# Limbaje Formale, Automate şi Compilatoare

Curs 1

2019-20

# Limbaje Formale, Automate şi Compilatoare - Curs 1

1 Prezentare curs

2 Limbaje formale

3 Mecanisme de generare a limbajelor: gramatici

4 Ierarhia lui Chomsky

5 Limbaje şi gramatici de tip 3 (regulate)

# Limbaje Formale, Automate şi Compilatoare

Titulari curs:

- O. Captarencu: otto@info.uaic.ro

  http://profs.info.uaic.ro/~otto/lfac.html
- A. Moruz:mmoruz@info.uaic.ro

# Sistem evaluare

- 7 seminarii, 6 laboratoare;
- AS = activitatea la seminar (max 10 puncte);
- AL = activitatea la laborator (max 10 puncte);
- T1,T2 teste scrise în săptămânile 8, respectiv în sesiune;

  Punctajul final se obţine astfel:

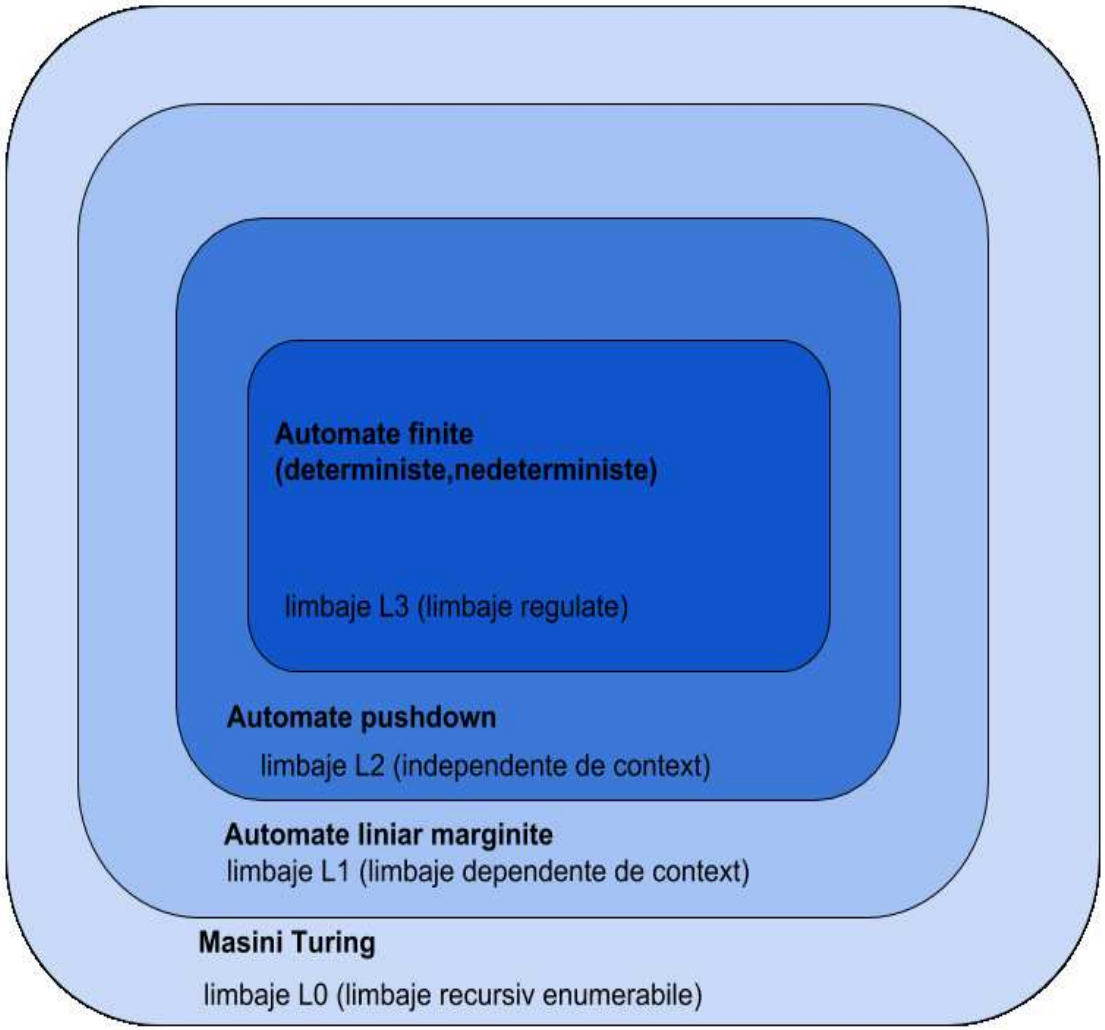
  P = 3 * AS + 3 * AL + 2 * T1 + 2 * T2
- Condiţii miminale de promovare: AS $\geq$ 5, AL $\geq$ 5, $T1 \geq 5$, $T2 \geq 5$;
- Punctaj minim pentru promovare: P $\geq$ 50;
- Nota finală (N) se va stabili astfel:

  dacă $P < 50$: $N = 4$, altfel: $N = round(P/10)$ (rotunjirea se face la cel mai apropiat întreg);

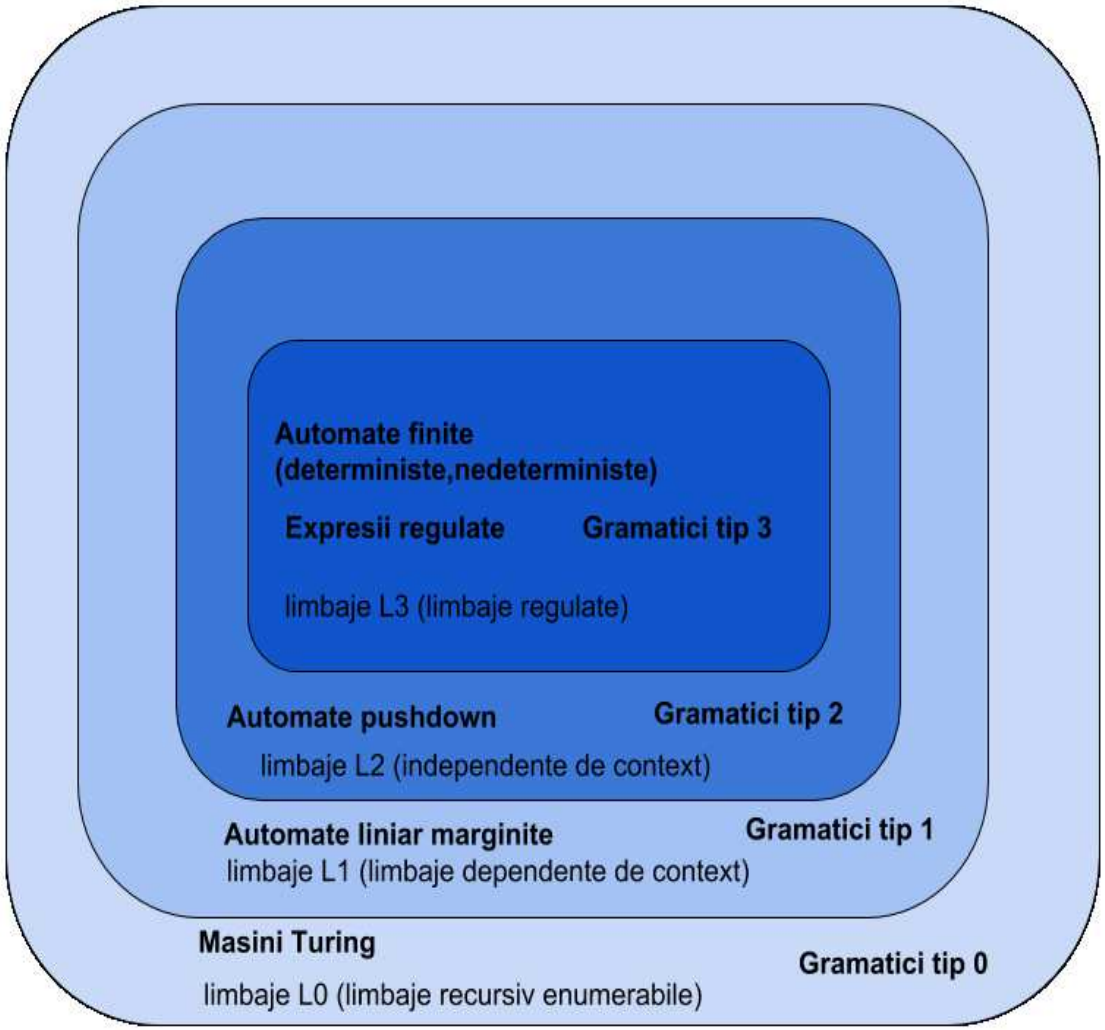
# Sistem evaluare

- AS = activitatea la seminar (max 10 puncte):
  - - media notelor de la două teste scrise, notate de la 0 la 10 (fără rotunjiri)
- AL = activitatea la laborator (max 10 puncte):
  - 1 test laborator, 1 proiect (note de la 0 la 10)
  - AL = media celor 2 note (fără rotunjiri)

# Tematica cursului (partea I)



# Tematica cursului (partea I)



# Tematica cursului (partea I)

- Limbaje şi gramatici
- Limbaje regulate; gramatici, automate , expresii regulate
- Limbaje independente de context; gramatici, automate pushdown

# Tematica cursului (partea II)

- Limbaje de programare: proiectare şi implementare
- Analiza lexicală
- Analiza sintactică
- Traducere în cod intermediar

# Bibliografie (selecţii)


1. A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman: Compilers: Principles, Techniques, and Tools. Boston: Addison-Wesley, 2007
2. Gh. Grigoras. Constructia compilatoarelor - Algoritmi fundamentali, Ed. Universitatii Al. I. ”Cuza Iasi”, ISBN 973-703-084-2, 274 pg., 2005
3. Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. (2006). Introduction to Automata Theory, Languages, and Computation (3rd ed.). Addison-Wesley
4. J. Toader - Limbaje formale şi automate, Editura Matrix Rom, Bucuresti, 1999.
5. J. Toader, S. Andrei - Limbaje formale şi teoria automatelor. Teorie şi practică, Editura Universitatii ”Al. I. Cuza”, Iasi, 2002.

# Limbaje Formale, Automate şi Compilatoare - Curs 1


1. Prezentare curs
2. Limbaje formale
3. Mecanisme de generare a limbajelor: gramatici
4. Ierarhia lui Chomsky
5. Limbaje şi gramatici de tip 3 (regulate)

# Alfabet, cuvânt, multţime de cuvinte

- Alfabet: V o mulţime finită (elementele lui V = simboluri )

# Alfabet, cuvânt, multţime de cuvinte

- Alfabet: V o mulţime finită (elementele lui V = simboluri )
- Cuvânt: şir finit de simboluri
  - cuvântul nul este notat cu $\epsilon$ sau $\lambda$.

# Alfabet, cuvânt, mulţime de cuvinte

- Alfabet: V o mulţime finită (elementele lui V = simboluri )
- Cuvânt: şir finit de simboluri
  - cuvântul nul este notat cu $\epsilon$ sau $\lambda$.
- Lungimea unui cuvânt $u$: numarul simbolurilor sale. Notaţie: $|u|$.

  $|\epsilon| = 0$

# Alfabet, cuvânt, mulţime de cuvinte

- Alfabet: V o mulţime finită (elementele lui V = simboluri )
- Cuvânt: şir finit de simboluri
  - cuvântul nul este notat cu $\epsilon$ sau $\lambda$.
- Lungimea unui cuvânt $u$: numarul simbolurilor sale. Notaţie: $|u|$.

  $|\epsilon| = 0$
- $V^*$ - multimea tuturor cuvintelor peste alfabetul V, inclusiv $\epsilon$.

  $\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

# Alfabet, cuvânt, mulţime de cuvinte

- Alfabet: V o mulţime finită (elementele lui V = simboluri )
- Cuvânt: şir finit de simboluri
  - cuvântul nul este notat cu $\epsilon$ sau $\lambda$.
- Lungimea unui cuvânt $u$: numarul simbolurilor sale. Notaţie: $|u|$.

  $|\epsilon| = 0$
- $V^*$ - multimea tuturor cuvintelor peste alfabetul V, inclusiv $\epsilon$.

  $\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
- $V^+$ - multimea tuturor cuvintelor nenule peste alfabetul V

  $\{0,1\}^+ = \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

# Operaţii pe cuvinte

- Concatenarea a doua cuvinte x, y: cuvântul $x \cdot y$ obţinut din simbolurile lui x, în ordinea în care apar, urmate de cele ale lui y de asemenea în ordinea în care apar:

  $x = 0100$, $y = 100$, $x \cdot y = 0100100$

  $x = 000$, $y = \epsilon$, $x \cdot y = 000$

# Operaţii pe cuvinte

- Concatenarea a doua cuvinte x, y: cuvântul $x \cdot y$ obţinut din simbolurile lui x, în ordinea în care apar, urmate de cele ale lui y de asemenea în ordinea în care apar:

  $x = 0100$, $y = 100$, $x \cdot y = 0100100$

  $x = 000$, $y = \epsilon$, $x \cdot y = 000$
- Concatenarea este asociativă

# Operaţii pe cuvinte

- Concatenarea a doua cuvinte x, y: cuvântul $x \cdot y$ obţinut din simbolurile lui x, în ordinea în care apar, urmate de cele ale lui y de asemenea în ordinea în care apar:

  $x = 0100$, $y = 100$, $x \cdot y = 0100100$

  $x = 000$, $y = \epsilon$, $x \cdot y = 000$
- Concatenarea este asociativă
- $(V^*, \cdot)$ este monoid ($\epsilon$ este element neutru), se numeşte monoidul liber generat de $V$.

# Operaţii pe cuvinte

- Concatenarea a doua cuvinte x, y: cuvântul $x \cdot y$ obţinut din simbolurile lui x, în ordinea în care apar, urmate de cele ale lui y de asemenea în ordinea în care apar:

  $x = 0100$, $y = 100$, $x \cdot y = 0100100$

  $x = 000$, $y = \epsilon$, $x \cdot y = 000$
- Concatenarea este asociativă
- $(V^*, \cdot)$ este monoid ($\epsilon$ este element neutru), se numeşte monoidul liber generat de $V$.
- Cuvântul $v$ este un prefix al cuvântului $u$ dacă $\exists w \in V^* : u = vw$; dacă $w \in V^+$ , atunci $v$ este un prefix propriu al lui $u$.

# Operaţii pe cuvinte

- Concatenarea a doua cuvinte x, y: cuvântul $x \cdot y$ obţinut din simbolurile lui x, în ordinea în care apar, urmate de cele ale lui y de asemenea în ordinea în care apar:

  $x = 0100$, $y = 100$, $x \cdot y = 0100100$

  $x = 000$, $y = \epsilon$, $x \cdot y = 000$
- Concatenarea este asociativă
- $(V^*, \cdot)$ este monoid ($\epsilon$ este element neutru), se numeşte monoidul liber generat de $V$.
- Cuvântul $v$ este un prefix al cuvântului $u$ dacă $\exists w \in V^* : u = vw$; dacă $w \in V^+$ , atunci $v$ este un prefix propriu al lui $u$.
- Cuvântul $v$ este un sufix al cuvântului $u$ dacă $\exists w \in V^* : u = wv$; dacă $w \in V^+$ , atunci $v$ este un sufix propriu al lui $u$.

- Fie $V$ un alfabet. O submulţime $L \subseteq V^*$ este un limbaj (formal) peste alfabetul $V$ (sau V-limbaj) dacă L are o descriere (matematică) finită.
- O descriere poate fi:

- Fie $V$ un alfabet. O submulţime $L \subseteq V^*$ este un limbaj (formal) peste alfabetul $V$ (sau V-limbaj) dacă L are o descriere (matematică) finită.
- O descriere poate fi:
  - neformală (în limbaj natural):
    - multimea cuvintelor peste alfabetul $\{0, 1\}$ care contin un numar par de 0.
    - $L = \{x \in V^+ : |x| \textit{ este par}\}$.
    - $\{a^n b^n | n \in N\}$.
    - $\{w \in \{0, 1\}^* | w \textit{ se termina in } 00\}$.

- Fie $V$ un alfabet. O submulţime $L \subseteq V^*$ este un limbaj (formal) peste alfabetul $V$ (sau V-limbaj) dacă L are o descriere (matematică) finită.
- O descriere poate fi:
  - neformală (în limbaj natural):
    - multimea cuvintelor peste alfabetul $\{0, 1\}$ care contin un numar par de 0.
    - $L = \{x \in V^+ : |x| \textit{ este par}\}$.
    - $\{a^n b^n | n \in N\}$.
    - $\{w \in \{0, 1\}^* | w \textit{ se termina in } 00\}$.
  - formală (descriere matematică):
    - o descriere inductivă a cuvintelor
    - o descriere generativă a cuvintelor (gramatică generativă)
    - o descriere a unei metode de recunoaştere a cuvintelor din limbaj (automat finit, automat pushdown, etc.)

# Operaţii cu limbaje

- Operatiile cu multimi (reuniune, intersectie etc)
- Produs de limbaje: $L_1 \cdot L_2 = \{u \cdot v | u \in L_1, v \in L_2\}$

  Exemplu:

  $L_1 = \{a^n, n \geq 1\}$, $L_2 = \{b^n, n \geq 1\}$

  $L_1 \cdot L_2 = \{a^n b^m, n \geq 1, m \geq 1\}$
- Iteraţia (produsul Kleene): $L^* = \bigcup_{n \geq 0} L^n$, unde:
  - $L^0 = \{\epsilon\}$
  - $L^{n+1} = L^n \cdot L$

  Exemplu:

  $L = \{a\}$, $L^0 = \{\epsilon\}, L^1 = L, L^2 = \{aa\}, \ldots, L^n = \{a^n\}$

  $L^* = \{a^n, n \geq 0\}$

# Limbaje Formale, Automate şi Compilatoare - Curs 1

1 Prezentare curs

2 Limbaje formale

3 Mecanisme de generare a limbajelor: gramatici

4 Ierarhia lui Chomsky

5 Limbaje şi gramatici de tip 3 (regulate)

# Gramatici

## Definiţie 1

*O gramatica este un sistem $G = (N, T, S, P)$, unde:*

- *$N$ şi $T$ sunt două alfabete disjuncte:*
  - *$N$ este multimea neterminalilor*
  - *$T$ este multimea terminalilor*
- *$S \in N$ este simbolul de start (neterminalul iniţial)*
- *$P$ este o multime finita de reguli (producţii) de forma $x \to y$, unde $x, y \in (N \cup T)^*$ şi $x$ conţine cel puţin un neterminal.*

# Derivare

## Definiţie 2

*Fie $G = (N, T, S, P)$ o gramatica şi $u, v \in (N \cup T)^*$.*

*Spunem că v este derivat direct (într-un pas) de la u prin aplicarea regulii $x \to y$, şi notăm $u \Rightarrow v$, dacă $\exists p, q \in (N \cup T)^*$ astfel încât $u = pxq$ şi $v = pyq$.*

# Derivare

## Definiţie 2

*Fie $G = (N, T, S, P)$ o gramatica şi $u, v \in (N \cup T)^*$.*

*Spunem că v este derivat direct (într-un pas) de la u prin aplicarea regulii $x \to y$, şi notăm $u \Rightarrow v$, dacă $\exists p, q \in (N \cup T)^*$ astfel încât $u = pxq$ şi $v = pyq$.*

- Daca $u_1 \Rightarrow u_2 \ldots \Rightarrow u_n, n > 1$, spunem ca $u_n$ este derivat din $u_1$ în $G$ şi notam $u1 \Rightarrow^+ u_n$.

# Derivare

## Definiţie 2

*Fie $G = (N, T, S, P)$ o gramatica şi $u, v \in (N \cup T)^*$.*

*Spunem că v este derivat direct (într-un pas) de la u prin aplicarea regulii $x \to y$, şi notăm $u \Rightarrow v$, dacă $\exists p, q \in (N \cup T)^*$ astfel încât $u = pxq$ şi $v = pyq$.*

- Daca $u_1 \Rightarrow u_2 \ldots \Rightarrow u_n, n > 1$, spunem ca $u_n$ este derivat din $u_1$ în $G$ şi notam $u1 \Rightarrow^+ u_n$.
- Scriem $u \Rightarrow^* v$ dacă $u \Rightarrow^+ v$ sau $u = v$.

# Limbaj generat

## Definiţie 3

*Limbajul generat de gramatica G este:*

$$L(G) = \{w \in T^* | S \Rightarrow^+ w\}$$

# Limbaj generat

## Definiţie 3

*Limbajul generat de gramatica G este:*

$$L(G) = \{w \in T^* | S \Rightarrow^+ w\}$$

## Definiţie 4

*Două gramatici $G_1$ şi $G_2$ sunt echivalente dacă $L(G_1) = L(G_2)$.*

# Exemplu

- $G = (N, T, S, P)$, $N = \{S, X, A\}$, $T = \{a, b\}$, P constă din:
  1. $S \to aXb$
  2. $aX \to aAb$
  3. $Xb \to bA$
  4. $aA \to aa$
  5. $A \to \epsilon$
- $L(G) = \{ab, abb, aabb\}$
- Gramatică echivalentă cu un singur neterminal ?
- Ce limbaj generează gramatica dacă sunt eliminate utlimele două reguli?

