FUNDAMENTELE PROIECTĂRII COMPILATOAARELOR

CURS 1

Gianina Georgescu

SCOPUL CURSULUI

- Să învățați despre structura unui compilator
- Să deprindeți unele cunoștințe de limbaje formale care constituie baza realizării unui compilator
- Să dobândiți abilități care să vă permită să proiectați un compilator
- Să folosiți cunoștințele dobândite în realizarea unui compilator sau a unei mari părți din acesta

Tehnicile de bază care vor fi învățate în timpul cursului pot fi utilizate în:

- construcția compilatoarelor
- arhitectura calculatoarelor
- teoria limbajelor
- algoritmică
- software engineering
- etc.

STRUCTURA SĂPTĂMÂNALĂ A CURSULUI (FPC)

- Nr. ore/săptămână: 4 (curs = 2 ore săptămânal; laborator = 2 ore la 2 săptămâni, seminar = 2 ore la 2 săptămâni)
- Semestrul: 6 / anul III de studiu: 10 cursuri
- Forma de examinare: examen
- Credite: 5

- **EVALUARE:** 50% laborator, 50% examen
- **NOTĂ:** vor fi notate răspunsurile la exercițiile din timpul seminariilor și/sau cursurilor cu 0,1 răspunsul. Punctajul obținut din răspunsuri se va adăuga la nota obținută din examen și laborator !

STRUCTURA CURSULUI

Motivație, scurt istoric. Structura unui compilator. Exemple.
Gramatici regulate. Automate finite. Expresii regulate
Analiza lexicală. Algoritmul Thompson. Transformarea directă a expresiilor regulate ăn AFD echivalent. Despre *flex*
Gramatici independente de context. Automate push-down. Translatoare stivă
Analiza sintactică. Metodele generale top-down și bottom-up
Parsere; algoritmul CYK
Gramatici si limbaje LL(1). Mulțimile FIRST, FOLLOW. Recursivitatea la stânga. Factorizarea stângă.
Proprietăți ale gramaticilor LL(1). Parserul recursiv descendent – algoritm.

STRUCTURA CURSULUI

Parser 1-predictiv pentru gramatici LL(1) – algoritm. Demonstrarea validitatii algoritmului
Algoritmul Earley. Analiza sintactică bottom up - metoda generală. Gramatici și limbaje LR(k), definiții, proprietăți.
Parser de tip deplasare-reducere pentru gramatici LR(1) – algoritm. Demonstrarea validității algoritmului pentru gramatici LR(1).
Parser SLR(1) – algoritm. Parser LALR(1) – algoritm. Revenirea din eroare în parsere de tip LR.
Analiza semantică. Gramatici atributate
Generarea codului

BIBLIOGRAFIE

- A. Aho, M. Lam, R. Sethi, J. Ullman, *Compilers: Priciples, Techniques & Tools*, 2007, Addyson Wesley
- A. Aaby, *Compiler Construction using Flex and Bison*, 2004,
- Bruno Preiss, *Lexical Analysis and Parsing using C++*, 2004

LIMBAJE PENTRU CALCULATOARE

- Limbaj cod-mașină (nivel 0)

  - adresele, numerele, instucțiunile: scrise în binar

  - foarte greu de folosit
- Limbaje de asamblare (nivel 1)

  - mnemonici pentru instrucțiuni, reprezentări în hexazecimal, referiri la adrese, regiștri etc.
- Limbaje de programare (nivel 2)

  - sunt independente de mașină, oferă facilități de prelucrare, învățare, depanare
- Limbaje specializate (pentru domenii restrânse)

PROCESOARE DE LIMBAJE

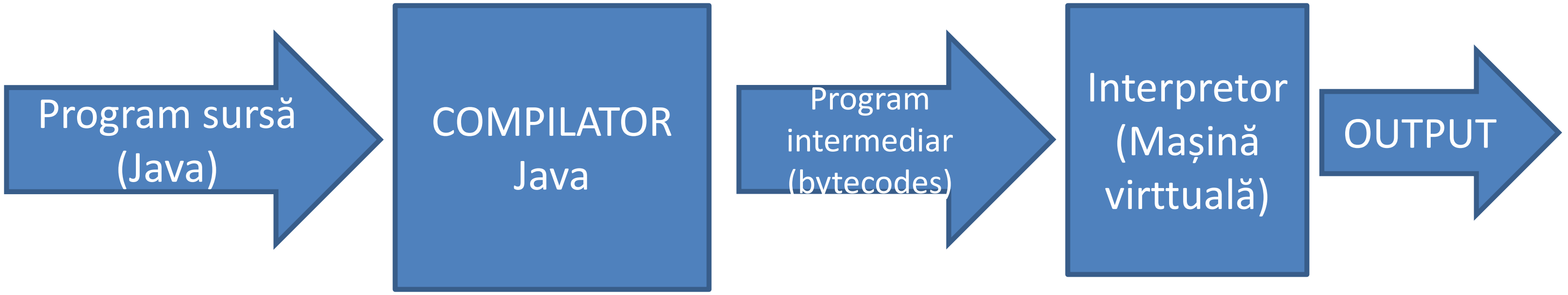
- COMPILATORUL: translatează un program scris într-un limbaj (de nivel înalt, specializat) într-o formă care poate fi executată de calculator (cod-mașină sau cod intermediar)

PROCESOARE DE LIMBAJE

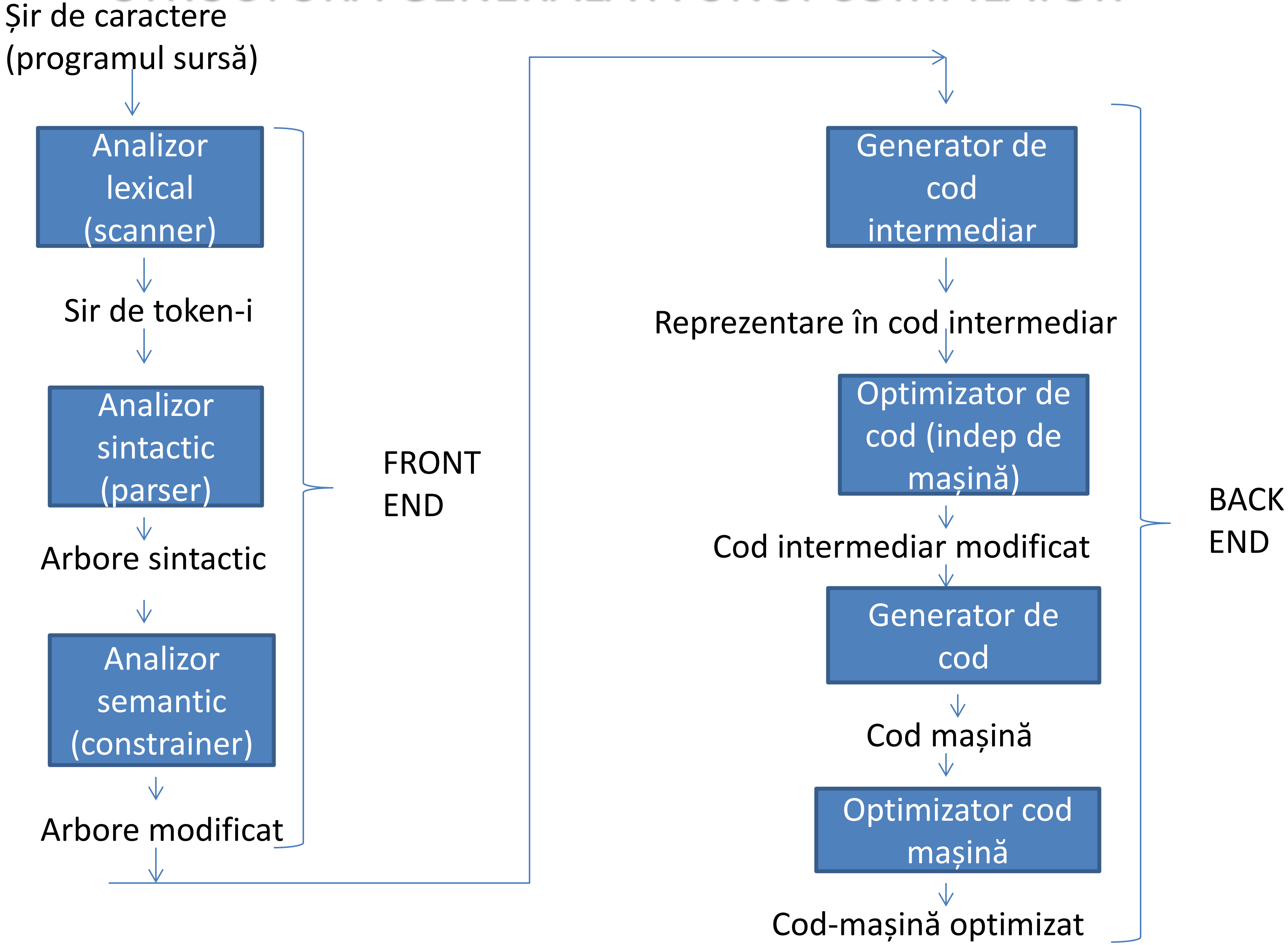
- ASAMBLORUL: translatează un program scris în limbaj de asamblare în cod-mașină
- INTERPRETORUL: nu produce un program țintă, ci execută direct instrucțiunile din programul sursă

