

# Limbaje Formale, Automate și Compilatoare

Curs 4

2021-22

# Curs 4



- 1 Gramatici de tip 3 și automate finite
- 2 Proprietăți de închidere pentru clasa limbajelor de tip 3
- 3 Expresii regulate
- 4 Automatul echivalent cu o expresie regulată
  - Algoritm

# Curs 4

- 1 Gramatici de tip 3 și automate finite
- 2 Proprietăți de închidere pentru clasa limbajelor de tip 3
- 3 Expresii regulate
- 4 Automatul echivalent cu o expresie regulată
  - Algoritm

# De la gramatici de tip 3 la automate finite

tip  $A \rightarrow uB$   $A, B \in N$   
 $A \rightarrow u$   $u \in T^*$  forma normală:  $A \rightarrow aB$   $a \in T$   $A, B \in N$   
 $A \xrightarrow{\text{sg. neputură}} a$ :  $S \rightarrow \epsilon$  ( $\xrightarrow{\text{sub stand}}$ )

- Pentru orice gramatică  $G$  de tip 3 (în formă normală) există un automat  $A$  (nedeterminist) astfel ca  $L(A) = L(G)$ :

$$G = (T, \Sigma, N, S, P)$$

$$P: 1. x \rightarrow ax$$

$$2. x \rightarrow b y$$

$$3. y \rightarrow by$$

$$4. y \rightarrow b$$

$$L(G) = \{a^n b^m \mid$$

$$n \geq 0$$

$$m \geq 0\}$$

↳

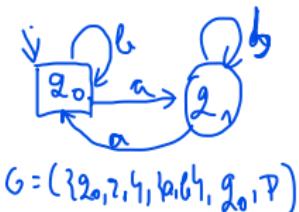
În gramatica $G$	În automatul $A$
$T$	$\Sigma = T$
$N$	$Q = N \cup \{f\}, F = \{f\}$
$S$	$q_0 = S$
$\xrightarrow{\text{en } \epsilon T} q \rightarrow ap \xrightarrow{\epsilon N} p$	$p \in \delta(q, a) \quad q \xrightarrow{a} p$
$q \rightarrow a$	$f \in \delta(q, a)$
dacă $S \rightarrow \epsilon$	se adaugă $S$ la $F$



# De la automate finite la gramatici de tip 3

$$\underline{L} = \{ w \in \{a, b\}^* \mid w \text{ conține un par alături}\}$$

- Pentru orice automat finit (nedeterminist) există o gramatică  $G$  de tip 3 astfel ca  $L(A) = L(G)$ :



$$P : \left\{ \begin{array}{l} q_0 \rightarrow b | q_0 \\ q_0 \rightarrow a q_1 \\ q_1 \rightarrow b q_1 \\ q_1 \rightarrow a | q_0 \\ q_0 \rightarrow b \\ q_1 \rightarrow a \end{array} \right.$$

În automatul A	În gramatica G
$\Sigma$	$T = \Sigma$
$Q$	$N = Q$
$q_0$	$S = q_0$
$p \in \delta(q, a)$	$q \rightarrow ap$
$\delta(q, a) \cap F \neq \emptyset$	$q \rightarrow a$
dacă $q_0 \in F$	se adaugă $q_0 \rightarrow \epsilon$

$$\left. \begin{array}{l} q_0 \rightarrow b \\ q_1 \rightarrow a \end{array} \right\} (\delta(q_0, b) \cap F \neq \emptyset) \quad \left. \begin{array}{l} q_0 \rightarrow a \\ q_1 \rightarrow b \end{array} \right\} (\delta(q_1, a) \cap F \neq \emptyset)$$

## Curs 4

- 1 Gramatici de tip 3 și automate finite
- 2 Proprietăți de închidere pentru clasa limbajelor de tip 3
- 3 Expresii regulate
- 4 Automatul echivalent cu o expresie regulată
  - Algoritm

# Închiderea la intersecție

$L = \{w \in \{a, b\}^* \mid w \text{ are un par alături de } a\}$  și  $L_1 = \{w : \text{par a}\}$