# Exemplu

- $L = \{a^n b^n | n \geq 1\}$
- Definiţia inductivă:
  - $ab \in L$
  - Daca $X \in L$, atunci $aXb \in L$
  - Nici un alt cuvânt nu face parte din L

# Exemplu

- $L = \{a^n b^n | n \geq 1\}$
- Definiţia inductivă:
  - $ab \in L$
  - Daca $X \in L$, atunci $aXb \in L$
  - Nici un alt cuvânt nu face parte din L
- Definiţia generativă:
  - $G = (\{X\}, \{a, b\}, X, P)$, unde $P = \{X \to aXb, X \to ab\}$
  - Derivarea cuvântului $a^3 b^3$:

    $X \Rightarrow aXb \Rightarrow a(aXb)b \Rightarrow aa(ab)bb$

# Exemplu

- $L = \{a^n b^n c^n | n \geq 1\}$
- $G = (N, T, S, P)$, $N = \{S, X\}$, $T = \{a, b, c\}$, P constă din:
  1. $S \to abc$
  2. $S \to aSXc$
  3. $cX \to Xc$
  4. $bX \to bb$
- Derivarea cuvântului $a^3 b^3 c^3$:

  $S \Rightarrow^{(2)} a\underline{S}Xc \Rightarrow^{(2)} aa\underline{S}XcXc \Rightarrow^{(1)} aaab\underline{cX}cXc \Rightarrow^{(3)}$

  $aaa\underline{bX}ccXc \Rightarrow^{(4)} aaabbc\underline{cX}c \Rightarrow^{(3)} aaabb\underline{cX}cc \Rightarrow^{(3)}$

  $aaab\underline{bX}ccc \Rightarrow^{(4)} aaabbbccc = a^3 b^3 c^3$

# Limbaje Formale, Automate şi Compilatoare - Curs 1


1. Prezentare curs
2. Limbaje formale
3. Mecanisme de generare a limbajelor: gramatici
4. Ierarhia lui Chomsky
5. Limbaje şi gramatici de tip 3 (regulate)

# Ierarhia lui Chomsky

1. **Gramatici de tip 0 (generale)**

   Nu exista restrictii asupra regulilor

# Ierarhia lui Chomsky

1. **Gramatici de tip 0 (generale)**

   Nu exista restrictii asupra regulilor

2. **Gramatici de tip 1 (dependente de context)**

   reguli de forma $pxq \to pyq$ unde $x \in N$, $y \neq \epsilon$, $p, q \in (N \cup T)^*$,

   $S \to \epsilon$, caz în care S nu apare în dreapta regulilor

# Ierarhia lui Chomsky

1. **Gramatici de tip 0 (generale)**

   Nu exista restrictii asupra regulilor

2. **Gramatici de tip 1 (dependente de context)**

   reguli de forma $pxq \to pyq$ unde $x \in N$, $y \neq \epsilon$, $p, q \in (N \cup T)^*$, $S \to \epsilon$, caz în care S nu apare în dreapta regulilor

3. **Gramatici de tip 2 (independente de context)**

   reguli de forma $A \to y$ unde $A \in N$ şi $y \in (N \cup T)^*$

# Ierarhia lui Chomsky

1. **Gramatici de tip 0 (generale)**

   Nu exista restrictii asupra regulilor

2. **Gramatici de tip 1 (dependente de context)**

   reguli de forma $pxq \to pyq$ unde $x \in N$, $y \neq \epsilon$, $p, q \in (N \cup T)^*$, $S \to \epsilon$, caz în care S nu apare în dreapta regulilor

3. **Gramatici de tip 2 (independente de context)**

   reguli de forma $A \to y$ unde $A \in N$ şi $y \in (N \cup T)^*$

4. **Gramatici de tip 3 (regulate)**

   reguli $A \to u$ sau $A \to uB$ unde $A, B \in N$ şi $u \in T^*$.

# Exemple

Tip 1: $pxq \to pyq$ unde $x \in N$, $y \neq \epsilon$, $p, q \in (N \cup T)^*$, $S \to \epsilon$

- $G = (N, T, S, P)$, $N = \{S, A, B\}$, $T = \{a, b, c\}$, $P$:

  (1) $S \to aaAc$

  (2) $aAc \to aAbBc$

  (3) $bB \to bBc$

  (4) $Bc \to Abc$

  (5) $A \to a$

  Gramatica tip 1

- $G = (N, T, S, P)$, $N = \{S, X\}$, $T = \{a, b, c\}$, $P$:

  (1) $S \to abc$

  (2) $S \to aSXc$

  (3) $cX \to Xc$ (nu este regulă de tip 1!, gramatica va fi de tip 0)

  (4) $bX \to bb$

# Exemple

Tip 2: $A \to y$ unde $A \in N$ şi $y \in (N \cup T)^*$

Tip3: $A \to u$ sau $A \to uB$ unde $A, B \in N$ şi $u \in T^*$.

- $G$:

  $(1) x \to axb$

  $(2) x \to \epsilon$

  (Gramatică tip 2)

- $G$:

  $(1) x \to ax$

  $(2) x \to bx$

  $(3) x \to \epsilon$

  (Gramatică tip 3)

# Exemple

- Fie
  $G = (\{E\}, \{a, +, -, (, )\}, E, \{E \to a, E \to (E + E), E \to (E - E)\})$
  .
  - Ce tip are gramatica G ?
  - Construiti derivari din E pentru cuvintele $(a + a)$ si $((a + a) - a)$
  - Cuvantul $(a + a - a)$ poate fi derivat din E?
  - Descrieti limbajul L(G)
- Fie $G = (\{A, B\}, \{a, b\}, A, \{A \to aA, A \to B, B \to bB, B \to \epsilon\})$
  - Ce tip are gramatica G ?
  - Descrieti limbajul L(G)

# Clasificarea limbajelor

- Un limbaj L este de tipul j daca exista o gramatica $G$ de tipul j astfel incat $L(G) = L$, unde $j \in \{0, 1, 2, 3\}$.
- Vom nota cu $\mathcal{L}_j$ clasa limbajelor de tipul j, unde $j \in \{0, 1, 2, 3\}$.
- Are loc: $\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$
- Incluziunile sunt stricte:
  - orice limbaj de tip $j + 1$ este si de tip $j \in \{0, 1, 2\}$
  - exista limbaje de tip $j$ care nu sunt de tip $j + 1$, $j \in \{0, 1, 2\}$

# Proprietăţi

- Fiecare din familiile $\mathcal{L}_j$ cu $0 \leq j \leq 3$ contine toate limbajele finite
- Fiecare din familiile $\mathcal{L}_j$ cu $0 \leq j \leq 3$ este inchisa la operatia de reuniune:

$$L_1, L_2 \in \mathcal{L}_j \Longrightarrow L_1 \cup L_2 \in \mathcal{L}_j,$$

$\forall j : 0 \leq j \leq 3$

# Notaţii alternative pentru gramatici de tip 2: BNF

## The syntax of C in Backus-Naur Form

```
<translation-unit> ::= {<external-declaration>}*

<external-declaration> ::= <function-definition>
                         | <declaration>

<function-definition> ::= {<declaration-specifier>}* <declarator> {<declaration>}* <compound-statemer

<declaration-specifier> ::= <storage-class-specifier>
                          | <type-specifier>
                          | <type-qualifier>

<storage-class-specifier> ::= auto
                            | register
                            | static
                            | extern
                            | typedef

<type-specifier> ::= void
                   | char
                   | short
                   | int
```

# gramatici DTD

- generează mulţimea documentelor XML cu o anumită structură (limbaj independent de context)

```
<!ELEMENT family (person)+>
<!ELEMENT person (name,addrres*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
```

# gramatici DTD

- Un ”cuvânt” din limbajul generat de gramtica DTD:

```
<?xml verson = "1.0">
<!DOCTYPE family SYSTEM "family.dtd">
<family>
    <person>
        <name>John</name>
        <address>First address</address>
        <address>Second address</address>
    </person>
   <person>
        <name>Sam</name>
   </person>
   <person>
        <name>Sarah</name>
        <address>First address</address>
    </person>
</family>
```

# XML Schema

- - rol similar gramaticilor DTD

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="family">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="address type = "xs:string" minOccurs = "0" maxOccures="unbounded">
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

# Limbaje Formale, Automate şi Compilatoare - Curs 1

1 Prezentare curs

2 Limbaje formale

3 Mecanisme de generare a limbajelor: gramatici

4 Ierarhia lui Chomsky

5 Limbaje şi gramatici de tip 3 (regulate)

# Gramatici de tip 3

- O gramatică $G = (N, T, S, P)$ este de tip 3 dacă regulile sale au forma: $A \to u$ sau $A \to uB$ unde $A, B \in N$ şi $u \in T^*$.
- Exemplu: $G = (\{D\}, \{0, 1, ..., 9\}, D, P)$

  Unde P este:

  $D \to 0D|1D|2D|\ldots|9D$

  $D \to 0|1|\ldots|9$

# Exemple

- Fie gramatica $G = (\{A, B\}, \{l, d\}, A, P)$ unde P este:
  $A \to lB,\ B \to lB|dB|\epsilon$ ($l$ = litera, $d$ = cifra)

# Exemple

- Fie gramatica $G = (\{A, B\}, \{l, d\}, A, P)$ unde P este:

  $A \to lB,\ B \to lB|dB|\epsilon$ ($l$ = litera, $d$ = cifra)

  $L(G)$: multimea identificatorilor
- Fie gramatica $G = (\{A, B\}, \{+, -, d\}, A, P)$ unde $P$ este:

  $A \to +dB| - dB|dB,\ B \to dB|\epsilon$ ($d$ = cifra)

# Exemple

- Fie gramatica $G = (\{A, B\}, \{I, d\}, A, P)$ unde P este:

  $A \to IB,\ B \to IB|dB|\epsilon$ ($I$ = litera, $d$ = cifra)

  $L(G)$: multimea identificatorilor

- Fie gramatica $G = (\{A, B\}, \{+, -, d\}, A, P)$ unde $P$ este:

  $A \to +dB| - dB|dB,\ B \to dB|\epsilon$ ($d$ = cifra)

  $L(G)$: multimea constantelor intregi

# Limbaje Formale, Automate şi Compilatoare

Curs 2

2019-20

# Curs 2

1. Forma normală pentru gramatici de tip 3
2. Proprietăţi de închidere pentru $\mathcal{L}_3$
3. Automate finite deterministe
4. Automate finite nedeterministe

# Forma normală

- O gramatică de tip 3 este in formă normală daca regulile sale sunt de forma $A \to a$ sau $A \to aB$, unde $a \in T$, si, eventual $S \to \epsilon$ ( caz in care S nu apare in dreapta regulilor).

- Pentru orice gramatica de tip 3 exista o gramatica echivalenta in forma normala.

# Forma normală

- Obtinerea gramaticii in forma normala echivalenta cu o gramatica de tip 3:
  - Se poate arata ca pot fi eliminate regulile de forma $A \to B$ (redenumiri) si cele de forma $A \to \epsilon$ (reguli de stergere), cu exceptia, eventual a regulii $S \to \epsilon$.
  - Orice regula de forma $A \to a_1 a_2 \dots a_n$ se inlocuieste cu $A \to a_1 B_1, B_1 \to a_2 B_2, \dots, B_{n-2} \to a_{n-1} B_{n-1}, B_{n-1} \to a_n$, $n > 1$, $B_1, \dots, B_{n-1}$ fiind neterminali noi.
  - Orice regula de forma $A \to a_1 a_2 \dots a_n B$ se inlocuieste cu $A \to a_1 B_1$, $B_1 \to a_2 B_2, \dots, B_{n-2} \to a_{n-1} B_{n-1}, B_{n-1} \to a_n B$, $n > 1$, $B_1, \dots, B_{n-1}$ fiind neterminali noi
  - Transformarile care se fac nu modifica limbajul generat de gramatica

# Curs 2

1 Forma normală pentru gramatici de tip 3

2 Proprietăţi de închidere pentru $\mathcal{L}_3$

3 Automate finite deterministe

4 Automate finite nedeterministe

Fie $L, L_1, L_2$ limbaje regulate: există gramaticile $G, G_1, G_2$ de tip 3 astfel ca $L = L(G), L_1 = L(G_1)$ şi $L_2 = L(G_2)$.

Atunci, următoarele limbaje sunt de asemenea regulate:

1. $L_1 \cup L_2$
2. $L_1 \cdot L_2$
3. $L^*$
4. $L_1 \cap L_2$
5. $L_1 \setminus L_2$

# Închiderea la reununiune

Fie $L, L_1, L_2$ limbaje de tip 3 (regulate).

Fie $G_1 = (N_1, T_1, S_1, P_1)$ si $G_2 = (N_2, T_2, S_2, P_2)$ gramatici de tip 3 cu $L_1 = L(G_1)$, $L_2 = L(G_2)$.

Presupunem $N_1 \cap N2 = \emptyset$

Închiderea la reuniune: se arata ca $L_1 \cup L_2 \in \mathcal{L}_3$:

Gramatica $G = (N_1 \cup N_2 \cup \{S\}, T_1 \cup T_2, S, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\})$ este de tip 3 si genereaza limbajul $L_1 \cup L_2$

# Închiderea la operaţia de produs

Fie $L_1, L_2$ limbaje de tip 3 (regulate).

Fie $G_1 = (N_1, T_1, S_1, P_1)$ si $G_2 = (N_2, T_2, S_2, P_2)$ gramatici de tip 3 cu $L_1 = L(G_1)$, $L_2 = L(G_2)$.

Presupunem $N_1 \cap N2 = \emptyset$

Gramatica $G = (N_1 \cup N_2, T_1 \cup T_2, S_1, P)$ unde $P$ consta din:

- regulile de forma $A \to uB$ din $P_1$ ($B \in N_1$)
- reguli $A \to uS_2$ pentru orice regula de forma $A \to u$ ($u \in T_1^*$) din $P_1$
- toate regulile din $P_2$

este de tip 3 si genereaza limbajul $L_1 L_2$.

# Exemplu

$L = \{uc^n, u \in \{a,b\}^+, n \geq 2\}$

$L = L_1 \cdot L_2$, unde: $L_1 = \{a,b\}^+$, $L_2 = \{c^n, n \geq 2\}$

$G1$ :

1. $S_1 \to aS_1$
2. $S_1 \to bS_1$
3. $S_1 \to a$
4. $S_1 \to b$

$G2$ :

1. $S_2 \to cS_2$
2. $S_2 \to cc$

$G = (\{S_1, S_2\}, \{a,b,c\}, S_1, P)$,

$P$ :

1. $S_1 \to aS_1$
2. $S_1 \to bS_1$
3. $S_1 \to aS_2$
4. $S_1 \to bS_2$
5. $S_2 \to cS_2$
6. $S_2 \to cc$

# Închiderea la operaţia de iteraţie

Fie $L$ limbaj de tip 3 (regulat).

Fie $G = (N, T, S, P)$ de tip 3 care genereaza $L$ ($L = L(G)$).

Presupunem ca simbolul de start S nu apare in partea dreapta a vreunei reguli.

Gramatica $G' = (N, T, S, P')$ unde $P'$ consta din

- reguli $A \to uB$ din P ($B \in N$)
- reguli $A \to uS$, pentru orice regula $A \to u$ din P ($u \in T^*$), diferită de $S \to \epsilon$
- regula $S \to \epsilon$

este de tip 3 si generează $L^*$

# Exemplu

$L = \{a^{n_1} b^{m_1} a^{n_2} b^{m_2} \ldots a^{n_k} b^{m_k}, n_i, m_i \geq 1 \forall i \in \{1, k\}, k \geq 0\}$

$L = \{a^n b^m, n \geq 1, m \geq 1\}^*$

$G$ :

1. $S \to x$
2. $x \to ax$
3. $x \to ay$
4. $y \to by$
5. $y \to b$

$G'$ :

1. $S \to x$
2. $x \to ax$
3. $x \to ay$
4. $y \to by$
5. $y \to bS$
6. $S \to \epsilon$

# Închiderea la intersecţie

Fie $L_1, L_2$ limbaje de tip 3 (regulate).

Fie $G_1 = (N_1, T_1, S_1, P_1)$ si $G_2 = (N_2, T_2, S_2, P_2)$ gramatici de tip 3, **în formă normală**, cu $L_1 = L(G_1)$, $L_2 = L(G_2)$.

Gramatica $G = (N_1 \times N_2, T_1 \cap T_2, (S_1, S_2), P)$, unde P constă din:

- $(S_1, S_2) \to \epsilon$, dacă $S_1 \to \epsilon \in P_1$ şi $S_2 \to \epsilon \in P_2$
- $(A_1, B_1) \to a(A_2, B_2)$, dacă $A_1 \to aA_2 \in P_1$ şi $B_1 \to aB_2 \in P_2$
- $(A_1, A_2) \to a$, dacă $A_1 \to a \in P_1$ şi $A_2 \to a \in P_2$

este de tip 3 şi generează limbajul $L_1 \cap L_2$

# Exemplu

$L(G1) = \{w \in \{0,1\}^*,$ w contine cel putin un simbol '0'$\}$,

$L(G2) = \{w \in \{0,1\}^*,$ w se termina cu '1'$\}$

$L(G) = \{w \in \{0,1\}^*,$ w contine cel putin un simbol '0' si se termina cu '1'$\}$

$G1$ :

1. $S_1 \to 1S_1$
2. $S_1 \to 0A$
3. $S_1 \to 0$
4. $A \to 1A$
5. $A \to 0A$
6. $A \to 1$
7. $A \to 0$

$G2$ :

1. $S_2 \to 0S_2$
2. $S_2 \to 1S_2$
3. $S_2 \to 1$

$G$

1. $(S_1, S2) \to 1(S_1, S_2)$
2. $(A, S_2) \to 1(A, S_2)$
3. $(S_1, S_2) \to 0(A, S_2)$
4. $(A, S_2) \to 0(A, S_2)$
5. $(A, S_2) \to 1$

# Exemplu

$L(G1) = \{w \in \{0,1\}^*,$ w contine cel putin un simbol '0'$\}$,

$L(G2) = \{w \in \{0,1\}^*,$ w se termina cu '1'$\}$

$L(G) = \{w \in \{0,1\}^*,$ w contine cel putin un simbol '0' si se termina cu '1'$\}$

$G1$ :

1. $S_1 \to 1S_1$
2. $S_1 \to 0A$
3. $S_1 \to 0$
4. $A \to 1A$
5. $A \to 0A$
6. $A \to 1$
7. $A \to 0$

$G2$ :

1. $S_2 \to 0S_2$
2. $S_2 \to 1S_2$
3. $S_2 \to 1$

$G$

1. $S \to 1S$
2. $X \to 1X$
3. $S \to 0X$
4. $X \to 0X$
5. $X \to 1$

# Curs 2

1 Forma normală pentru gramatici de tip 3

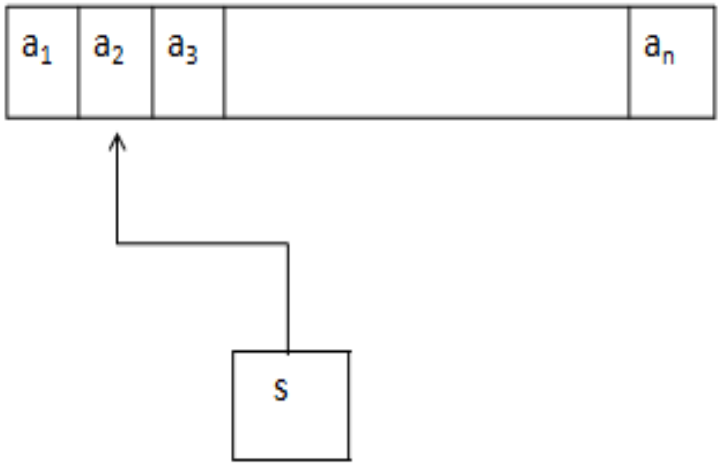
2 Proprietăţi de închidere pentru $\mathcal{L}_3$

3 Automate finite deterministe

4 Automate finite nedeterministe

# Automate finite

- Mecanism de recunoaştere (acceptare) pentru limbaje
- Limbaje de tip 3
- Mulţime finită de stări



# Automate finite

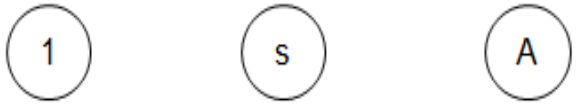
## Definiţie 1

*Un automat finit determinist este un 5-uplu* $A = (Q, \Sigma, \delta, q_0, F)$*, unde:*

- $Q$ *şi* $\Sigma$ *sunt mulţimi finite, nevide, numite mulţimea stărilor respectiv alfabetul de intrare*
- $q_0 \in Q$ *este starea iniţială*
- $F \subseteq Q$ *este mulţimea stărilor finale*
- $\delta$ *este o funcţie ,* $\delta : Q \times \Sigma \to Q$*, numită funcţia de tranziţie*

# Reprezentare prin diagrame(grafuri) de tranziţie

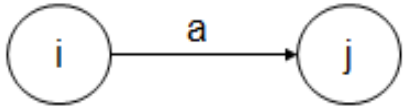
Stări:




Stare iniţială:

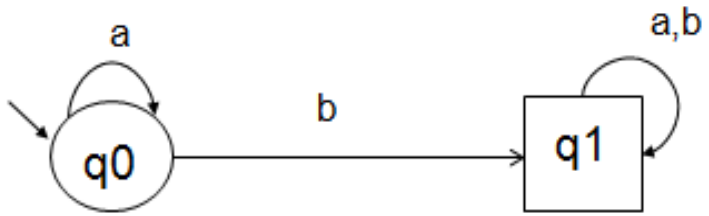

Stări finale:


Funcţia de tranziţie:



# Reprezentare prin matricea de tranziţie

$A = (\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_1\})$




| Intrare / Stare $\delta$ | a | b |
|---|---|---|
| q0 | q0 | q1 |
| q1 | q1 | q1 |

# Limbajul acceptat

- Extensia lui $\delta$ la cuvinte $\hat{\delta} : Q \times \Sigma^* \to Q$
  1. $\hat{\delta}(q, \epsilon) = q, \forall q \in Q$;
  2. $\hat{\delta}(q, ua) = \delta(\hat{\delta}(q, u), a)), \forall q \in Q, \forall u \in \Sigma^*, \forall a \in \Sigma$.

