Limbaje formale si translatoare - introducere

February 26, 2024

Outline

Administrativ

- Examen final - 60%: ≥ 4 pentru a se aduna laboratorul
- Laborator - 40%: ≥ 5 pt intrare in examen
  - Colocviu Lex si Yacc
  - Proiect (individual sau echipe de 2 studenti)
  - teme
- puncte suplimentare: Kahoot
- anca.marginean@cs.utcluj.ro
- Registration key Moodle - *lex&YaccX*, $X = 1..9$
  - Camelia Pintea - 1,2
  - Flaviu Cojocaru - 1
  - Adrian Burzo - 2
  - Alex Garleanu - 1
  - Stefan Popescu - 2
  - Constantin Senila - 1

- Syntax errors: Java vs python
- see BNF(BackusNaur form) for JAVA
- see AST for Javascript: tokens, AST tree

Bibliografie

- Curs introductiv - Capitolul 1. Michael Scott “Programming language pragmatics” third edition
- **I.A.Letia, E.S.Chifu “Limbaje formale si translatoare“**

- J.E. Hopcroft, R. Motwani, J.D. Ullman ”Introduction to Automata Theory, Languages, and Computation“
- A.V. Aho, M.S. Lam, R. Sethi, J.D. Ullman : ”Compilers: Principles, Techniques, and Tools“ (dragon book)
- M. Sipster: ”Introduction to the theory of computation“ (third edition)‘

Outline

```
primesTo m = sieve [2..m]
             where
             sieve (x:xs) = x : sieve (xs \\ [x,x+x..m])
             sieve [] = []
```

```
import java.util.LinkedList;
import java.util.BitSet;

public class Sieve{
    public static LinkedList<Integer> sieve(int n){
        LinkedList<Integer> primes = new LinkedList<Integer>
            ();
        BitSet nonPrimes = new BitSet(n+1);

        for (int p = 2; p <= n ; p = nonPrimes.nextClearBit(
            p+1)) {
            for (int i = p * p; i <= n; i += p)
                nonPrimes.set(i);
            primes.add(p);
        }
        return primes;
    }
}
```

Limbaje formale si translatoare

- Un limbaj e un set legal de propozitii

Limbaje formale si translatoare

- Un limbaj e un set legal de propozitii
- O propozitie e o secventa de simboluri

Limbaje formale si translatoare

- Un limbaj e un set legal de propozitii
- O propozitie e o secventa de simboluri
- Un simbol poate fi un caracter, un cuvant, un semn de punctuatie, ...

Limbaje formale si translatoare

- Un limbaj e un set legal de propozitii
- O propozitie e o secventa de simboluri
- Un simbol poate fi un caracter, un cuvant, un semn de punctuatie, ...
- Un limbaj formal este un limbaj definit de un set finit de reguli neambigue care delimiteaza propozitiile legale de cele ilegale

Limbaje formale si translatoare

- Un limbaj e un set legal de propozitii
- O propozitie e o secventa de simboluri
- Un simbol poate fi un caracter, un cuvant, un semn de punctuatie, ...
- Un limbaj formal este un limbaj definit de un set finit de reguli neambigue care delimiteaza propozitiile legale de cele ilegale

Obiective curs

- Cum se poate descrie un limbaj formal?
- Cum se poate recunoaste si prelucra un limbaj?
- Automate finite deterministe si nedeterministe, automate stiva pentru parsare
- + Elemente introductive de procesare a limbajului natural

Donalt Knuth(1938-): programming - "the art of telling another human being what one wants the computer to do"

Limbaje formale si translatoare - definire limbaj de programare si implementare

- specificam limbajul de programare folosind modele formale - gramatici si automate
- transformam aceste modele formale intr-o implementare

Implementare limbaje de programare: 3 strategii

- interpretare - $source \xrightarrow{interpret} actions/results$
- compilare - $source \xrightarrow{translates}$
  $machine\ /assembly\ language \xrightarrow{execute} actions/results$
- hibrid
  - Just-In-Time(JIT) compilers - interpreteaza parti din program, compileaza alte parti in timpul executiei
  - compilatoare care translateaza programul in alte limbaje de programare(precum C) sau limbaj intermediar (Java bytecode) pentru care exista un translator sau compilator

Observatie: strategia este specifica implementarii unui limbaj, si nu unui limbaj (exista interpretoare C si compilatoare Lisp)

Outline

Evolutia limbajelor de programare

Machine language - instructiuni in binar sau hexazecimal care controleaza direct unitatea centrala de procesare (CPU)

```
55 89 e5 53 83 ec 04 83 e4 f0 e8 31 00 00 00 89 c3 e8 2a 00
00 00 39 c3 74 10 8d b6 00 00 00 00 39 c3 7e 13 29 c3 39 c3
75 f6 89 1c 24 e8 6e 00 00 00 8b 5d fc c9 c3 29 d8 eb eb 90
```

Listing 1: gcd in hexazecimal

Limbaj de asamblare

- one-to-one correspondence: mnemonics - machine language instructions
- programare dependenta de setul de instructiuni al masinii

```
 pushl   %ebp                    jle   D
 movl    %esp, %ebp              subl   %eax, %ebx
 pushl   %ebx              B:    cmpl   %eax, %ebx
 subl    $4, %esp                jne   A
 andl    $-16, %esp        C:    movl   %ebx, (%esp)
 call    getint                  call   putint
 movl    %eax, %ebx              movl   -4(%ebp), %ebx
 call    getint                  leave
 cmpl    %eax, %ebx              ret
 je      C                 D:    subl   %ebx, %eax
A:cmpl   %eax, %ebx              jmp   B
```

Limbaj masina - limbaj asamblare

```
55 89 e5 53 83 ec 04 83 e4 f0 e8 31 00 00 00 89 c3 e8 2a 00
00 00 c9 c3 74 10 8d b6 00 00 00 00 39 c3 7e 13 29 c3 39 c3
75 f6 89 1c 24 e8 6e 00 00 00 8b 5d fc c9 c3 29 d8 eb eb 90
```

Corespondenta one-to-one *cmpl* %*eax*, %*ebx* e reprezentat de secventa 39 c3

1950 Fortran - first high-level programming language

*FOR*mula *TRAN*slator; Rapid urmat de Lisp si Algol
expresii aritmetice, If, Do, Goto

```
10    if (a .EQ. b) goto 20
      if (a .LT. b) then
          a = b - a
      else
          b = a - b
      endif
      goto 10
20  end
```

Evolutie

- COBOL(1959) - Common Business-Oriented Language; type declarations, record types, file manipulation
- LISP - McCarthy, MIT, 1958; functional: recursive
- APL (array manipulation) - IBM, 1960; imperative, matrix-centric; symbols α
- Algol, Pascal(1970), Clu, Modula, Ada - Imperative, block-structured language, formal syntax definition, structured programming
- SNOBOL, Icon - string processing languages
- 1964 - BASIC - Beginner's All-purpose Symbolic Instruction Code; programming for the masses; la Dartmouth College
  - 1975 - Altair BASIC (Bill Gates, Paul Allen) - Micro-Soft
- C(1969)- procedural, imperative
- Simula, Smalltalk, C++, Java (1991), C# - object oriented
- ML, Miranda, Haskell(1990) - functional languages with types
- sh, awk(1977), perl, tcl, python(1989), php - scripting langs
- SQL(1974 IBM) - database queries
- Prolog(1972) - logic programming language

Clasificarea limbajelor

declarative	
functional	Lisp/Scheme, ML, Haskell
dataflow	Id, Val
logic, constraint-based	Prolog, spreadsheets
template-based	XSLT
imperative	
von Neumann	C, Ada, Fortran, . . .
scripting	Perl, Python, PHP, . . .
object-oriented	Smalltalk, Eiffel, Java, . . .

Declarativ - *ce* trebuie sa faca

Imperativ - *cum* trebuie sa faca

Preluat din Michael Scott "Programming language pragmatics" third edition

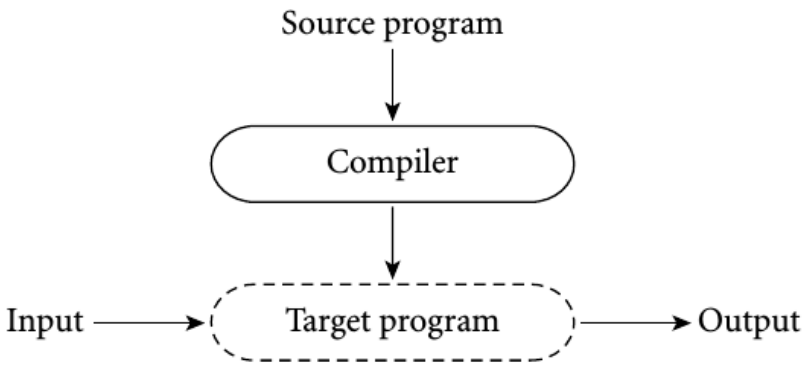
Popularitatea Limbajelor

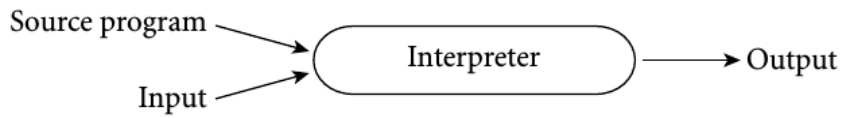
- TIOBE index - “The Importance of Being Earnest“ `https://www.tiobe.com/tiobe-index/`
- PYPL Index : The PYPL PopularitY of Programming Language Index `http://pypl.github.io/PYPL.html`

Outline

Compilator vs Interpretor

- Compilator: translatarea dintr-un limbaj de nivel inalt in limbaj asamblare sau masina
- Object code - rezultatul compilarii



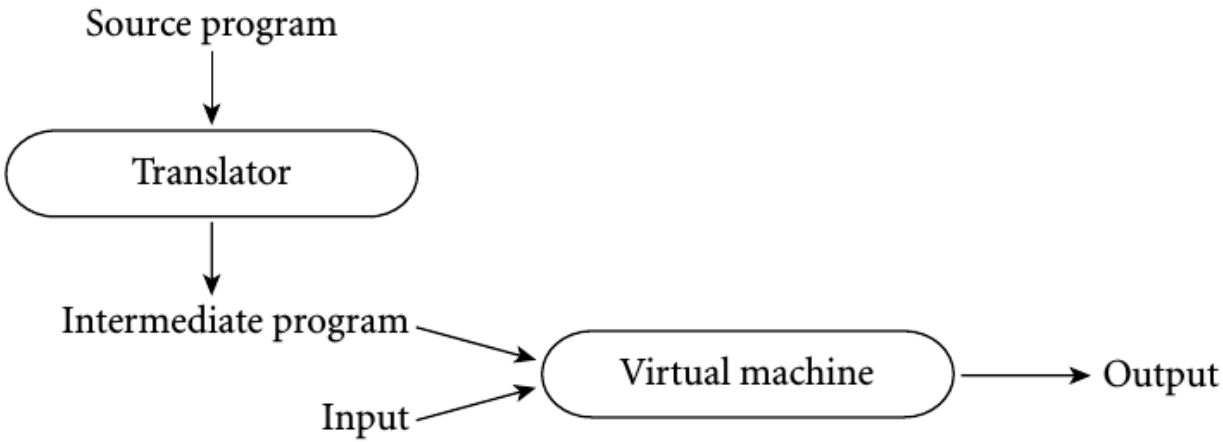


Compilatorul nu face parte din executie; interpretorul citeste instructiuni mai mult sau mai putin una cate una si le executa

Comparatie

- Interpretarea - mai mare flexibilitate si mesaje de eroare mai bune
- Compilarea - performanta mai buna (decizii la momentul compilarii)
  exemplu: GHC optimizare tail-recursive calls - apelul recursiv e ultimul statements din functie

Combinare compilare - interpretare



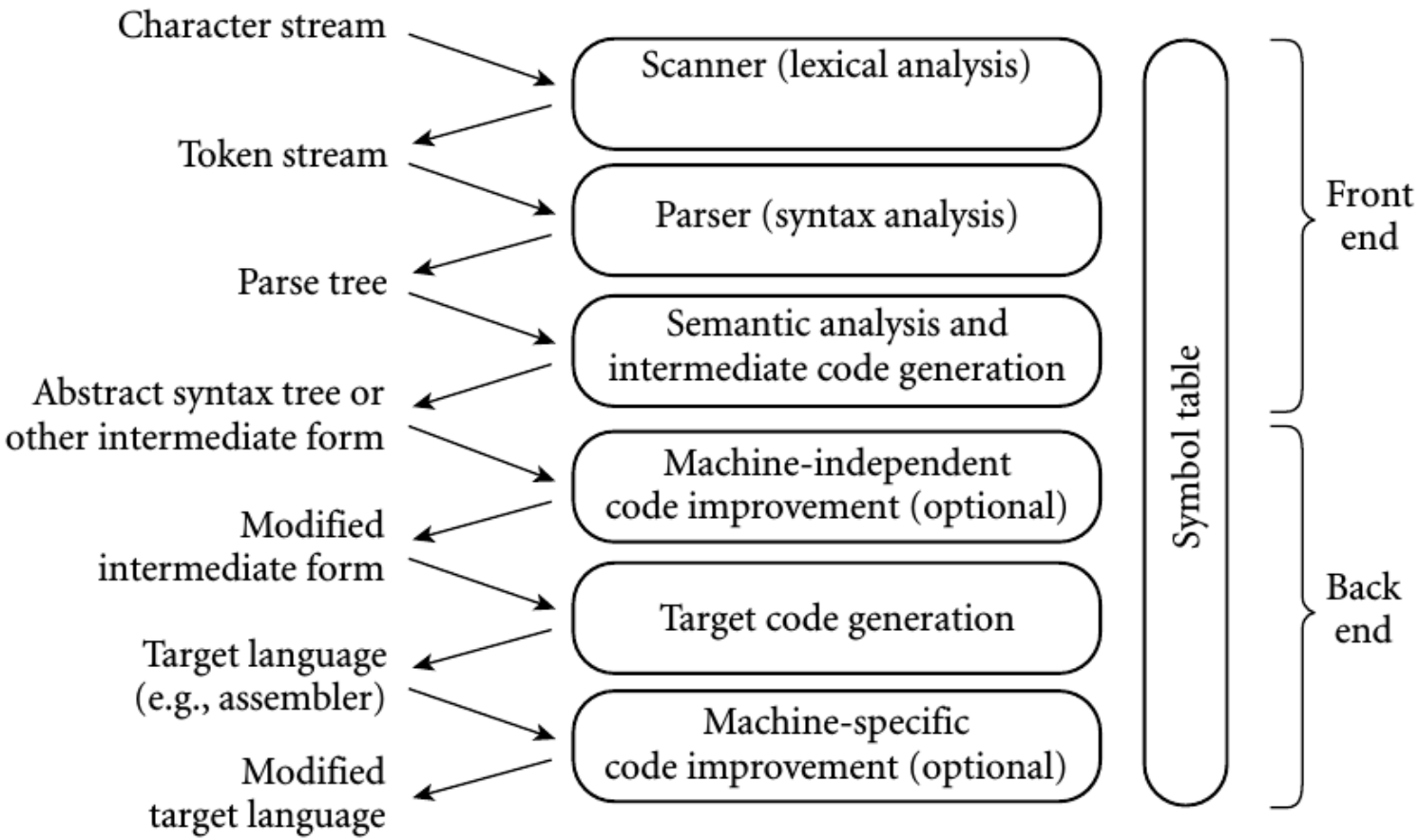
- Java bytecode - rezultatul compilarii codului sursa; executat in Java virtual machine (JVM)

Preprocesare - eliminarea comentariilor, spatiile

Observatii

Asamblor - translatare din asamblare in cod masina (initial folosind o mapare 1 la 1)
Compilator - translatare din limbaj de nivel inalt in asamblare sau cod masina; substantial mai complicat decat asamblorul (nu exista mapare 1 la 1 intre sursa si target)