PROCESOARE DE LIMBAJE

- COMPILATORUL HIBRID: este o combinație între un compilator și un interpretor



STRUCTURA GENERALĂ A UNUI COMPILATOR



LEGĂTURA DINTRE TEORIA COMPILĂRII ȘI LIMBAJELE FORMALE

ANALIZA LEXICALĂ → Formalizată cu ajutorul expresiilor regulate

ANALIZORUL LEXICAL (SCANNER-ul) → Implementat cu ajutorul automatelor finite

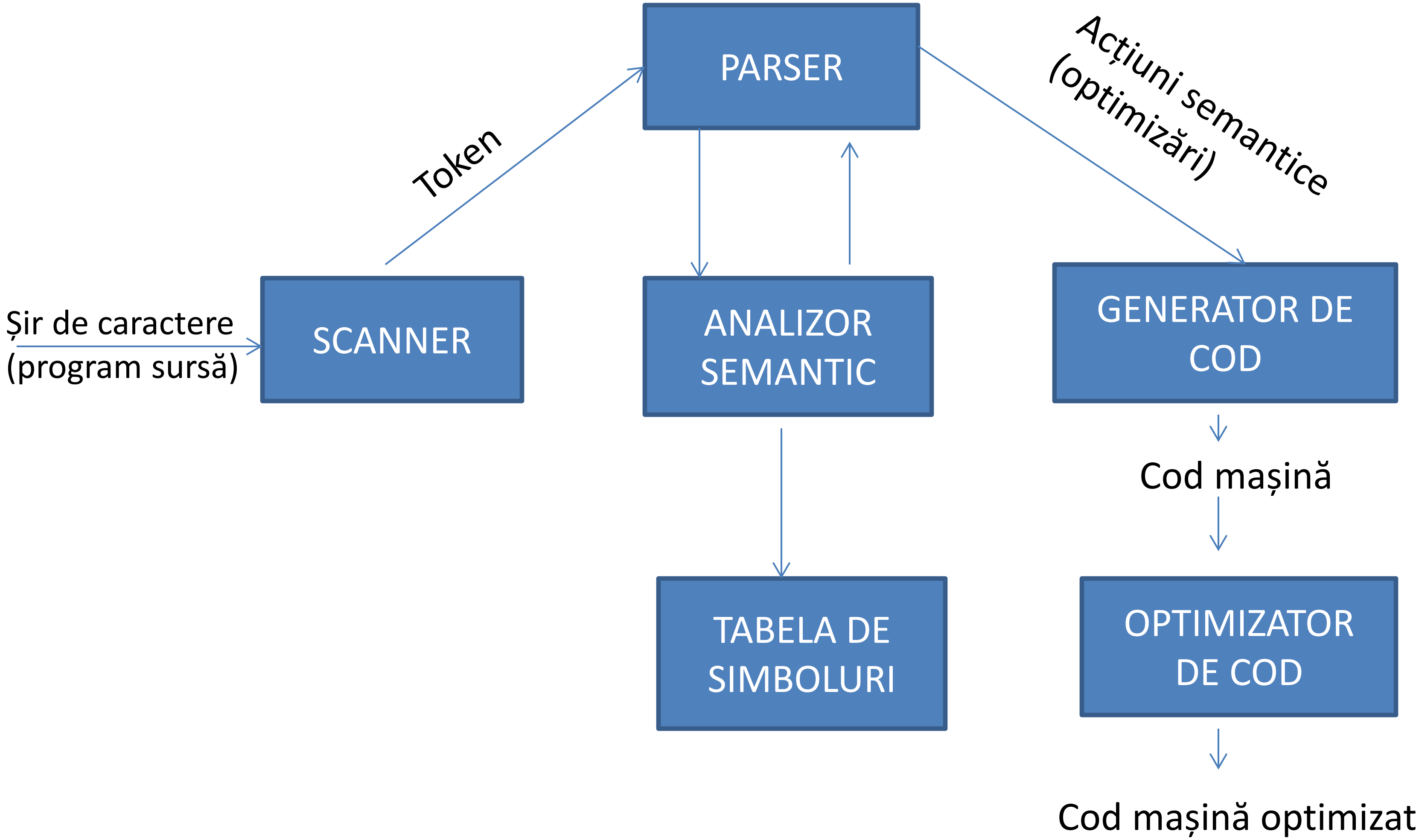
LEGĂTURA DINTRE TEORIA COMPILĂRII ȘI LIMBAJELE FORMALE

ANALIZA SINTACTICĂ → Formalizată cu ajutorul gramaticilor independente de context

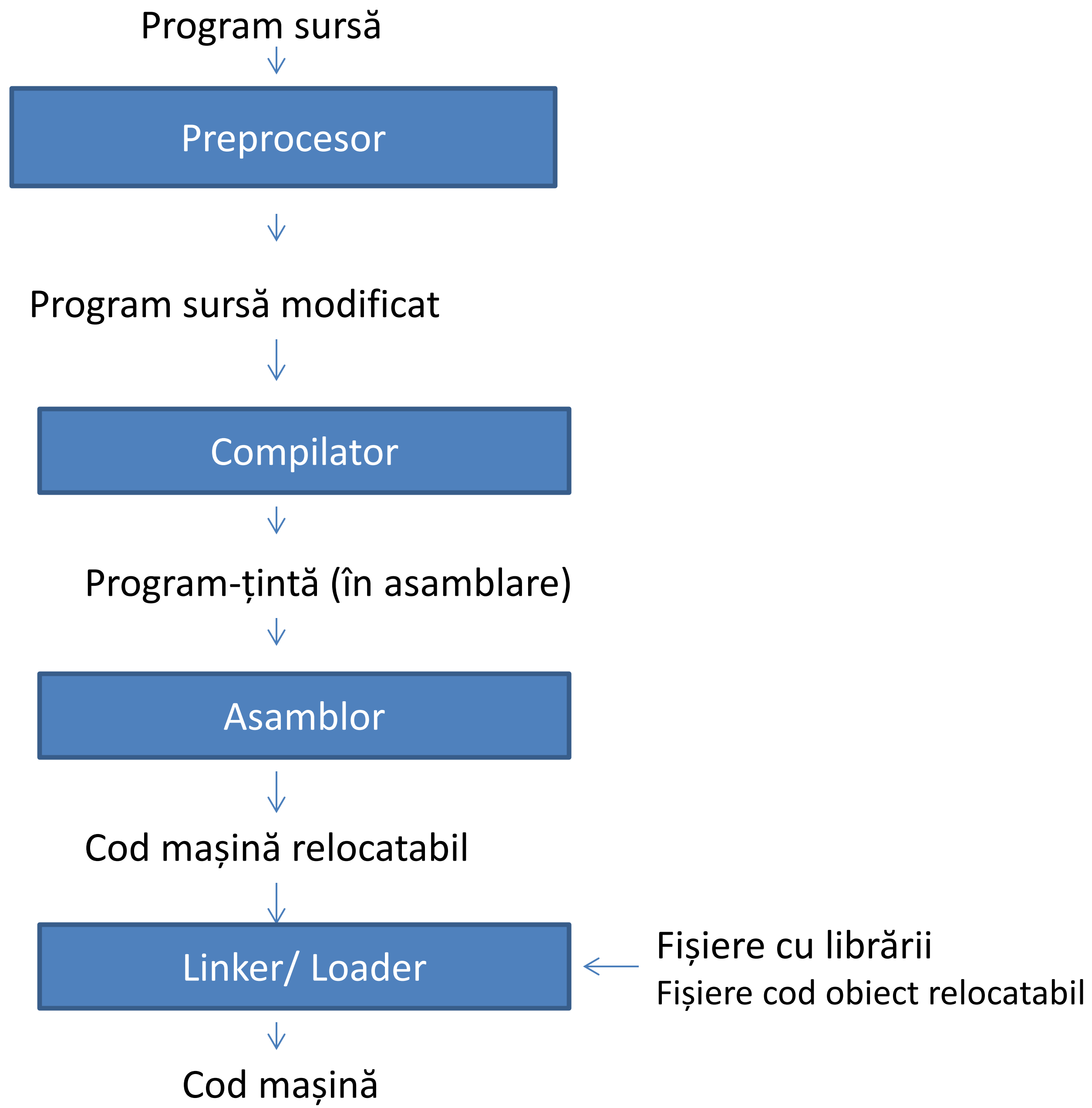
ANALIZORUL SINTACTIC (PARSER-ul) → Implementat cu ajutorul automatelor push-down (deterministe pentru parser-e de tip LL sau LR)