- **Observaţii:**
  - $\hat{\delta}(q, a) = \delta(q, a), \forall q \in Q, \forall a \in \Sigma$
  - $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v), \forall q \in Q, \forall u, v \in \Sigma^*$

# Limbajul acceptat

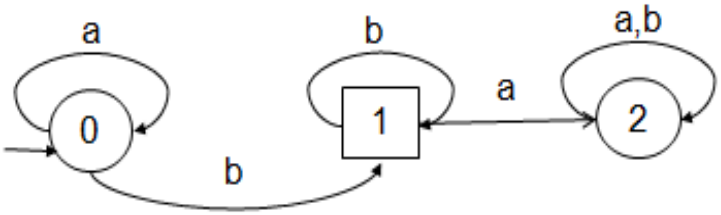
**Definiţie 2**

*Limbajul acceptat (recunoscut)* *de automatul* $A = (Q, \delta, \Sigma, q_0, F)$ *este mulţimea :*

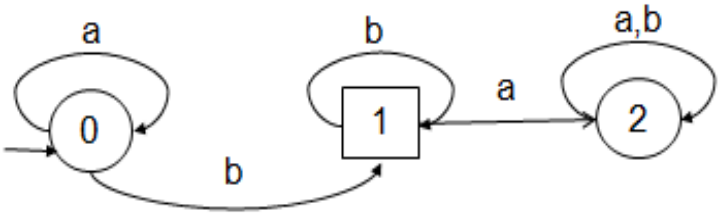
$$L(A) = \{w | w \in \Sigma^*, \hat{\delta}(q_0, w) \in F\}.$$

- Un cuvânt $w$ este recunoscut de un automat $A$ dacă, după citirea în întregime a cuvântului $w$, automatul (pornind din starea iniţială $q_0$ ) ajunge într-o stare finală.
- $\hat{\delta}(q, a) = \delta(q, a), \forall q \in Q, \forall a \in \Sigma$. Din acest motiv, $\hat{\delta}$ va fi notată de asemenea cu $\delta$.
- Două automate $A$ şi $A'$ sunt echivalente, dacă $L(A) = L(A')$

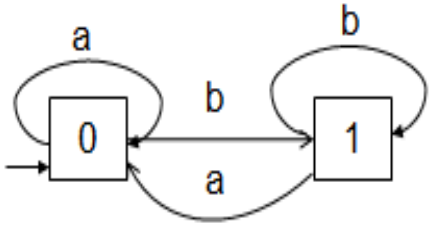
# Exemple



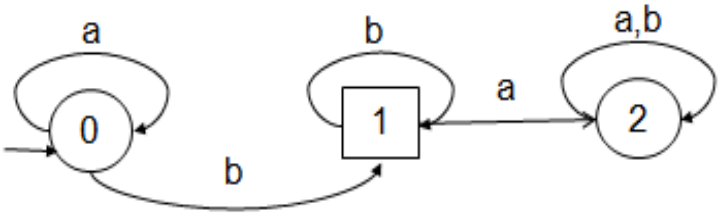
# Exemple



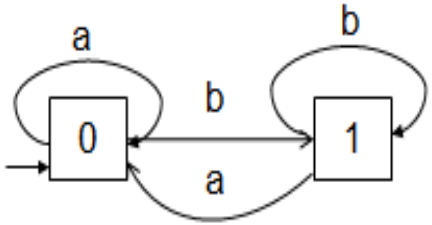

$$L(A) = \{a^n b^m | n \geq 0, m \geq 1\}$$



# Exemple




$$L(A) = \{a^n b^m | n \geq 0, m \geq 1\}$$




$$L(A) = \{a, b\}^*$$

# Exemple

Automate deterministe pentru:

- $L = \{w \in \{0,1\}^* | w$ conţine un număr par de $0\}$
- $L = \{w \in \{0,1\}^* | w$ se termina cu $11\}$

# Curs 2

1 Forma normală pentru gramatici de tip 3

2 Proprietăţi de închidere pentru $\mathcal{L}_3$

3 Automate finite deterministe

4 Automate finite nedeterministe

# Automate finite nedeterministe

**Definiţie 3**

*Un automat finit nedeterminist este un 5-uplu* $A = (Q, \Sigma, \delta, q_0, F)$, *unde:*

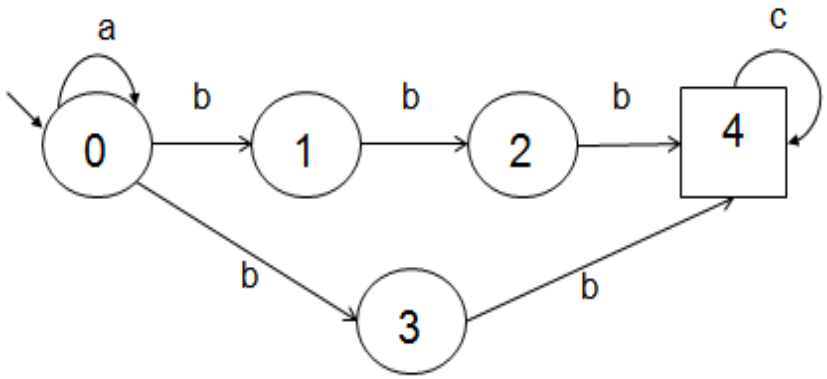
- $Q$, $\Sigma$, $q_0$ şi $F$ *sunt definite ca în cazul automatelor finite deterministe*
- $\delta$ *este o funcţie ,* $\delta : Q \times \Sigma \to 2^Q$, *numită funcţia de tranziţie*

- **Observaţie:**

  A este automat determinist, dacă

$$|\delta(q, a)| = 1, \forall q \in Q, \forall a \in \Sigma$$

# Exemple




| Intrare / Stare | a | b | c |
|---|---|---|---|
| 0 | {0} | {1,3} | Φ |
| 1 | Φ | {2} | Φ |
| 2 | Φ | {4} | Φ |
| 3 | Φ | {4} | Φ |
| 4 | Φ | Φ | {4} |

# Extensia lui $\delta$ la cuvinte

- Fie $S$ mulţime de stări. Notăm $\delta(S, a) = \bigcup_{q \in S} \delta(q, a)$.
- Extensia lui $\delta$ la cuvinte $\hat{\delta} : Q \times \Sigma^* \to 2^Q$
  1. $\hat{\delta}(q, \epsilon) = \{q\}, \forall q \in Q$;
  2. $\hat{\delta}(q, ua) = \delta(\hat{\delta}(q, u), a), \forall q \in Q, \forall u \in \Sigma^*, \forall a \in \Sigma$.

# Extensia lui $\delta$ la cuvinte

- Fie $S$ mulţime de stări. Notăm $\delta(S, a) = \bigcup_{q \in S} \delta(q, a)$.
- Extensia lui $\delta$ la cuvinte $\hat{\delta} : Q \times \Sigma^* \to 2^Q$
  1. $\hat{\delta}(q, \epsilon) = \{q\}, \forall q \in Q$;
  2. $\hat{\delta}(q, ua) = \delta(\hat{\delta}(q, u), a), \forall q \in Q, \forall u \in \Sigma^*, \forall a \in \Sigma$.

- **Observaţii:**
  - $\hat{\delta}(q, a) = \delta(q, a), \forall q \in Q, \forall a \in \Sigma$
  - $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v), \forall q \in Q, \forall u, v \in \Sigma^*$.

# Limbajul acceptat

**Definiţie 4**

*Limbajul acceptat (recunoscut)* *de automatul finit nedeterminist* $A = (Q, \Sigma, \delta, q_0, F)$ *este mulţimea :*

$$L(A) = \{w | w \in \Sigma^*, \hat{\delta}(q_0, w) \cap F \neq \emptyset\}.$$

- Un cuvânt $w$ este recunoscut de un automat A dacă, după citirea în întregime a cuvântului $w$, automatul (pornind din starea iniţială $q_0$ ) poate să ajungă într-o stare finală.

# Determinism = Nedeterminism

**Teorema 1**

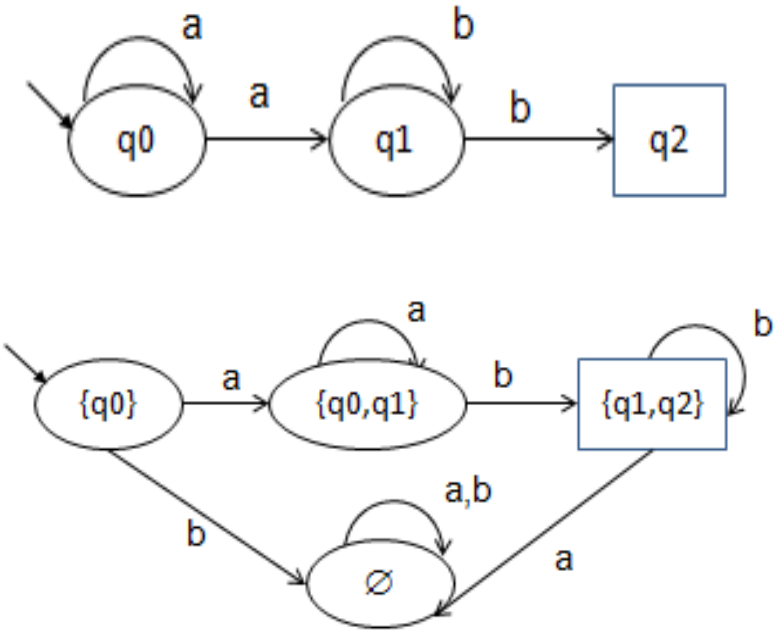
*Pentru orice automat nedeterminist A, există unul determinist A′ echivalent.*

Dacă $A = (Q, \Sigma, \delta, q_0, F)$ atunci $A' = (2^Q, \Sigma, \delta', Q_0, F')$ unde:

- $Q_0 = \{q_0\}$
- $F' = \{S | S \subseteq Q, S \cap F \neq \emptyset\}$
- $\delta'(S, a) = \bigcup_{s \in S} \delta(s, a)$

- Pentru aplicaţii se construiesc doar stările accesibile din starea iniţială

# Exemplu



# Determinism = Nedeterminism

Dacă $A = (Q, \Sigma, \delta, q_0, F)$ atunci $A' = (2^Q, \Sigma, \delta', Q_0, F')$ unde:

- $Q_0 = \{q_0\}$
- $F' = \{S | S \subseteq Q, S \cap F \neq \emptyset\}$
- $\delta'(S, a) = \bigcup_{s \in S} \delta(s, a)$

Au loc:

# Determinism = Nedeterminism

Dacă $A = (Q, \Sigma, \delta, q_0, F)$ atunci $A' = (2^Q, \Sigma, \delta', Q_0, F')$ unde:

- $Q_0 = \{q_0\}$
- $F' = \{S | S \subseteq Q, S \cap F \neq \emptyset\}$
- $\delta'(S, a) = \bigcup_{s \in S} \delta(s, a)$

Au loc:

- $\delta'(S, w) = \bigcup_{s \in S} \delta(s, w), \forall w \in \Sigma^*$

# Determinism = Nedeterminism

Dacă $A = (Q, \Sigma, \delta, q_0, F)$ atunci $A' = (2^Q, \Sigma, \delta', Q_0, F')$ unde:

- $Q_0 = \{q_0\}$
- $F' = \{S | S \subseteq Q, S \cap F \neq \emptyset\}$
- $\delta'(S, a) = \bigcup_{s \in S} \delta(s, a)$

Au loc:

- $\delta'(S, w) = \bigcup_{s \in S} \delta(s, w), \forall w \in \Sigma^*$
- $\delta'(Q_0, w) = \delta'(\{q_0\}, w) = \bigcup_{s \in \{q_0\}} \delta(s, w) = \delta(q0, w)$

# Determinism = Nedeterminism

Dacă $A = (Q, \Sigma, \delta, q_0, F)$ atunci $A' = (2^Q, \Sigma, \delta', Q_0, F')$ unde:

- $Q_0 = \{q_0\}$
- $F' = \{S | S \subseteq Q, S \cap F \neq \emptyset\}$
- $\delta'(S, a) = \bigcup_{s \in S} \delta(s, a)$

Au loc:

- $\delta'(S, w) = \bigcup_{s \in S} \delta(s, w), \forall w \in \Sigma^*$
- $\delta'(Q_0, w) = \delta'(\{q_0\}, w) = \bigcup_{s \in \{q_0\}} \delta(s, w) = \delta(q0, w)$
- $w \in L(A') \Leftrightarrow$
  $\delta'(Q_0, w) \in F' \Leftrightarrow \delta'(Q_0, w) \cap F \neq \emptyset \Leftrightarrow \delta(q_0, w) \cap F \neq \emptyset$
  $\Leftrightarrow w \in L(A)$

# Limbaje Formale, Automate şi Compilatoare

Curs 3

2019-20

# Structura cursului

1. Automate finite cu $\epsilon$-tranziţii

2. Automatul determinist minimal

# Curs 3

1 Automate finite cu $\epsilon$-tranziţii

2 Automatul determinist minimal

# Automate finite cu $\epsilon$-tranziţii

## Definiţie 1

*Un automat finit cu $\epsilon$-tranziţii este un 5-uplu $A = (Q, \Sigma, \delta, q_0, F)$, unde:*

- *$Q$, $\Sigma$, $q_0$ şi $F$ sunt definite ca în cazul automatelor finite deterministe*
- *$\delta$ este o funcţie , $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$, numită funcţia de tranziţie*

**Observaţie:**

- A este automat nedeterminist, dacă $\delta(q, \epsilon) = \emptyset, \forall q \in Q$
- A este automat determinist, dacă, în plus:

$$|\delta(q, a)| = 1, \forall q \in Q, \forall a \in \Sigma$$

# Extensia lui $\delta$ la cuvinte

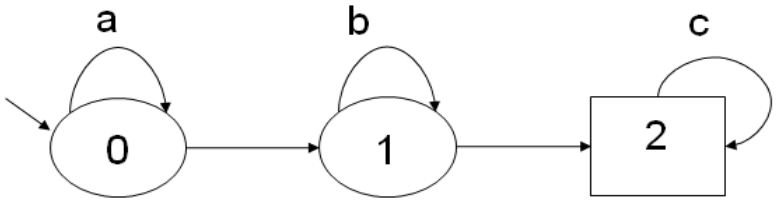
- $Cl(q)$-mulţimea stărilor la care se poate ajunge prin $\epsilon$-tranziţii:
  - $q \in Cl(q)$
  - $q' \in Cl(q) \Rightarrow \delta(q', \epsilon) \subseteq Cl(q)$
- Dacă $S \subseteq Q$, atunci notăm:

$$Cl(S) = \bigcup_{q \in S} Cl(q)$$

- Extensia lui $\delta$ la cuvinte: $\hat{\delta} : Q \times \Sigma^* \to 2^Q$
  1. $\hat{\delta}(q, \epsilon) = Cl(q), \forall q \in Q$;
  2. $\hat{\delta}(q, ua) = Cl(\delta(\hat{\delta}(q, u), a))), \forall q \in Q, \forall u \in \Sigma^*, \forall a \in \Sigma$.

# Extensia lui $\delta$ la cuvinte

- $\hat{\delta}(q, a) = Cl(\delta(Cl(q), a)), \forall q \in Q, \forall a \in \Sigma$




- În cazul automatelor cu $\epsilon$ - tranziţii vom păstra notaţia $\hat{\delta}$ pentru extensie pentru că, în general, $\hat{\delta}(q, \epsilon) \neq \delta(q, \epsilon)$ şi $\hat{\delta}(q, a) \neq \delta(q, a), a \in \Sigma$.
- $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v), \forall q \in Q, \forall u, v \in \Sigma^*$

# Limbajul acceptat

**Definiţie 2**

*Limbajul acceptat (recunoscut) de automatul cu $\epsilon$-tranziţii $A = (Q, \Sigma, \delta, q_0, F)$ este mulţimea :*

$$L(A) = \{w | w \in \Sigma^*, \hat{\delta}(q_0, w) \cap F \neq \emptyset\}.$$

- Un cuvânt $w$ este recunoscut de un automat A dacă, după citirea în întregime a cuvântului $w$, automatul (pornind din starea iniţială $q_0$ ) poate să ajungă într-o stare finală.