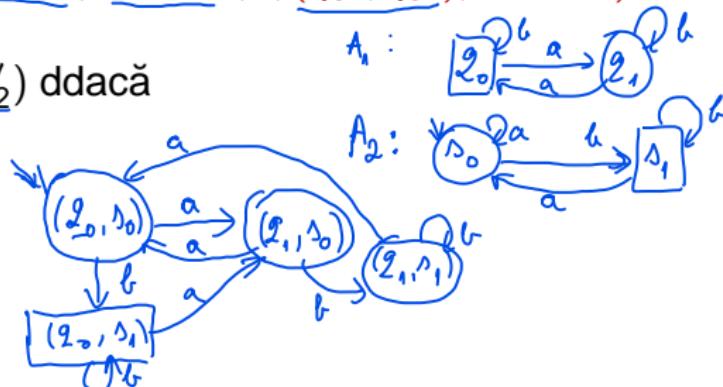
- Dacă  $L_1, L_2 \in \mathcal{L}_3$ , atunci  $L_1 \cap L_2 \in \mathcal{L}_3$      $L = L_1 \cap L_2$      $L_1 = \{w : \text{par a}\}$

Fie  $A_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, F_1)$  și  $A_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, F_2)$  automate deterministe astfel încât  $L_1 = L(A_1)$  și  $L_2 = L(A_2)$ .  
 Automatul  $A$  (determinist) care recunoaște  $L_1 \cap L_2$ :

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

$$\delta((\underline{q_1, q_2}), a) = (\underline{q'_1, q'_2}) \text{ dacă}$$

- $\delta_1(q_1, a) = q'_1$
- $\delta_2(q_2, a) = q'_2$



# Închiderea la diferență $(L_1 \setminus L_2)$

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

$L = \{ w \in \{a, b, c\}^* \mid w \text{ nu este "abc"} \}$

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

Automatul  $A'$  care recunoaște  $\underline{\bar{L}} = \overline{L(A)}$ :

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

# Închiderea la diferență

- 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 recunoaște  $\overline{L} = \overline{L(A)}$ :

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

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

## Închiderea la produs

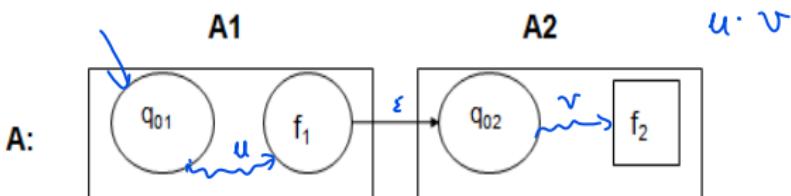


- Fie  $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, \{f_1\})$  și  $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, \{f_2\})$  automate cu o singură stare finală astfel încât  $L_1 = L(A_1)$  și  $L_2 = L(A_2)$ .

Automatul  $A$  (cu  $\epsilon$ -tranzitii) care recunoaște  $\underline{L_1 \cdot L_2}$ :

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

$$\delta(f_1, \epsilon) = \{q_{02}\}$$

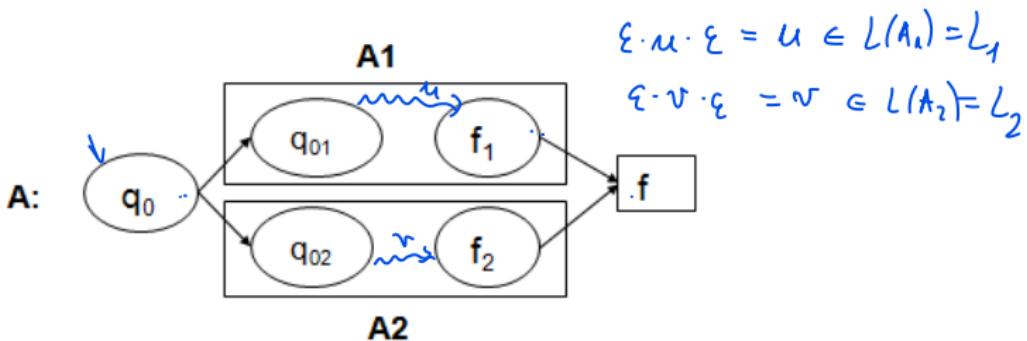


# Închiderea la reuniune

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

Automatul  $A$  (cu  $\epsilon$ -tranzitii) care recunoaște  $\underline{L_1 \cup L_2}$ :