Fazele unui compilator



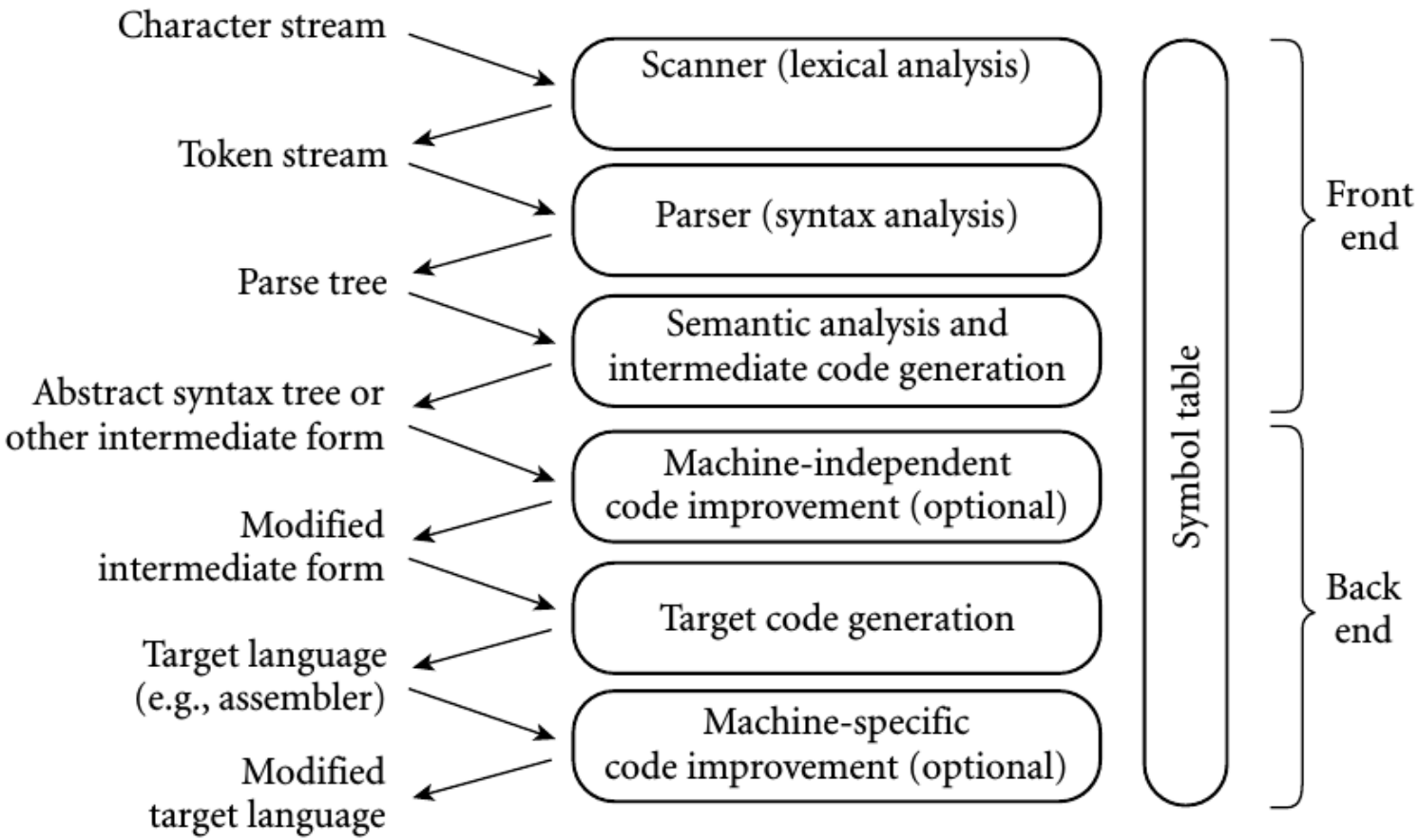
```
int main() {
  int i = getint(), j = getint();
  while (i != j) {
    if (i > j) i = i - j;
    else j = j - i;
  }
  putint(i);
}
```

Scanare

- Scanarea (analiza lexicala) - citeste caracterele si le grupeaza in token-i (cea mai mica unitate cu sens a unui program)
- elimina de obicei si comentariile si spatiile albe
- reduce dimensiunea inputului pentru parser (numarl de caractere e mult mai mare decat cel de tokeni)

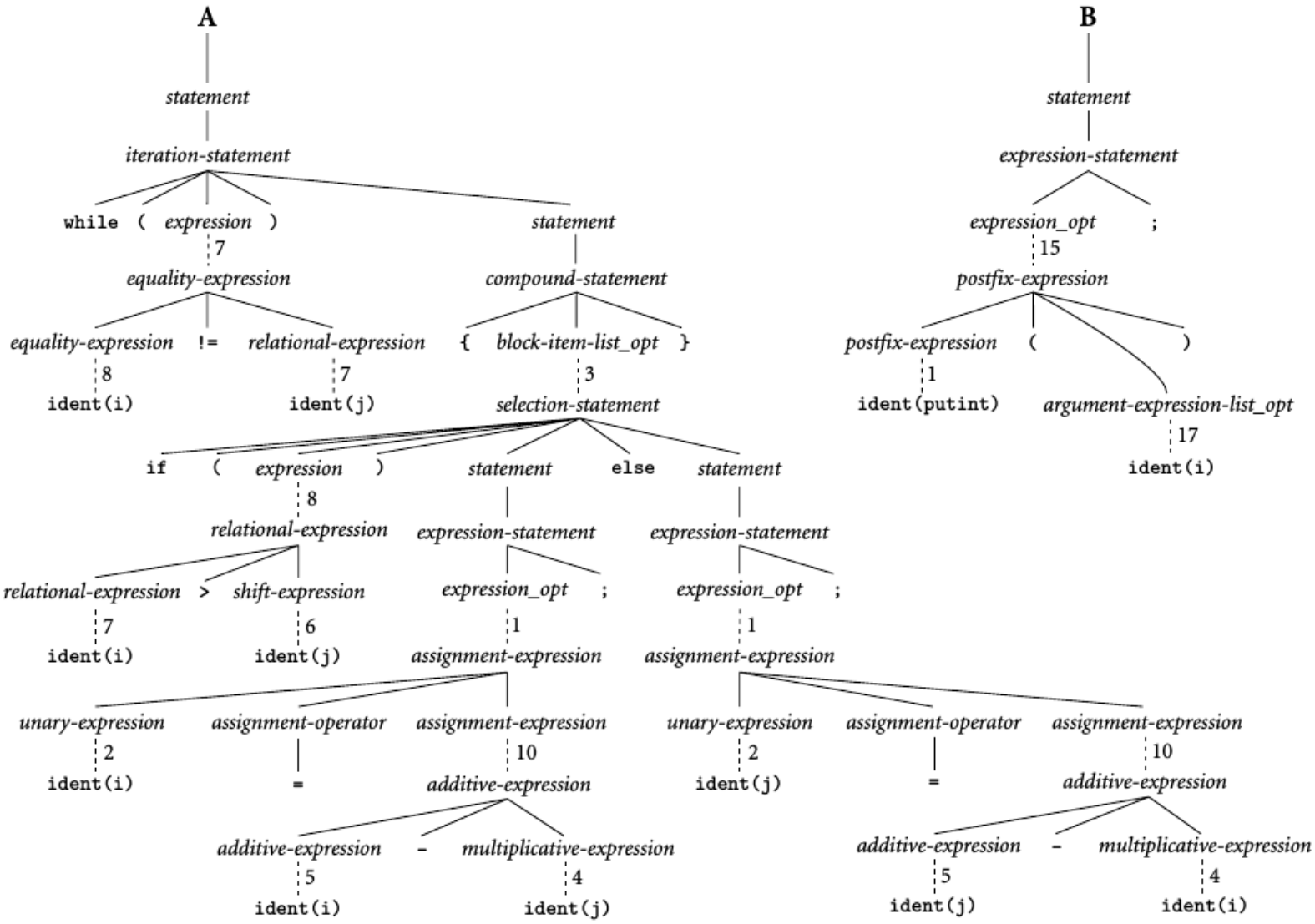
$$
\begin{array}{cccccccc}
int & main & (&) & \{ & int & i & = \\
getint & (&) & , & j & = & getint & (\\
) & ; & while & (& i & != & j &) \\
\{ & if & (& i & > & j &) & i \\
= & i & - & j & ; & else & j & = \\
j & - & i & ; & \} & putint & (& i \\
) & ; & \} & & & & &
\end{array}
$$

Fazele unui compilator - parsare



Parsare

- Parsarea - organizeaza token-ii in arbori de parsare (parse tree)
- cum formeaza tokenii un program
- analizor sintactic - derivator sau parser
- Arbore de parsare(derivare): radacina e programul, frunzele sunt token-ii de la analizorul lexical


A
statement
iteration-statement
while (expression)
7
equality-expression
equality-expression != relational-expression
8
ident(i)
7
ident(j)
statement
compound-statement
{ block-item-list_opt }
3
selection-statement
if (expression) statement else statement
8
relational-expression
relational-expression > shift-expression
7
ident(i)
6
ident(j)
expression-statement
expression_opt ;
1
assignment-expression
unary-expression assignment-operator assignment-expression
2
ident(i)
=
10
additive-expression
additive-expression - multiplicative-expression
5
ident(i)
4
ident(j)
expression-statement
expression_opt ;
1
assignment-expression
unary-expression assignment-operator assignment-expression
2
ident(j)
=
10
additive-expression
additive-expression - multiplicative-expression
5
ident(j)
4
ident(i)
B
statement
expression-statement
expression_opt ;
15
postfix-expression
postfix-expression ( )
1
ident(putint)
argument-expression-list_opt
17
ident(i)

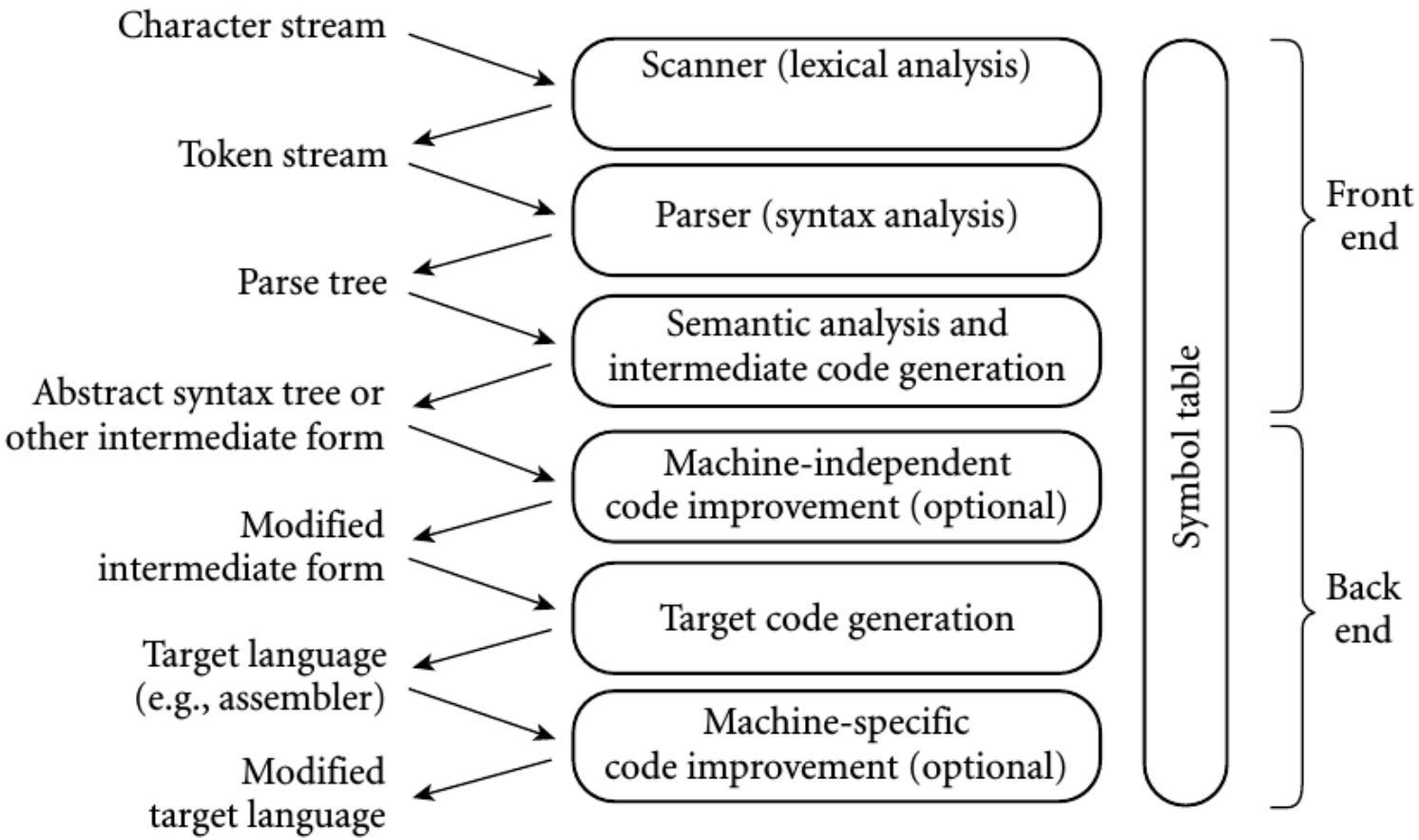
Observatii: Analizor lexical + sintactic

- Scanarea (analiza lexicala) si parsarea (analiza sintactica) - verifica daca toti token-ii sunt corecti (well formed) si secventa de token-i corespunde sintaxei limbajului

Intrebare - **cum definim limbajul**?

- Set de productii (gramatici), diagrame sintactice, BNF (Backus Naur Form)