# Automatul determinist echivalent

**Teorema 1**

*Pentru orice automat $A$ cu $\epsilon$ - tranziţii există un automat $A'$ determinist echivalent cu $A$*

Dacă $A = (Q, \Sigma, \delta, q_0, F)$ atunci $A' = (Q', \Sigma, \delta', q_0', F')$ unde:

- $Q' = 2^Q$
- $q_0' = Cl(q_0)$
- $\delta'(S, a) = Cl(\bigcup_{s \in S} \delta(s, a)) \quad S \in Q', a \in \Sigma$
- $S \in F' \Leftrightarrow S \cap F \neq \emptyset$

# Automatul determinist echivalent

**Teorema 1**

*Pentru orice automat $A$ cu $\epsilon$ - tranziţii există un automat $A'$ determinist echivalent cu $A$*

Dacă $A = (Q, \Sigma, \delta, q_0, F)$ atunci $A' = (Q', \Sigma, \delta', q_0', F')$ unde:

- $Q' = 2^Q$
- $q_0' = Cl(q_0)$
- $\delta'(S, a) = Cl(\bigcup_{s \in S} \delta(s, a)) \quad S \in Q', a \in \Sigma$
- $S \in F' \Leftrightarrow S \cap F \neq \emptyset$

Au loc:

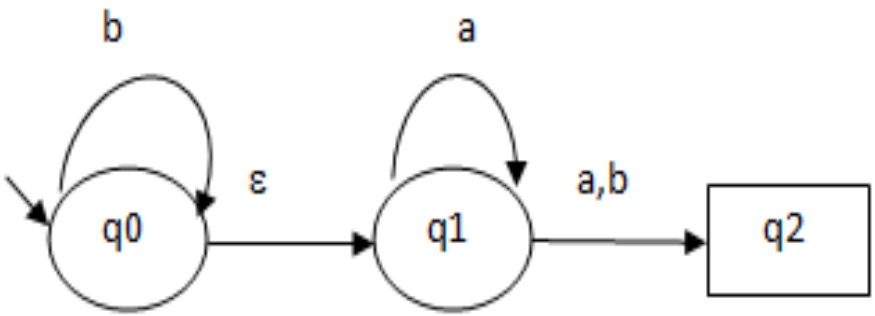
- $\delta'(q_0', w) = \hat{\delta}(q_0, w), \forall w \in \Sigma^*$
- $L(A') = L(A)$

# Automatul determinist echivalent - algoritm

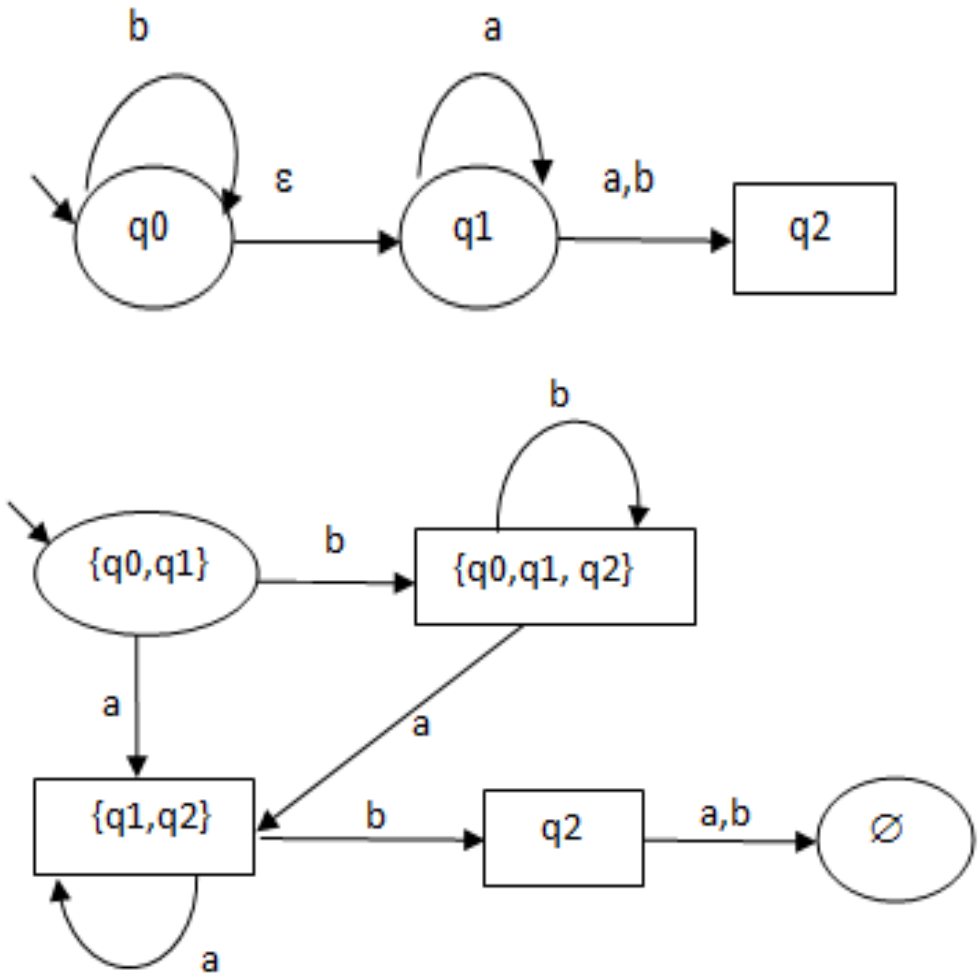
- Intrare: Automatul A (cu $\epsilon$ - tranziţii) ; $Cl(S)$
- Ieşire: Automatul determinist $A' = (Q', \Sigma, \delta', q_0', F')$, echivalent cu A.

$q_0' = Cl(\{q_0\})$; $Q' = \{q_0'\}$ ;
$marcat(q_0') = false$; $F' = \emptyset$ ;
if $(q_0' \cap F \neq \emptyset)$ then $F' = F' \cup \{q_0'\}$ ;
while $(\exists S \in Q' \&\& !marcat(S))$ {
    for $(a \in \Sigma)$ {
        $S' = Cl(\bigcup_{s \in S} \delta(s, a))$;
        $\delta'(S, a) = S'$ ;
        if $(S' \notin Q')$ {
            $Q' = Q' \cup \{S'\}$;
            $marcat(S') = false$;
            if $(S' \cap F \neq \emptyset)$ then $F' = F' \cup \{S'\}$ ;
        }
    }
    $marcat(S) = true$;
}

# Exemplu



# Exemplu



# Curs 3

1 Automate finite cu $\epsilon$-tranziţii

2 Automatul determinist minimal

# Stări accesibile

- Fie $A = (Q, \Sigma, \delta, q_0, F)$ automat finit determinist

Starea $q$ este accesibilă în $A$ dacă există un cuvânt $w \in \Sigma^*$ astfel încât $q = \delta(q_0, w)$.

# Stări inseparabile

Fie $A = (Q, \Sigma, \delta, q_0, F)$ un automat finit determinist.

## Definiţie 3

*Stările $q_1$ şi $q_2$ sunt inseparabile în raport cu $F$, (notat $q_1 \rho q_2$) ddacă*

$$\forall w \in \Sigma^* : \delta(q_1, w) \in F \Leftrightarrow \delta(q_2, w) \in F$$

# Stări inseparabile

Fie $A = (Q, \Sigma, \delta, q_0, F)$ un automat finit determinist.

**Definiţie 3**

*Stările $q_1$ şi $q_2$ sunt inseparabile în raport cu $F$, (notat $q_1 \rho q_2$) ddacă*

$$\forall w \in \Sigma^* : \delta(q_1, w) \in F \Leftrightarrow \delta(q_2, w) \in F$$

- Dacă există $w \in \Sigma^*$ cu $\delta(q_1, w) \in F$ şi $\delta(q_2, w) \notin F$ (sau invers), stările $q_1$ şi $q_2$ sunt separabile (de către $w$), şi notăm $q_1$ $\underline{sep}$ $q_2$
- $q_1$ $\underline{sep}$ $q2 \Leftrightarrow \neg q_1 \rho q_2$.

# Stări inseparabile

Fie $A = (Q, \Sigma, \delta, q_0, F)$ un automat finit determinist.

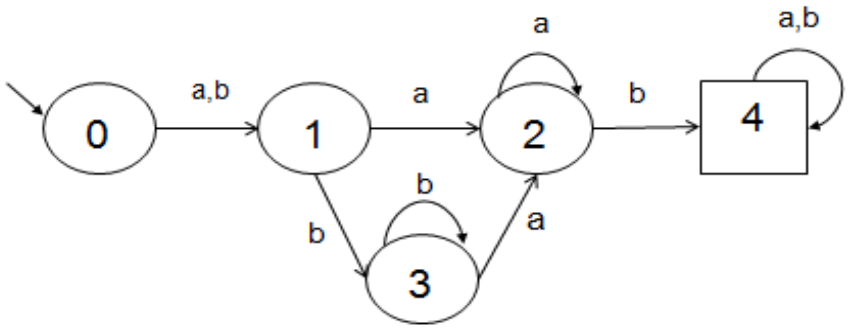
**Definiţie 3**

*Stările $q_1$ şi $q_2$ sunt inseparabile în raport cu $F$, (notat $q_1 \rho q_2$) ddacă*

$$\forall w \in \Sigma^* : \delta(q_1, w) \in F \Leftrightarrow \delta(q_2, w) \in F$$

- Dacă există $w \in \Sigma^*$ cu $\delta(q_1, w) \in F$ şi $\delta(q_2, w) \notin F$ (sau invers), stările $q_1$ şi $q_2$ sunt separabile (de către $w$), şi notăm $q_1$ $\underline{sep}$ $q_2$
- $q_1$ $\underline{sep}$ $q2 \Leftrightarrow \neg q_1 \rho q_2$.
- **Observaţie:** dacă $q_1 \in F$ şi $q_2 \notin F$, atunci $q_1$ $\underline{sep}$ $q_2$

# Exemplu



# Automat minimal

**Observaţii:**

- Relatia $\rho$ este relaţie de echivalenţă.
- $\exists a \in \Sigma : \delta(p, a) \underline{sep}\ \delta(q, a) \Longrightarrow p \underline{sep}\ q$.

# Automat minimal

**Observaţii:**

- Relatia $\rho$ este relaţie de echivalenţă.
- $\exists a \in \Sigma : \delta(p, a)$ *sep* $\delta(q, a) \Longrightarrow p$ *sep* $q$.

**Teorema 2**

*Fie A un automat determinist cu toate stările accesibile. Daca toate stările din A sunt separabile în raport cu F, atunci nu există un alt automat $A'$ cu număr mai mic de stări şi $L(A) = L(A')$.*

# Automatul minimal

Fie $A = (Q, \Sigma, \delta, q_0, F)$ un automat finit determinist si relaţia $\rho$.

- Dacă $\forall q_1, q_2 \in Q : q_1 \ \underline{sep} \ q_2$, atunci A este minimal.
- Altfel, automatul minimal:

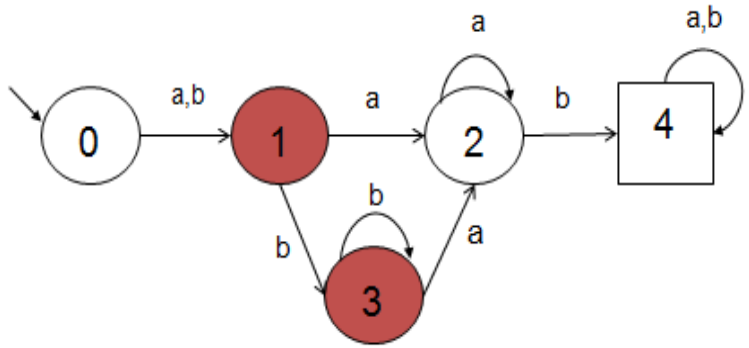
$$A_\rho = (Q/\rho, \Sigma, \delta_\rho, [q_0], F/\rho)$$

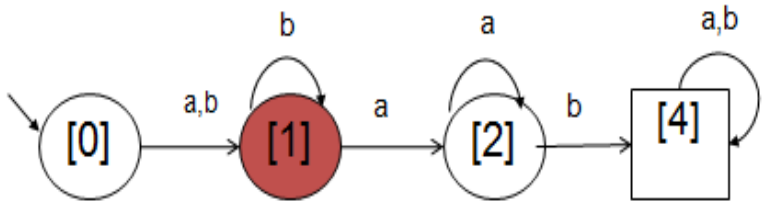
  - $Q/\rho$ - clasele de echivalenţă ale relaţiei $\rho$:

$$Q/\rho = \{[q] | q \in Q\}$$

  - $\delta_\rho([q], a) = [\delta(q, a)]$
  - $[q_0]$ clasa de echivalenţă în care se află starea $q_0$
  - $F/\rho = \{[q] | q \in F\}$

# Exemplu



# Automatul minimal

Fie automatul minimal: $A_\rho = (Q/\rho, \Sigma, \delta_\rho, [q_0], F/\rho)$

- $Q/\rho$ - clasele de echivalenţă ale relaţiei $\rho$:
- $\delta_\rho([q], a) = [\delta(q, a)]$
- $[q_0]$ clasa de echivalenţă în care se află starea $q_0$
- $F/\rho = \{[q] | q \in F\}$

**Teorema 3**

*Fie automatul determinist A, cu toate stările accesibile. Automatul $A_\rho$ construit ca mai sus este automatul cu număr minim de stări care acceptă limbajul $L(A)$.*

# Algoritm pentru determinarea relaţiei $\rho$

- Fie $A = (Q, \Sigma, \delta, q_0, F)$, $Q = \{q_0, q_1, \ldots, q_n\}$
- Tablou $separabil[q_i, q_j]$:
  - $separabil[q_i, q_j] = 1$ ddacă $q_i$ $\underline{sep}$ $q_j$ ($separabil[q_i, q_j] = 0$ ddacă $q_i \rho q_j$)
  - iniţial $separabil[q_i, q_j] = 1$ ddacă $q_i \in F, q_j \notin F$ (sau invers)
  - Pentru stările $q_i, q_j$, dacă există $a \in \Sigma$ cu $\delta(q_i, a)$, $\delta(q_j, a)$ separabile, atunci $q_i$, $q_j$ vor fi separarbile, adică :
    dacă $separabil[q_i, q_j] = 0$ şi există $a \in \Sigma$ cu $separabil[\delta(q_i, a), \delta(q_j, a)] = 1$, atunci $separabil[q_i, q_j] = 1$

# Algoritm pentru determinarea relaţiei $\rho$

- *lista*$[p, r]$ : $(p \neq r)$
  - definită pentru perechi de stări cu *separabil*$[p, r] = 0$

# Algoritm pentru determinarea relaţiei $\rho$

- $lista[p, r] : (p \neq r)$
  - definită pentru perechi de stări cu $separabil[p, r] = 0$
  - $lista[p, r] = \{(q_i, q_j) | separabil[q_i, q_j] = 0 \ \wedge \ exista\ a \in \Sigma : p = \delta(q_i, a),\ r = \delta(q_j, a),\ (q_i, q_j) \neq (p, r)\}$

# Algoritm pentru determinarea relaţiei $\rho$

- Se iniţializează tabloul *separabil* ($separabil[q_i, q_j] = 1$, dacă $q_i \in F$, $q_j \notin F$ sau invers)
- Pentru orice $q_i$, $q_j$ $(0 \leq i < j \leq n)$ cu $separabil[q_i, q_j] = 0$ :

# Algoritm pentru determinarea relaţiei $\rho$

- Se iniţializează tabloul *separabil* (*separabil*$[q_i, q_j] = 1$, dacă $q_i \in F$, $q_j \notin F$ sau invers)
- Pentru orice $q_i$, $q_j$ $(0 \leq i < j \leq n)$ cu *separabil*$[q_i, q_j] = 0$ :
  - Dacă există $a \in \Sigma$ cu *separabil*$[\delta(q_i, a), \delta(q_j, a)] = 1$, atunci:
    - *separabil*$[q_i, q_j] = 1$
    - trebuie modificat tabloul *separabil* pentru toate perechile de stări a căror separabilitate depinde de $q_i$, $q_j$ (perechile de stări din *lista*$[q_i, q_j]$)

# Algoritm pentru determinarea relaţiei $\rho$

- Se iniţializează tabloul *separabil* ($separabil[q_i, q_j] = 1$, dacă $q_i \in F$, $q_j \notin F$ sau invers)
- Pentru orice $q_i$, $q_j$ ($0 \leq i < j \leq n$) cu $separabil[q_i, q_j] = 0$ :
  - Dacă există $a \in \Sigma$ cu $separabil[\delta(q_i, a), \delta(q_j, a)] = 1$, atunci:
    - $separabil[q_i, q_j] = 1$
    - trebuie modificat tabloul *separabil* pentru toate perechile de stări a căror separabilitate depinde de $q_i$, $q_j$ (perechile de stări din $lista[q_i, q_j]$)
  - Altfel (pentru orice $a \in \Sigma$ are loc $separabil[\delta(q_i, a), \delta(q_j, a)] = 0$):
    - pentru orice $a \in \Sigma$ cu $\delta(q_i, a) \neq \delta(q_j, a)$ adaugă $(q_i, q_j)$ la $lista[\delta(q_i, a), \delta(q_j, a)]$

# Algoritm pentru determinarea relaţiei $\rho$

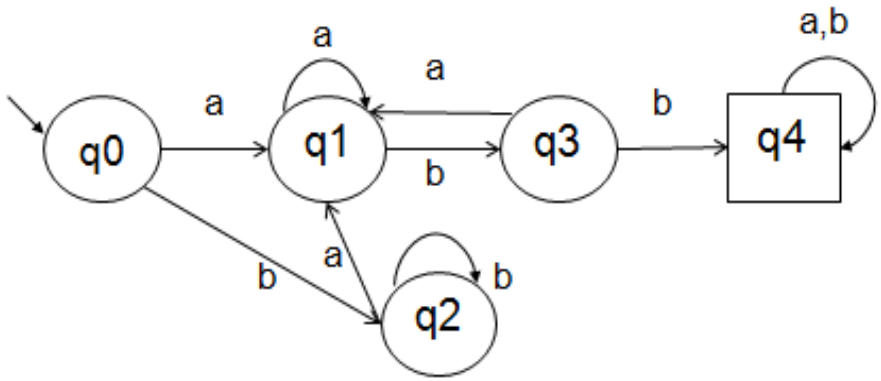
```
//initializarea tablourilor,
se marchează perechile F × (Q − F) si (Q − F) × F
1.for (i=0; i<=n-1; i++)
2.     for (j=i+1,j<=n; j++) {
3.          lista[qi,qj]=∅;
4.          if ((qi ∈ F && qj ∉ F) || (qi ∉ F && qj ∈ F))
5.             separabil[qi,qj]=1;
6.          else
7.             separabil[qi,qj]=0;
8.     }
```

```
9.for (i=0; i<=n-1; i++)
10.    for (j=i+1,j<=n; j++) {
       //se selecteaza doar starile inseparabile
11.        if (separabil[qi,qj]==0) {
              //daca exista a astfel incat δ(qi,a) sep δ(qj,a)
              //inseamna ca qi si qj sunt separabile
12.           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
13.              update_separabil(qi,qj);
14.           }
15.           else {
16.                for (a ∈ Σ : δ(qi,a) ≠ δ(qj,a)&&  (qi,qj) ≠ (δ(qi,a),δ(qj,a)))
17.                    adauga (qi,qj) la lista[δ(qi,a),δ(qj,a)]
18.           }
19.        }
20.    }
```

# Algoritm pentru determinarea relaţiei $\rho$

```
// 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);
        }
}
```

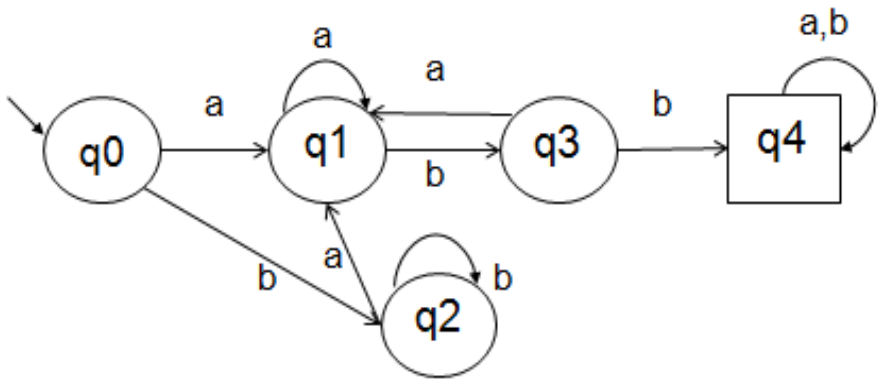
# Exemplu




| q1 | q2 | q3 | q4 | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | q0 |
| | 0 | 0 | 1 | q1 |
| | | 0 | 1 | q2 |
| | | | 1 | q3 |

| δ | a | b |
|---|---|---|
| q0 | q1 | q2 |
| q1 | q1 | q3 |
| q2 | q1 | q2 |
| q3 | q1 | q4 |
| q4 | q4 | q4 |

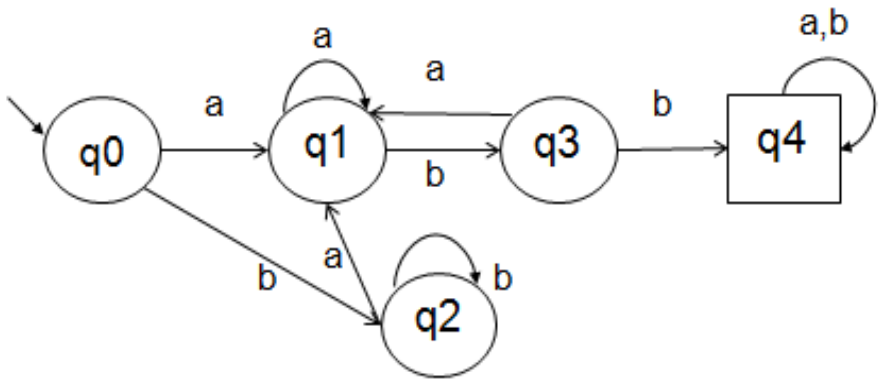
# Exemplu




| q1 | q2 | q3 | q4 | |
|---|---|---|---|---|
| 0 | 0 | (0) 1 | 1 | q0 |
| | 0 | 0 | 1 | q1 |
| | | 0 (q0,q1) | 1 | q2 |
| | | | 1 | q3 |

| δ | a | b |
|---|---|---|
| q0 | q1 | q2 |
| q1 | q1 | q3 |
| q2 | q1 | q2 |
| q3 | q1 | q4 |
| q4 | q4 | q4 |

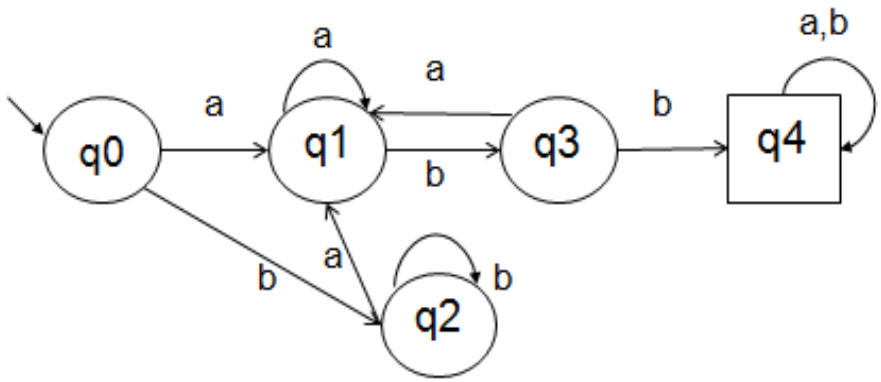
# Exemplu




| q1 | q2 | q3 | q4 | |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | q0 |
| | 0 | (0) 1 | 1 | q1 |
| | | 0<br>(q1,q2)<br>(q0,q1) | 1 | q2 |
| | | | 1 | q3 |

| δ | a | b |
|---|---|---|
| q0 | q1 | q2 |
| q1 | q1 | q3 |
| q2 | q1 | q2 |
| q3 | q1 | q4 |
| q4 | q4 | q4 |