ANALIZA SEMANTICĂ → Formalizată cu ajutorul gramaticilor atributate

STRUCTURA DINAMICĂ A UNUI COMPILATOR



SISTEM DE PROCESARE A UNUI LIMBAJ



EXEMPLU

Fie instrucțiunea> poz=init+rata*60

poz :token <id,1>

// id - tipul token-ului;

// 1 – poziția în tabela de simboluri;

= : token <=>

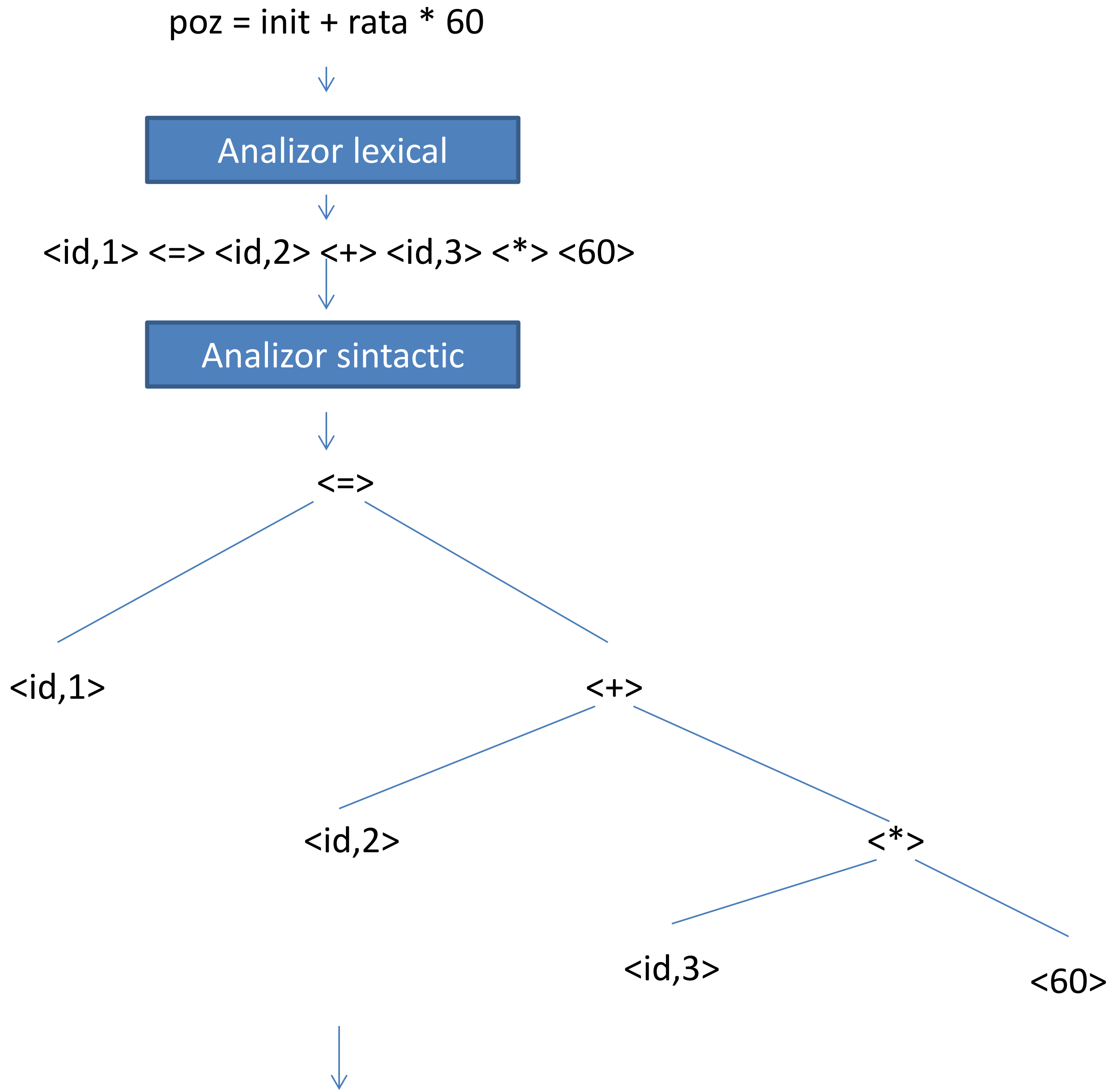
init :token <id,2>

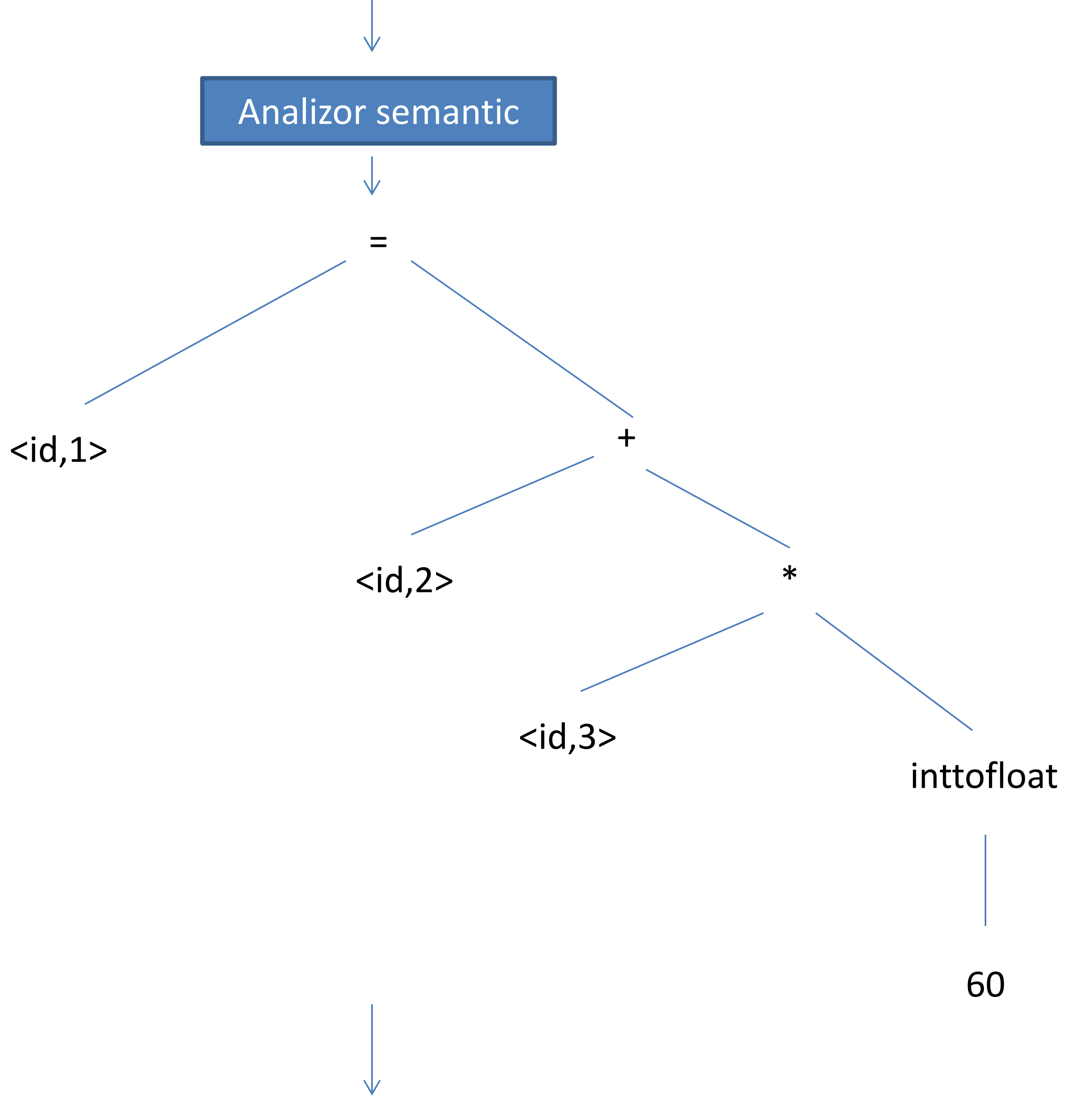
\+ : token <+>

rata:token <id,3>

* : token <*>

60 : token <60>


poz = init + rata * 60
Analizor lexical
<id,1> <=> <id,2> <+> <id,3> <*> <60>
Analizor sintactic
<=>
<id,1>
<+>
<id,2>
<*>
<id,3>
<60>

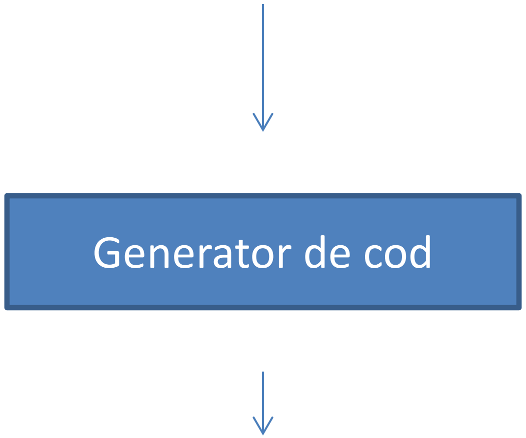

Analizor semantic
=
<id,1>
+
<id,2>
*
<id,3>
inttofloat
60

Generator cod intermediar

t1 = intofloat(60)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3

Optimizator de cod

t1 = id3 * 60.0
id1 = id2 + t1




```
LDF     R2,id3
MULF  R2,R2,#60
LDF     R1,id1
ADDF   R1,R1,R2
STF      id1,R1
```

LDF   R2, id3   - încarcă valoarea de tip float de la adresa lui id3 în registrul R2

MULF R2,R2,#60 - înmulțește ca valori de tip float numărul de la adresa conținută de R2 cu 60 și pune rezultatul la adresa R2

ADDF - adunare de numere de tip float....

STF   id1,R1   stochează la adresa lui id1 ceea ce găsești la adresa conținută de R1, ca float

ELEMENTE DE LIMBAJE FORMALE ȘIRURI ȘI ALFABETE

Un alfabet este o mulțime finită și nevidă de elemente numite litere sau simboluri.

Exemple:

$\{0,1\}$ alfabetul cifrelor binare

$\{0,1,2,3,4,5,6,7,8,9\}$ alfabetul cifrelor zecimale

$\{a, b, c, d\}$

Șiruri (cuvinte) peste un alfabet

Fie Σ un alfabet. Un șir peste Σ este orice secvență finită de elemente alăturate din Σ.

Exemplu: dacă $\Sigma = \{\mathrm{a}, \mathrm{b}\}$, atunci urmatoarele

$aabab$ este șir peste Σ de lungime 5

bab este șir peste Σ de lungime 3

Notație: vom nota șirurile peste Σ cu x, y, z

Șiruri (cuvinte) peste un alfabet

Lungimea unui șir x este egală cu numărul simbolurilor (literelor) lui x și se notează cu $|x|$. De exemplu, $|aabab| = 5$.

Există un unic șir de lungime 0 peste Σ, numit șirul nul sau șirul vid, notat cu λ sau cu ϵ.

Astfel, $|\lambda| = 0$ (respectiv $|\epsilon| = 0$).

Șiruri (cuvinte) peste un alfabet

Vom scrie a^n pentru un șir de lungime n format doar din a-uri. De exemplu, $a^5 = aaaaa$, $a^1 = a$, iar $a^0 = \lambda$. Formal, a^n este definit inductiv:

$$a^0 = \lambda$$
$$a^{n+1} = a^n a$$

Mulțimea tuturor șirurilor peste alfabetul Σ este notată cu Σ^*. De exemplu:

$$\{a, b\}^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\},$$
$$\{a\}^* = \{\lambda, a, aa, aaa, aaaa, \ldots\} = \{a^n \mid n \geq 0\}.$$

Șiruri (cuvinte) peste un alfabet

Prin convenție:

$$\emptyset^* = \{\lambda\},$$

unde $\emptyset$ este mulțimea vidă.

Observație: există diferențe între mulțimi și șiruri.
De exemplu:

- $\{a, b\} = \{b, a\}$, dar $ab \neq ba$
- $\{a, a, b\} = \{a, b\}$, dar $aab \neq ab$
- $\emptyset, \{\lambda\}, \lambda$ sunt 3 entități distincte

Operații cu șiruri

Concatenarea a două șiruri $x = a_1 a_2 \dots a_m$ și $y = b_1 b_2 \dots b_n$ este șirul notat cu xy obținut prin alăturarea literelor lui x și ale lui y în această ordine:

$$xy = a_1 a_2 \dots a_m b_1 b_2 \dots b_n$$

unde $a_1, \dots, a_m, b_1, \dots, b_n$ sunt litere peste același alfabet

Dacă $x = ab, y = bba$ sunt două șiruri peste $\{a, b\}$, atunci $xy = abbba$.

Observații:

- În general, $xy \neq yx$
- Concatenarea este asociativă: $(xy)z = x(yz)$
- Șirul vid este element neutru pentru concatenare: $\lambda x = x\lambda = x$
- $|xy| = |x| + |y|$
- $a^m a^n = a^{m+n}, \forall m, n \geq 0$
- $\hat{x} = a_m \dots a_1$ desemnează răsturnatul lui $x = a_1 \dots a_m$
- Evident, $\widehat{xy} = \hat{y}\hat{x}$.

Operații cu șiruri

Pentru un șir x vom nota cu x^n șirul obținut prin concatenarea a n copii ale lui x. De exemplu: $(aab)^4 = aabaabaabaab, (aab)^1 = aab$, $(aab)^0 = \lambda$. Formal, x^n este definit inductiv:

$$x^0 = \lambda$$
$$x^{n+1} = x^n x$$

Dacă $a \in \Sigma, x \in \Sigma^*$ notăm cu $|x|_a$ numărul aparițiilor lui a în x. Astfel, pentru $\Sigma = \{a, b, c\}$: $|abacc|_a = 2, |abacc|_b = 1, |abacc|_c = 2$

Operații cu șiruri

Un prefix al șirului x este un șir inițial al lui x, adică un șir y pentru care există șirul z astfel încât $x = yz$.

De exemplu, $abaab$ este un prefix pentry $abaababa$.

Șirul vid este prefix pentru orice șir, și fiecare șir este prefix pentru el însuși.

Un prefix y al lui x este prefix propriu pentru x dacă $y \neq \lambda, y \neq x$.

Operații cu mulțimi de șiruri. Limbaje

Fie Σ un alfabet și $M_1, M_2 \subseteq \Sigma^*$.

Reuniunea, intersecția, diferența dintre M_1, M_2 se definesc ca pentru mulțimi.

Complementara față de Σ^*a lui M_1 este:$\Sigma^* - M_1$

Concatenarea lui M_1, M_2 este definită prin

$$M_1 \cdot M_2 = M_1 M_2 = \{xy | x \in M_1, y \in M_2\}$$

Numim limbaj peste un alfabet Σ orice submulțime $L \subseteq \Sigma^*$

Operații cu limbaje

Fie Σ un alfabet și $L \subseteq \Sigma^*$ un limbaj.