Fazele unui compilator - analiza semantica

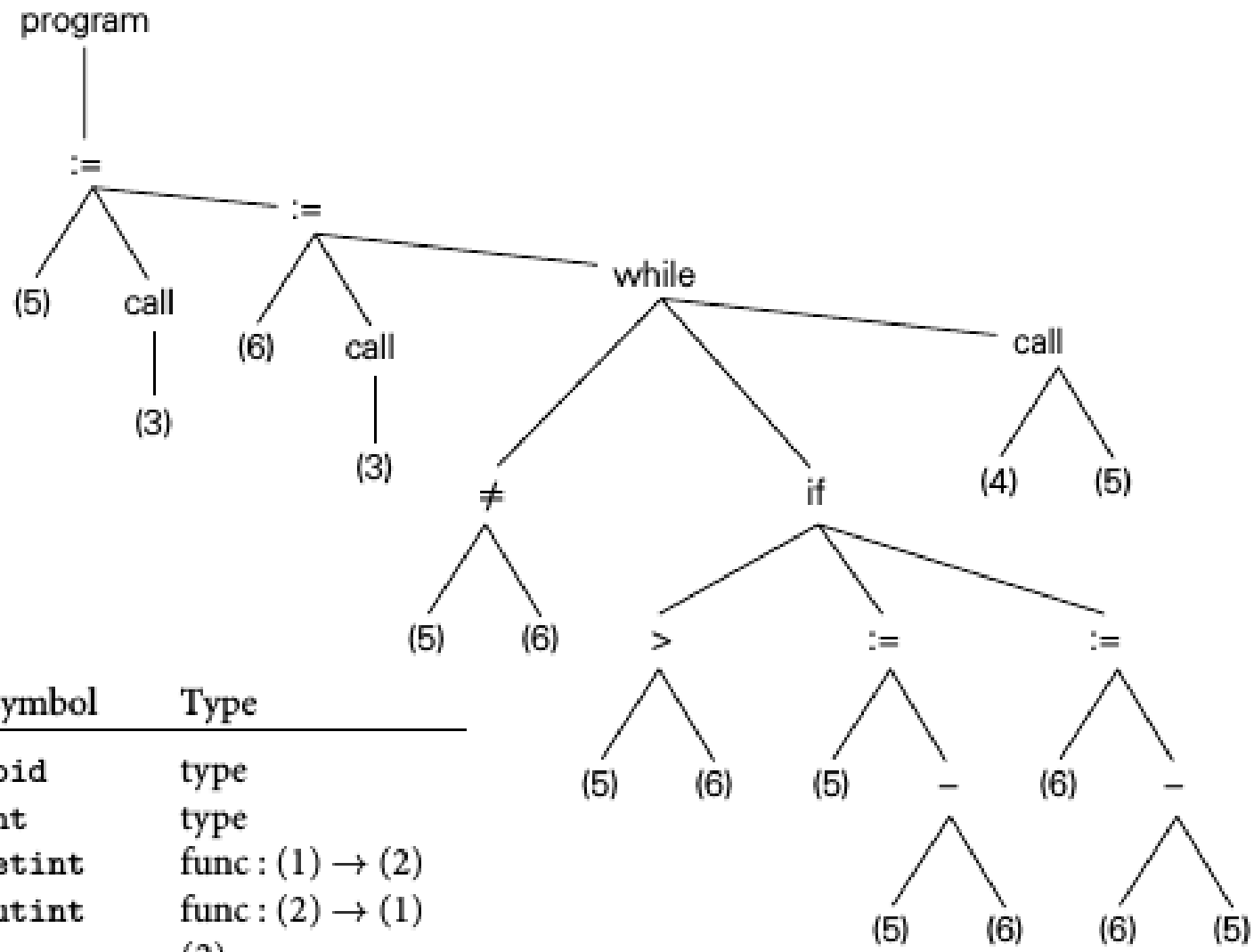


Analiza semantica si Generarea de cod intermediar

Descoperirea sensului programului

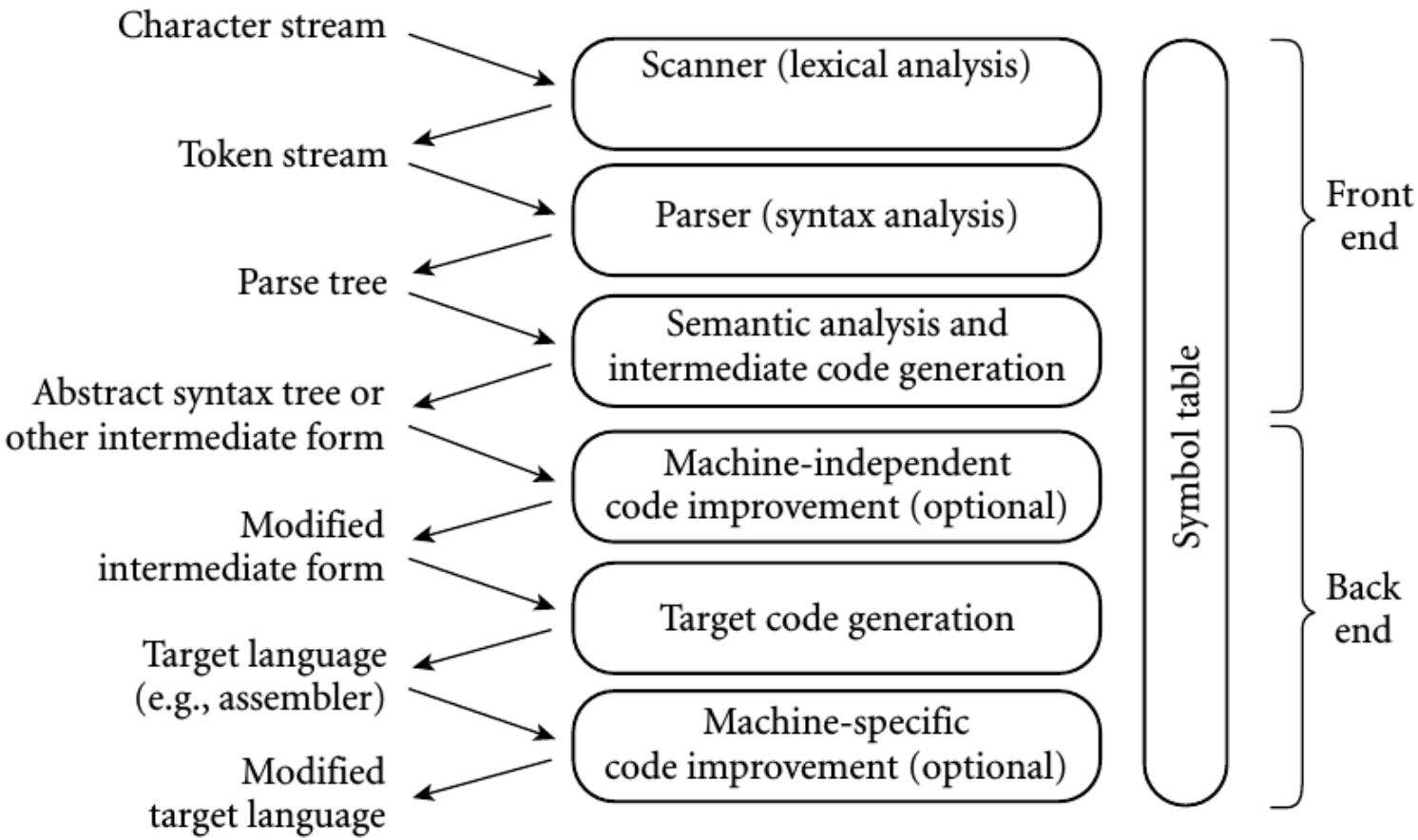
- folosirea aceluiasi identificator - aceeasi entitate; tabela de simboluri (symbol table)
- tipurile identificatorilor si ale expresiilor
- verificarea unor reguli semantice: ex - nu se aduna un string cu un intreg, procedurile sunt chemate cu nr corect de argumente (static rules)
- Observatie: exista si reguli semantice care nu pot fi verificate la momentul compilarii, ci doar la momentul rularii; ex: nu se acceseaza un element dintr-un array din afara limitelor acestuia (dynamic rules)

In multe compilatoare: actiuni la momentul identificarii unui pas particular intr-o regula gramaticala (semantic action routines) Arbore sintactic(abstract syntax tree, sau syntax tree) - mai concis decat arborele de derivare (parse tree)(uneori numit concrete syntax tree)



Index	Symbol	Type
1	`void`	type
2	`int`	type
3	`getint`	func : (1) → (2)
4	`putint`	func : (2) → (1)
5	`i`	(2)
6	`j`	(2)

Fazele unui compilator - Optimizare si generare cod



Generarea de cod target. Optimizare

Pornind de la arborele sintactic si tabela de simboluri - cod limbaj de asamblare sau masina
Optimizare cod - independent sau dependent de masina

Front-end. Back-end

Front-end - verifica daca un program este corect scris in termenii sintaxei si semanticii limbajului de programare
Back end - responsabil cu translatarea sursei in code target (asamblare/masina)

Alt exemplu

$$position = initial + rate * 60$$

1. Analiza lexicala

Alt exemplu

$$position = initial + rate * 60$$

1. Analiza lexicala

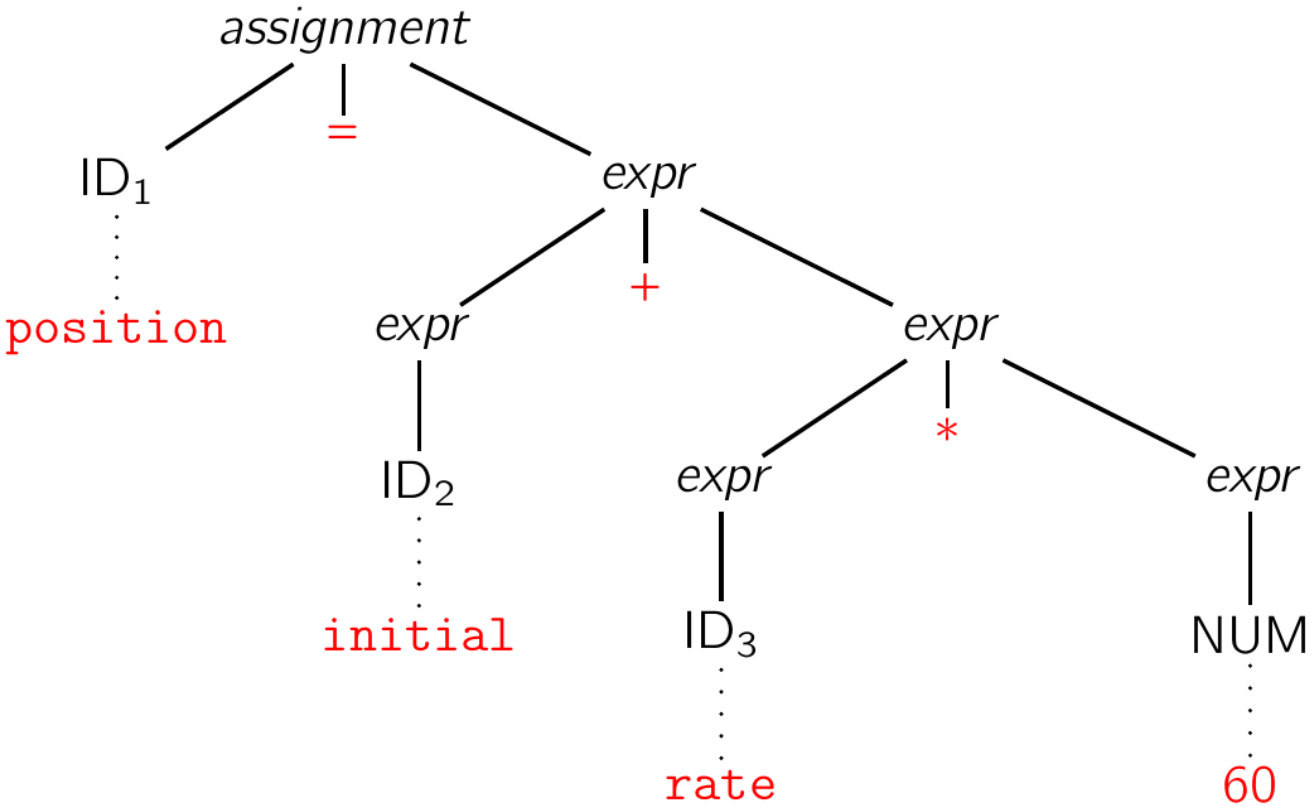
Lexeme	Tokeni
position	ID
=	=
initial	ID
+	+
rate	ID
*	*
60	NUM

2. Parsare/Analiza sintactica

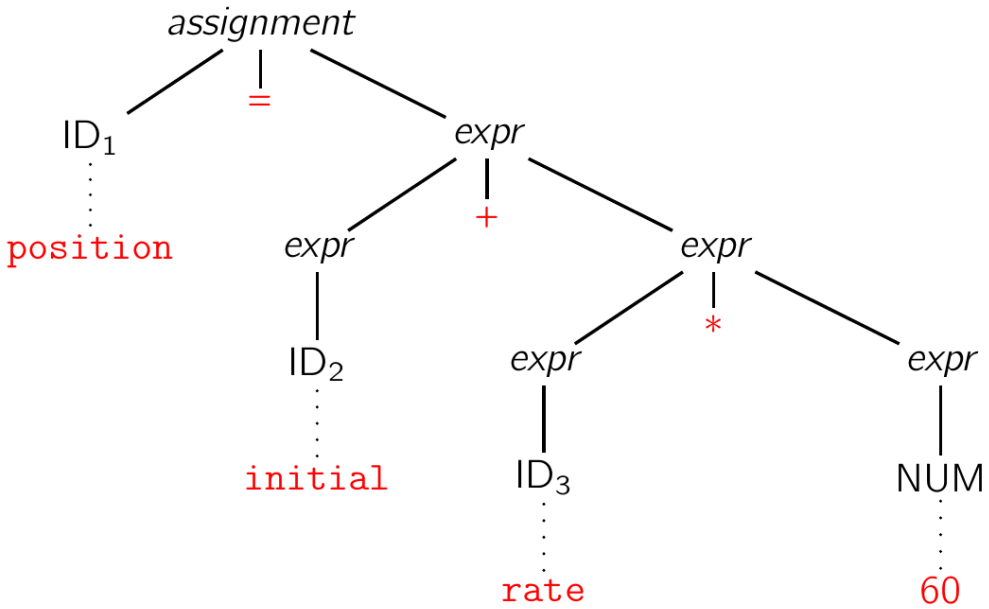
```
<assignment> -> ID "=" <expr>
<expr>       -> ID | NUM |<expr><op><expr>| (< expr >)
<op>         -> + | - | * | /
```

2. Parsare/Analiza sintactica

```
<assignment> -> ID "=" <expr>
<expr>       -> ID | NUM |<expr><op><expr>| (< expr >)
<op>         -> + | - | * | /
```



3. Analiza semantica: verificare de tipuri si alte informatii
Generare cod intermediar din arborele de parsare




```
temp1 = inttoreal(60)
temp2 = id3 * temp1
temp3 = id2 + temp2
id1 = temp3
```

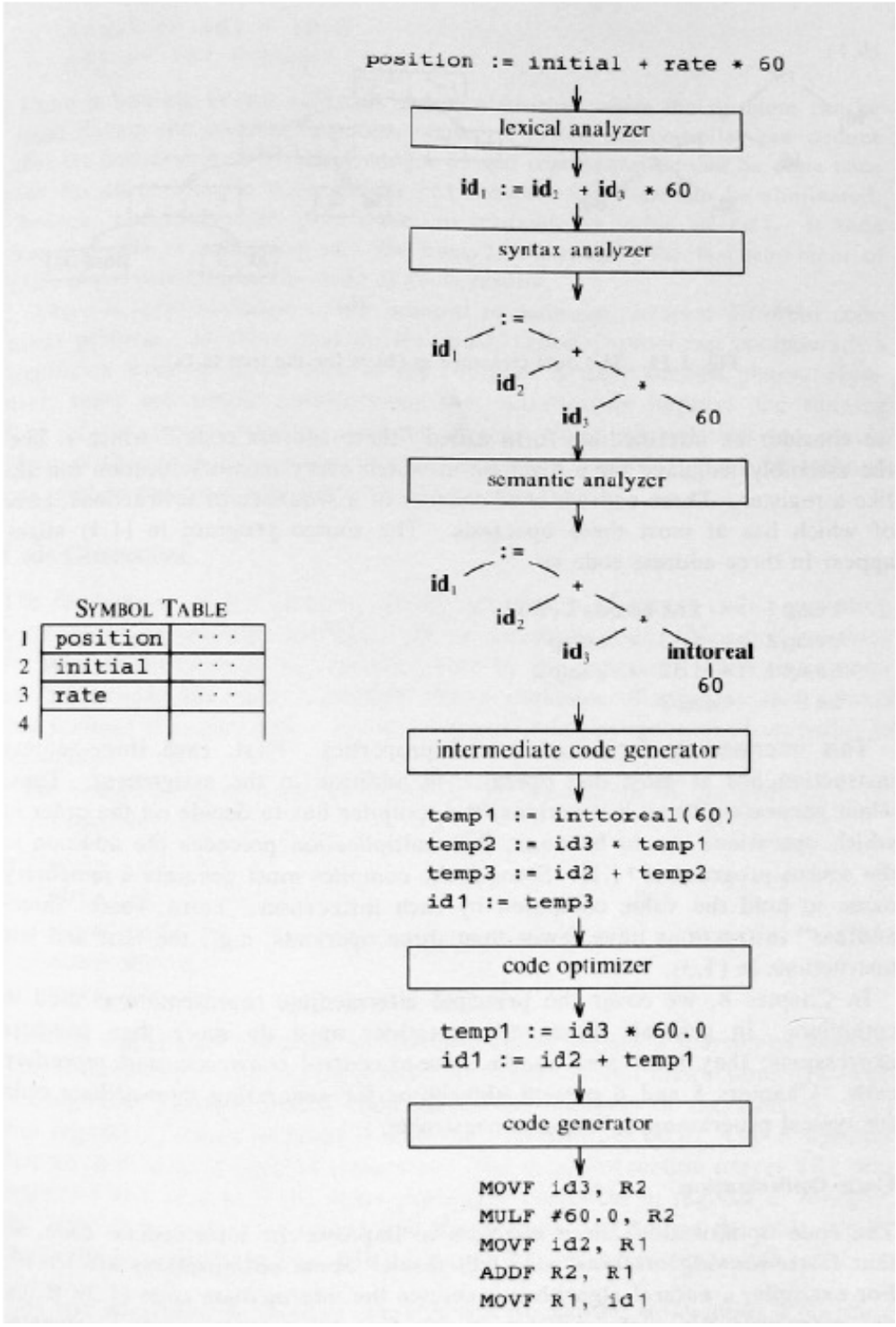
4. Exemplu de optimizare

```
temp1 = inttoreal(60)
temp2 = id3 * temp1
temp3 = id2 + temp2
id1 = temp3
```

```
temp1 = id3*60.0
id1 = id2+ temp1
```