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

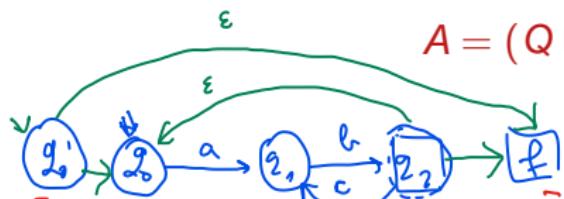


# Închiderea la iterare

- Fie  $A = (Q, \Sigma, \delta, q_{01}, \{f\})$  automat cu o singură stare finală astfel încât  $L(A) = L$ .

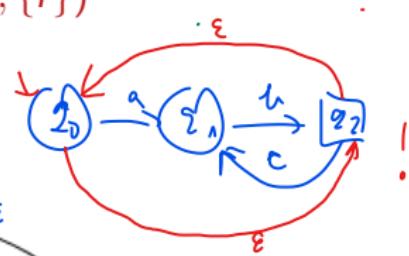
$$L^* \stackrel{\text{def}}{=} \frac{u_1}{\epsilon L} \frac{u_2}{\epsilon L} \dots \frac{u_m}{\epsilon L} \stackrel{\epsilon}{\in} \Sigma^*$$

Automatul  $A$  (cu  $\epsilon$ -tranzitii) care recunoaște  $L^*$  ( $= L(A)^*$ ):



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

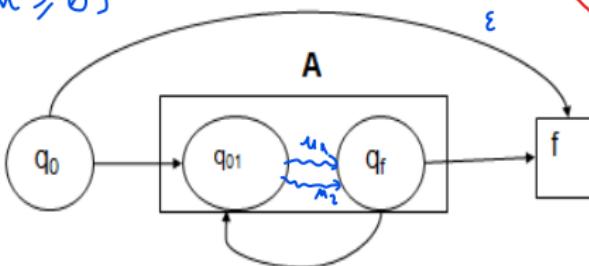
$$L = \{ ab(c\bar{b})^m, m \geq 0 \}$$



$$(c\bar{b})^n \notin L^*$$

$$L^*$$

$$ab \xrightarrow{\epsilon} ab \xrightarrow{\epsilon} ab \xrightarrow{\epsilon} \dots \quad A':$$



# Curs 4

- 1 Gramatici de tip 3 și automate finite
- 2 Proprietăți de închidere pentru clasa limbajelor de tip 3
- 3 **Expresii regulate**
- 4 Automatul echivalent cu o expresie regulată
  - Algoritm

# Expresii regulate - definiție

- Reprezentarea limbajelor de tip 3 prin expresii algebrice

## Definiție 1

Dacă  $\Sigma$  este un alfabet atunci o expresie regulată peste  $\Sigma$  se definește inductiv astfel:

$$L(\phi) = \emptyset \quad L(\epsilon) = \{\epsilon\} \quad L(a) = \{a\}$$

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

$$L((E^*)) = (L(E))^*$$

# Expresii regulate - definiție



- Reprezentarea limbajelor de tip 3 prin expresii algebrice

## Definiție 1

Dacă  $\Sigma$  este un alfabet atunci o expresie regulată peste  $\Sigma$  se definește inductiv astfel:

- $\emptyset, \epsilon, a$  ( $a \in \Sigma$ ) 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  $L(E_1) \cup L(E_2)$
- $(E_1 : E_2)$  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 \*, ·, |