Definim inductiv $L^n, n \geq 0$, astfel:

$$L^0 = \{\lambda\}$$
$$L^{n+1} = L^n L$$

Definim L^* prin:

$$L^* = \bigcup_{n \geq 0} L^n = L^0 \cup L \cup \cdots \cup L^n \cup \cdots$$

Observăm că pentru orice $L, \lambda \in L^*$

Notăm cu $L^+ = L^* - \{\lambda\}$

Exemple

1) Fie $\Sigma = \{a, b\}$ un alfabet și $L \subseteq \Sigma^*$ un limbaj.
   Pentru $L = \{aa, bba\}$ avem:

$$L^0 = \{\lambda\}$$

$$L^1 = \{aa, bba\}$$

$$L^2 = \{aaaa, aabba, bbaaa, bbabba\}$$

$$\dots$$

$$L^* = \{\lambda, aa, bba, aaaa, aabba, bbaaa, bbabba, a^6 \dots\}$$

2) Definim recursiv limbajul L al șirurilor peste $\{a, b\}$ care încep cu a și au lungime pară:

- Baza: $aa, ab \in L$
- Pasul recursiv: dacă $x \in L$, atunci $xaa, xab, xba, xbb \in L$
- Închiderea: orice șir x din L poate fi obținut plecând de la elementele de bază, aplicând de un număr finit de ori pasul recursiv

Exemple

3) Definim recursiv limbajul L al șirurilor peste $\{a, b\}$ în care fiecare apariție a lui b este imediat precedată de un simbol a. De exemplu, $\lambda, a, abaaba \in L, abb \notin L$

- Baza: $\lambda \in L$
- Pasul recursiv: dacă $x \in L$, atunci $xa, xab \in L$
- Închiderea: orice șir x din L poate fi obținut plecând de la elementul de bază, aplicând de un număr finit de ori pasul recursiv

Exerciții

Descrieți recursiv limbajele:

$$AnBn = \{a^n b^n | n \geq 0\}$$

$$Pal = \{w \in \{a, b\}^* | w \text{ este palindrom}\}$$

$$ParBal = \{w \in \{(,)\}^* | \text{ parantezele din } w \text{ sunt balansate}\}$$

Expresii regulate

Fie Σ un alfabet. Definim o expresie regulată astfel:

(i) ϕ este exp reg peste Σ care descrie limbajul vid, ϕ

(ii) λ este exp reg peste Σ care descrie limbajul $\{\lambda\}$

(iii) $\forall\ a \epsilon \Sigma$, a este expresie regulată peste Σ care descrie limbajul $\{a\}$

Fie p, q expresii regulate peste Σ care descriu respectiv limbajele P, Q. Atunci:

(iv) $p|q, pq$ (notat uneori $p \cdot q$), p^* sunt expresii regulate care descriu respectiv limbajele $P\ U\ Q, P \cdot Q = PQ, P^*$

(v) (p) este exp. reg. peste Σ care descrie limbajul P

Pentru p expresie regulată, notăm cu $L(p)$ limbajul descris de p

Operatorii utilizați de expresiile regulate

Operatorii de bază pentru expresiile regulate sunt:

- ”|” (uneori ” $+$ ”) pentru reuniune;
- ”$\cdot$ ” pentru concatenare. De cele mai multe ori punctul este omis
- ” $*$ ” pentru iterația Kleene; este operator unar
- Precedența celor 3 operatori este crescătoare de sus în jos ($*$ are prioritatea cea mai mare)
- Parantezele sunt utilizate pentru a modifica precedența operatorilor.
- În plus, pentru simplificarea scrierii expresiilor, au fost introduși mulți alți operatori (+, ?, ^, $ etc.)

PRECEDENȚA OPERATORILOR

$$(R)$$

$$R^*$$

$$R_1R_2$$

$$R_1 \mid R_2$$

Parantezele au cea mai mare precedență.

Expresia ab*c|d poate fi scrisă și ca ((a(b*))c)|d

EXEMPLUL 1 DE EXPRESIE REGULATĂ

EXPRESIE REGULATĂ PESTE {0,1} CE DESCRIE ȘIRURI CE CONȚIN 00 CA SUBȘIR; CÂTEVA ȘIRURI CARE SE POTRIVESC EXPRESIEI

(0 | 1)*00(0 | 1)*

11011100101
0000
111110111100011111

EXEMPLUL 2 DE EXPRESIE REGULATĂ

EXPRESIE REGULATĂ PESTE {0,1} CE DESCRIE ȘIRURI DE LUNGIME 4 ;
CÂTEVA ȘIRURI CARE SE POTRIVESC EXPRESIEI

(0|1)(0|1)(0|1)(0|1)

0000
1010
1111
1000

SCRIEREA SIMPLIFICATĂ A EXPRESIEI DIN EXEMPLUL 2

EXPRESIE REGULATĂ PESTE {0,1} CE DESCRIE ȘIRURI DE LUNGIME 4 ; CÂTEVA ȘIRURI CARE SE POTRIVESC EXPRESIEI

AICI SE FOLOSEȘTE OPERATORUL {}. DACA R ESTE O EXPRESIE REGULATĂ ATUNCI:

- R{2,5} inseamna 2 pănă la cel mult 5 apariții ale lui R
- R{4,} înseamnă cel puțin 4 apariții ale lui R
- R{4} înseamnă exact 4 apariții ale lui R

(0|1){4}

0000
1010
1111
1000

SCRIEREA SIMPLIFICATĂ A EXPRESIEI DIN EXEMPLUL 3

EXPRESIE REGULATĂ CE DESCRIE ȘIRURI PESTE {0,1} CU CEL MULT UN 0, CU AJUTORUL OPERATORULUI ‘?’ CÂTEVA ȘIRURI CARE SE POTRIVESC EXPRESIEI

$$1^*(0 \mid \varepsilon)1^*$$

11110111
111111
0111
0

SCRIERIEA SIMPLIFICATĂ A EXPRESIEI DIN EXEMPLUL 3

EXPRESIE REGULATĂ CE DESCRIE ȘIRURI PESTE {0,1} CU CEL MULT UN 0

SE FOLOSEȘTE OPERATORUL '?':
R? ARE SEMNIFICAȚIA: CEL MULT O APARIȚIE A EXPRESIEI R
CÂTEVA ȘIRURI CARE SE POTRIVESC EXPRESIEI

1*0?1*

11110111
111111
0111
0

EXEMPLUL 4 DE EXPRESIE REGULATĂ

EXPRESIE REGULATĂ CE DESCRIE ȘIRURI PESTE {a,@,.} CARE REPREZINTĂ ADRESE DE MAIL (a este o literă oarecare)

aa* (.aa*)* @ aa*.aa* (.aa*)*

cs143@cs.stanford.edu
first.middle.last@mail.site.org

SCRIERAREA SIMPLIFICATĂ A EXPRESIEI DIN EXEMPLUL 4

EXPRESIE REGULATĂ CE DESCRIE ȘIRURI PESTE {a, @, .} CARE REPREZINTĂ ADRESE DE MAIL (a este o literă oarecare)

SE FOLOSEȘTE OPERATORUL '+'
R^+ ARE SEMNIFICAȚIA: CEL PUȚIN O APARIȚIE A EXPRESIEI R

$a^+ \ (.aa^*)^* \ @ \ aa^*.aa^* \ (.aa^*)^*$

$a^+(.a^+)^*@a^+(.a^+)^+$

EXEMPLUL 5 DE EXPRESIE REGULATĂ

EXPRESIE REGULATĂ CE DESCRIE ȘIRURI PESTE {+,-,0,1,2,3,4,5,6,7,8,9} CARE REPREZINTĂ NUMERE PARE

(+|-)?[0123456789]*[02468]

42
+1370
-3248
-9999912

AICI OPERATOUL [] DESEMNEAZĂ O CLASĂ DE CARACTERE. [0123456789] DESEMNEAZĂ **UN** CARACTER CE POATE FI 0,1,2,3,4,5,6,7,8 SAU 9

SCRIEREA SIMPLIFICATĂ A EXPRESIEI DIN EXEMPLUL 5

EXPRESIE REGULATĂ CE DESCRIE ȘIRURI PESTE {+,-,0,1,2,3,4,5,6,7,8,9} CARE REPREZINTĂ NUMERE PARE

SE FOLOSEȘTE OPERATORUL CLASĂ DE CARACTERE CU INTERVAL. AICI [0-9] ARE SEMNIFICAȚIA **UN** CARACTER DIN INTERVALUL CUPRINS ÎNTRE CODUL ASCII AL LUI 0 ȘI CODUL ASCII AL LUI 9

(+|-)?[0-9]*[02468]

42
+1370
-3248
-9999912

ALTE EXEMPLE DE EXPRESII REGULATE

6. Limbajul șirurilor peste $\{a, b\}^*$ ce conțin un număr impar de a. Care dintre expresiile de mai jos este corectă? Explicați de ce.