5. Generare de cod

```
MOVF id3, R2  The F stands for floating-point instruction
MULF #60.0, R2  The # means that 60.0 is a constant
MOVF id2, R1    The first and second operand of each
    instruction
ADDF R2, R1     specify a source and a destination
MOVF R1, id1
```


position := initial + rate * 60
lexical analyzer
id1 := id2 + id3 * 60
syntax analyzer
:=
id1
+
id2
*
id3
60
semantic analyzer
:=
id1
+
id2
*
id3
inttoreal
60
intermediate code generator
temp1 := inttoreal(60)
temp2 := id3 * temp1
temp3 := id2 + temp2
id1 := temp3
code optimizer
temp1 := id3 * 60.0
id1 := id2 + temp1
code generator
MOVF id3, R2
MULF #60.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1
SYMBOL TABLE
1 position ...
2 initial ...
3 rate ...
4

Outline

Esolang - esoteric programming language

Brainfuck - 8 caractere

```
++++++++++[>+++++++>++++++++++>+++<<<-]>++.>+.+++++++
..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.
```

Listing 2: "Hello world"

LOLCODE

```
HAI 1.2
CAN HAS STDIO?
PLZ OPEN FILE "LOLCATS.TXT"?
    AWSUM THX
        VISIBLE FILE
    O NOES
        INVISIBLE "ERROR!"
KTHXBYE
```

LOLCODE

```
HAI 1.2
BTW Greets a friend
I HAS A animal
GIMMEH animal
BOTH SAEM animal AN "cat"
O RLY?
        YA RLY
                VISIBLE "Hello cat"
                VISIBLE "Nice to meet you"
        MEBBE BOTH SAEM animal AN "mouse"
                VISIBLE "Hello mouse"
                VISIBLE "Nice to eat you"
        NO WAI
                VISIBLE "Hello stranger"
OIC
KTHXBYE
```

Exemple de limbaje

- Geometric figure drawing
- Music manipulation
- Table manipulation
- Finance langauge
- A graph language
- text-based adventure game

`http://www.cs.columbia.edu/~sedwards/classes.html`

`http://www.99-bottles-of-beer.net/toplist.html`

- See the toplist
- brainfuck code, dna#,
- lisp, haskell, pyton, c++

Semantic - observatii

- Ceva poate fi sintactic corect, dar fara sens

  The rock jumped thorugh the hairy planet

- Sau ambiguu

  The chickens are ready to eat.

- Instrumente pentru reprezentare
  - Siruri de rescriere
  - Gramatici - ierarhia lui Chomsky
  - Derivari si arbori de derivare
- Gramatici regulate si automate finite
  - Automate finite
  - Diagrame de stare si expresii regulate
- Gramatici independente de context si automate stiva: Automate stiva
- Analiza sintactica descendenta:
  - LL(k)
  - eliminare recursivitate stanga
  - Factorizare stanga
  - gramatici LL(k) tari
  - Derivator LL(1) - segmente de program
- Analiza sintactica ascendenta
  - LR(k)
  - Derivator LR(0) - functia de tranzitie
  - SLR(1)
- Elemente de Procesare a limbajului natural - NLP

Rezumat

Motivatie

Evolutia limbajelor de programare

Compilare - Interpretare. Fazele unui compilator

Limbaje ezoterice

Exemplu de proiect ani anteriori

Limbaj propriu pentru desenare (autor Timotei Molcut)
Link actualizat pt 99 bottles
Fun facts

Elemente de limbaje formale. Instrumente pentru reprezentare

- Gramaticile formale, in particular gramaticile independente de context, sunt uneltele cele mai utilizate pentru a reprezenta clar structura programelor, sub forma arborilor de derivare.
- Pornind de la gramatici, se pot specifica automate care accepta programe concrete.
- Automatele pot fi mdificate pentru a genera o codificare acceptabila din arborii de derivare.

Outline

Siruri si sisteme de rescriere

- Un limbaj = set de stringuri
- Definirea formala a limbajului = raspuns formal pt "Care stringuri sunt admise de catre Limbaj?"

Siruri si sisteme de rescriere cont

Alfabet (vocabular) V - un set de simboluri
ex: `V=1,2,3,4,5,6,7,8,9,0`

String un string peste alfabetul V = secventa(sir finit) de simboluri din alfabetul V
ex: 2018
Stringul vid: ε

$$\varepsilon\chi = \chi\varepsilon = \chi$$

V^* - multimea tuturor stringurilor peste V
$V^+ = V^* \setminus \{\varepsilon\}$

Limbaj L peste V este orice subset al lui V^*

Propozitii - elementele limbajului
Numarul de propozitii dintr-un limbaj poate fi infinit

Exemple:

$$V = \{a, b, c, ...a\}; \quad L = \{cuvintele\ limbii\ engleze\}$$

$$V = \{0, 1\}; \quad L = \{\varepsilon, 01, 010, 0101, 01010, 010101, ...\}$$

Cum putem defini propozitiile unui limbaj? Ne trebuie o reprezentare formala

proces de generare

Derivare

Relatie de derivare $\Rightarrow^+$ binara, tranzitiva pe V^*

$$L = \{\chi | \zeta \Rightarrow^+ \chi,\ \zeta \textit{ un anumit sir din } V^*\}$$

Sistem formal $(V, \Rightarrow^+)$

definire derivare prin enumerare?? – NU

Productii Un sir finit de perechi (σ, τ) de siruri din V^*
Definesc relatia de derivare

Generam un string pornind de la alt string
Inchiderea tranzitiva a relatiei finite descrise de catre productii = relatia de derivare

Derivare cont.

Sistem de rescriere (V, P), V vocabular, P set finit de productii $\sigma \to \tau$, $\sigma, \tau \in V^*$

Derivare directa $\Rightarrow$ Un sir χ este derivabil direct din π: $\pi \Rightarrow \chi$ daca exista sirurile $\sigma, \tau, \mu, \nu \in V^*$ a.i.
$\sigma \to \tau \in P$,
$\pi = \mu\sigma\nu$,
$\chi = \mu\tau\nu$.

Derivare $\Rightarrow^+$ Un sir χ este derivabil din sirul π: $\pi \Rightarrow^+ \chi$ daca exista sirurile $\rho_0, ..., \rho_n \in V^*, n \geq 1$ a.i.
$\pi = \rho_0$, $\chi = \rho_n$,
$\rho_{i-1} \Rightarrow \rho_i, i = \overline{1, n}$.

Secventa $\rho_0, ..., \rho_n$ = derivare de lungime n.
χ reductibil direct la π daca χ derivabil direct din π

Expresii aritmetice: Fie sistemul de rescriere (V, P)

unde

- $V = \{+, *, (,), i, E, T, F\}$
- cu productiile P
  1. $E \to T$
  2. $E \to E + T$
  3. $T \to F$
  4. $T \to T * F$
  5. $F \to i$
  6. $F \to (E)$

Derivari cu lungimea lor:

$E \Rightarrow T$	1
$T \Rightarrow T * F$	1
$T * F \Rightarrow T * i$	1
$E \Rightarrow^* T * i$	3
$TiE \Rightarrow^* iii$	5
$E \Rightarrow i + i * i$	8

Outline

Gramatici

Definitie generativa a unui limbaj: Un cuadruplu (T, N, Z, P) este o gramatica pentru limbajul $L(G)$

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

daca

- T si N disjuncte, formeaza impreuna vocabularul
- $(T \cup N, P)$ este un sistem de rescriere
- $Z \in N$

Doua gramatici sunt echivalente daca $L(G_1) = L(G_2)$

- T - multimea terminalelor
- N - multimea nonterminalelor sau a variabilelor sintactice
- Z - nonterminal anume, simbol de start
- in limbaj - sirurile derivabile din simbolul de start si care constau doar din terminalelor

Gramatica G genereaza limbajul L

Exemplul 1

Fie gramatica $G = (\{0, 1\}, \{S\}, S, P = \{S \to 0S1, S \to \varepsilon\})$
Care dintre stringurile de mai jos $\in L(G)$?

- 01
- 010
- 1
- 01010
- ε
- 0^n1^n

Exemplul 1

Fie gramatica $G = (\{0,1\}, \{S\}, S, P = \{S \to 0S1, S \to \varepsilon\})$
Care dintre stringurile de mai jos $\in L(G)$?

- 01
- 010
- 1
- 01010
- ε
- 0^n1^n

$$S \Rightarrow^* 0^n1^n$$

$$L(G) = \{0^n1^n | n \geq 0\}$$

Exemplul 2

Fie gramatica $G = (T, N, S, P)$ unde

- $T = \{a, b\}$
- $N = \{S, A, B\}$
- cu productiile P
  1. $S \to AB$
  2. $A \to aA$
  3. $A \to \varepsilon$
  4. $B \to bB$
  5. $B \to \varepsilon$

Care stringuri apartin limbajului L(G)?

- ε
- a
- b
- $aaabb$

Exemplul 2

Fie gramatica $G = (T, N, S, P)$ unde

- $T = \{a, b\}$
- $N = \{S, A, B\}$
- cu productiile P
  1. $S \to AB$
  2. $A \to aA$
  3. $A \to \varepsilon$
  4. $B \to bB$
  5. $B \to \varepsilon$

Care stringuri apartin limbajului L(G)?

- ε
- a
- b
- $aaabb$

$$S \Rightarrow^1 AB \Rightarrow^2 aAB \Rightarrow^2 aaAB \Rightarrow^2 aaaAB$$
$$\Rightarrow^3 aaaB \Rightarrow^4 aaabB \Rightarrow^4 aaabbB \Rightarrow^5 aaabb$$

Exemplul 3

Fie gramatica $G = (T, N, S, P)$ unde

- $T = \{a\}$
- $N = \{S, N, Q, R\}$
- cu productiile P
  1. $S \to QNQ$
  2. $QN \to QR$
  3. $RN \to NNR$
  4. $RQ \to NNQ$
  5. $N \to a$
  6. $Q \to \varepsilon$

Expresii aritmetice:

$G_1 = (T, N, E, P)$

- $T = \{+, *, (,), i\}$
- $N = \{E, T, F\}$
- cu productiile P
  1. $E \to T$
  2. $E \to E + T$
  3. $T \to F$
  4. $T \to T * F$
  5. $F \to i$
  6. $F \to (E)$

derivare pt i; dar pt i*i?

Expresii aritmetice: Gramatici echivalente

Doua gramatici sunt echivalente daca $L(G_1) = L(G_2)$

$G_1 = (T, N, E, P)$

- $T = \{+, *, (,), i\}$
- $N = \{E, T, F\}$
- cu productiile P
  1. $E \to T$
  2. $E \to E + T$
  3. $T \to F$
  4. $T \to T * F$
  5. $F \to i$
  6. $F \to (E)$

$G_2 = (T, N, E, P)$

- $T = \{+, *, (,), i\}$
- $N = \{E, E', T, T', F\}$
- cu productiile P
  1. $E \to T$
  2. $E \to Te'$
  3. $E' \to +T$
  4. $E' \to +TE'$
  5. $T \to F$
  6. $T \to FT'$
  7. $T' \to *F$
  8. $T' \to *FT'$
  9. $F \to i$
  10. $F \to (E)$

Outline

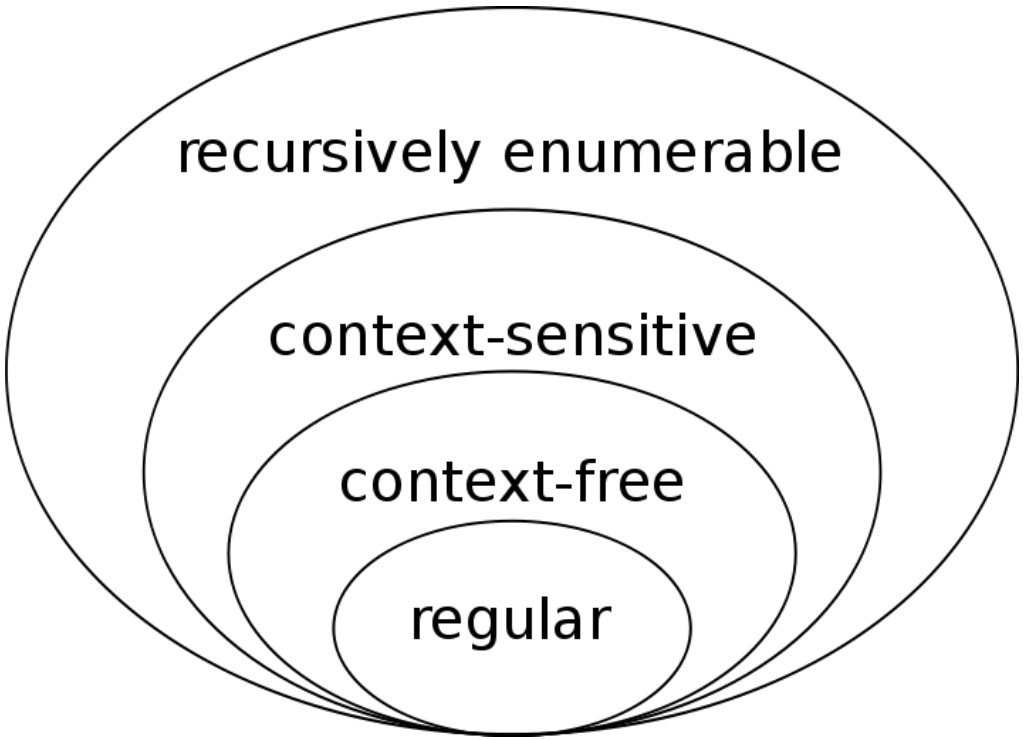
Ierarhia lui Chomsky

Conceptul de Clasificarea gramaticilor - 1950, **Noam Chomsky** - parintele lingvisticii moderne, unul dintre fondatorii stiintelor cognitive
- Modalitate de a descrie complexitatea structurala a unor propozitii particulare din limbajul natural

- Limbajele clasificate in functie de gramatica care le genereaza: constrangeri asupra productiilor gramaticii definesc diferite clase de gramatici/limbaje

Ierarhia lui Chomsky

- Gramatica de tip 0
- Gramatica de tip 1 - dependenta de context
- Gramatica de tip 2 - independenta de context
- Gramatica de tip 3 - regulata



Gramatica de Tip 0

$G = (T, N, Z, P)$

- cele mai generale gramatici
- fiecare productie are forma

$$\sigma \to \tau,\ \sigma \in V^+,\ \tau \in V^*$$

Gramatica din exemplul 3 este de Tip 0 (si nu si de tip 1,2,3)
$RN \to NNR$
$RQ \to NNQ$

Gramatica de Tip 1 - Dependente de context

$G = (T, N, Z, P)$

- fiecare productie are forma

$$\mu X \nu \to \mu \chi \nu, \quad \mu, \nu \in V^*, X \in N, \chi \in V^+$$

Context-sensitive (dependenta de context) - contextul lui X

Gramatica de Tip 1 - Dependente de context

$G = (T, N, Z, P)$

- fiecare productie are forma

$$\mu X \nu \to \mu \chi \nu, \quad \mu, \nu \in V^*, X \in N, \chi \in V^+$$

Context-sensitive (dependenta de context) - contextul lui X

Gramatica de Tip 2 - Independente de context

$G = (T, N, Z, P)$

- fiecare productie are forma

$$X \to \chi, X \in N, \chi \in V^*$$

Gramatica de Tip 2 - Independente de context

$G = (T, N, Z, P)$

- fiecare productie are forma

$$X \to \chi, X \in N, \chi \in V^*$$

- Context-free grammars - suficient de puternice pentru a descrie sintaxa limbajelor de programare
- permit construirea unor algoritmi eficienti de parsare: determina daca un string este sau nu generat din gramatica

gramatica din Exemplul 1 este Context-free grammar.