**Exemple**

$$L((a|b)^*) = L(a|b) = \underbrace{(L(a) \cup L(b))^*}_{\{a,b\}^*} \quad L(\overbrace{(a|b)}^{E_1} | \overbrace{(c|d)}^{E_2}) = L(a|b) \cup L(c|d)$$

- $\underline{(a|b)|(c|d)} \rightarrow \{a, b, c, d\}$

$$(L(a) \cup L(b)) \cup (L(c) \cup L(d))$$

- $(0|1) \cdot (0|1) \rightarrow \{00, 01, 10, 11\}$

$$L(\overbrace{(0|1)}^{E_1} \cdot \overbrace{(0|1)}^{E_2}) = L(0|1) \cdot L(0|1)$$

- $a^*b^* \rightarrow \{a^n b^k | n, k \geq 0\}$

$$L(a^* \cdot b^*) = L(a^*) \cdot L(b^*) \\ = (L(a))^* \cdot (L(b))^* = \{0,1\}^* \cdot \{0,1\}^*$$

- $\underline{(0|1|2|...|9)(0|1|2|...|9)^*}$  descrie multimea intregilor fara semn

$$(1|2|..|9) \cdot (0|1|..|9)^* | \square$$

- $(a|b|c|...|z)(a|b|c|...|z|0|1|2|...|9)^*$  descrie multimea identificatorilor

$$\{a^n, n > 1\} L(a|a^*) = L(a) \cup L(a^*) = \{a\} \cup \{a^n, n > 0\} = a^*$$

Două expresii regulate  $E_1, E_2$  sunt echivalente, și scriem  $E_1 = E_2$  dacă

$$L(E_1) = L(E_2)$$

$$a \cdot a^* \quad a^* \cdot a \quad \{a, b\}^+ : (a|b) \cdot (a|b)^*$$

$$a | a^* = a^* \\ a \cdot a^* \equiv a^* \cdot a$$

## Proprietăți

$$\alpha^* \mid (\alpha\alpha)^* = \underbrace{(\alpha\alpha)^*}_{\{\alpha^n, n \geq 0\} \cup \{\alpha^{2k}, k \geq 0\}} \alpha^* \quad \{ \alpha^m u \mid m \geq 0, u \in \Sigma, \Sigma^*\}^* = (\alpha|u)^*$$

$$L((p|q)|r) = L(p| (q|r)) \Leftarrow$$

$$\bullet (p|q)|r = p|(q|r) \quad (L(p) \cup L(q)) \cup L(r) = L(p) \cup (L(q) \cup L(r))$$

$$\bullet (pq)r = p(qr) \quad \alpha^* \cdot \beta^* \cdot \gamma^*$$

$$\bullet p|q = q|p \quad L(p) \cup L(q) = L(q) \cup L(p)$$

$$\bullet p \cdot \epsilon = \epsilon \cdot p = p \quad L(p \cdot \epsilon) = L(p) \cdot L(\epsilon) = L(p) \cdot \{\epsilon\}^* = L(p)$$

$$\bullet p|\emptyset = p|p = p \quad L(p|\emptyset) = L(p) \cup \emptyset = L(p)$$