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

EXEMPLE DE EXPRESII REGULATE

7. Limbajul șirurilor peste $\{a, b\}^*$ care se terminǎ în b și nu conțin aa.

$$(b\ |ab)^*(b|ab)$$

8. Limbajul șirurilor peste $\{a, b\}^*$ care conțin un număr par de b și a.

$$\big(aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ab \mid ba)\big)^*$$

Reguli algebrice pentru expresii regulate, utile pentru simplificarea exp. reg.

Fie p, q, r expresii regulate peste același alfabet.

$$p|(q|r) = (p|q)|r$$

$$p|q = q|p$$

$$p(qr) = (pq)r$$

$$\lambda p = p\lambda = p$$

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

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

$$\emptyset p = p\emptyset = \emptyset$$

$$\lambda|pp^* = p^* = \lambda|p^*p$$

Reguli algebrice pentru expresii regulate, utile pentru simplificarea exp. reg.

Fie p, q, r expresii regulate peste același alfabet.

$$(pq)^*p = p(qp)^*$$
$$(p^*q)^*p^* = (p|q)^*$$
$$p^*(qp^*)^* = (p|q)^*$$
$$(\lambda|p)^* = p^*$$
$$pp^* = p^*p$$
$$(p^*)^* = p^*$$

Notă: egalitățile de mai sus, între expresii regulate, vor fi considerate în termenii următori: limbajele descrise de cele 2 expresii între care avem semnul ”=” reprezintă unul și același limbaj

EXERCIȚII CU EXPRESII REGULATE

Găsiți o expresie regulată pentru fiecare dintre următoarele mulțimi de șiruri peste $\{a, b\}^*$.

a) Limbajul șirurilor care conțin exact 2 de a.

b) Limbajul șirurilor care conțin cel puțin 2 de a.

c) Limbajul șirurilor care nu se termină cu ab.

d) Limbajul șirurilor care încep sau se încheie cu aa sau bb.

e) Limbajul șirurilor care nu conțin subșirul aa.

f) Limbajul șirurilor care conțin un număr par de a.

g) Limbajul șirurilor care nu conțin mai mult de o apariție a lui aa. (Șirul aaa ar trebui văzut ca având 2 apariții ale lui aa.)

EXERCIȚII CU EXPRESII REGULATE

h) Limbajul șirurilor în care fiecare a este imediat urmat de bb.
i) Limbajul șirurilor care conțin atât bb cât și aba ca subșiruri.
j) Limbajul șirurilor care nu conțin aaa ca subșir.
k) Limbajul șirurilor care nu conțin subșirul bba.
l) Limbajul șirurilor care conțin atât bab cât și aba ca subșiruri.
m) Limbajul șirurilor în care numărul de a este par iar numărul de b este impar

AUTOMATE FINITE

AUTOMATE FINITE NEDETERMINISTE CU λ-tranziții

Numim automat finit nedeterminist cu λ-tranziții (AFN_λ) o structură de forma:

$A = (Q, \Sigma, \delta, s, F)$, unde:

Q mulțimea stărilor (finită, nevidă)

Σ alfabetul automatului

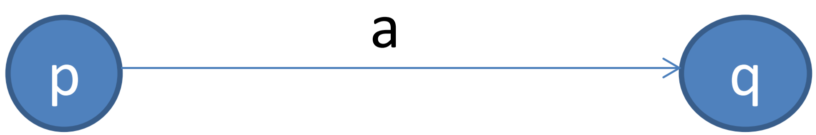
δ: $Q \times (\Sigma \cup \{\lambda\} \to 2^Q$ funcția de tranziție

$s \in Q$ starea inițială a automatului

$F \subseteq Q$ mulțimea stărilor finale

AUTOMATE FINITE NEDETERMINISTE CU λ-tranzitii

- Notație grafică pentru $q \epsilon \delta(p, a)$



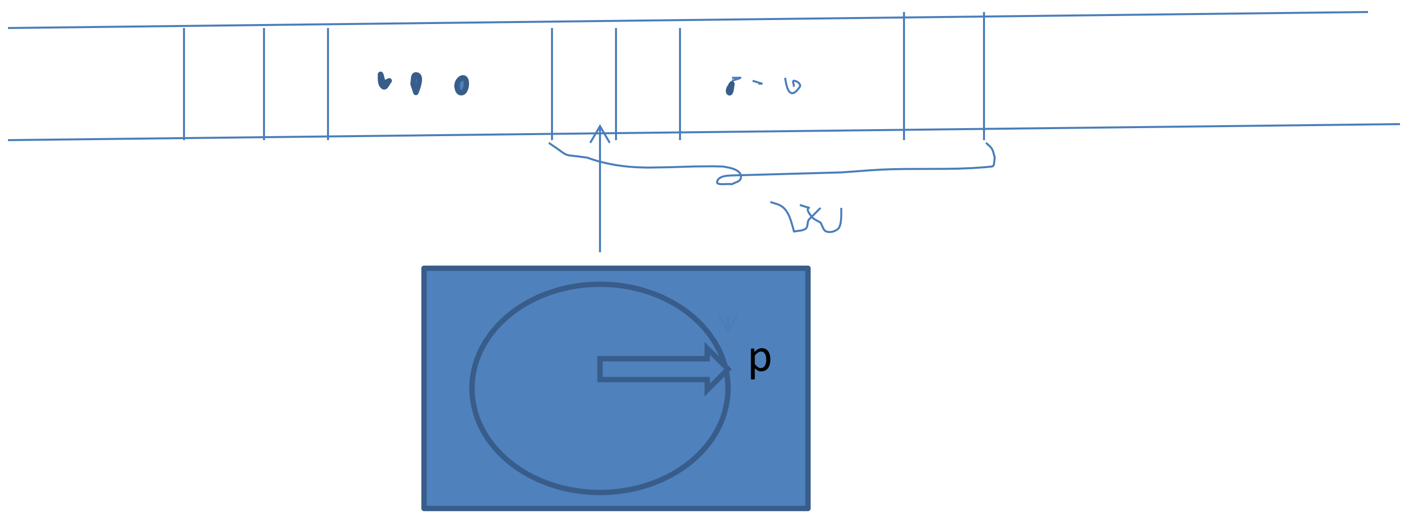
- Starea inițială a automatului o marcăm printr-un arc care intră, iar o stare finală o notăm cu un cerc dublu sau printr-un arc care iese.

AUTOMATE FINITE NEDETERMINISTE CU λ-tranzitii

Descriere instantanee (instanță a lui A):

$(p, w), p \epsilon Q$ starea curentă

$w \epsilon \Sigma^*$ șirul curent de pe banda de intrare



AUTOMATE FINITE NEDETERMINISTE CU λ-tranzitii

➢ O **<u>mișcare a lui A</u>** este definită prin:

$(p, aw) \to (q, w)$ dacă și numai dacă

$q \epsilon \delta(p, a)$, unde $p, q \epsilon Q, a \epsilon \Sigma \cup \{\lambda\}, w \epsilon \Sigma^*$

În acest caz spunem că A trece din starea p în starea q, citind din intrare simbolul a, care poate fi și λ. Aici aw reprezintă șirul aflat pe banda de intrare. După ce a efectuat mișcarea, pe banda de intrare rămâne w.

În cazul în care $a = \lambda$, atunci intrarea rămâne neschimbată, automatul doar își schimbă starea.