$S \to 0S1$ DA
$QN \to QR$ si $RN \to NNR$ NU

Gramatica de Tip 3 - Regulate

$G = (T, N, Z, P)$

- fiecare productie are forma

$$X \to t, \quad X \in N, \ t \in T \cup \{\varepsilon\}$$

sau

$$X \to tY, \ X, Y \in N, t \in T$$

Gramatica de Tip 3 - Regulate

$G = (T, N, Z, P)$

- fiecare productie are forma

$$X \to t, \quad X \in N, \ t \in T \cup \{\varepsilon\}$$

sau

$$X \to tY, \ X, Y \in N, t \in T$$

- Regular grammars: de obicei folosite pt a defini structura lexicala a limbajelor de programare

Ierarhia lui Chomsky - rezumat

tip 0	$\sigma \to \tau$	$\sigma \in V^+,\ \tau \in V^*$
dependenta de context	$\mu X \nu \to \mu \chi \nu$	$\mu, \nu \in V^*, X \in N, \chi \in V^+$
independenta de context	$X \to \chi$	$X \in N, \chi \in V^*$
regulata	$X \to t$,	$X \in N,\ t \in T \cup \{\varepsilon\}$ *sau*
	$X \to tY$,	$X, Y \in N, t \in T$

Ierharhia Chomsky - concluzii

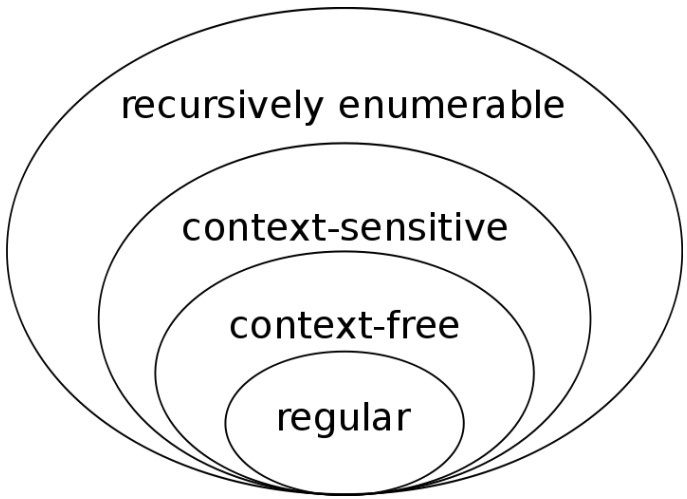
- productia $S \to \varepsilon$ productii ε
  Admise in gramaticile independente de context, regulate - limbajele se pot descrie printr=o gramatica si fara productia ε
- Every Regular Language is Context-Free, every Context-Free Language is Context-Sensitive and every Context-Sensitive Language is a Type 0 Language.
- fiecare simbol din vocabular apare in derivarea cel putin a unei propozitii; (nu exista simboluri inutile)

Ierarhia lui Chomsky - concluzii

- Gramatica de tip 0
- Gramatica de tip 1 - dependenta de context
- Gramatica de tip 2 - independenta de context
- Gramatica de tip 3 - regulata

Pt compilatoare: gramaticile regulate si independente de context

- Simboluri fundamentale ale limbajului (identificatori, constante..): gramatici regulate
- Structura programului: gramatici independente de context



Outline

Derivare si arbori - G1 - expresii aritmetice

P= $(E \to T,\ E \to E+T,\ T \to F,\ T \to T*F,\ F \to i,\ F \to (E))$

Derivari pentru i+i*i

Derivare stanga	Arbitrar	Derivare dreapta
E	E	E
$E+T$	$E+T$	$E+T$
$T+T$	$E+T*F$	$E+T*F$
$F+T$	$T+T*F$	$E+T*i$
$i+T$	$T+F*T$	$E+F*i$
$i+T*F$	$T+F*i$	$E+i*i$
$i+F*F$	$F+F*i$	$T+i*i$
$i+i*F$	$i+F*i$	$F+i*i$
$i+i*i$	$i+i*i$	$i+i*i$

Derivare si arbori - G1 - expresii aritmetice

P= $(E \to T,\ E \to E+T,\ T \to F,\ T \to T*F,\ F \to i,\ F \to (E))$

Derivari pentru i+i*i

Derivare stanga	Arbitrar	Derivare dreapta
E	E	E
$E+T$	$E+T$	$E+T$
$T+T$	$E+T*F$	$E+T*F$
$F+T$	$T+T*F$	$E+T*i$
$i+T$	$T+F*T$	$E+F*i$
$i+T*F$	$T+F*i$	$E+i*i$
$i+F*F$	$F+F*i$	$T+i*i$
$i+i*F$	$i+F*i$	$F+i*i$
$i+i*i$	$i+i*i$	$i+i*i$

La ce se refera Structura conferita de gramatica unui sir?

- ? Secventa pasilor de derivare
- ? Relatia ce arata din ce subsir este derivat un anumit nonterminal

i*i este derivat tot timpul din T

Derivare si arbori - G1 - expresii aritmetice

P= $(E \to T,\ E \to E+T,\ T \to F,\ T \to T*F,\ F \to i,\ F \to (E))$

Derivari pentru i+i*i

Derivare stanga	Arbitrar	Derivare dreapta
E	E	E
$E+T$	$E+T$	$E+T$
$T+T$	$E+T*F$	$E+T*F$
$F+T$	$T+T*F$	$E+T*i$
$i+T$	$T+F*T$	$E+F*i$
$i+T*F$	$T+F*i$	$E+i*i$
$i+F*F$	$F+F*i$	$T+i*i$
$i+i*F$	$i+F*i$	$F+i*i$
$i+i*i$	$i+i*i$	$i+i*i$

La ce se refera Structura conferita de gramatica unui sir?

- NU Secventa pasilor de derivare
- DA Relatia ce arata din ce subsir este derivat un anumit nonterminal

i*i este derivat tot timpul din T

Derivare cont.

- $T \Rightarrow^{+} i * i$ - unitate semantica: operatorul $*$ se aplica operanzilor i
- gramatica - structura semantica relevanta fiecarei propozitii din limbaj
- Daca $E \rightarrow E + T, T \rightarrow T * F$ schimbam in $E \rightarrow E * T, T \rightarrow T + F$
  multimea de siruri va fi aceeasi cu G_1, dar structura propozitiilor va fi alta: adunarile mai prioritare decat inmultirile

Derivari cont.

Fie gramatica $G = (T, N, Z, P)$.

- Sirul $\chi \in V^+$ este o fraza pentru X a lui $\mu\chi\nu$ daca si numai daca

$$Z \Rightarrow^* \mu X \nu \Rightarrow^* \mu\chi\nu$$

unde $\mu, \nu \in V^*, X \in N$

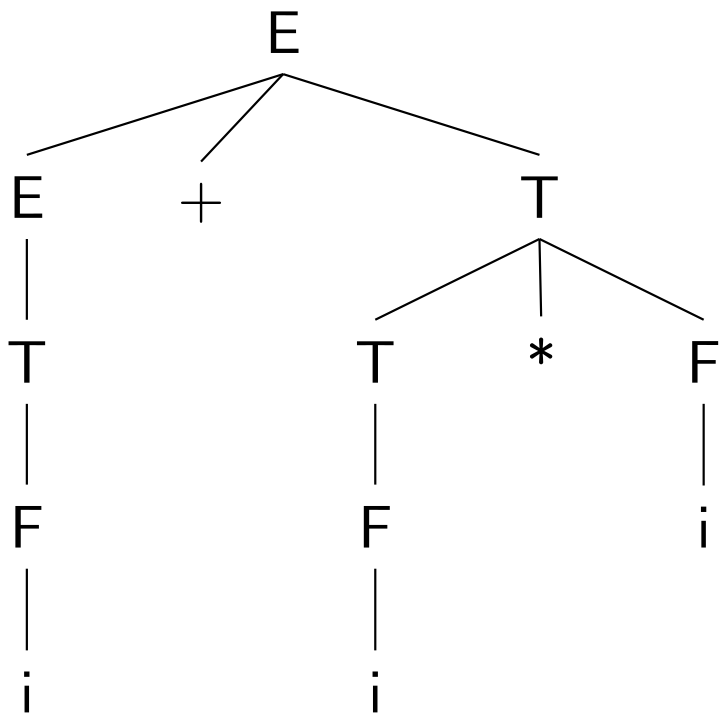
- Sirul $\chi \in V^+$ este o fraza simpla a lui $\mu\chi\nu$ daca si numai daca

$$Z \Rightarrow^* \mu X \nu \Rightarrow \mu\chi\nu$$

Subsirurile derivate din nonterminale singulare se numesc fraze.
Obs: fraza nu consta numai din terminale:
$E \Rightarrow^* E + T \Rightarrow^* E + F * F$. Deci $F * F$ derivat din T

Set de fraze - Arborele de derivare

- Toate cele trei derivari pt gramatica expresiilor aritmetice sunt echivalente: confera acelasi set de fraze.
- Arborele de derivare - reprezentarea intregului set de derivari echivalente; structura frazala



- din arbore de parsare: orice sir din orice derivare a unei propozitii - taietura = nr minim de noduri care intrerup calea de la radacina catre frunze
- exemplu: T, +, T, *, F, E, +, T

Arbori de derivare

- parse tree - metoda vizuala de a descrie orice derivare dintr-o gramatica independenta de context (Context-free grammar CFG)
- fiecare nod are un label
- radacina este simbolul de start al gramaticii
- daca un nod n, etichetat cu A are cel putin un descendent, A este in N
- daca nodurile $n_1, n_2, ...n_k$ sunt descendentii unui nod n, cu etichetele $A_1, A_2, ...A_k$ atunci

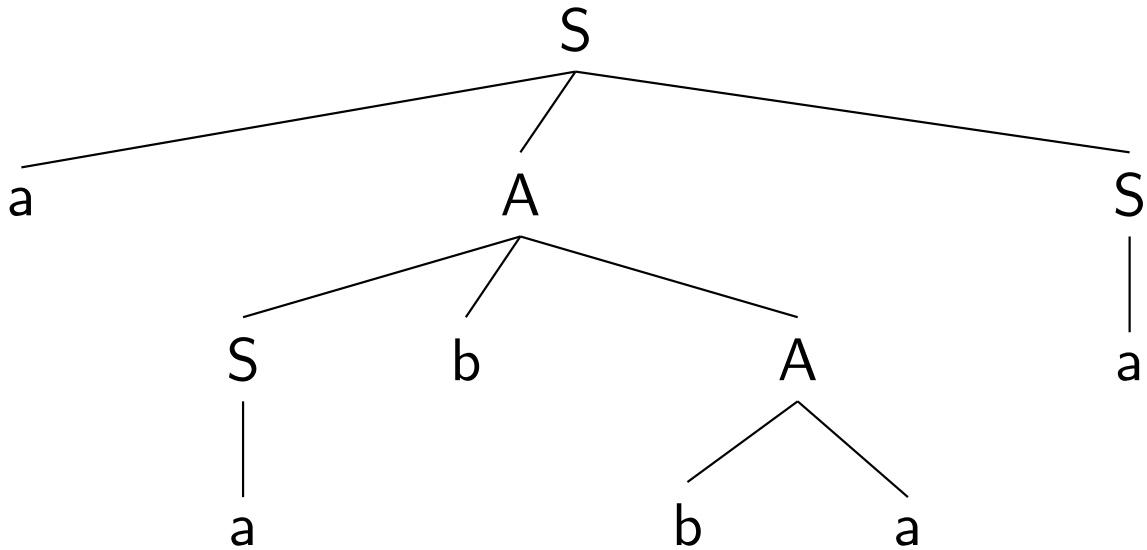
$$A \rightarrow A_1, A_2, ..., A_k$$

este o productie in P

Exemplu

Fie $G = (\{a, b\}, \{S, A\}, S, P)$

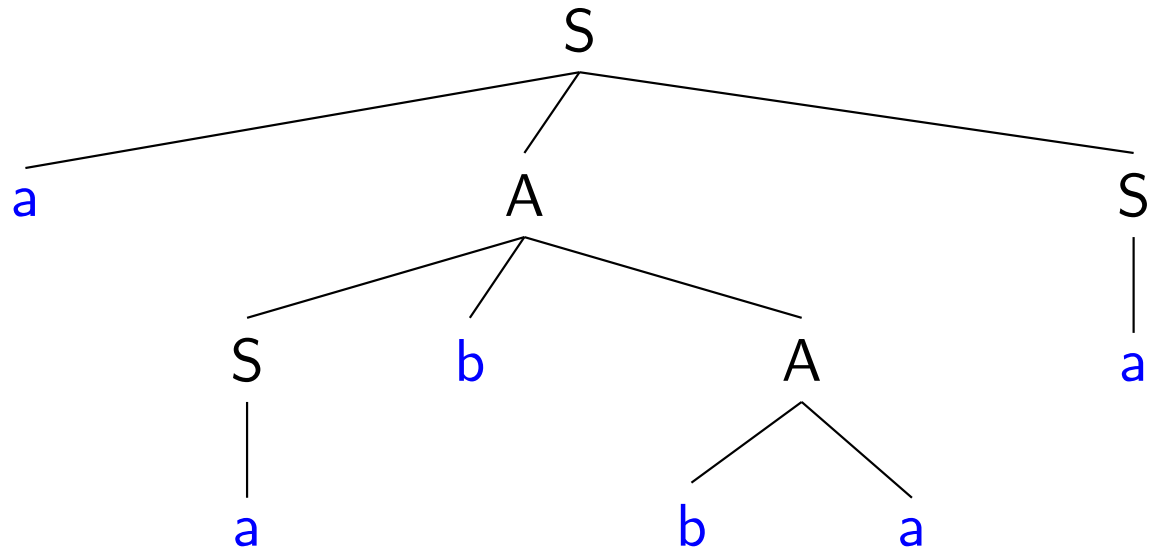
- $S \to aAS$
- $S \to a$
- $A \to SbA$
- $A \to ba$
- $A \to SS$



Exemplu

Fie $G = (\{a, b\}, \{S, A\}, S, P)$

- $S \to aAS$
- $S \to a$
- $A \to SbA$
- $A \to ba$
- $A \to SS$



$$S \Rightarrow^* aabbaa$$

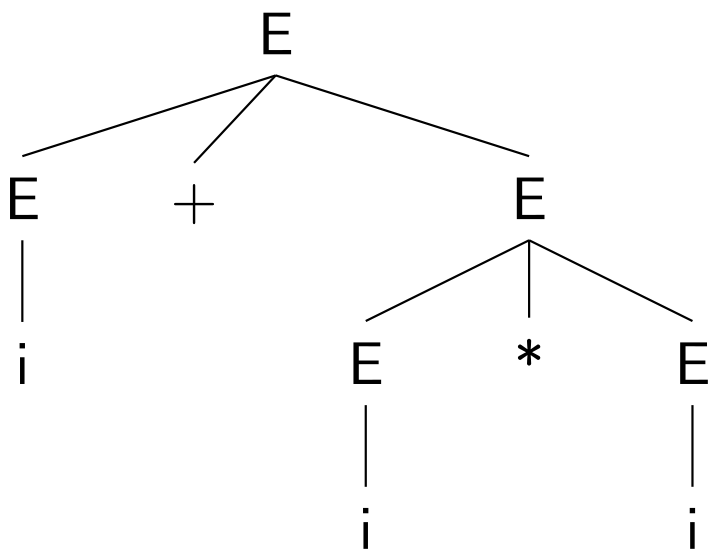
$S \to aAS \to aSbAS \to aabAS \to aabbaS \to aabbaa$

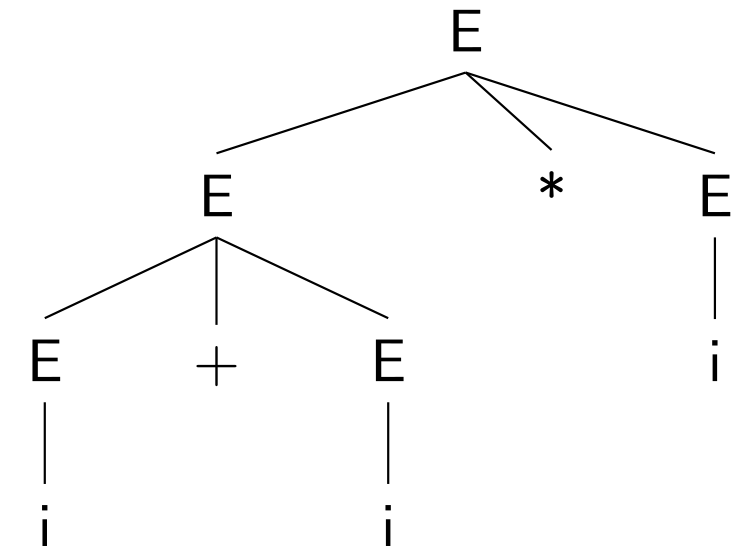
Stanga la dreapta frunzele: propozitie (rezultatul arborelui de derivare)

Ambiguitate

Fie $G_4 = (\{+, *, i\}, \{E\}, E, P)$

- $E \to E + E$
- $E \to E * E$
- $E \to i$





2+3*4??