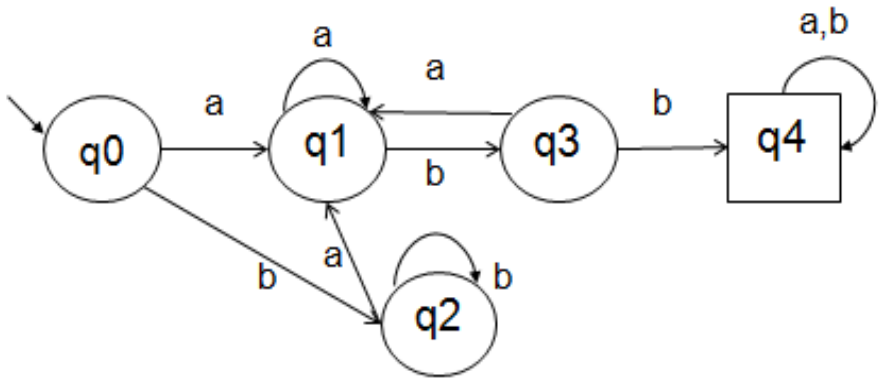
# Exemplu




| q1 | q2 | q3 | q4 | |
|---|---|---|---|---|
| (0) 1 | 0 | 1 | 1 | q0 |
| | (0) 1 | 1 | 1 | q1 |
| | | (0)1<br>(q1,q2)<br>(q0,q1) | 1 | q2 |
| | | | 1 | q3 |

| δ | a | b |
|---|---|---|
| q0 | q1 | q2 |
| q1 | q1 | q3 |
| q2 | q1 | q2 |
| q3 | q1 | q4 |
| q4 | q4 | q4 |

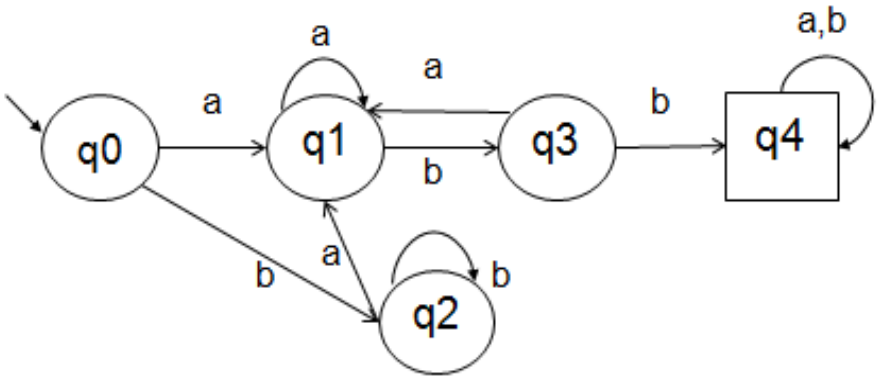
# Exemplu

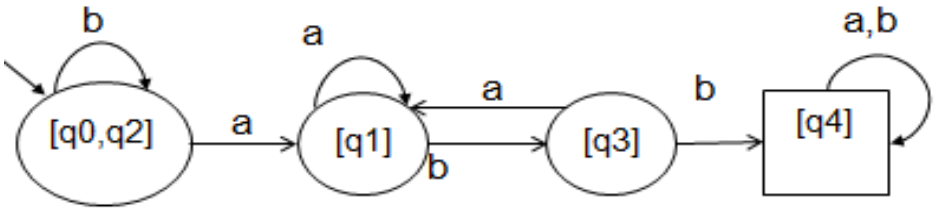



| q1 | q2 | q3 | q4 | |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | q0 |
| | 1 | 1 | 1 | q1 |
| | | 1<br>(q1,q2)<br>(q0,q1) | 1 | q2 |
| | | | 1 | q3 |

| δ | a | b |
|---|---|---|
| q0 | q1 | q2 |
| q1 | q1 | q3 |
| q2 | q1 | q2 |
| q3 | q1 | q4 |
| q4 | q4 | q4 |

# Exemplu



# Limbaje Formale, Automate şi Compilatoare

Curs 4

2019-20

# Curs 4

1. Corectitudinea algoritmului pentru determinarea relaţiei $\rho$
2. Gramatici de tip 3 şi automate finite
3. Proprietăţi de închidere pentru clasa limbajelor de tip 3
4. Expresii regulate
5. Automatul echivalent cu o expresie regulată
   - Algoritm

# Curs 4

1. Corectitudinea algoritmului pentru determinarea relaţiei $\rho$
2. Gramatici de tip 3 şi automate finite
3. Proprietăţi de închidere pentru clasa limbajelor de tip 3
4. Expresii regulate
5. Automatul echivalent cu o expresie regulată
   - Algoritm

# Algoritm pentru determinarea relaţiei $\rho$

```
//initializarea tablourilor,
se marchează perechile F × (Q − F) si (Q − F) × F
1.for (i=0; i<=n-1; i++)
2.     for (j=i+1,j<=n; j++) {
3.          lista[qi,qj]=∅;
4.          if ((qi ∈ F && qj ∉ F) || (qi ∉ F && qj ∈ F))
5.             separabil[qi,qj]=1;
6.          else
7.             separabil[qi,qj]=0;
8.     }
```

```
9.for (i=0; i<=n-1; i++)
10.    for (j=i+1,j<=n; j++) {
       //se selecteaza doar starile inseparabile
11.        if (separabil[qi,qj]==0) {
              //daca exista a astfel incat δ(qi,a) sep δ(qj,a)
              //inseamna ca qi si qj sunt separabile
12.           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
13.              update_separabil(qi,qj);
14.           }
15.           else {
16.                for (a ∈ Σ : δ(qi,a) ≠ δ(qj,a)&&  (qi,qj) ≠ (δ(qi,a),δ(qj,a)))
17.                    adauga (qi,qj) la lista[δ(qi,a),δ(qj,a)]
18.           }
19.        }
20.    }
```

# Algoritm pentru determinarea relaţiei $\rho$

```
// 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

1. Corectitudinea algoritmului pentru determinarea relaţiei $\rho$
2. Gramatici de tip 3 şi automate finite
3. Proprietăţi de închidere pentru clasa limbajelor de tip 3
4. Expresii regulate
5. Automatul echivalent cu o expresie regulată
   - Algoritm

# De la gramatici de tip 3 la automate finite

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

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

# De la automate finite la gramatici de tip 3

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

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

# Curs 4

1. Corectitudinea algoritmului pentru determinarea relaţiei $\rho$
2. Gramatici de tip 3 şi automate finite
3. Proprietăţi de închidere pentru clasa limbajelor de tip 3
4. Expresii regulate
5. Automatul echivalent cu o expresie regulată
   - Algoritm

# 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 $\overline{L} = \overline{L(A)}$:

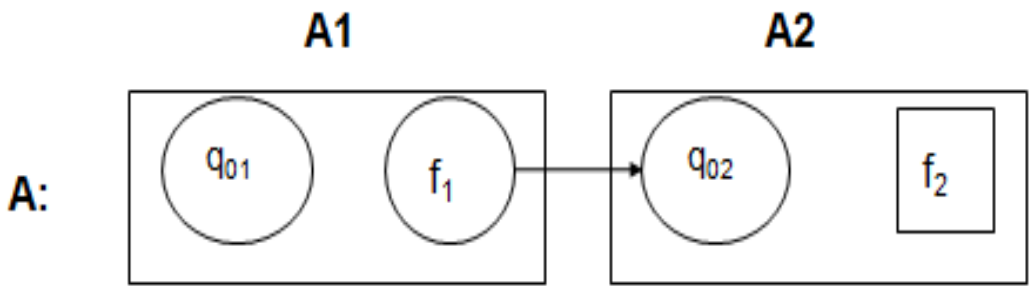
$$A' = (Q, \Sigma, \delta, q_0, Q \setminus F)$$

- 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 $\epsilon$-tranziţii) care recunoaşte $L_1 \cdot L_2$:

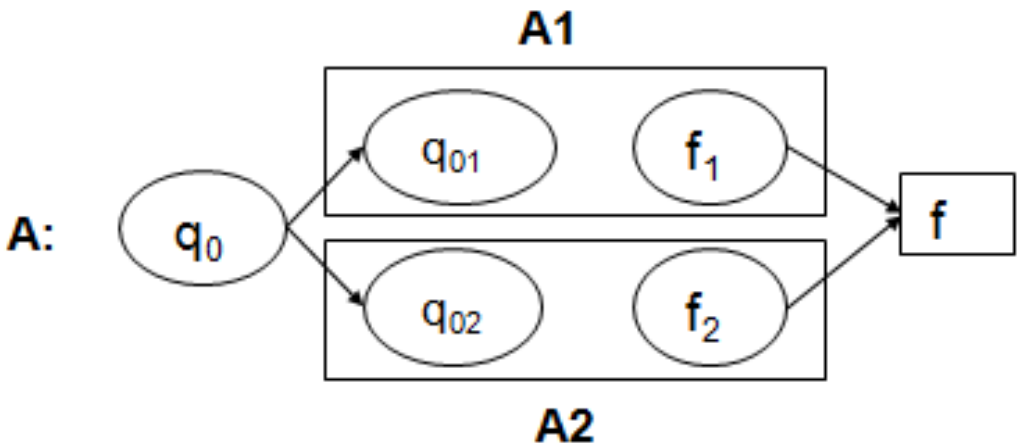
$$A = (Q_1 \cup Q_2, \Sigma, \delta, q_{01}, \{f_2\})$$



# 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 $\epsilon$-tranziţii) care recunoaşte $L_1 \cup L_2$:

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

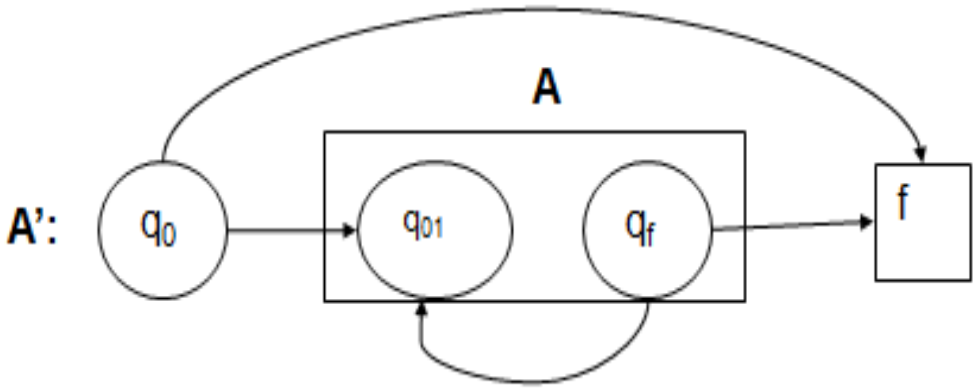


# Închiderea la iteraţie

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

  Automatul $A$ (cu $\epsilon$-tranziţii) care recunoaşte $L^*$ $(= L(A)^*)$:

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



# Curs 4

1. Corectitudinea algoritmului pentru determinarea relaţiei $\rho$
2. Gramatici de tip 3 şi automate finite
3. Proprietăţi de închidere pentru clasa limbajelor de tip 3
4. Expresii regulate
5. Automatul echivalent cu o expresie regulată
   - Algoritm

# Expresii regulate - definiţie

- Reprezentarea limbajelor de tip 3 prin expresii algebrice

## Definiţie 1

*Dacă $\Sigma$ este un alfabet atunci o expresie regulată peste $\Sigma$ se defineşte 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* $L(E_1) \cup L(E2)$
  - $(E_1 \cdot E_2)$ *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 $\Sigma$ este un alfabet atunci o expresie regulată peste $\Sigma$ se defineşte 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* $L(E_1) \cup L(E2)$
  - $(E_1 \cdot E_2)$ *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**

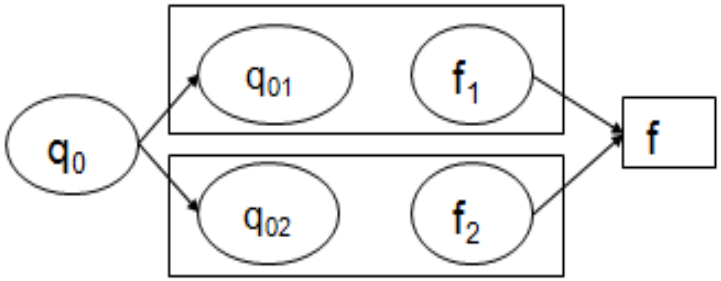
*Pentru orice expresie regulată $E$ peste $\Sigma$ există un automat finit (cu $\epsilon$ - tranziţii) $A$, astfel încât $L(A) = L(E)$.*

*Demonstratie*: inducţie structurală.

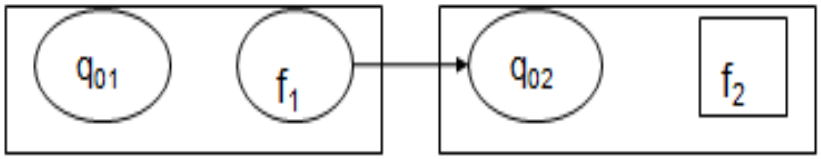
- Dacă $E \in \{\emptyset, \epsilon, a\}$ ($a \in \Sigma$) atunci automatul corespunzător este respectiv:

# Demonstraţie

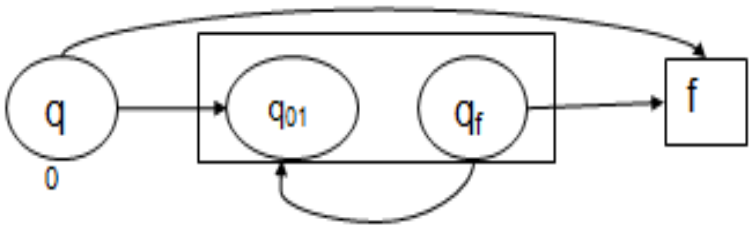
- $E = E_1 | E_2$




- $E = E_1 E_2$




- $E = E_1^*$



# 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 $tree(r, tS, tD)$

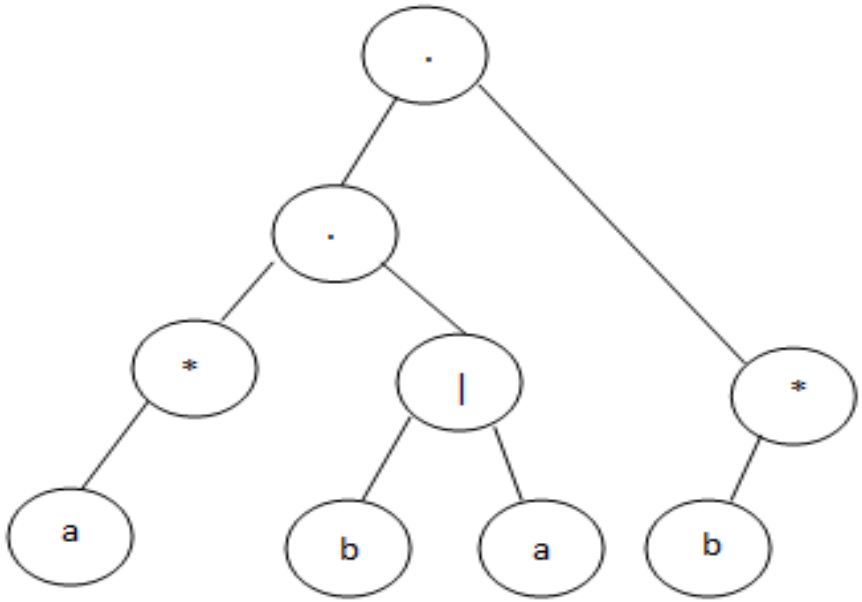
# Algoritm

```
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



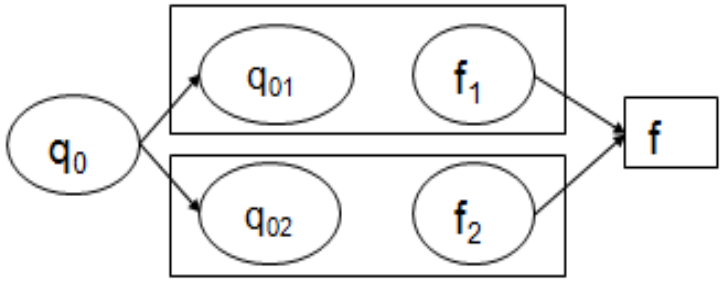

$a^* \cdot (b|a) \cdot b^*$

# Curs 4

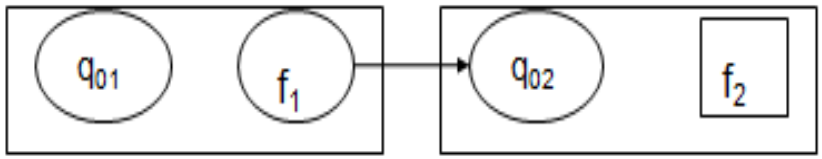
1. Corectitudinea algoritmului pentru determinarea relaţiei $\rho$
2. Gramatici de tip 3 şi automate finite
3. Proprietăţi de închidere pentru clasa limbajelor de tip 3
4. Expresii regulate
5. Automatul echivalent cu o expresie regulată
   - Algoritm

# Automatul echivalent cu o expresie regulată

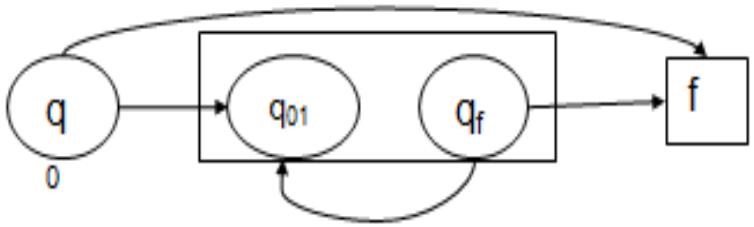

- $E = E_1 | E_2$




- $E = E_1 E_2$




- $E = E_1^*$



# Observaţii

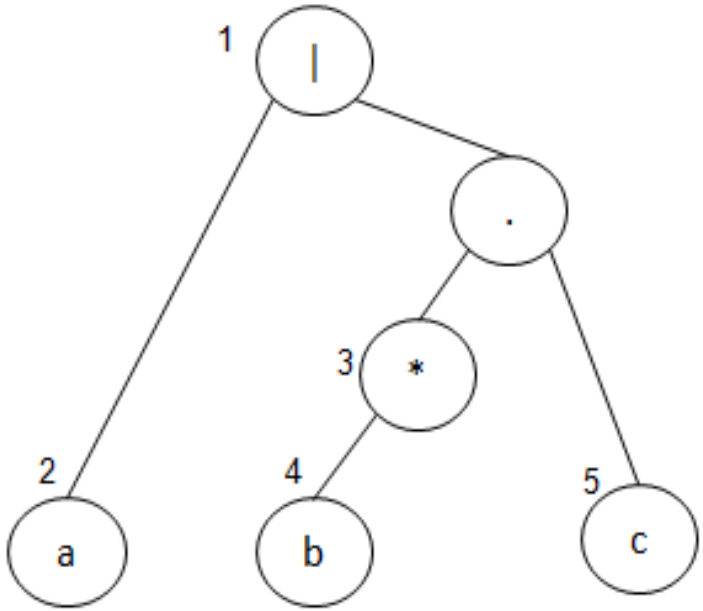
- pentru orice apariţie a unui simbol din $\Sigma$, cât şi pentru $\epsilon$, dacă acesta apare explicit în $E$, este nevoie de 2 stări în automatul construit.
- fiecare din apariţiile operatorilor $|$ şi $*$ dintr-o expresie regulată $E$ introduce două noi stări în automatul construit
- operatorul $\cdot$ 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 $\epsilon$ - 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:

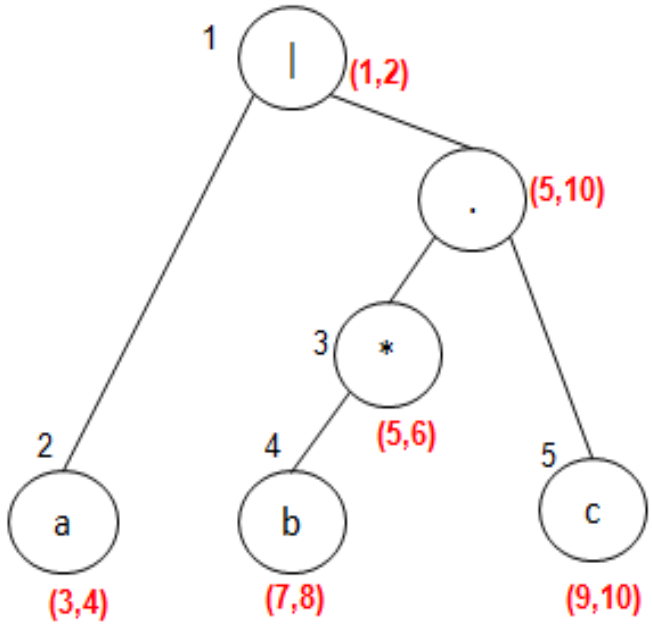
   - Dacă nodul are numărul $k$ (de la pasul 2) atunci:

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

   - 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 $\cdot$ :

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

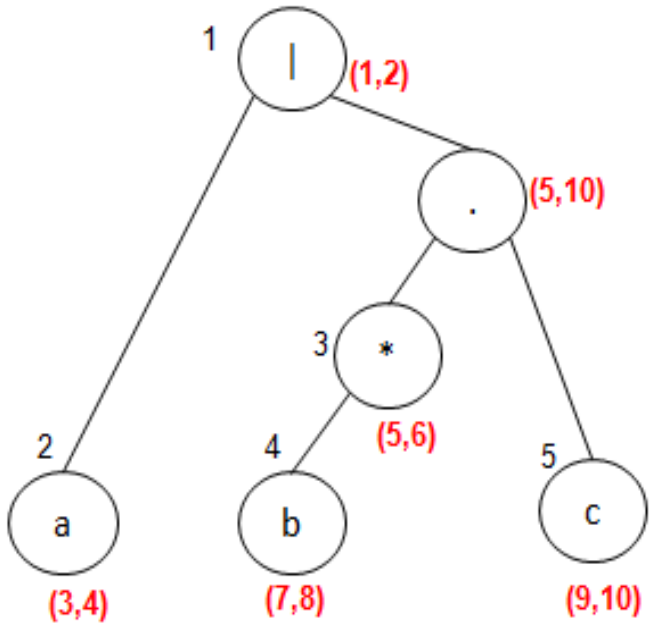
   - Dacă N este etichetat cu $*$ (D nu există în acest caz):

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

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

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

# Exemplu




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

# Exemplu

| $\delta$ | a | b | c | $\epsilon$ |
|---|---|---|---|---|
| 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 $\epsilon$ - 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.