$$\bullet \emptyset \cdot p = p \cdot \emptyset = \emptyset$$

$$p \cdot \epsilon = p$$

$$p \mid p = p$$

$$\bullet p(q|r) = pq|pr$$

$$p \mid \epsilon = p \quad ?$$

$$p \cdot p = p$$

$$\bullet (p|q)r = pr|qr$$

$$p=a \quad a|\epsilon \stackrel{?}{=} a \\ \{aa\}^* \neq \{ab\}^*$$

$$p=a \quad \{aa\}^* \neq \{ab\}^*$$

$$\bullet \epsilon|pp^* = p^*$$

$$p=a^* \quad a^* \mid \epsilon = a^* \quad ?$$

$$p=a^* \quad ?$$

$$\bullet \epsilon|p^*p = p^*$$

$$\epsilon \mid a \cdot a^* = a^* \quad (p^*)^* = p^*$$

$$a^* \cdot a^* = a^*$$

# De la o expresie regulată la automatul finit

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

a a b

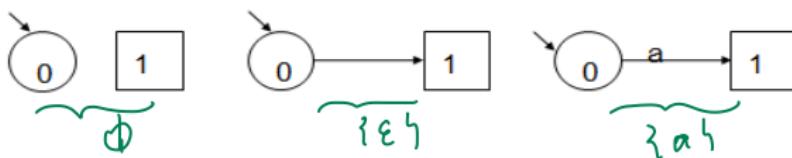
## Teorema 1

Pentru orice expresie regulată  $E$  peste  $\Sigma$  există un automat finit (cu  $\epsilon$ -tranzitii)  $A$ , astfel încât  $L(A) = L(E)$ .

Demonstratie: inducție structurală.

- St. inițială  $\Rightarrow$  st. finală
- o singură stare finală

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



$\emptyset :$

$\{ε\} :$

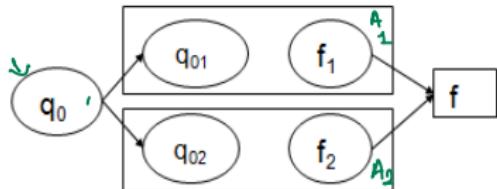
# Demonstrație

- $E = E_1 | E_2$

$$\begin{array}{l} A_1 \approx E_1 \\ A_2 \approx E_2 \end{array} \quad L(A_1) = L(E_1)$$

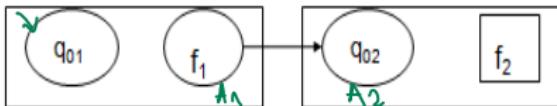
$$L(A_2) = L(E_2)$$

$$\begin{aligned} A : L(A) \sqsupseteq L(E) &= L(E_1) \cup L(E_2) \\ &= \underline{L(A_1)} \cup \underline{L(A_2)} \end{aligned}$$



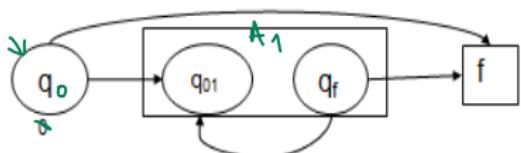
- $E = E_1 \cdot E_2$

¶ A :  $L(A) = L(E) = L(E_1) \cdot L(E_2)$   
 $= L(A_1) \cdot L(A_2)$



- $E = E_1^*$

A  $L(A) = L(E) = (L(E_1))^*$   
 $= (L(A_1))^*$



# Reprezentarea expresiilor regulate sub formă de arbore

- Intrare:** Expresia regulată  $E = e_0e_1 \dots e_{n-1}$

Precedența operatorilor:

$$\text{prec}(\text{!}) = 1, \text{prec}(\cdot) = 2, \text{prec}(\ast) = 3 (\text{prec}(\text{()}) = 0).$$

- Ieșire:** Arborele asociat:  $t$ .

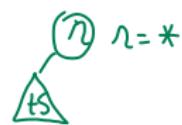
- Metoda:** Se consideră două stive:

- STIVA1 stiva operatorilor
- STIVA2 stiva arborilor (care va conține arborii parțiali construiți)
- Metoda  $\text{tree}(r, tS, tD)$



$\text{tree}(n, \text{NULL}, \text{NULL})$

$\text{tree}(n, +S, \text{NULL})$



# Algoritm

```

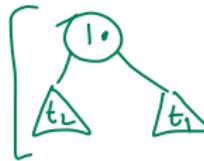
i = 0;
while(i < n) {
    c = ei;
    switch(c) {
        case '(': { STIVA1.push(c); break; }
        case simbol (din alfabet): { STIVA2.push(tree(c,NULL,NULL)); break; }
        case operator: {
            while (prec(STIVA1.top())>=prec(c))
                build_tree();
            STIVA1.push(c);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