Ambiguitate

"Look at the dog with one eye"

- O propozitie este ambigua daca derivarile sale pot fi descrise princ el putin doi arbori de derivare disctinti.
- O gramatica este ambigua dava in limbajul generat exista cel putin o propozitie ambigua
- O gramatica este ambigua daca genereaza mai mult de o derivare cea mai din stanga pentru vreo propozitie

Rezumat

Siruri si sisteme de rescriere

Gramatici

Ierarhia lui Chomsky

Arbori de derivare

Exemplu

Fie $G = (\{the, a, reads, walks, kid, robot\},$
$\{S, NounPhrase, Predicate, Article, Noun, Verb\}, S, P)$

- $S \rightarrow NounPhrase\ Predicate$
- $NounPhrase \rightarrow Article\ Noun$
- $Predicate \rightarrow Verb$
- $Article \rightarrow the$
- $Article \rightarrow a$
- $Verb \rightarrow reads$
- $Verb \rightarrow walks$
- $Noun \rightarrow kid$
- $Noun \rightarrow robot$

Gramatici regulate. Automate finite

Outline

Introducere in automate. Automate deterministe finite

- Letia & Chifu 2.2: 2.2.1
- capitolul 1.1: Finite automata, FORMAL DEFINITION OF A FINITE AUTOMATON , EXAMPLES OF FINITE AUTOMATA, FORMAL DEFINITION OF COMPUTATION "Introduction to the Theory of computation" 3rd edition, Michael Sipser
- Introduction to Automata Theory, Languages, and Computation sections 2.1, 2.2, Ullman

Automate finite

- modele pentru calculatoare cu extrem de putina memorie
- colectie finita de stari cu reguli de tranzitie care determina trecerea dintr-o stare in alta

Reprezentarea FA - State diagram

- noduri
- arce - indica tranzitia starilor
- etichete (labels) pe arce care definesc ce cauzeaza tranzitia

Exemplu: Recunoasterea cuvintelor care se termina in ".ing"

ingest, reading

Automat $\rightarrow$ Cod

1. citeste urmatorul input
2. decide starea urmatoare
3. sari la inceputul codului pentru acea stare

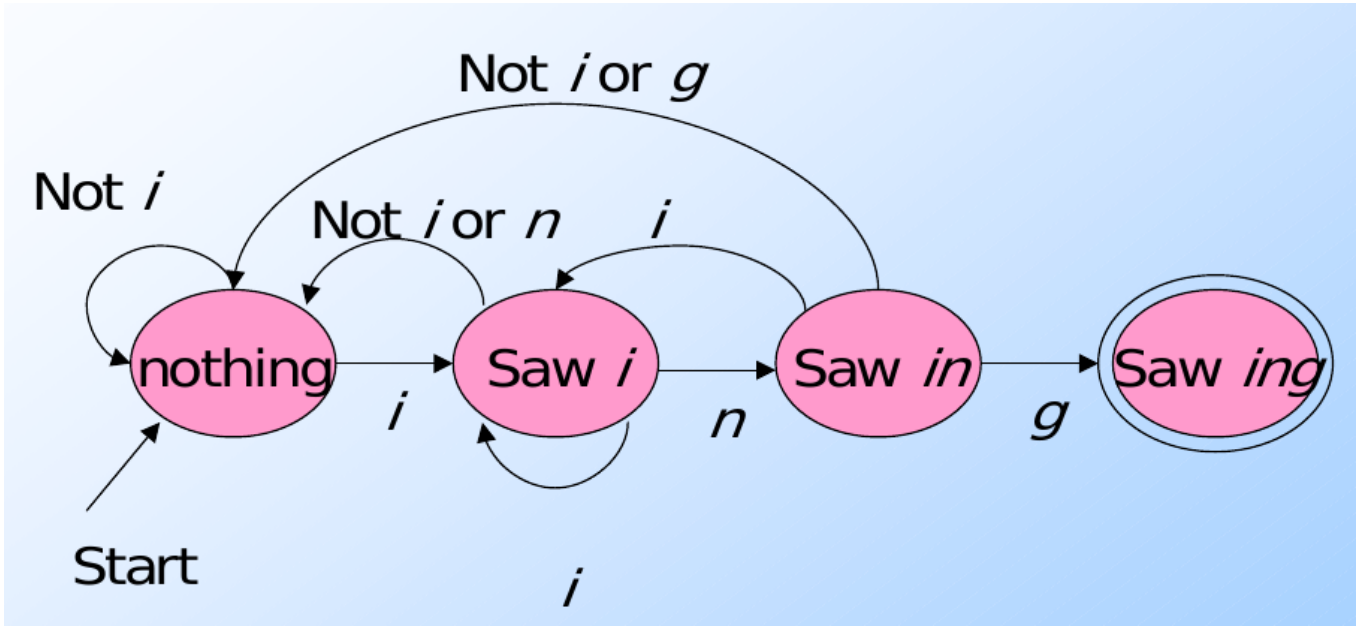
```
2: /* i seen /*
 c = getNextInput();
 if (c=='n') goto 3;
 else if (c=='i') goto 2;
 else goto 1;
3:  /* "in" seen */
...
```

Automat $\rightarrow$ Cod

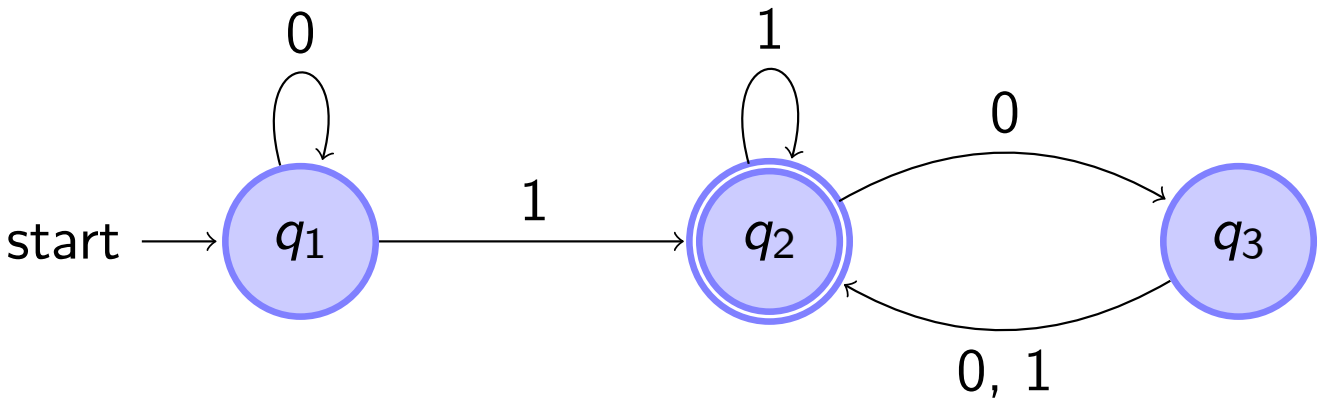
1. citeste urmatorul input
2. decide starea urmatoare
3. sari la inceputul codului pentru acea stare

```
2: /* i seen /*
  c = getNextInput();
  if (c=='n') goto 3;
  else if (c=='i') goto 2;
  else goto 1;
3:  /* "in" seen */
...
```

de fapt: expresii regulate .*ing



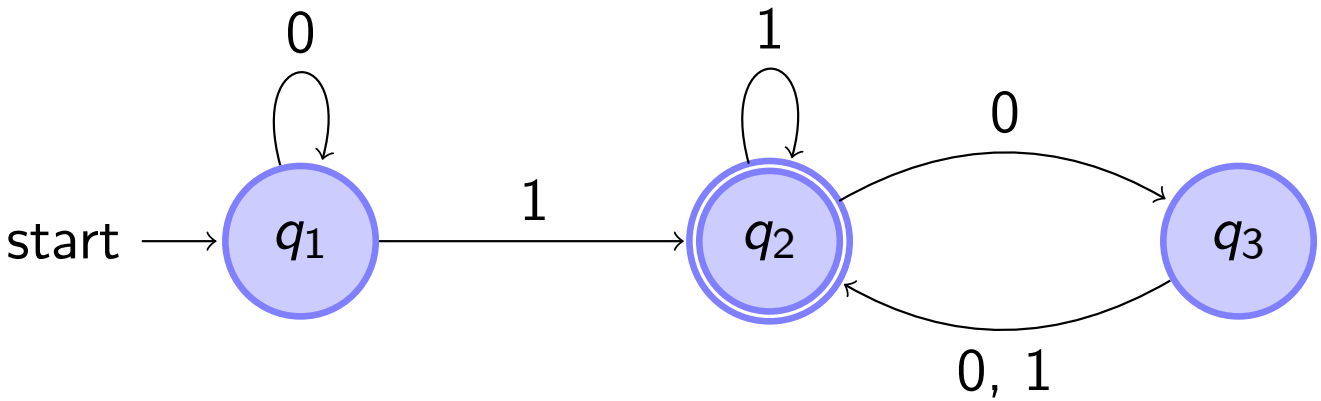
Exemplu: Automat



- 3 stari; start state, accept state
- transitions

Automatul primeste un input string si produce *accept* sau *reject*.
fie 1101:

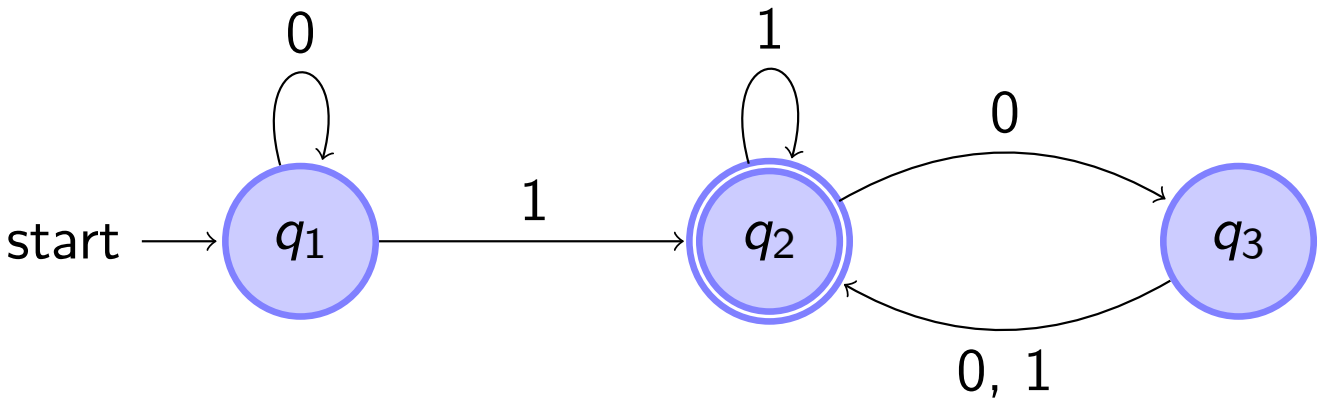
1. Start in q_0
2. Citeste 1 si urmeaza tranzitia q_1 to q_2
3. Citeste 1 si urmeaza tranzitia q_2 to q_2
4. Citeste 0 si urmeaza tranzitia q_2 to q_3
5. Citeste 1 si urmeaza tranzitia q_3 to q_2
6. *accept* deoarece se afla in starea accept q_2 la sfarsitul input-ului



- Accepta 1, 01, 11, 0101010101?
- Dar 100, 0100, 110000, 0101000000?
- dar 0, 10, 101000?

care sunt toate stringurile pe care automatul le accepta?
Setul tuturor sirurilor recunoscute de an automat A: $L(A)$

L(A) =?



- Accepta 1, 01, 11, 0101010101? DA
- Dar 100, 0100, 110000, 0101000000? Da
- dar 0, 10, 101000? le respinge

care sunt toate stringurile pe care automatul le accepta?
Setul tuturor sirurilor recunoscute de an automat A: $L(A)$

L(A) = $\{w|w$ contine cel putin un 1 si se termina cu un numar par de 0-uri dupa ultimul 1$\}$

Table of Contents

Automat finit determinist (Deterministic Finite Automaton) - *formal* definition

$$(\Sigma, Q, \delta, q_0, F)$$

- un alfabet de intrare Σ- set de simboluri
- un set finit de stari Q
- o functie de tranzitie δ
- o stare de start q_0
- un set de stari finale $F \subseteq Q$ (final state, accepting states)

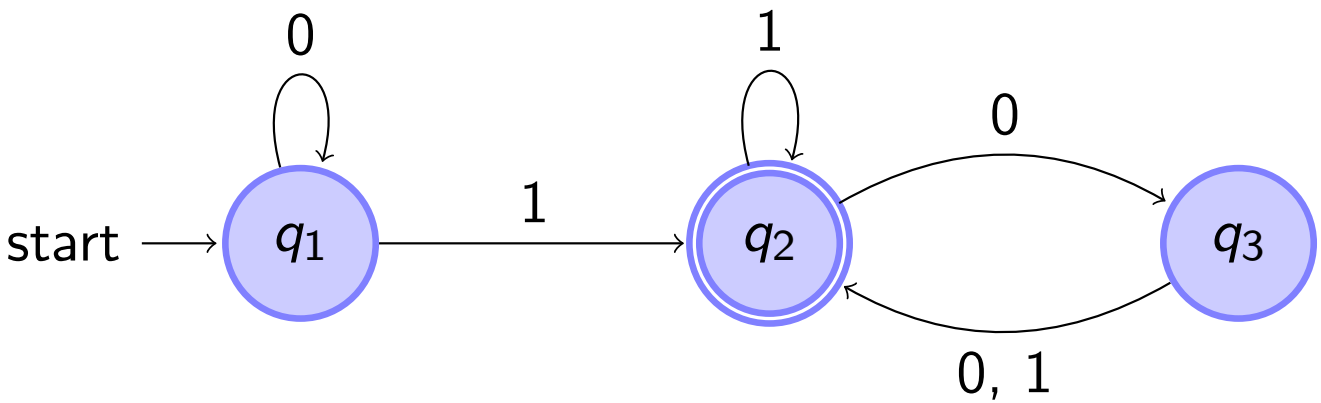
Functia de tranzitie $\delta : Q \times \Sigma \rightarrow Q$: $\delta(q,\ a)$ starea in care automatul DFA trece cand este in starea q si primeste ca input a.