# Limbaje Formale, Automate şi Compilatoare

Curs 5

2019-20

# Curs 5

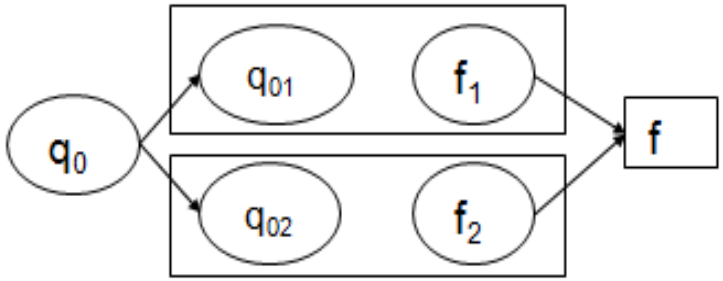
1. Automatul echivalent cu o expresie regulată
   - Algoritm
2. Gramatici şi limbaje independente de context
3. Forma redusă pentru gramatici independente de context
4. Eliminarea regulilor de ştergere şi a redenumirilor

# Curs 5

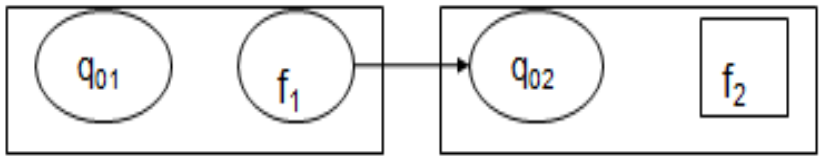
1. Automatul echivalent cu o expresie regulată
   - Algoritm
2. Gramatici şi limbaje independente de context
3. Forma redusă pentru gramatici independente de context
4. Eliminarea regulilor de ştergere şi a redenumirilor

# Automatul echivalent cu o expresie regulată

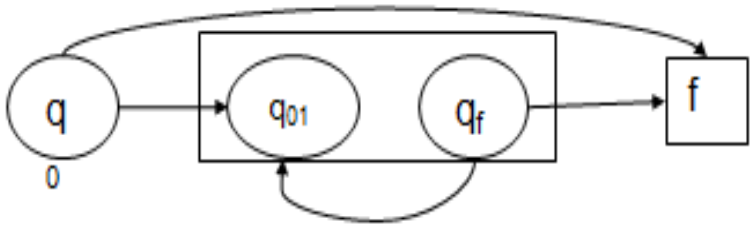

- $E = E_1 | E_2$




- $E = E_1 E_2$




- $E = E_1^*$



# Observaţii

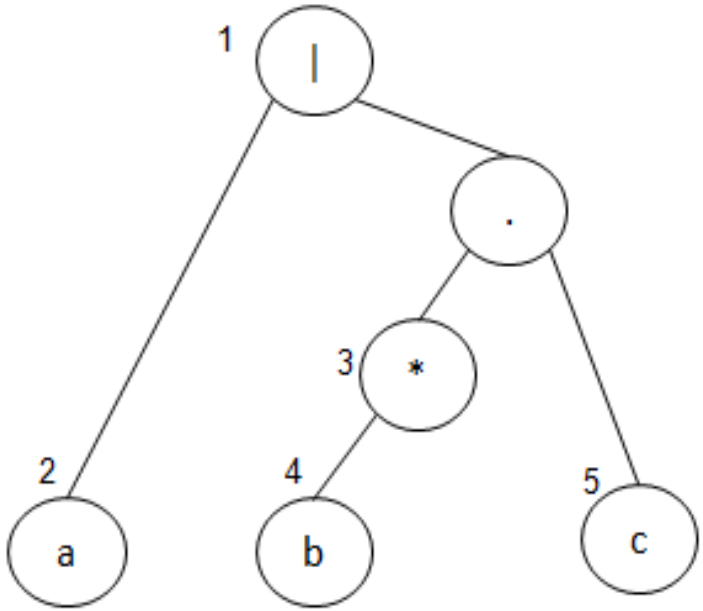
- pentru orice apariţie a unui simbol din $\Sigma$, cât şi pentru $\epsilon$, dacă acesta apare explicit în $E$, este nevoie de 2 stări în automatul construit.
- fiecare din apariţiile operatorilor $|$ şi $*$ dintr-o expresie regulată $E$ introduce două noi stări în automatul construit
- operatorul $\cdot$ 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 $\epsilon$ - 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:

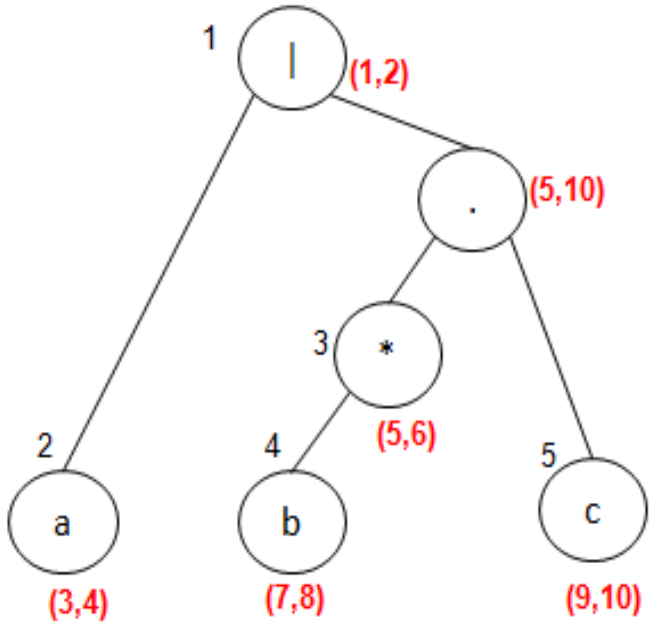
   - Dacă nodul are numărul $k$ (de la pasul 2) atunci:

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

   - 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 $\cdot$ :

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

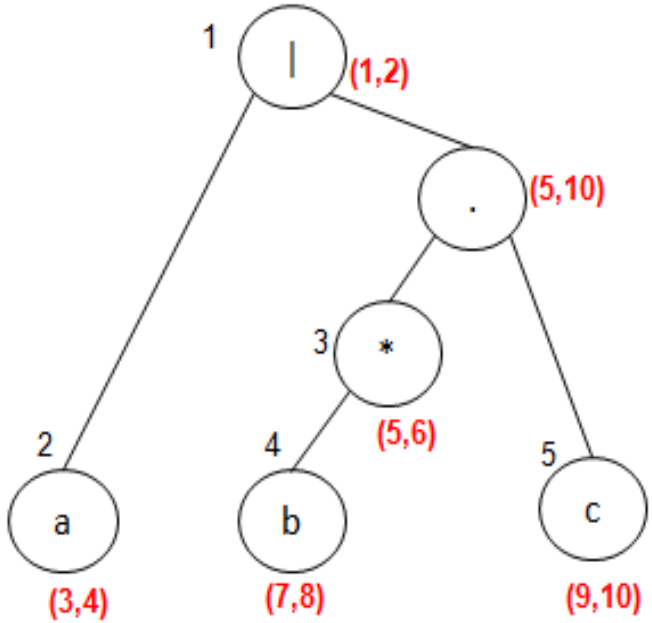
   - Dacă N este etichetat cu $*$ (D nu există în acest caz):

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

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

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

# Exemplu




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

# Exemplu

| $\delta$ | a | b | c | $\epsilon$ |
|---|---|---|---|---|
| 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 1**

*Algoritmul descris este corect: automatul cu $\epsilon$ - 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

1. Automatul echivalent cu o expresie regulată
   - Algoritm
2. Gramatici şi limbaje independente de context
3. Forma redusă pentru gramatici independente de context
4. Eliminarea regulilor de ştergere şi a redenumirilor

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

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 $\epsilon$ ;*
- *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 $A \to X_1 X_2 \ldots X_n$ este o regulă.*
  *Dacă A are un succesor etichetat cu $\epsilon$ (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* $w$*.*

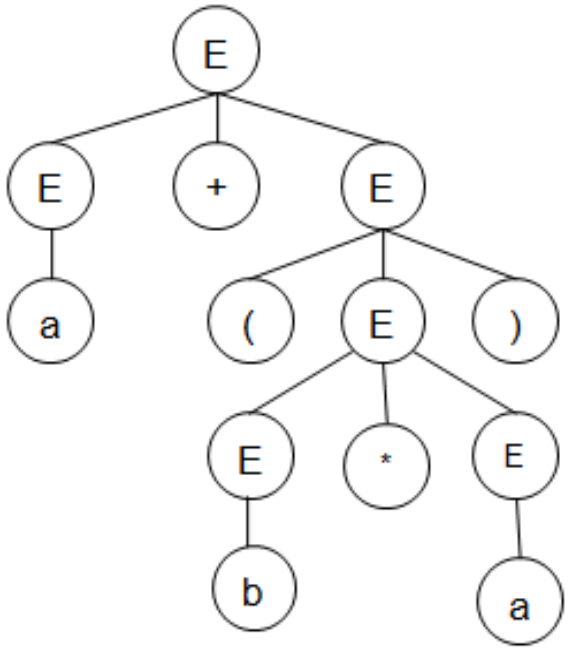
# Exemplu

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

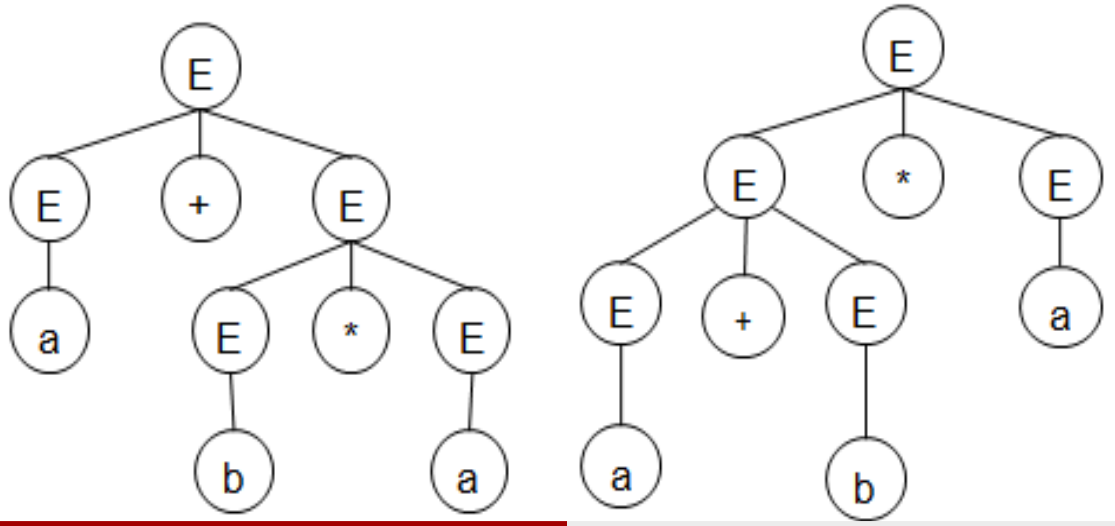
# Ambiguitate

## 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.*

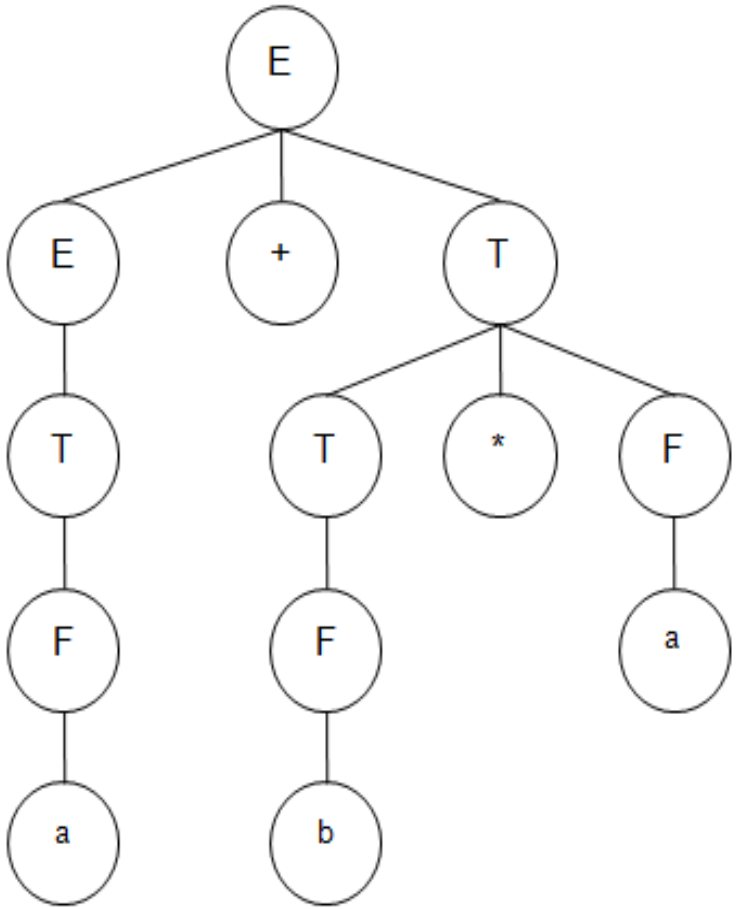
- Echivalent: *w* are 2 derivări extrem stângi(drepte) distincte.
- Problema ambiguităţii gramaticilor de tip 2 este nedecidabilă: nu există un algoritm care pentru o gramatică oarecare *G* să testeze dacă *G* este sau nu ambiguă

# 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

1. Automatul echivalent cu o expresie regulată
   - Algoritm
2. Gramatici şi limbaje independente de context
3. Forma redusă pentru gramatici independente de context
4. Eliminarea regulilor de ştergere şi a redenumirilor

# Simboluri inutile

- Un simbol $X$ din $N \cup T$ este accesibil dacă există o derivare de forma $S \Rightarrow^+ \alpha X \beta$
- Un simbol $A$ din $N$ este productiv dacă există o derivare de forma $A \Rightarrow^+ w, w \in T^*$
- Un simbol este inutil dacă este inaccesibil sau neproductiv

# Gramatici în formă redusă

**Definiţie 4**

*O gramatică este în formă redusă, dacă nu conţine simboluri inutile.*

- Orice limbaj independent de context poate fi generat de o gramatică în formă redusă.

# Eliminarea simbolurilor inutile

- Pentru orice gramatică independentă de context $G$ există o gramatică $G'$ de acelaşi tip în formă redusă echivalentă cu $G$.
- Pentru eliminarea simbolurilor inutile:
  - Se determină şi apoi se elimină simbolurile neproductive şi toate regulile ce conţin măcar unul dintre acestea.
  - Se determină apoi se elimină simbolurile inaccesibile şi toate regulile aferente.

# Eliminarea simbolurilor neproductive - algoritm

- Intrare: $G = (N, T, S, P)$
- Ieşire: $G' = (N', T, S, P')$, $L(G') = L(G)$, $N'$ conţine doar simboluri productive

$N_0 = \emptyset;\ \ i = 0;$
do {
$\quad i = i + 1;$
$\quad N_i = N_{i-1} \cup \{A | A \to \alpha \in P, \alpha \in (N_{i-1} \cup T)^*\};$
} while $N_i \neq N_{i-1}$;
$N' = N_i;$
$P' = \{A \to \alpha \in P | A \in N', \alpha \in (N' \cup T)^*\};$

- Un simbol $A$ este productiv ddacă $A \in N'$
- Consecinţă: $L(G) \neq \emptyset$ ddacă $S \in N'$

# Exemplu

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

- $S \to a|aA|bC$
- $A \to aAB$
- $B \to bac$
- $C \to aSb$

Gramatica $G'$ cu toate simbolurile productive:

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

- $S \to a|bC$
- $B \to bac$
- $C \to aSb$

# Eliminarea simbolurilor inaccesibile

- Intrare: $G = (N, T, S, P)$
- Ieşire: $G' = (N', T', S, P')$, $L(G') = L(G)$, $N', T'$ conţin doar simboluri accesibile

$V_0 = \{S\}$; $i = 0$;
do {
$i = i + 1$;
$V_i = V_{i-1} \cup \{X | X \in N \cup T, \ \exists A \to \alpha X \beta \in P, A \in (V_{i-1} \cap N)\}$;
} while $V_i \neq V_{i-1}$;
$N' = V_i \cap N$;
$T' = V_i \cap T$;
$P' = \{A \to \alpha \in P | A \in N', \alpha \in (N' \cup T')^*\}$;

- $X$ accesibil ddacă $X \in V_i$

# Exemplu

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

- $S \to a|aA|bC$
- $A \to aAB$
- $B \to bac$
- $C \to aSb$
- Eliminarea simbolurilor neproductive duce la:

$$G' = (\{S, B, C\}, \{a, b, c\}, S, \{S \to a|bC, B \to bac, C \to aSb\})$$

- Eliminarea simbolurilor inaccesibile duce la:

$$G' = (\{S, C\}, \{a, b\}, S, \{S \to a|bC, C \to aSb\})$$

- Ce se întâmplă dacă se aplică algoritmii în ordinea inversă?

