1 \ldots a_l a_{l+1} \ldots a_j) \Rightarrow$

$$\delta(q_0, x) = \delta(q_0, xy)$$

$xy^i z \in L(A), \forall i \geq 0$:

- $i = 0$: $\delta(q_0, xz) = \delta(\delta(q0, x), z) = \delta(\delta(q0, xy), z) = \delta(q0, xyz) \in F$ $(w = xyz \in L = L(A))$
- Presupunem că $xy^i z \in L$, pentru orice $i \leq r$ şi demonstrăm pentru $i = r + 1$

Curs 5

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 1

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 $A \to X_1X_2 \dots X_n$ este o regulă.*
  Dacă A are un succesor etichetat cu ϵ (pentru regula $A \to \epsilon$), nodul etichetat cu A nu mai are alţi succesori.

Arbori sintactici

Definiţie 2

- *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.

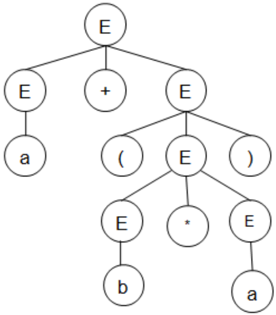
Exemplu

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



Ambiguitate

Definiţie 3

O gramatică G este ambiguă dacă există un cuvânt w în $L(G)$ care are 2 arbori de derivare distincţi.

- Echivalent: w are 2 derivări extrem stângi(drepte) distincte.

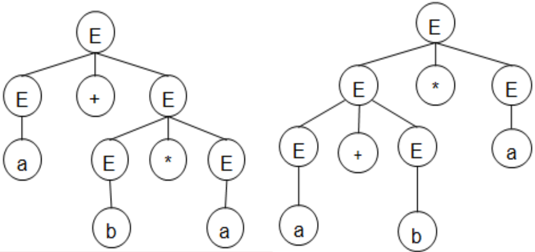
Ambiguitate

Definiţie 3

O gramatică G este ambiguă dacă există un cuvânt w în L(G) care are 2 arbori de derivare distincţi.

- Echivalent: *w* are 2 derivări extrem stângi(drepte) distincte.

Gramatica precedentă este ambiguă: cuvântul $a + b * a$ are 2 arbori de derivare:



Ambiguitate

Definiţie 3

O gramatică G este ambiguă dacă există un cuvânt w în L(G) care are 2 arbori de derivare distincţi.

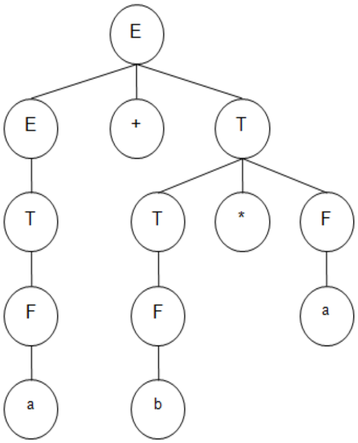
- Echivalent: w are 2 derivări extrem stângi(drepte) distincte.
- Problema ambiguităţii gramaticilor de tip 2 este nedecidabilă: nu există un algoritm care pentru o gramatică oarecare G să testeze dacă G este sau nu ambiguă

Exemplu: o gramatică echivalentă neambiguă

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

- $E \to E + T$
- $E \to T$
- $T \to T * F$
- $T \to F$
- $F \to (E)$
- $F \to a|b$

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



Curs 5

Eliminarea regulilor de ştergere

- Intrare: $G = (N, T, S, P)$
- Ieşire: $G' = (N, T, S, P')$, $L(G') = L(G)$, P' nu conţine reguli de ştergere (reguli de forma $A \to \epsilon$)

$N_0 = \{A | A \in N,\ \ A \to \epsilon \in P\};\ \ i = 0;$

```
do {
```

$\quad i = i + 1;$

$\quad N_i = N_{i-1} \cup \{X | X \in N,\ \ \exists X \to \alpha \in P, \alpha \in N_{i-1}^*\};$

```
} while
```
$N_i \neq N_{i-1};$

$N_\epsilon = N_i;$

Eliminarea regulilor de ştergere

- Intrare: $G = (N, T, S, P)$
- Ieşire: $G' = (N, T, S, P')$, $L(G') = L(G)$, P' nu conţine reguli de ştergere (reguli de forma $A \rightarrow \epsilon$)

$N_0 = \{A | A \in N,\ \ A \rightarrow \epsilon \in P\};\ i = 0;$

do $\{$

$\quad i = i + 1;$

$\quad N_i = N_{i-1} \cup \{X | X \in N,\ \ \exists X \rightarrow \alpha \in P, \alpha \in N_{i-1}^*\};$

$\}$ while $N_i \neq N_{i-1};$

$N_\epsilon = N_i;$

Are loc:

- $N_0 \subseteq N_1 \ldots \subseteq N_i \subseteq N_{i+1} \subseteq \ldots N_\epsilon \subseteq N$
- $A \in N_\epsilon \Longleftrightarrow A \Rightarrow^+ \epsilon$

Eliminarea regulilor de ştergere

P' se obţine din P astfel:

- în fiecare regulă $A \to \alpha \in P$ se pun în evidenţă simbolurile din N_ϵ ce apar în α:

$$\alpha = \alpha_1 X_1 \alpha_2 X_2 \ldots \alpha_n X_n \alpha_{n+1}, \; X_i \in N_\epsilon$$

- se înlocuieşte fiecare regulă de acest fel cu mulţimea de reguli de forma

$$A \to \alpha_1 Y_1 \alpha_2 Y_2 \ldots \alpha_n Y_n \alpha_{n+1} \text{ unde } Y_i = X_i \text{ sau } Y_i = \epsilon$$

  în toate modurile posibile (2^n)
- se elimină toate regulile de ştergere
- pentru a obţine cuvântul nul (dacă S este în N_ϵ) se adaugă S' simbol de start nou şi regulile $S' \to S$, $S' \to \epsilon$

Exemplu

$G = (\{S, A, B, C\}, \{a, b, c\}, S, P)$, unde P:

- $S \to aAbC|BC$
- $A \to aA|aB$
- $B \to bB|C$
- $C \to cC|\epsilon$

$G' = (\{S', S, A, B, C\}, \{a, b, c\}, S', P')$ unde P':

- $S' \to S|\epsilon$
- $S \to aAbC|aAb|B|C$
- $A \to aA|aB|a$
- $B \to bB|b|C$
- $C \to cC|c$