Setul tuturor sirurilor recunoscute de un automat A:
$L(A) = \{w | A\ accepta\ w\}$

Descrierea formala a automatului:

$D_1 = (\{0,1\}, \{q_1, q_2, q_3\}, \delta, q_1, \{q_2\})$

	δ	0	1
$\rightarrow$	q_1	q_1	q_2
$*$	q_2	q_3	q_2
	q_3	q_2	q_2



Acceptare 011 ?$\exists \delta(q_1, 0)$:

$$\delta(q_1, 0) = q_1; \delta(q_1, 1) = q_2; \delta(q_2, 1) = q_2 \in F$$

s-a gasit secventa de stari: q_1, q_1, q_2

Definitie formala a calculului

Fie $A = (\Sigma, Q, \delta, q_0, F)$ si $w = w_1 w_2 ... w_n$, $w_i \in \Sigma$. Automatul recunoaste w daca exista o secventa $r_0, r_1, ..., r_n \in Q$:

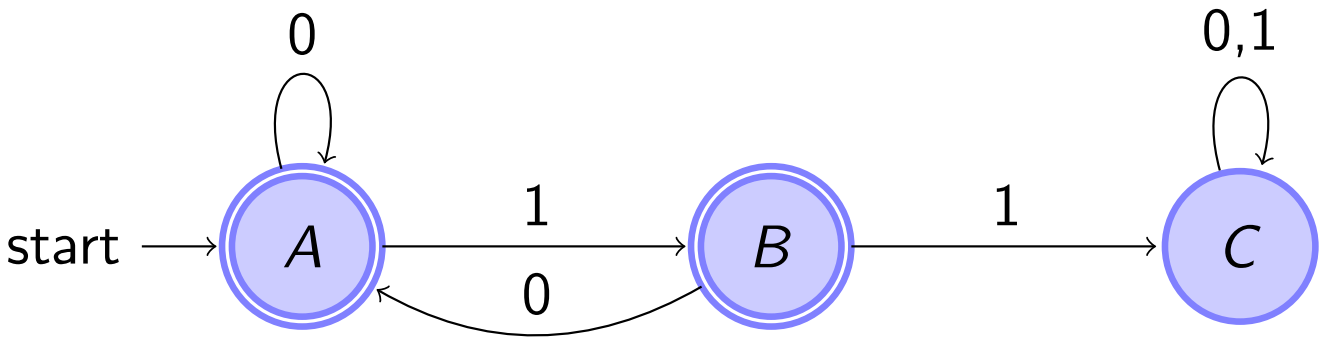
Definitie formala a calculului

Fie $A = (\Sigma, Q, \delta, q_0, F)$ si $w = w_1 w_2 ... w_n$, $w_i \in \Sigma$. Automatul recunoaste w daca exista o secventa $r_0, r_1, ..., r_n \in Q$:

- $r_0 = q_0$
- $\delta(r_i, w_{i+1}) = r_{i+1}$ pt $i = 0, ..., n-1$
- $r_n \in F$

Exemplu:

Ce accepta?



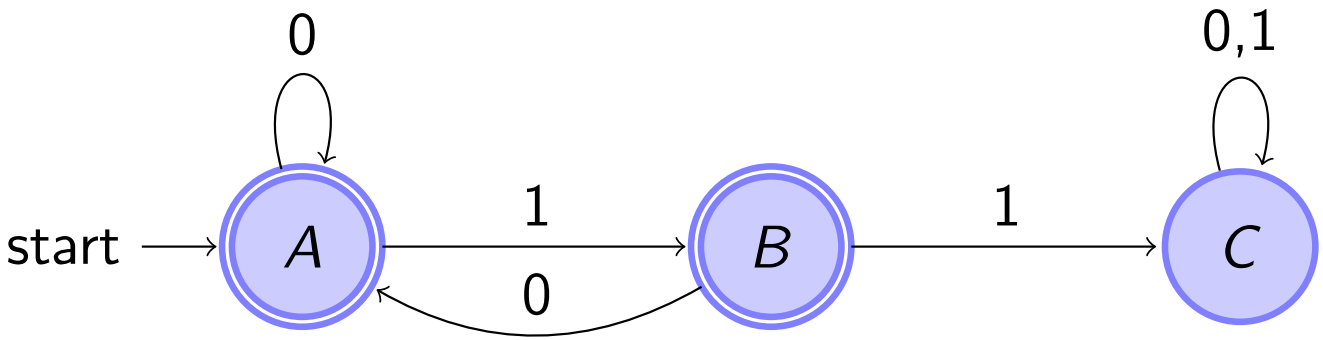
Extended $\hat{\delta}$

$$\hat{\delta}(A, 011) = \delta(\delta(\delta(A, 0), 1), 1) =$$

$$\delta(\delta(A, 1), 1) = \delta(B, 1) = C$$

Exemplu:

Ce accepta?



Extended $\hat{\delta}$

$$\hat{\delta}(A, 011) = \delta(\delta(\delta(A, 0), 1), 1) =$$

$$\delta(\delta(A, 1), 1) = \delta(B, 1) = C$$

Accepta toate stringurile care nu includ doua simboluri consecutive 1

Table of Contents

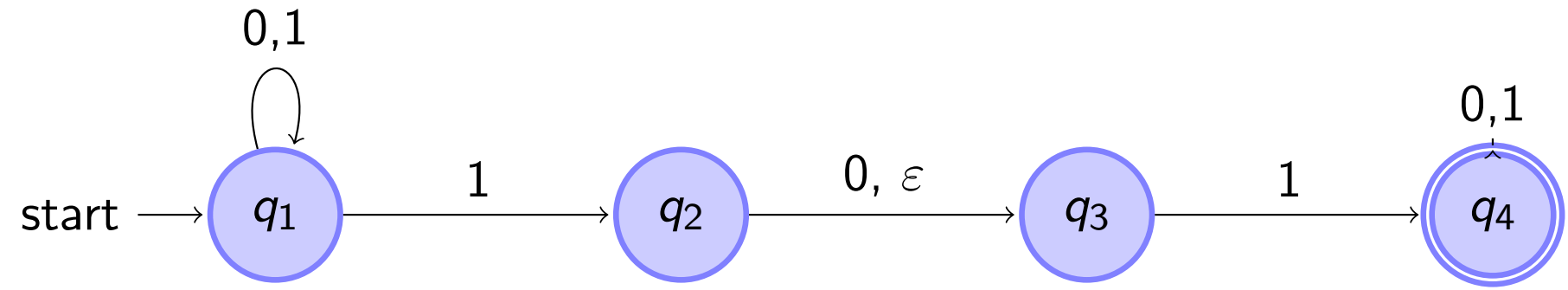
Automat finit nedeterminist - *formal* definition

$$(\Sigma, Q, \delta, q_0, F)$$

- un alfabet de intrare Σ- set de simboluri
- un set finit de stari Q
- o functie de tranzitie δ
- o stare de start q_0
- un set de stari finale $F \subseteq Q$ (final state, accepting states)

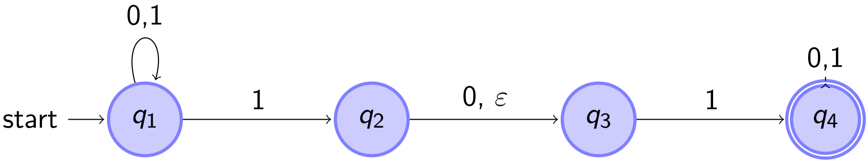
Functia de tranzitie $\delta : Q \times \Sigma_\varepsilon \to P(Q)$: $\delta(q,\ a)$ starea/starile in care automatul NFA poate trece cand este in starea q si primeste ca input a.

Exemplu - automat nedeterminist



Input: 010110

Exemplu - automat nedeterminist



Input: 010110 Calcul nedeterminist: accept/reject

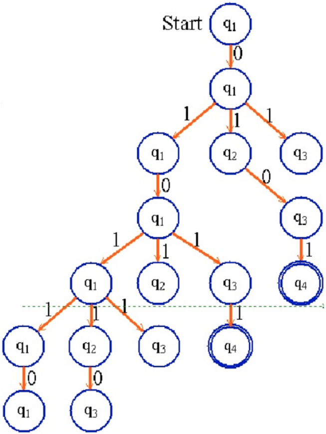


Table of Contents

Outline

Automat finit - definitie formala ca sistem de rescriere

Automat finit (finite automaton, finite state acceptor):

$$A = (T, Q, R, q_0, F)$$

- Q set nevid - setul starilor interne
- $(T \cup Q, R)$ sistem de rescriere; $T \cap Q = \emptyset$
- $q_0 \in Q$ - starea initiala
- $F \subseteq Q$ - stari finale
- fiecare element din R are forma $qt \to q'$, $q, q' \in Q, t \in T$

Definitie formala *Automat finit*

- automatul A accepta/recunoaste setul de stringuri

$$L(A) = \{\tau \in T^* | q_0\tau \Rightarrow^* q, q \in F\}$$

- Doua automate A si A' sunt echivalente daca si numai daca $L(A) = L(A')$

Interpretare

- masina care citeste la intrare un input string; citeste simbol cu simbol si isi schimba starea interna
- automatul se afla in starea q cand sirul curent din derivare este $q\tau$
- automatul face o tranzitie din q in q' daca $\tau = t\chi$ si ?? $\in R$ $q\tau = qt\chi \Rightarrow ???$
- fiecare tranzitie sterge un simbol din stringul de intrare

Definitie formala *Automat finit*

- automatul A accepta/recunoaste setul de stringuri

$$L(A) = \{\tau \in T^* | q_0\tau \Rightarrow^* q, q \in F\}$$

- Doua automate A si A' sunt echivalente daca si numai daca $L(A) = L(A')$

Interpretare

- masina care citeste la intrare un input string; citeste simbol cu simbol si isi schimba starea interna
- automatul se afla in starea q cand sirul curent din derivare este $q\tau$
- automatul face o tranzitie din q in q' daca $\tau = t\chi$ si $qt \rightarrow q' \in R$ $q\tau = qt\chi \Rightarrow q'\chi$
- fiecare tranzitie sterge un simbol din stringul de intrare

Exemplu automat vazut ca sistem de rescriere

$$A = (T = \{0, 1\}, Q = \{q_0, q_1\}, R, q_0, F = \{q_1\})$$

$$\begin{aligned} R = \{ & q_0 1 \to q_1 \\ & q_0 0 \to q_0 \\ & q_1 1 \to q_0 \\ & q_1 0 \to q_1 \\ \} & \end{aligned}$$

Intrebare: 1001 apartine limbajului automatului? Dar 10?
?Exista derivarea

$$q_0 1001 \Rightarrow^* q_1$$

Table of Contents

Teorema

- Pentru fiecare gramatica regulata G exista un automat finit A a.i. $L(A) = L(G)$.

Reamintire: gramatica regulata (din ierarhia lui Chomsky)
$G = (T, N, Z, P)$

- fiecare productie are forma

$$X \to t, \quad X \in N, \ t \in T \cup \{\varepsilon\}$$

sau

$$X \to tY, \ X, Y \in N, t \in T$$

Construirea AF pentru gramatica **regulata** G

- **Algoritm** Construirea automatului $A = (T, N \cup \{f\}, R, Z, F)$, $f \notin N$ pentru gramatica $G = (T, N, Z, P)$.
  1. daca $X \to t \in P$, $X \in N, t \in T$, atunci $Xt \to f \in R$
  2. daca $X \to tY \in P$, $X, Y \in N, t \in T$, atunci $Xt \to Y \in R$
  3. $F = \{f\} \cup \{X | X \to \varepsilon \in P\}$

Gramatica pentru constante reale - gramatica regulata

Fie gramatica G_3

- $T = \{n, ., +, -, E\}$
- $N = \{C, F, I, X, S, U\}$
- $P = \{$

$$C \to n, C \to nF, C \to .I,$$
$$F \to .I, F \to ES,$$
$$I \to n, I \to nX,$$
$$X \to ES,$$
$$S \to n, S \to +U, S \to -U,$$
$$U \to n\}$$

Exemple de derivare:

- $C \Rightarrow n$
- $C \Rightarrow .I \Rightarrow .n$
- $C \Rightarrow nF \Rightarrow n.I \Rightarrow n.nX \Rightarrow n.nES \Rightarrow n.nE + U \Rightarrow n.nE + n$

FA pentru G_3

Gramatica regulata

- $T = \{n, ., +, -, E\}$
- $N = \{C, F, I, X, S, U\}$
- $P = \{$

$C \to n, C \to nF, C \to .I,$
$F \to .I, F \to ES,$
$I \to n, I \to nX,$
$X \to ES,$
$S \to n, S \to +U, S \to -U,$
$U \to n\}$

Automat finit

- $T = \{n, ., +, -, E\}$
- $Q = \{C, F, I, X, S, U, q\}$
- $P = \{$

$Cn \to q, Cn \to F, C. \to I,$
$F. \to I, FE \to S,$
$In \to q, In \to X,$
$XE \to S,$
$Sn \to q, S+ \to U, S- \to U,$
$Un \to q\}$

- $q_0 = C$
- $F = \{q\}$

Derivare $n.n$

Gramatica

$$C \Rightarrow nF \Rightarrow n.I \Rightarrow n.n$$

Automat

$$Cn.n \Rightarrow F.n \Rightarrow In \Rightarrow q$$

Pentru orice $Z\tau\chi \Rightarrow^* X\chi \Rightarrow^* q$, $\tau, \chi \in T^*$, $X \in N$, $\tau\chi \in L(A)$, $q \in F$,
starea X specifica nonterminalul din G care ar fi trebuit utilizat pentru derivarea lui χ

Derivare $n.n$

Gramatica

$$C \Rightarrow nF \Rightarrow n.I \Rightarrow n.n$$

Automat

$$Cn.n \Rightarrow F.n \Rightarrow In \Rightarrow q$$

Pentru orice $Z\tau\chi \Rightarrow^* X\chi \Rightarrow^* q$, $\tau, \chi \in T^*$, $X \in N$, $\tau\chi \in L(A)$, $q \in F$,
starea X specifica nonterminalul din G care ar fi trebuit utilizat pentru derivarea lui χ

Derivare $n.n$

Gramatica

$$C \Rightarrow nF \Rightarrow n.I \Rightarrow n.n$$

Automat

$$Cn.n \Rightarrow F.n \Rightarrow In \Rightarrow q$$

Pentru orice $Z\tau\chi \Rightarrow^* X\chi \Rightarrow^* q$, $\tau, \chi \in T^*$, $X \in N$, $\tau\chi \in L(A)$, $q \in F$,
starea X specifica nonterminalul din G care ar fi trebuit utilizat pentru derivarea lui χ

Proprietati ale automatului

Pentru orice $Z\tau\chi \Rightarrow^* X\chi \Rightarrow^* q$, $\tau, \chi \in T^*$, $X \in N$, $\tau\chi \in L(A)$, $q \in F$,
starea X specifica nonterminalul din G care ar fi trebuit utilizat pentru derivarea lui χ
Demonstratie prin inductie

- daca $\tau\chi \in L(G)$. afirmatia este adevarata pentru Z stare initiala
- proprietatea ramana adevarata pana la starea finala Q, care nu genereaza alte simboluri

Fiecare propozitie din $L(G)$ apartine lui $L(A)$ si invers

Evitarea backtrackingului

- Automatul generat este nedeterminist: stare *l* cu inputul *n* sunt mai multe tranzitii posibile
- la implementare: backtracking necesar in cazul unei decizii incorecte
- motive pentru evitarea backtrackingului:
  - timpul necesar parsarii unui string cu backtracking poate creste exponential cu lungimea stringului
  - daca automatul nu accepta stringul, stringul va fi recunoscut drept incorect. Pinpointingul (tratarea erorilor) devine dificial cu backtracking
  - deoarece in compilator, tranzitiilor de stare le sunt asociate anumite actiuni, la revenire ar trebui anularea acelor actiuni

Automat finit determinist

Un automat este determinist daca fiecare derivare poate fi continuata prin cel mult o mutare.
$\rightarrow$ Determinist daca Partile stanga ale tuturor productiilor sunt distincte

Poate fi definit prin Tabelul de stare (state table): q, t contine q' daca si numai daca $qt \rightarrow q' \in R$
Backtrackingul poate fi intotdeauna evitat cand se recunosc stringuri pentru limbaje regulate

Automat finit determinist (deterministic finite automaton

Pentru orice gramatica regulata G, exista un automat finit determinist A (DFA) a.i. $L(A) = L(G)$

Algoritm construire DFA

Idee: construim un automat pentru gramatica $G = (T, N, Z, P)$ a.i. in timpul acceptarii unei propozitii din $L(G)$, starea la fiecare pas sa mentioneze elementul N utilizat pentru a deriva restul stringului.