```

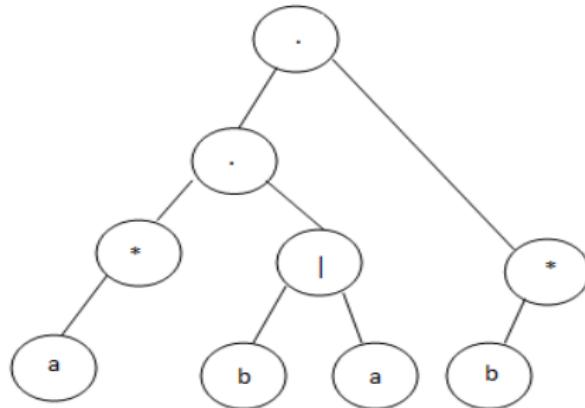
(c)

# 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;
    }
}
```



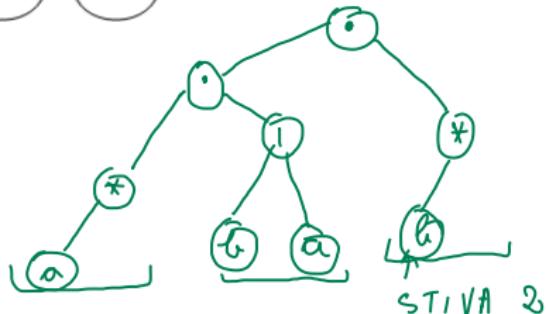
# Exemplu



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

~~\*/\|•\*~~

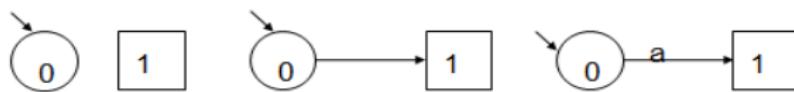
STIV 1



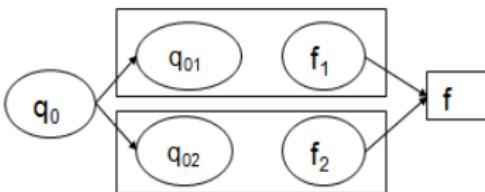
# Curs 4

- 1 Gramatici de tip 3 și automate finite
- 2 Proprietăți de închidere pentru clasa limbajelor de tip 3
- 3 Expresii regulate
- 4 Automatul echivalent cu o expresie regulată
  - Algoritm

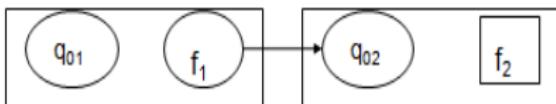
# Automatul echivalent cu o expresie regulată



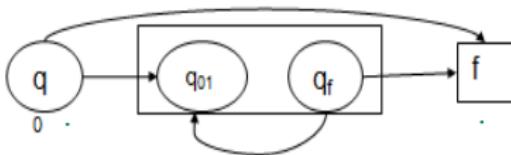
$(a^* \cdot b)l \cdot c$   
1 o stăru



- $E = E_1 | E_2$



- $E = E_1 E_2$



- $E = E_1^*$

## Observații

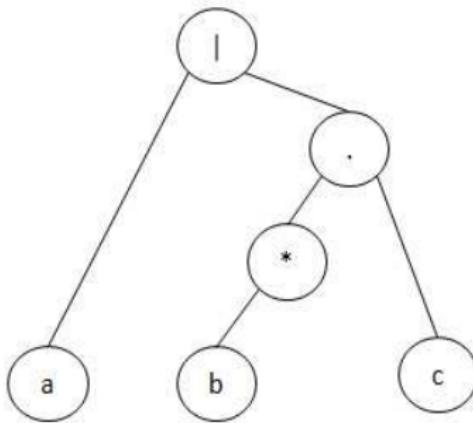
- pentru orice apariție a unui simbol din  $\Sigma$ , cât și pentru  $\epsilon$ , dacă acesta apare explicit în  $E$ , este nevoie de 2 stări în automatul construit.
- fiecare din aparițiile operatorilor  $|$  și  $*$  dintr-o expresie regulată  $E$  introduce două noi stări în automatul construit
-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

# Algoritm

- **Intrare:** Expresia regulată  $E$  cu  $n$  simboluri dintre care  $m$  sunt paranteze și apariții ale operatorului produs;
  - **Ieșire:** Automatul (cu  $p = 2(n - m)$  stări) cu  $\epsilon$  - tranziții echivalent cu  $E$
  - Fie o metodă `generateState()` care generează o stare nouă la fiecare apel (stările: numere crescătoare incepând de la 1)
- 1 Se construiește arborele atașat expresiei  $E$ ;
  - 2 Se parcurge arborele în postordine
    - Pentru fiecare nod  $N$ , se vor genera (dacă este cazul) și memora două stări,  $N.i$ ,  $N.f$ , care reprezintă starea inițială respectiv finală a automatului  $A_N$  echivalent cu expresia  $E_N$  corespunzătoare subarborelui cu rădăcina  $N$
    - Automatul  $A_N$  va fi construit pe baza automatelor construite la pașii anteriori, utilizând Teorema 1
  - 3 Starea inițială a automatului este  $N.i$ , starea finală  $N.f$ , unde  $N$  este nodul rădăcină;



# Exemplu



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



## 2. Se parcurge arborele obținut în postordine.

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

## 2. Se parcurge arborele obținut în postordine.

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

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

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

## 2. Se parcurge arborele obținut în postordine.

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

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

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

- Dacă N este etichetat cu |:  
 $N.i = generateState(), N.f = generateState(),$