AUTOMATE FINITE NEDETERMINISTE CU λ-tranziții

- Notăm cu $\rightarrow^*$ închiderea reflexivă și tranzitivă a relației $\rightarrow$
- $\rightarrow^*$ înseamnă 0 sau mai multe mișcări ale automatului A
- **Limbajul recunoscut de *A*** este:
  $L(A) = \{w\epsilon\Sigma^* \mid (s, w) \rightarrow^* (q, \lambda), q\epsilon F\}$
- Spunem că două automate A_1, A_2 sunt **echivalente** dacă $L(A_1) = L(A_2)$

Exemplul 1: AFN_λ care recunoaște limbajul $\{w \in \{a,b\}^* | w$ conține ca subșir pe $abba$ sau pe $aab\}$

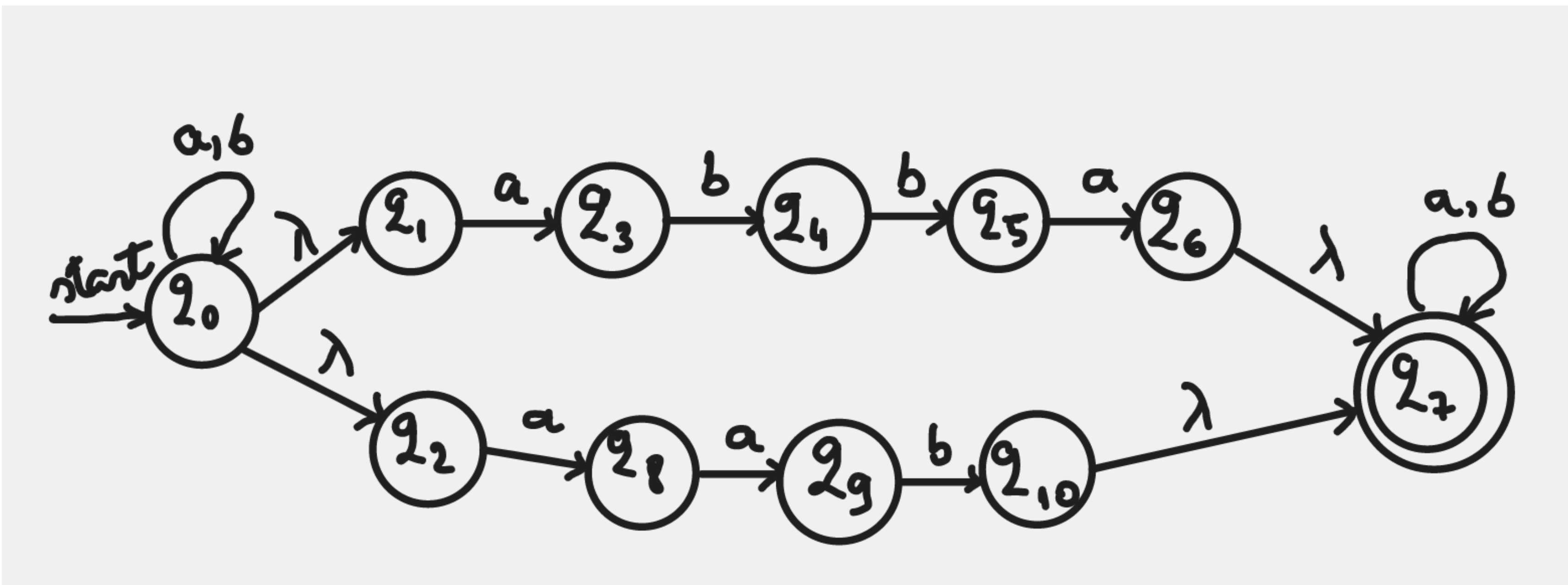


Diagrama de tranziție a stărilor automatului finit nedeterminist cu λ- tranziții din Exemplul 1. Fiecărei pereche (*stare, simbol*), $simbol \in \{a, b, \lambda\}$, îi corespunde o mulțime de stări.

	a	b	λ
q_0	$\{q_0\}$	$\{q_0\}$	$\{q_1, q_2\}$
q_1	$\{q_3\}$	$\emptyset$	$\emptyset$
q_2	$\{q_8\}$	$\emptyset$	$\emptyset$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\emptyset$	$\{q_5\}$	$\emptyset$
q_5	$\{q_6\}$	$\emptyset$	$\emptyset$
q_6	$\emptyset$	$\emptyset$	$\{q_7\}$
q_7	$\{q_7\}$	$\{q_7\}$	$\emptyset$
q_8	$\{q_9\}$	$\emptyset$	$\emptyset$
q_9	$\emptyset$	$\{q_{10}\}$	$\emptyset$
q_{10}	$\emptyset$	$\emptyset$	$\{q_7\}$

AUTOMATE FINITE NEDETERMINISTE

Definiție. Numim automat finit nedeterminist (AFN) o structură de forma:

$$A = (Q, \Sigma, \delta, s, F), \text{ unde:}$$

Q mulțimea stărilor (finită, nevidă)

Σ alfabetul automatului

$\delta: Q \times \Sigma \rightarrow 2^{Q}$ funcția de tranziție

$s \in Q$ starea inițială a automatului

$F \subseteq Q$ mulțimea stărilor finale

AUTOMATE FINITE NEDETERMINISTE

- Mișcare a lui A:

  $(p, aw) \ \rightarrow \ (q, w)$ dacă și numai dacă

  $q\epsilon\delta(p, a)$, unde $p, q\epsilon Q, a\epsilon\Sigma, w\epsilon\Sigma^*$
- Notăm cu$\rightarrow^*$ închiderea reflexivă și tranzitivă a relației $\rightarrow^*$
- Limbajul recunoscut de A este:

  $$\mathsf{L}(A) \ = \ \{w\epsilon\Sigma^* |\, (s, w) \rightarrow^* (q, \lambda), q\epsilon F\}$$

Exemplul 2: AFN care recunoaște limbajul $\{w \in \{a, b\}^* | w$ conține ca subșir pe $abba$ sau pe $aab\}$

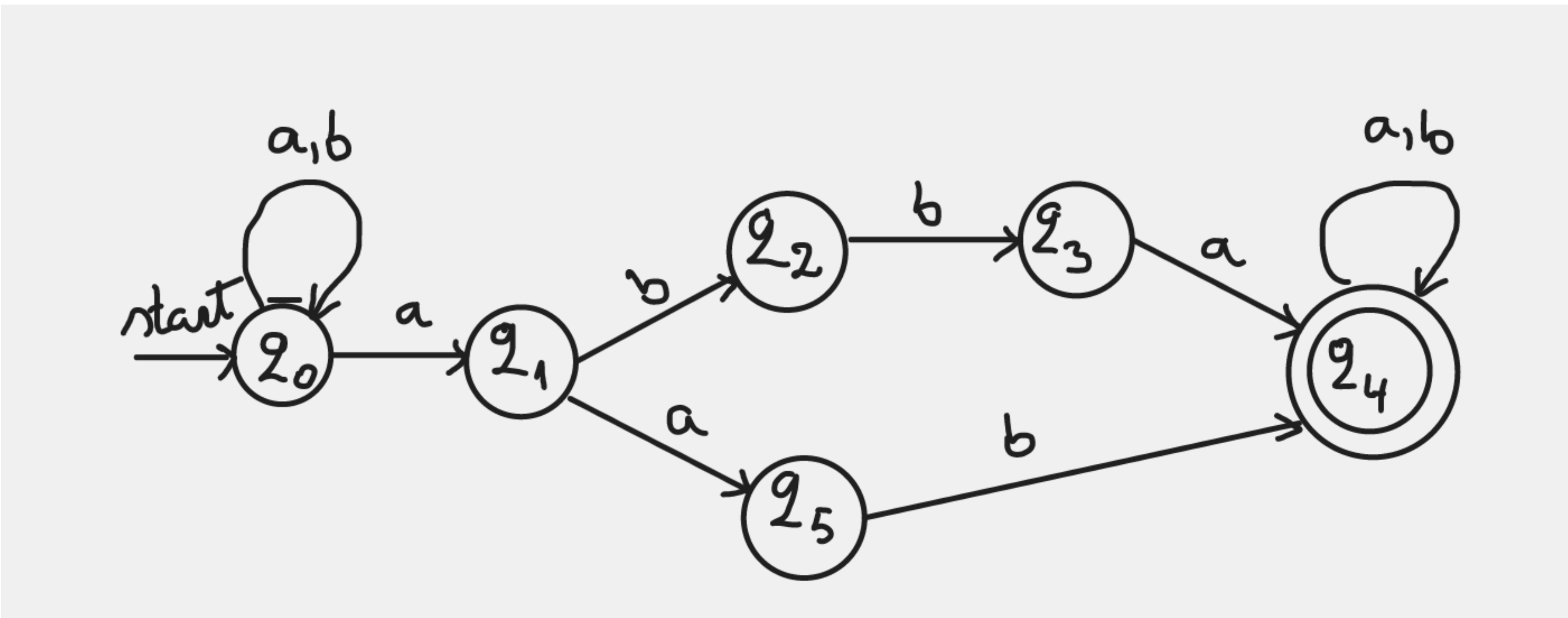


Diagrama de tranziție a stărilor automatului finit nedeterminist din Exemplul 2. Fiecărei pereche $(stare, simbol), simbol \in \{a, b\}$, îi corespunde o mulțime de stări.