# Curs 5

1. Automatul echivalent cu o expresie regulată
   - Algoritm
2. Gramatici şi limbaje independente de context
3. Forma redusă pentru gramatici independente de context
4. Eliminarea regulilor de ştergere şi a redenumirilor

# 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 $A \to \epsilon$)

$N_0 = \{A | A \in N, \ \ A \to \epsilon \in P\}; \ \ i = 0;$
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 $N_i \neq N_{i-1}$;
$N_\epsilon = N_i;$

Are loc:

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

# Eliminarea regulilor de ştergere

$P'$ se obţine din $P$ astfel:

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

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

- 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 $N_\epsilon$) 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 $P'$:

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

# Limbaje Formale, Automate şi Compilatoare

Curs 6

2019-20

# Curs 6

1. Eliminarea redenumirilor din gramatici de tip 2
2. Forma normală Chomsky
3. Problema recunoaşterii: algoritmul Cocke Younger Kasami
4. Automate pushdown
5. Legătura dintre automatele pushdown şi limbajele de tip 2
6. Automate pushdown deterministe

# Curs 6

1 Eliminarea redenumirilor din gramatici de tip 2

2 Forma normală Chomsky

3 Problema recunoaşterii: algoritmul Cocke Younger Kasami

4 Automate pushdown

5 Legătura dintre automatele pushdown şi limbajele de tip 2

6 Automate pushdown deterministe

# Eliminarea redenumirilor ($A \to B,\ A, B \in N$)

- Intrare: $G = (N, T, S, P)$
- Ieşire: $G' = (N, T, S, P')$, $L(G') = L(G)$, $P'$ nu conţine redenumiri

```
for(A ∈ N){
    N_0 = {A};  i = 0;
    do{
        i = i + 1;
        N_i = N_{i-1} ∪ {C | C ∈ N,  ∃B → C ∈ P, B ∈ N_{i-1}};
    } while N_i ≠ N_{i-1};
    N_A = N_i; //N_A = {X ∈ N | A ⇒* X}
}
P' = {X → α ∈ P | α ∉ N}
for(X → α_1|α_2|...|α_n ∈ P')
    for(A ∈ N  &&  X ∈ N_A, X ≠ A)
        P' = P' ∪ {A → α_1|α_2|...|α_n}
```

# Exemplu

$G = (\{x, y, z\}, \{a, b, c\}, x, P)$, unde P:

- $x \to y|ax|a$
- $y \to z|by|b$
- $z \to cz|c$

$N_x = \{x, y, z\}$, $N_y = \{y, z\}$, $N_z = \{z\}$

Gramatica echivalentă fără redenumiri $G' = (\{x, y, z\}, \{a, b, c\}, x, P')$ unde $P'$:

- $x \to ax|a|by|b|cz|c$
- $y \to by|b|cz|c$
- $z \to cz|c$

# Curs 6


1. Eliminarea redenumirilor din gramatici de tip 2
2. Forma normală Chomsky
3. Problema recunoaşterii: algoritmul Cocke Younger Kasami
4. Automate pushdown
5. Legătura dintre automatele pushdown şi limbajele de tip 2
6. Automate pushdown deterministe

# Forma normală Chomsky

## Definiţie 1

*O gramatică este în formă normală Chomsky dacă regulile sale au forma:*

$A \to BC$, $A \to a$ *( şi eventual* $S \to \epsilon$*)* $(A, B, C \in N$ *şi* $a \in T)$.

## Teorema 1

*Orice limbaj independent de context poate fi generat de o gramatică în formă normală Chomsky.*

# Demonstraţie

- Se elimină regulile de ştergere şi redenumirile

# Demonstraţie

- Se elimină regulile de ştergere şi redenumirile
- Se elemină regulile care nu sunt în formă normală Chomsky: Dacă $A \to x_1 x_2 \ldots x_n$, $n > 1$ este o astfel de regulă atunci o înlocuim cu $A \to Y_1 Y_2 \ldots Y_n$ unde:
  - $Y_i = x_i$, dacă $x_i \in N$ (neterminalii rămân la fel)
  - $Y_i = x_a$ dacă $x_i = a \in T$ ($x_a$ este neterminal nou) şi se adaugă regula $x_a \to a$

# Demonstraţie

- Se elimină regulile de ştergere şi redenumirile
- Se elemină regulile care nu sunt în formă normală Chomsky: Dacă $A \to x_1 x_2 \dots x_n$, $n > 1$ este o astfel de regulă atunci o înlocuim cu $A \to Y_1 Y_2 \dots Y_n$ unde:
  - $Y_i = x_i$, dacă $x_i \in N$ (neterminalii rămân la fel)
  - $Y_i = x_a$ dacă $x_i = a \in T$ ($x_a$ este neterminal nou) şi se adaugă regula $x_a \to a$
- O regulă de forma $A \to Y_1 Y_2 \dots Y_n$ , dacă $n > 2$, o înlocuim cu:
  - $A \to Y_1 Z_1$
  - $Z_1 \to Y_2 Z_2$
  - .....
  - $Z_{n-3} \to Y_{n-2} Z_{n-2}$
  - $Z_{n-2} \to Y_{n-1} Y_n$,

    unde $Z_1, Z_2, \dots, Z_{n-2}$ sunt neterminali noi.

# Exemplu

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

- $S \to aSb|cAc$
- $A \to cA|c$

Gramatica echivalentă în formă normală Chomsky

$G = (\{S, A, x_a, x_b, Z_1, Z_2\}, \{a, b, c\}, S, P')$, unde $P'$:

- $S \to x_a Z_1 | x_c Z_2$
- $Z_1 \to S x_b$
- $Z_2 \to A x_c$
- $A \to x_c A | c$
- $x_a \to a$
- $x_b \to b$
- $x_c \to c$

# Curs 6

1 Eliminarea redenumirilor din gramatici de tip 2

2 Forma normală Chomsky

3 Problema recunoaşterii: algoritmul Cocke Younger Kasami

4 Automate pushdown

5 Legătura dintre automatele pushdown şi limbajele de tip 2

6 Automate pushdown deterministe

# Algoritmul Cocke Younger Kasami (CYK)

- Problema recunoaşterii în gramatici de tip 2: dată o gramatică de tip 2 si un cuvânt w, să se decidă dacă $w \in L(G)$
- Problema recunoaşterii în gramatici în formă normală Chomsky se poate rezolva cu algoritmul CYK în timp $O(n^3)$.
- Dacă $w = a_1 a_2 \ldots a_n$ atunci se constuiesc mulţimile

$$V_{ij} = \{A | A \Rightarrow^+ a_i a_{i+1} \ldots a_{i+j-1}\}$$

inductiv pentru $j = 1, \ldots, n$

$$w \in L(G) \Leftrightarrow S \in V_{1n}$$

# Algoritmul Cocke Younger Kasami

- Pentru $j = 1$:
  - $V_{i1} = \{A | A \Rightarrow^+ a_i\} = \{A | \exists A \to a_i \in P\}$
- Pentru $j > 1$, $V_{ij}$:
  - Dacă $A \Rightarrow^+ a_i a_{i+1} \ldots a_{i+j-1}$:

    $A \Rightarrow BC \Rightarrow^+ a_i a_{i+1} \ldots a_{i+j-1}$ şi

    $B \Rightarrow^+ a_i a_{i+1} \ldots a_{i+k-1}$ $(B \in V_{ik})$

    $C \Rightarrow^+ a_{i+k} a_{i+k+1} \ldots a_{i+j-1}$ $(C \in V_{i+k\ j-k})$

    unde $1 \leq i \leq n+1-j$, $1 \leq k \leq j-1$
  - $V_{ij} = \bigcup_{k=1}^{j-1} \{A | A \to BC \in P, B \in V_{ik}, C \in V_{i+k\ j-k}\}$

# Algoritmul Cocke Younger Kasami

- Notaţie:

$$\{A | A \to BC \in P, B \in V_{ik}, C \in V_{i+k\ j-k}\} = V_{ik} \circ V_{i+k\ j-k}$$

- Atunci:

  pentru $2 \leq j \leq n, 1 \leq i \leq n+1-j$ :

$$V_{ij} = \bigcup_{k=1}^{j-1} (V_{ik} \circ V_{i+k\ j-k})$$

# Algoritmul Cocke Younger Kasami

- Intrare: $G = (N, T, S, P)$ în formă normală Chomsky, $w = a_1 a_2 \dots a_n$
- Ieşire: $w \in L(G)$?

```
for(i=1; i<=n; i++)
    V_{i1} = {A | ∃ A → a_i ∈ P};
for(j=2; j<=n; j++)
    for (i=1; i<=n+1-j; i++){
        V_{ij} = ∅;
        for(k=1; k<=j-1; k++)
            V_{ij} = V_{ij} ∪ (V_{ik} ∘ V_{i+k  j-k});
    }
if(S ∈ V_{1n})  w ∈ L(G) else  w ∉ L(G)
```

# Exemplu

$G = (\{S, X, Y, Z\}, \{a, b, c\}, S, P)$, unde P:

- $S \to XY$
- $X \to XY|a$
- $Y \to YZ|a|b$
- $Z \to c$

$w = abc$

# Exemplu

$G = (\{S, X, Y, Z\}, \{a, b, c\}, S, P)$, unde P:

- $S \to XY$
- $X \to XY|a$
- $Y \to YZ|a|b$
- $Z \to c$

$w = abc$

| | | |
|---|---|---|
| $V_{11} = \{X, Y\}$ | $V_{12} = \{S, X\}$ | $V_{13} = \{S, X\}$ |
| $V_{21} = \{Y\}$ | $V_{22} = \{Y\}$ | |
| $V_{31} = \{Z\}$ | | |

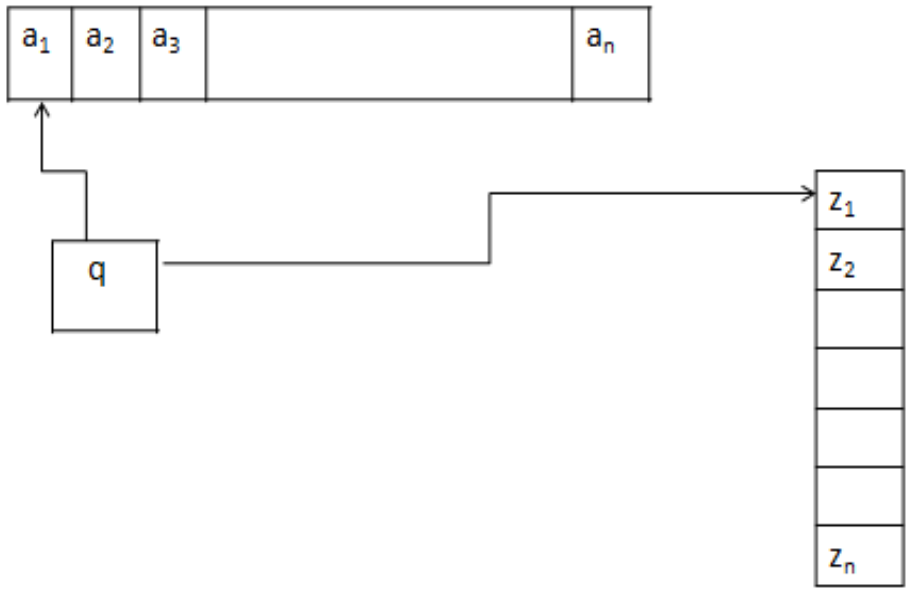
$S \in V_{13} \Leftrightarrow abc \in L(G)$

# Curs 6


1. Eliminarea redenumirilor din gramatici de tip 2
2. Forma normală Chomsky
3. Problema recunoaşterii: algoritmul Cocke Younger Kasami
4. Automate pushdown
5. Legătura dintre automatele pushdown şi limbajele de tip 2
6. Automate pushdown deterministe

# Automate pushdown

- Automat finit + memorie pushdown (stiva)
- Model fizic:



# Automate pushdown-definiţie

## Definiţie 2

*Un automat pushdown este un 7-uplu:* $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$*:*

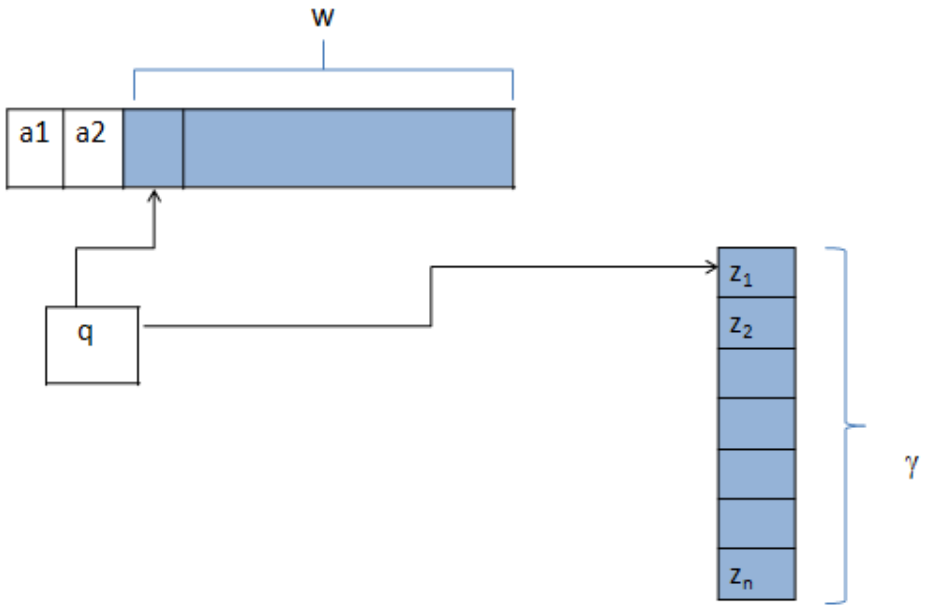
- $Q$ *este mulţimea (finită) a stărilor*
- $\Sigma$ *este alfabetul de intrare*
- $\Gamma$ *este alfabetul memoriei pushdown (stivei)*
- $q_0 \in Q$ *este starea iniţială*
- $z_0 \in \Gamma$ *este simbolul iniţial din stivă*
- $F \subseteq Q$ *este mulţimea stărilor finale*
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$

Modelul este nedeterminist

# Configuraţia unui automat pushdown

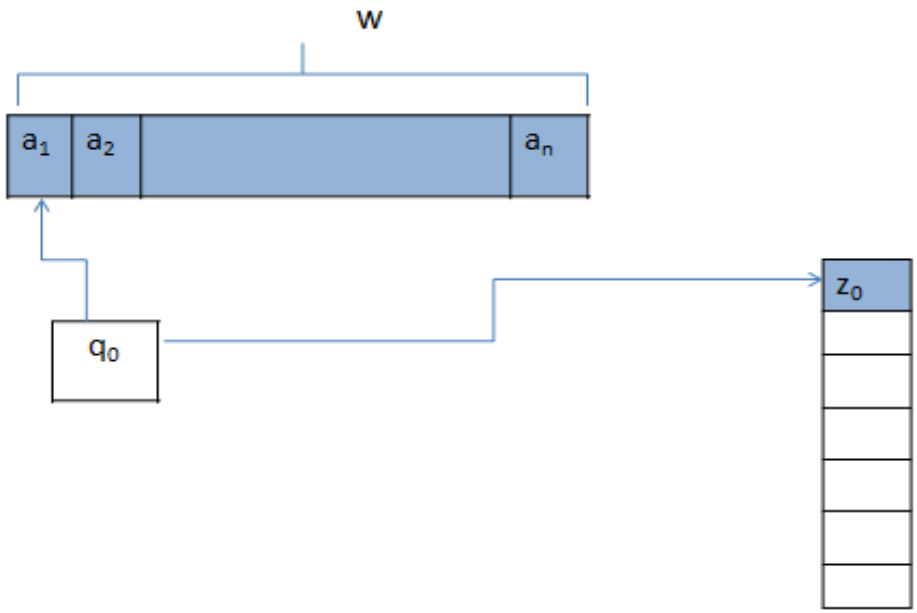
**Configuraţie**: $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$

1 : $\gamma$ (primul simbol din $\gamma$) reprezintă vârful stivei



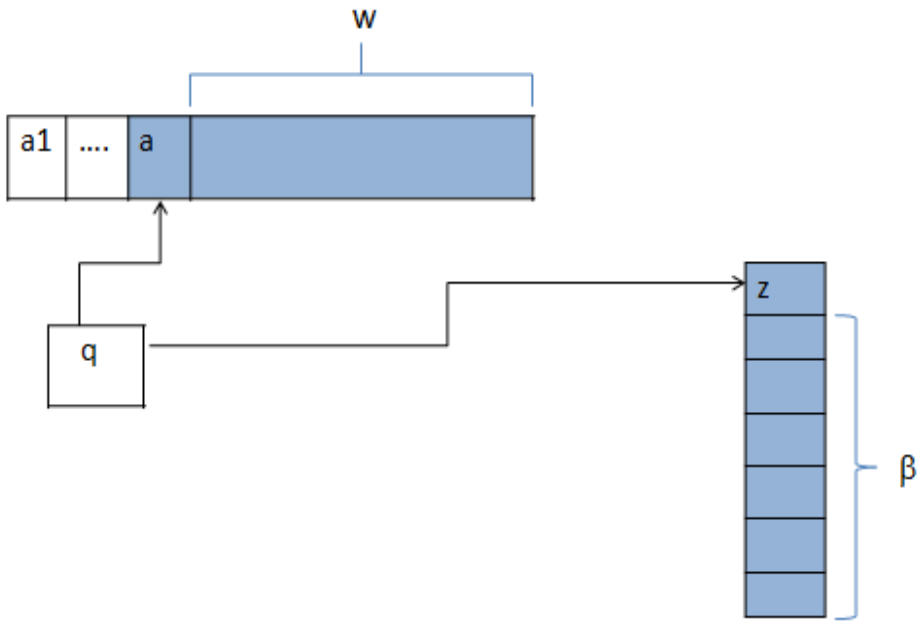
# Automate pushdown

**Configuraţie iniţială**: $(q_0, w, z_0) \in Q \times \Sigma^* \times \Gamma^*$



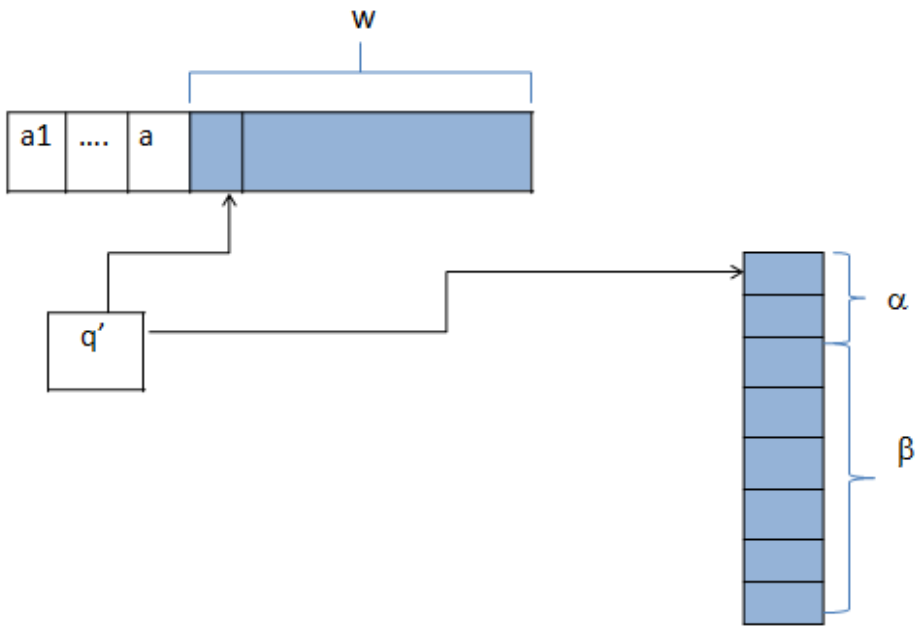
# Relaţia de tranziţie între configuraţii

- Configuraţia curentă $(q, aw, z\beta)$ şi $(q', \alpha) \in \delta(q, a, z)$
  $(q, q' \in Q,\ a \in \Sigma \cup \{\epsilon\},\ z \in \Gamma,\ \alpha, \beta \in \Gamma^*)$



# Relaţia de tranziţie între configuraţii

- $(q, aw, z\beta) \vdash (q', w, \alpha\beta)$



# Relaţia de tranziţie între configuraţii

Fie $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ un automat pushdown.