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

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

## 2. Se parcurge arborele obținut în postordine.

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

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

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

- Dacă N este etichetat cu |:  
 $N.i = generateState(), N.f = generateState(),$

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

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

- Dacă N este etichetat cu · :  
 $N.i = S.i, N.f = S.f,$

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

## 2. Se parcurge arborele obținut în postordine.

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

- Dacă N este etichetat cu a (deci este frunza):  
 $N.i = \text{generateState}()$ ,  $N.f = \text{generateState}()$ ,

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

- Dacă N este etichetat cu |:  
 $N.i = \text{generateState}()$ ,  $N.f = \text{generateState}()$ ,

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

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

- Dacă N este etichetat cu · :  
 $N.i = S.i$ ,  $N.f = S.f$ ,

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

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

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

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

2. Se parcurge arborele obținut în postordine.

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

- Dacă N este etichetat cu a (deci este frunza):  
 $N.i = \text{generateState}(), N.f = \text{generateState}(),$

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

- Dacă N este etichetat cu |:  
 $N.i = \text{generateState}(), N.f = \text{generateState}(),$

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

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

- Dacă N este etichetat cu .:  
 $N.i = S.i, N.f = S.f,$

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

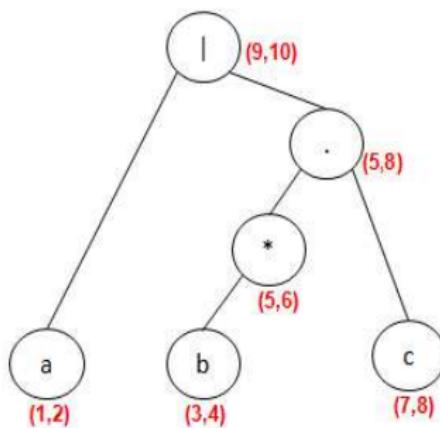
- Dacă N este etichetat cu \* (D nu există în acest caz):  
 $N.i = \text{generateState}(), N.f = \text{generateState}(),$

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

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

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

# Exemplu



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

# Exemplu

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

# Corectitudinea algoritmului

## Teorema 2

*Algoritmul descris este corect: automatul cu  $\epsilon$  - tranzitii obtinut este echivalent cu expresia regulata E.*

### Demonstratie:

- Tranzitiile care se definesc in algoritm urmaresc constructia din teorema 1.

Automatul obtinut este echivalent cu expresia data la intrare.