	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_4\}$	$\{q_2\}$
q_2	$\emptyset$	$\{q_3\}$
q_3	$\{q_5\}$	$\emptyset$
q_4	$\emptyset$	$\{q_5\}$
$\leftarrow q_5$	$\{q_5\}$	$\{q_5\}$

AUTOMATE FINITE DETERMINISTE (AFD)

- Definiție. Numim automat finit determinist (AFD) o structură de forma:

$$A = (Q, \Sigma, \delta, s, F), \text{ unde}$$

Q mulțimea stărilor (finită, nevidă)

Σ alfabetul automatului

δ: $Q \times \Sigma \hookrightarrow Q$ parțial definită

$s \in Q$ starea inițială a automatului

$F \subseteq Q$ mulțimea stărilor finale

AUTOMATE FINITE DETERMINISTE

- Spunem că AFD A este total dacă

  δ: $Q \; x \; \Sigma \; \rightarrow \; Q$ definită total (ca funcție)
- Pe mulțimea instanțelor lui A definim:

  $(p, aw) \; \rightarrow \; (q, w)$ dacă și numai dacă

  $q = \delta(p, a)$, unde $p, q \epsilon Q, a \epsilon \Sigma, w \epsilon \Sigma^*$
- Notăm cu $\rightarrow^*$ inchiderea reflexivă și tranzitivă a relației $\rightarrow$
- **Observație**. Orice AFD poate fi completat la un AFD total echivalent prin adăugarea unei stări nefinale, q. Pentru toate stările p pentru care avem $\delta(p, a) = \emptyset$ pentru un simbol a, vom face tranziție de la p la q etichetată cu a. Din q vom face tranziții în ea însăși pentru toate simbolurile automatului. În felul acesta se obține un AFD total, echivalent cu cel inițial.

AUTOMATE FINITE DETERMINISTE

- Mișcare a lui A

  $(p, aw) \rightarrow (q, w)$ dacă și numai dacă

  $q = \delta(p, a)$, unde $p, q \epsilon Q, a \epsilon \Sigma, w \epsilon \Sigma^*$
- Notăm cu $\rightarrow^*$ închiderea reflexivă și tranzitivă a relației $\rightarrow$
- Limbajul recunoscut de A este:

  $L(A) = \{w \epsilon \Sigma^* | (s, w) \rightarrow^* (q, \lambda), q \epsilon F\}$

Exemplul 3: AFD total care recunoaște limbajul $\{w \in \{a, b\}^* | w \text{ conține ca subșir pe } abba \text{ sau } aab\}$

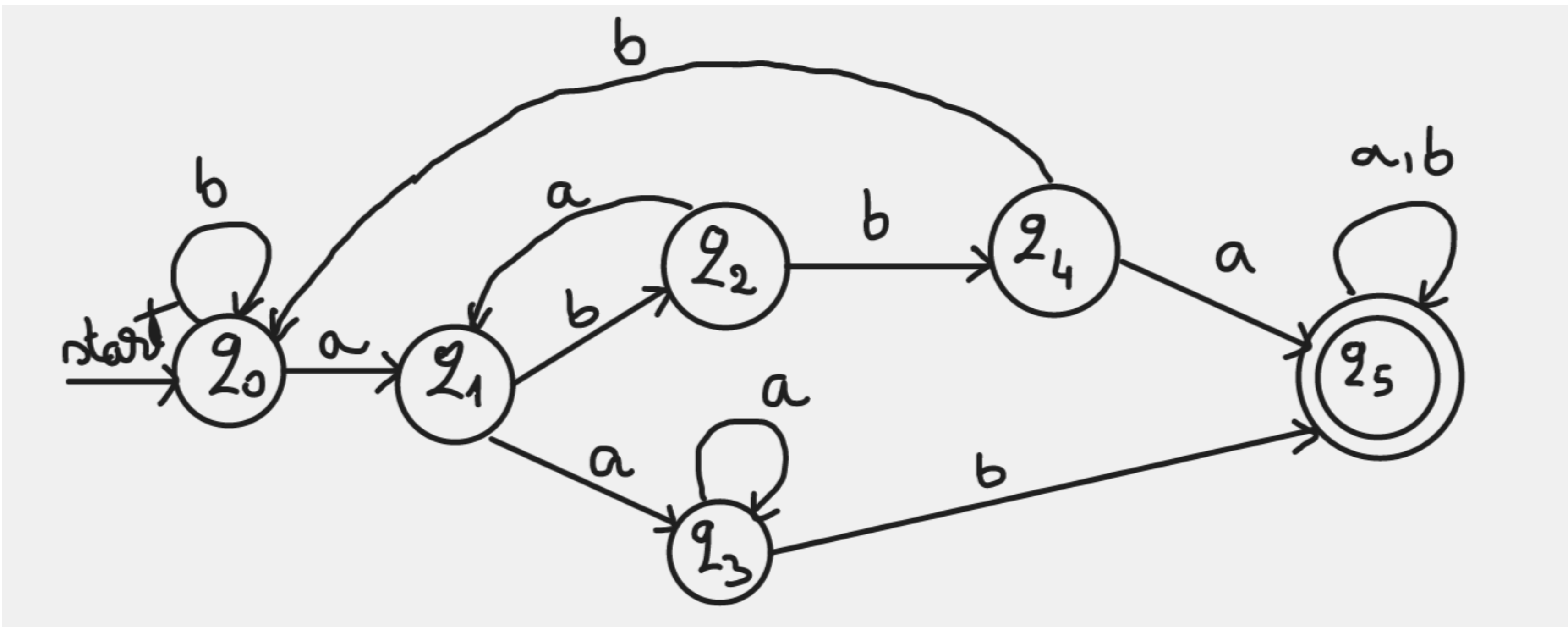


Diagrama de tranziție a stărilor automatului determinist complet din Exemplul 3. Fiecărei pereche (*stare, simbol*), $simbol \in \{a, b\}$, îi corespunde o unică stare.

	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_3	q_2
q_2	q_1	q_4
q_3	q_3	q_5
q_4	q_5	q_0
$\leftarrow q_5$	q_5	q_5

EXERCIȚII CU AUTOMATE FINITE

Găsiți un automat finit (AF) pentru fiecare dintre următoarele mulțimi de șiruri peste $\{a, b\}^*$:

a) Limbajul șirurilor care conțin exact 2 de a.
b) Limbajul șirurilor care conțin cel puțin 2 de a.
c) Limbajul șirurilor care nu se termină cu ab.
d) Limbajul șirurilor care încep sau se încheie cu aa sau bb.
e) Limbajul șirurilor care nu conțin subșirul aa.
f) Limbajul șirurilor care conțin un număr par de a.
g) Limbajul șirurilor care nu conțin mai mult de o apariție a lui aa. (Șirul aaa ar trebui văzut ca având 2 apariții ale lui aa)

EXERCIȚII CU AUTOMATE FINITE

h) Limbajul șirurilor în care fiecare a este imediat urmat de bb.

i) Limbajul șirurilor care conțin atât bb cât și aba ca subșiruri.

j) Limbajul șirurilor care nu conțin aaa ca subșir.

k) Limbajul șirurilor care nu conțin subșirul bba.

l) Limbajul șirurilor care conțin atât bab cât și aba ca subșiruri.

m) Limbajul șirurilor în care numărul de a este par iar numărul de b este impar

RELAȚIILE DINTRE FAMILIILE DE LIMBAJE RECUNOSCUTE DE DIFERITELE TIPURI DE AUTOMATE FINITE ȘI CELE DESCRISE DE EXPRESIILE REGULATE

$$L_{AFN\lambda} = L_{AFN} = L_{AFD} = L_{ExpReg}$$

- Care este deosebirea dintre un AFN și un AFD?
- Dar între un AFD și un AFD total?
- Care este relația dintre familiile recunoscute de un AFD și un AFD total? Dar între familiile recunoscute de un AFD și un AFNλ ?
- Care este deosebirea dintre un AFNλ și un AFN? Care este relația dintre familiile de limbaje recunoscute de cele 2 tipuri de automate?