- Relaţia de tranziţie între configuraţii:
  $(q, aw, z\beta) \vdash (q', w, \alpha\beta)$ dacă $(q', \alpha) \in \delta(q, a, z)$
  $(q, q' \in Q,\ a \in \Sigma \cup \{\epsilon\},\ z \in \Gamma,\ \alpha, \beta \in \Gamma^*)$

- Calcul: închiderea reflexivă şi tranzitivă a relaţiei de mai sus: dacă $C_1, \ldots, C_n$ configuraţii astfel încât:

$$C_1 \vdash C_2 \vdash \ldots \vdash C_n$$

se scrie: $C_1 \vdash^+ C_n$ dacă $n \geq 2$, $C_1 \vdash^* C_n$, dacă $n \geq 1$

# Limbajul recunoscut

## Prin stări finale (dacă $F \neq \emptyset$)

$$L(M) = \{w \in \Sigma^* | (q_0, w, z_0) \vdash^* (q, \epsilon, \gamma),\ q \in F,\ \gamma \in \Gamma^*\}$$

## Prin golirea stivei (dacă $F = \emptyset$)

$$L_\epsilon(M) = \{w \in \Sigma^* | (q_0, w, z_0) \vdash^* (q, \epsilon, \epsilon),\ q \in Q\}$$

# Exemplu

Automat care recunoaşte limbajul $\{a^n b^n | n \geq 1\}$:

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, z\}, \delta, q_0, z, \{q_2\})$$

1. $\delta(q_0, a, z) = \{(q_0, az)\}$
2. $\delta(q_0, a, a) = \{(q_0, aa)\}$
3. $\delta(q_0, b, a) = \{(q_1, \epsilon)\}$
4. $\delta(q_1, b, a) = \{(q_1, \epsilon)\}$
5. $\delta(q_1, \epsilon, z) = \{(q_2, \epsilon)\}$

# Exemple

- Un automat pushdown ce recunoaşte limbajul $\{waw^R | w \in \{0,1\}^*\}$

# Exemple

- Un automat pushdown ce recunoaşte limbajul $\{waw^R | w \in \{0,1\}^*\}$
  - Fiecare 0 sau 1 citit se introduce în stivă
  - $a$ la intrare produce pregătirea scoaterii a câte un simbol din stiva dacă el coincide cu cel din intrare

# Exemple

- Un automat pushdown ce recunoaşte limbajul $\{waw^R | w \in \{0,1\}^*\}$
  - Fiecare 0 sau 1 citit se introduce în stivă
  - $a$ la intrare produce pregătirea scoaterii a câte un simbol din stiva dacă el coincide cu cel din intrare

$M = (\{q_0, q_1, q_2\}, \{0, 1, a\}, \{0, 1, z\}, \delta, q_0, z, \{q_2\})$

1. $\delta(q_0, i, z) = \{(q_0, iz)\}, (i \in \{0, 1\})$
2. $\delta(q_0, i, j) = \{(q_0, ij)\}, (i, j \in \{0, 1\})$
3. $\delta(q_0, a, i) = \{(q_1, i)\}$
4. $\delta(q_1, i, i) = \{(q_1, \epsilon)\}$
5. $\delta(q_1, \epsilon, z) = \{(q_2, \epsilon)\}$

# Exemple

- Un automat pushdown ce recunoaşte limbajul $\{waw^R | w \in \{0,1\}^*\}$
  - Fiecare 0 sau 1 citit se introduce în stivă
  - $a$ la intrare produce pregătirea scoaterii a câte un simbol din stiva dacă el coincide cu cel din intrare

$M = (\{q_0, q_1, q_2\}, \{0, 1, a\}, \{0, 1, z\}, \delta, q_0, z, \{q_2\})$

1. $\delta(q_0, i, z) = \{(q_0, iz)\}, (i \in \{0,1\})$
2. $\delta(q_0, i, j) = \{(q_0, ij)\}, (i, j \in \{0,1\})$
3. $\delta(q_0, a, i) = \{(q_1, i)\}$
4. $\delta(q_1, i, i) = \{(q_1, \epsilon)\}$
5. $\delta(q_1, \epsilon, z) = \{(q_2, \epsilon)\}$

- Un automat pushdown ce recunoaşte limbajul $\{ww^R | w \in \{0,1\}^*\}$ ?

# Exemple

- Un automat pushdown ce recunoaşte limbajul $\{waw^R | w \in \{0,1\}^*\}$
  - Fiecare 0 sau 1 citit se introduce în stivă
  - $a$ la intrare produce pregătirea scoaterii a câte un simbol din stiva dacă el coincide cu cel din intrare

$M = (\{q_0, q_1, q_2\}, \{0,1,a\}, \{0,1,z\}, \delta, q_0, z, \{q_2\})$

1. $\delta(q_0, i, z) = \{(q_0, iz)\}, (i \in \{0,1\})$
2. $\delta(q_0, i, j) = \{(q_0, ij)\}, (i,j \in \{0,1\})$
3. $\delta(q_0, a, i) = \{(q_1, i)\}$
4. $\delta(q_1, i, i) = \{(q_1, \epsilon)\}$
5. $\delta(q_1, \epsilon, z) = \{(q_2, \epsilon)\}$

- Un automat pushdown ce recunoaşte limbajul $\{ww^R | w \in \{0,1\}^*\}$ ?
- Un automat pushdown ce recunoaşte limbajul $\{ww | w \in \{0,1\}^*\}$ ?

# Echivalenţa definiţiilor privind recunoaşterea

## Teorema 2

*Pentru orice automat pushdown $M$ cu $F = \emptyset$, există un automat pushdown $M'$ cu stări finale astfel ca $L(M') = L_\epsilon(M)$.*

# Echivalenţa definiţiilor privind recunoaşterea

**Teorema 2**

*Pentru orice automat pushdown $M$ cu $F = \emptyset$, există un automat pushdown $M'$ cu stări finale astfel ca $L(M') = L_\epsilon(M)$.*

Dacă $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, \emptyset)$, considerăm
$M' = (Q \cup \{q_f, q_0'\}, \Sigma, \Gamma \cup \{z_0'\}, \delta', q_0', z_0', \{q_f\})$ cu $\delta'$:

# Echivalenţa definiţiilor privind recunoaşterea

**Teorema 2**

*Pentru orice automat pushdown $M$ cu $F = \emptyset$, există un automat pushdown $M'$ cu stări finale astfel ca $L(M') = L_\epsilon(M)$.*

Dacă $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, \emptyset)$, considerăm
$M' = (Q \cup \{q_f, q_0'\}, \Sigma, \Gamma \cup \{z_0'\}, \delta', q_0', z_0', \{q_f\})$ cu $\delta'$:

1. $\delta'(q_0', \epsilon, z_0') = \{(q_0, z_0 z_0')\}$ (fără să citească niciun simbol, $M'$ trece în configuraţia iniţială a lui $M$)

# Echivalenţa definiţiilor privind recunoaşterea

**Teorema 2**

*Pentru orice automat pushdown $M$ cu $F = \emptyset$, există un automat pushdown $M'$ cu stări finale astfel ca $L(M') = L_\epsilon(M)$.*

Dacă $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, \emptyset)$, considerăm
$M' = (Q \cup \{q_f, q_0'\}, \Sigma, \Gamma \cup \{z_0'\}, \delta', q_0', z_0', \{q_f\})$ cu $\delta'$:

1. $\delta'(q_0', \epsilon, z_0') = \{(q_0, z_0 z_0')\}$ (fără să citească niciun simbol, $M'$ trece în configuraţia iniţială a lui $M$)
2. $\delta'(q, a, z) = \delta(q, a, z)$, $\forall q \in Q,\ a \in \Sigma \cup \{\epsilon\},\ z \in \Gamma$ ($M'$ face aceleaşi tranziţii ca şi $M$)

# Echivalenţa definiţiilor privind recunoaşterea

**Teorema 2**

*Pentru orice automat pushdown $M$ cu $F = \emptyset$, există un automat pushdown $M'$ cu stări finale astfel ca $L(M') = L_\epsilon(M)$.*

Dacă $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, \emptyset)$, considerăm
$M' = (Q \cup \{q_f, q_0'\}, \Sigma, \Gamma \cup \{z_0'\}, \delta', q_0', z_0', \{q_f\})$ cu $\delta'$:

1. $\delta'(q_0', \epsilon, z_0') = \{(q_0, z_0 z_0')\}$ (fără să citească niciun simbol, $M'$ trece în configuraţia iniţială a lui $M$)
2. $\delta'(q, a, z) = \delta(q, a, z)$, $\forall q \in Q,\ a \in \Sigma \cup \{\epsilon\},\ z \in \Gamma$ ($M'$ face aceleaşi tranziţii ca şi $M$)
3. $\delta'(q, \epsilon, z_0') = \{(q_f, \epsilon)\}$, $\forall q \in Q$ ($M'$ va trece în starea finală doar dacă stiva lui $M$ este vidă)

# Echivalenţa definiţiilor privind recunoaşterea

## Teorema 3

*Pentru orice automat pushdown $M$ cu $F \neq \emptyset$, există un automat pushdown $M'$ cu $F = \emptyset$ astfel ca $L_\epsilon(M') = L(M)$.*

# Echivalenţa definiţiilor privind recunoaşterea

**Teorema 3**

*Pentru orice automat pushdown $M$ cu $F \neq \emptyset$, există un automat pushdown $M'$ cu $F = \emptyset$ astfel ca $L_\epsilon(M') = L(M)$.*

Dacă $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$, considerăm

$M' = (Q \cup \{q_\epsilon, q_0'\}, \Sigma, \Gamma \cup \{z_0'\}, \delta', q_0', z_0', \emptyset)$

# Demonstraţie

$M' = (Q \cup \{q_\epsilon, q_0'\}, \Sigma, \Gamma \cup \{z_0'\}, \delta', q_0', z_0', \emptyset)$, cu $\delta'$:

# Demonstraţie

$M' = (Q \cup \{q_\epsilon, q_0'\}, \Sigma, \Gamma \cup \{z_0'\}, \delta', q_0', z_0', \emptyset)$, cu $\delta'$:

1. $\delta'(q_0', \epsilon, z_0') = \{(q_0, z_0 z_0')\}$ (fără să citească niciun simbol, $M'$ trece în configuraţia iniţială a lui $M$)

# Demonstraţie

$M' = (Q \cup \{q_\epsilon, q_0'\}, \Sigma, \Gamma \cup \{z_0'\}, \delta', q_0', z_0', \emptyset)$, cu $\delta'$:

1. $\delta'(q_0', \epsilon, z_0') = \{(q_0, z_0 z_0')\}$ (fără să citească niciun simbol, $M'$ trece în configuraţia iniţială a lui $M$)
2. a) $\delta'(q, a, z) = \delta(q, a, z)$, $\forall q \in Q,\ a \in \Sigma,\ z \in \Gamma$ ($M'$ face aceleaşi tranziţii ca şi $M$, pentru orice simbol întâlnit)
   b) $\delta'(q, \epsilon, z) = \delta(q, \epsilon, z)$, dacă $q \in Q \setminus F,\ z \in \Gamma$ (se fac aceleaşi $\epsilon$-tranziţii ca în M, dacă starea nu este finală )
   c) $\delta'(q, \epsilon, z) = \delta(q, \epsilon, z) \cup \{(q_\epsilon, \epsilon)\}$, $q \in F,\ z \in \Gamma$ (daca M ajunge într-o stare finală, $M'$ poate trece într-o stare specială )

# Demonstraţie

$M' = (Q \cup \{q_\epsilon, q_0'\}, \Sigma, \Gamma \cup \{z_0'\}, \delta', q_0', z_0', \emptyset)$, cu $\delta'$:

1. $\delta'(q_0', \epsilon, z_0') = \{(q_0, z_0 z_0')\}$ (fără să citească niciun simbol, $M'$ trece în configuraţia iniţială a lui $M$)
2. a) $\delta'(q, a, z) = \delta(q, a, z)$, $\forall q \in Q,\ a \in \Sigma,\ z \in \Gamma$ ($M'$ face aceleaşi tranziţii ca şi $M$, pentru orice simbol întâlnit)
   b) $\delta'(q, \epsilon, z) = \delta(q, \epsilon, z)$, dacă $q \in Q \setminus F,\ z \in \Gamma$ (se fac aceleaşi $\epsilon$-tranziţii ca în M, dacă starea nu este finală )
   c) $\delta'(q, \epsilon, z) = \delta(q, \epsilon, z) \cup \{(q_\epsilon, \epsilon)\}$, $q \in F,\ z \in \Gamma$ (daca M ajunge într-o stare finală, $M'$ poate trece într-o stare specială )
3. $\delta'(q, \epsilon, z_0') = \{(q_\epsilon, \epsilon)\}$, dacă $q \in F$ ( cazul 2(c), în situaţia în care în stivă este $z_0'$)

# Demonstraţie

$M' = (Q \cup \{q_\epsilon, q_0'\}, \Sigma, \Gamma \cup \{z_0'\}, \delta', q_0', z_0', \emptyset)$, cu $\delta'$:

1. $\delta'(q_0', \epsilon, z_0') = \{(q_0, z_0 z_0')\}$ (fără să citească niciun simbol, $M'$ trece în configuraţia iniţială a lui $M$)
2. a) $\delta'(q, a, z) = \delta(q, a, z), \forall q \in Q,\ a \in \Sigma,\ z \in \Gamma$ ($M'$ face aceleaşi tranziţii ca şi $M$, pentru orice simbol întâlnit)
   b) $\delta'(q, \epsilon, z) = \delta(q, \epsilon, z)$, dacă $q \in Q \setminus F,\ z \in \Gamma$ (se fac aceleaşi $\epsilon$-tranziţii ca în M, dacă starea nu este finală )
   c) $\delta'(q, \epsilon, z) = \delta(q, \epsilon, z) \cup \{(q_\epsilon, \epsilon)\}$, $q \in F,\ z \in \Gamma$ (daca M ajunge într-o stare finală, $M'$ poate trece într-o stare specială )
3. $\delta'(q, \epsilon, z_0') = \{(q_\epsilon, \epsilon)\}$, dacă $q \in F$ ( cazul 2(c), în situaţia în care în stivă este $z_0'$)
4. $\delta'(q_\epsilon, \epsilon, z) = \{(q_\epsilon, \epsilon)\}$, dacă $z \in \Gamma \cup \{z_0'\}$ ( $M'$ rămâne în starea $q_\epsilon$ şi se extrage vârful stivei)

# Curs 6

1 Eliminarea redenumirilor din gramatici de tip 2

2 Forma normală Chomsky

3 Problema recunoaşterii: algoritmul Cocke Younger Kasami

4 Automate pushdown

5 Legătura dintre automatele pushdown şi limbajele de tip 2

6 Automate pushdown deterministe

# Automatul pushdown echivalent cu o gramatică de tip 2

**Teorema 4**

*Pentru orice gramatică $G$ există un automat pushdown $M$ fără stări finale astfel încât $L_\epsilon(M) = L(G)$*

# Automatul pushdown echivalent cu o gramatică de tip 2

**Teorema 4**

*Pentru orice gramatică G există un automat pushdown M fără stări finale astfel încât* $L_\epsilon(M) = L(G)$

- Fie $G = (N, T, S, P)$
- Construim $M = (\{q\}, T, N \cup T, \delta, q, S, \emptyset)$ unde:
  1. $\delta(q, \epsilon, A) = \{(q, \beta) | A \to \beta \in P\}, \forall A \in N$
  2. $\delta(q, a, a) = \{(q, \epsilon)\}, \forall a \in T$
  3. $\delta(q, x, y) = \emptyset$, în restul cazurilor
- $w \in L(G) \Leftrightarrow S \Rightarrow^+ w \Leftrightarrow (q, w, S) \vdash^+ (q, \epsilon, \epsilon) \Leftrightarrow w \in L_\epsilon(M)$
- M simulează derivările extrem stângi din G

# Exemplu

- $G = (\{x\}, \{a, b\}, x, \{x \to axb, x \to ab\})$
- Automatul pushdown echivalent:

$$M = (\{q\}, \{a, b\}, \{a, b, x\}, \delta, q, x, \emptyset)$$

1. $\delta(q, \epsilon, x) = \{(q, axb), (q, ab)\}$
2. $\delta(q, a, a) = \{(q, \epsilon)\}$
3. $\delta(q, b, b) = \{(q, \epsilon)\}$

# Gramatica echivalentă cu un automat pushdown

**Teorema 5**

*Pentru orice automat pushdown $M$ există o gramatică $G$ astfel încât $L(G) = L_\epsilon(M)$*

# Gramatica echivalentă cu un automat pushdown

## Teorema 5

*Pentru orice automat pushdown $M$ există o gramatică $G$ astfel încât $L(G) = L_\epsilon(M)$*

- Fie $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, \emptyset)$
- Construim $G = (N, \Sigma, S, P)$ astfel:
  - $N = \{[qzp] | p, q \in Q, z \in \Gamma\} \cup \{S\}$
  - P conţine toate regulile de forma:
    - $S \to [q_0 z_0 q]$, $\forall q \in Q$
    - dacă $(p, \epsilon) \in \delta(q, a, z)$, atunci:
      $[qzp] \to a$
    - dacă $(p, z_1 z_2 \dots z_m) \in \delta(q, a, z)$, atunci, pentru orice secvenţă de stări $q_1, \dots, q_m \in Q$ :
      $[qzq_m] \to a[pz_1 q_1][q_1 z_2 q_2] \dots [q_{m-1} z_m q_m]$
- Are loc: $[qzp] \Rightarrow^+ w \Leftrightarrow (q, w, z) \vdash^+ (p, \epsilon, \epsilon)$

# Curs 6

1 Eliminarea redenumirilor din gramatici de tip 2

2 Forma normală Chomsky

3 Problema recunoaşterii: algoritmul Cocke Younger Kasami

4 Automate pushdown

5 Legătura dintre automatele pushdown şi limbajele de tip 2

6 Automate pushdown deterministe

# Automate pushdown deterministe

## Definiţie 3

*Automatul pushdown* $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ *este determinist dacă funcţia de tranziţie* $\delta : Q \times (\Gamma \cup \{\epsilon\}) \times \Gamma \longrightarrow 2^{Q \times \Gamma^*}$ *îndeplineşte condiţiile:*

1. $|\delta(q, a, z)| = 1, \forall a \in \Sigma \cup \{\epsilon\},\ \forall q \in Q,\ \forall z \in \Gamma$
2. *Dacă* $\delta(q, \epsilon, z) \neq \emptyset$ *atunci* $\delta(q, a, z) = \emptyset, \forall a \in \Sigma$

Un automat pushdown determinist poate avea $\epsilon$-tranziţi

# Automate pushdown deterministe

## Definiţie 3

*Automatul pushdown* $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ *este determinist dacă funcţia de tranziţie* $\delta : Q \times (\Gamma \cup \{\epsilon\}) \times \Gamma \longrightarrow 2^{Q \times \Gamma^*}$ *îndeplineşte condiţiile:*

1. $|\delta(q, a, z)| = 1, \forall a \in \Sigma \cup \{\epsilon\},\ \forall q \in Q,\ \forall z \in \Gamma$
2. *Dacă* $\delta(q, \epsilon, z) \neq \emptyset$ *atunci* $\delta(q, a, z) = \emptyset, \forall a \in \Sigma$

Un automat pushdown determinist poate avea $\epsilon$-tranziţii

$M = (\{q_0, q_1, q_2\}, \{0, 1, a\}, \{0, 1, z\}, \delta, q_0, z, \{q_2\})$

1. $\delta(q_0, i, z) = \{(q_0, iz)\}, (i \in \{0, 1\})$
2. $\delta(q_0, i, j) = \{(q_0, ij)\}, (i, j \in \{0, 1\})$
3. $\delta(q_0, a, i) = \{(q_1, i)\}$
4. $\delta(q_1, i, i) = \{(q_1, \epsilon)\}$
5. $\delta(q_1, \epsilon, z) = \{(q_2, \epsilon)\}$

# $\mathcal{L}_{2DET}$ - Limbaje de tip 2 deterministe

$\mathcal{L}_{2DET} = \{L | \exists M$ automat pushdown deterministe astfel ca $L = L(M)\}$.

- Clasa $\mathcal{L}_{2DET}$ este o clasă proprie a clasei de limbaje $\mathcal{L}_2$ ($\mathcal{L}_{2DET} \subset \mathcal{L}_2$).
- $\{ww^R | w \in \{0,1\}^*\} \in \mathcal{L}_2 \setminus \mathcal{L}_{2DET}$

# $\mathcal{L}_{2DET}$ - Limbaje de tip 2 deterministe

$\mathcal{L}_{2DET} = \{L | \exists M$ automat pushdown determinist astfel ca $L = L(M)\}$.

- Clasa $\mathcal{L}_{2DET}$ este o clasă proprie a clasei de limbaje $\mathcal{L}_2$ ($\mathcal{L}_{2DET} \subset \mathcal{L}_2$).
- $\{ww^R | w \in \{0,1\}^*\} \in \mathcal{L}_2 \setminus \mathcal{L}_{2DET}$
  $M = (\{q_0, q_1, q_2\}, \{0,1\}, \{0,1,z\}, \delta, q_0, z, \{q_2\})$
  1. $\delta(q_0, i, z) = \{(q_0, iz)\}, (i \in \{0,1\})$
  2. $\delta(q_0, i, j) = \{(q_0, ij)\}, (i, j \in \{0,1\},\ i \neq j)$
  3. $\delta(q_0, i, i) = \{(q_0, ii),\ (q_1, \epsilon)\}$
  4. $\delta(q_1, i, i) = \{(q_1, \epsilon)\}$
  5. $\delta(q_1, \epsilon, z) = \{(q_2, \epsilon)\}$

# $\mathcal{L}_{2DET}$ - Limbaje de tip 2 deterministe

**Definiţie 4**

*O gramatică G este deterministă dacă:*

- *Orice regulă este de forma* $A \to a\alpha$, *unde* $a \in T$ *iar* $\alpha \in (N \cup T)^*$
- *Pentru orice* $A \in N$, *dacă* $A \to a\alpha$ , $A \to b\alpha'$ *sunt reguli, atunci* $a \neq b$

Pentru orice gramatică deterministă $G$ există un automat pushdown determinist $M$ astfel ca $L(G) = L(M)$