Daca $X \to tU$ si $X \to tV \in P$,
atunci cand t este urmatorul simbol, restul stringului poate fi derivat atat din U cat si din V
dar pentru a avea DFA, R trebuie sa contina o singura productie $Xt \to q'$
deci starea q' trebuie sa contina un set de nonterminale - acelea care puteau fi utilizate pentru derivarea restului sirului

Algoritm construire DFA pt G=(T,N,Z,P)

$$A = (T, Q, R, q_0, F), q \textit{ reprezinta } N_q \subseteq N \cup \{f\}, f \notin N$$

1. initial $Q = \{q_0\}$ si $R = \emptyset, N_{q_0} = \{Z\}$
2. pentru $q \in Q$ netratat se efectueaza pasii 3-5 pentru fiecare $t \in T$
3. fie $next(q, t) = \{U | \exists X \in N_q \text{ a.i. } X \to tU \in P\}$
4. daca exista un $X \in N_q$ a.i. $X \to t \in P$, atunci adauga f la $next(q, t)$ daca nu era deja adaugat;
   daca exista $X \in N_q$ a.i. $X \to \varepsilon \in P$ atunci adauga f la N_q
5. daca $next(q, t) \neq \emptyset$, atunci fie q' starea ce reprezinta $N_{q'} = next(q, t)$. Adauga q' la Q si $qt \to q'$ in R
6. daca toate starile din Q au fost considerate, atunci $F = \{q | f \in N_q\}$ si terminat; altfel continua cu pasul 2

DFA

	n	.	$+$	$-$	E	N
q_0	q_1	q_2				$\{C\}$
q_1		q_2			q_3	$\{f, F\}$
q_2	q_4					$\{I\}$
q_3	q_5		q_6	q_6		$\{S\}$
q_4					q_3	$\{f, X\}$
q_5						$\{f\}$
q_6	q_5					$\{U\}$

- $T = \{n, ., +, -, E\}$, $F = \{q_1, q_4, q_5\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$
- $P =$

$$q_0 n \to q_1, q_0 . \to q_2,$$
$$q_1 . \to q_2, q_1 E \to q_3,$$
$$q_2 n \to q_4$$
$$q_3 n \to q_5, q_3 + \to q_6, q_3 - \to q_6,$$
$$q_4 E \to q_3$$
$$q_6 n \to q_5\}$$

DFA

	n	.	$+$	$-$	E	N
q_0	q_1	q_2				$\{C\}$
q_1		q_2			q_3	$\{f, F\}$
q_2	q_4					$\{I\}$
q_3	q_5		q_6	q_6		$\{S\}$
q_4					q_3	$\{f, X\}$
q_5						$\{f\}$
q_6	q_5					$\{U\}$

- $T = \{n, ., +, -, E\}$, $F = \{q_1, q_4, q_5\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$
- $P =$

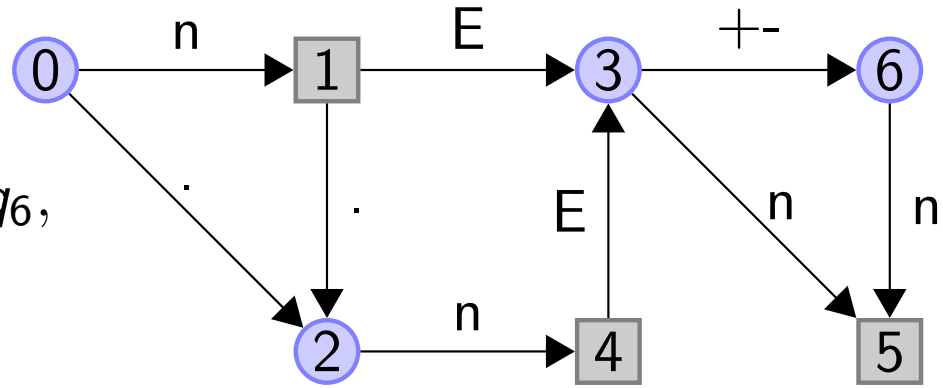
$$q_0 n \to q_1, q_0 . \to q_2,$$
$$q_1 . \to q_2, q_1 E \to q_3,$$
$$q_2 n \to q_4$$
$$q_3 n \to q_5, q_3 + \to q_6, q_3 - \to q_6,$$
$$q_4 E \to q_3$$
$$q_6 n \to q_5\}$$

Diagrama de stare

Fie $T = \{n, ., +, -, E\}$, $F = \{q_1, q_4, q_5\}$
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$P = \{q_0 n \to q_1, q_0 . \to q_2,$
$q_1 . \to q_2, q_1 E \to q_3,$
$q_2 n \to q_4$
$q_3 n \to q_5, q_3 + \to q_6, q_3 - \to q_6,$
$q_4 E \to q_3$
$q_6 n \to q_5\}$

Diagrama de stare



cale care incepe in q_0 si se termina intr-o stare finala $\in L(A)$

n	.n	n.n
nEn	nE+n	nE-n
.nEn	.nE+n	.nE-n
n.nEn	n.nE+n	n.nE-n

Diagrama de stare

Fie $A = (T, Q, R, q_0, F)$ un automat finit,

- $D = \{(q, q') | \exists t, qt \to q' \in R\}$,
- $f : (q, q') \to \{t | qt \to q' \in R)$ o mapare de la D la $P(T)$

Graful directionat (Q, D) cu etichetele muchiilor $f((q, q'))$ este diagrama de stare a automatului A

Pentru fiecare automat finit A exista o gramatica regulata G a.i. $L(A) = L(G)$

Din automatul $A = (T, Q, R, q_0, F)$ construim gramatica $G = (T, Q, q_0, P)$:

$$P = \{q \to tq' | qt \to q' \in R\} \cup \{q \to \varepsilon | q \in F\}$$

Gramatici pentru automat

$$F = \{q_1, q_4, q_5\} \quad P = \{ \quad q_0 n \to q_1, q_0 . \to q_2,$$
$$q_1 . \to q_2, q_1 E \to q_3,$$
$$q_2 n \to q_4$$
$$q_3 n \to q_5, q_3 + \to q_6, q_3 - \to q_6,$$
$$q_4 E \to q_3$$
$$q_6 n \to q_5\}$$

Productii gramatica

$$q_0 \to nq_1|.q_2,$$
$$q_1 \to .q_2|Eq_3|\varepsilon,$$
$$q_2 \to nq_4$$
$$q_3 \to nq_5| + q_6| - q_6,$$
$$q_4 \to Eq_3|\varepsilon$$
$$q_5 \to \varepsilon$$
$$q_6 \to nq_5\}$$

Productii gramatica fara productii ε

$$q_0 \to n|nq_1|.q_2,$$
$$q_1 \to .q_2|Eq_3,$$
$$q_2 \to n|nq_4$$
$$q_3 \to n| + q_6| - q_6,$$
$$q_4 \to Eq_3$$
$$q_6 \to n\}$$

Gramatici pentru automat

$$F = \{q_1, q_4, q_5\} \quad P = \{ \quad q_0 n \to q_1, q_0 . \to q_2,$$
$$q_1 . \to q_2, q_1 E \to q_3,$$
$$q_2 n \to q_4$$
$$q_3 n \to q_5, q_3 + \to q_6, q_3 - \to q_6,$$
$$q_4 E \to q_3$$
$$q_6 n \to q_5\}$$

Productii gramatica

$$q_0 \to nq_1|.q_2,$$
$$q_1 \to .q_2|Eq_3|\varepsilon,$$
$$q_2 \to nq_4$$
$$q_3 \to nq_5| + q_6| - q_6,$$
$$q_4 \to Eq_3|\varepsilon$$
$$q_5 \to \varepsilon$$
$$q_6 \to nq_5\}$$

Productii gramatica fara productii ε

$$q_0 \to n|nq_1|.q_2,$$
$$q_1 \to .q_2|Eq_3,$$
$$q_2 \to n|nq_4$$
$$q_3 \to n| + q_6| - q_6,$$
$$q_4 \to Eq_3$$
$$q_6 \to n\}$$

- Pentru orice gramatica regulata G, exista un automat finit A a.i. $L(A) = L(G)$
- Pentru fiecare automat finit A exista o gramatica regulata G a.i. $L(A) = L(G)$

Gramaticile regulate si automatele finite sunt echivalente

DFA vs NFA

Ambele automate, deterministe si nedeterministe, sunt capabile sa recunoasca toate limbajele regulate:

$$L(NFA) = L(DFA)$$

Diferenta principala: spatiu vs timp:

- DFA sunt mai rapide decat NFA
- DFA sunt exponential mai mari decat NFA

FA sunt folosite ca mdoele pentru:

- software for designing digital circuits
- lexical analyzer of a compiler
- software for verifying finite state systems, such as communication protocols: exemplul cu planeta

Rezumat

Introducere in automate. Automate deterministe finite

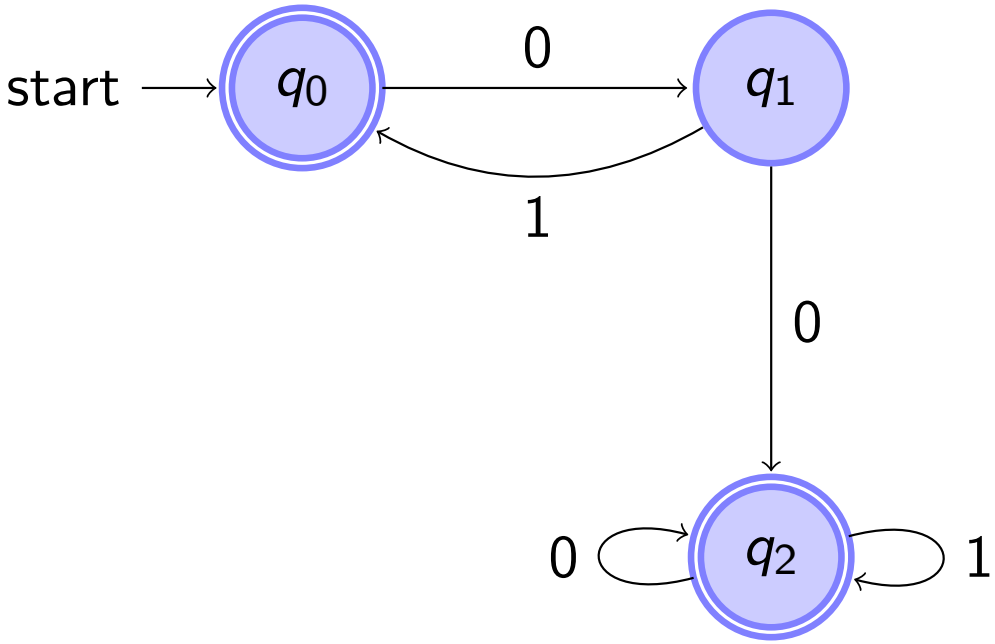
- DFA - deterministic finite automaton
- NFA - nondeterministic finite automaton

Gramatici regulate si automate finite

- Automate finite vazute ca sisteme de rescriere
- Automate finite pentru gramatici regulate

Extended Example - ullman slides

- On a distant planet, there are three species, a, b, and c. Any two different species can mate. If they do:
  1. The participants die.
  2. Two children of the third species are born.
- The planet fails if at some point all individuals are of the same species. Then, no more breeding can take place.
- State = sequence of three integers the numbers of individuals of species a, b, and c.


start
q_0
q_1
q_2
0
1
0
0
1

Expresii regulate

March 27, 2023

Outline

Gramatici regulate 2. Expresii regulate

Regular expression - expresii regulate

Expresiile regulate descriu limbajele regulate

Fie V un vocabular si simbolurile $E, \varepsilon, +, *, (,) \notin V$.
Un string ρ peste $V \cup \{E, \varepsilon, +, *, (,)\}$ este o expresie regulata peste V daca

1. ρ este un simbol peste V sau unul dintre simbolurile E, ε, sau
2. ρ este de forma $(X + Y)$, (XY), $(X)^*$, unde X si Y sunt expresii regulate.

Descriere expresii regulate

- $E = \emptyset$ este limbajul empty
- $\varepsilon = \{\varepsilon\}$ este limbajul format din stringul empty
- $v, v \in V$ descrie limbajul $\{v\}$
- $(X + Y) = \{w | w \in X \ sau \ w \in Y\}$
- $XY = \{\chi\gamma | \chi \in X \ si \ \gamma \in Y\}$
- operatorul * inchidere (Kleene closure):

$$X^* = \varepsilon + X + XX + XXX + ...$$

expresii regulate 2

- Parantezele se pot omite
- * este operator unar, cu prioritate mai mare decat oricare operator binar
- $+$ are prioritate mai mica decat concatenarea

$W + XY^*$ este echivalent cu $(W \ + \ (X \ (Y^*) \ ) \ )$

Exemple expresii regulate

- $01 = \{01\}$

Exemple expresii regulate

- $01 = \{01\}$
- $01 + 0 = \{01, 0\}$ - in lex |

Exemple expresii regulate

- $01 = \{01\}$
- $01 + 0 = \{01, 0\}$ - in lex |
- $0(1 + 0) = \{01, 00\}$

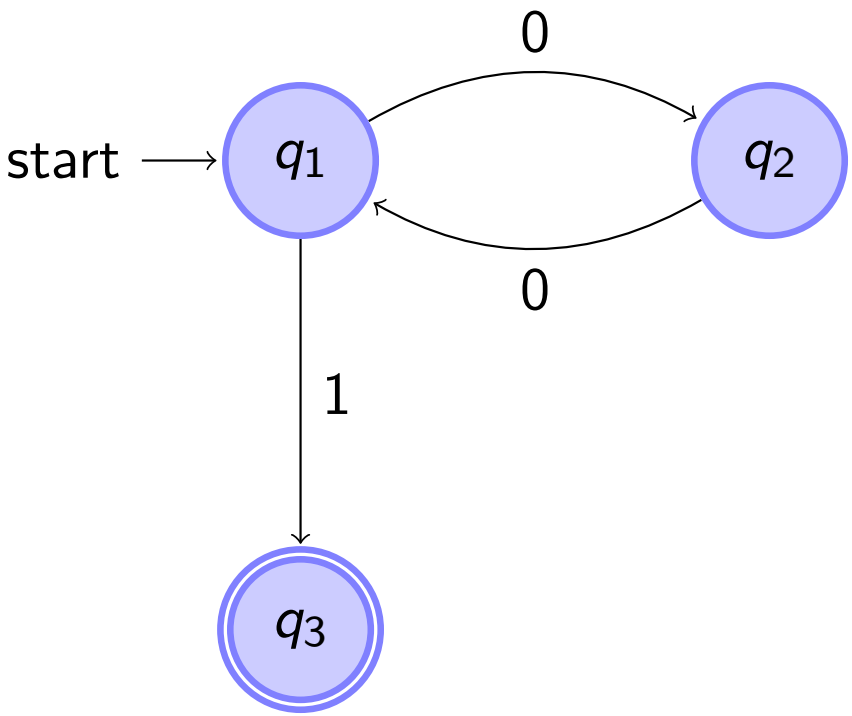
Exemple expresii regulate

- $01 = \{01\}$
- $01 + 0 = \{01, 0\}$ - in lex |
- $0(1 + 0) = \{01, 00\}$
- $0^* = \{\varepsilon, 0, 00, 000, ...\}$

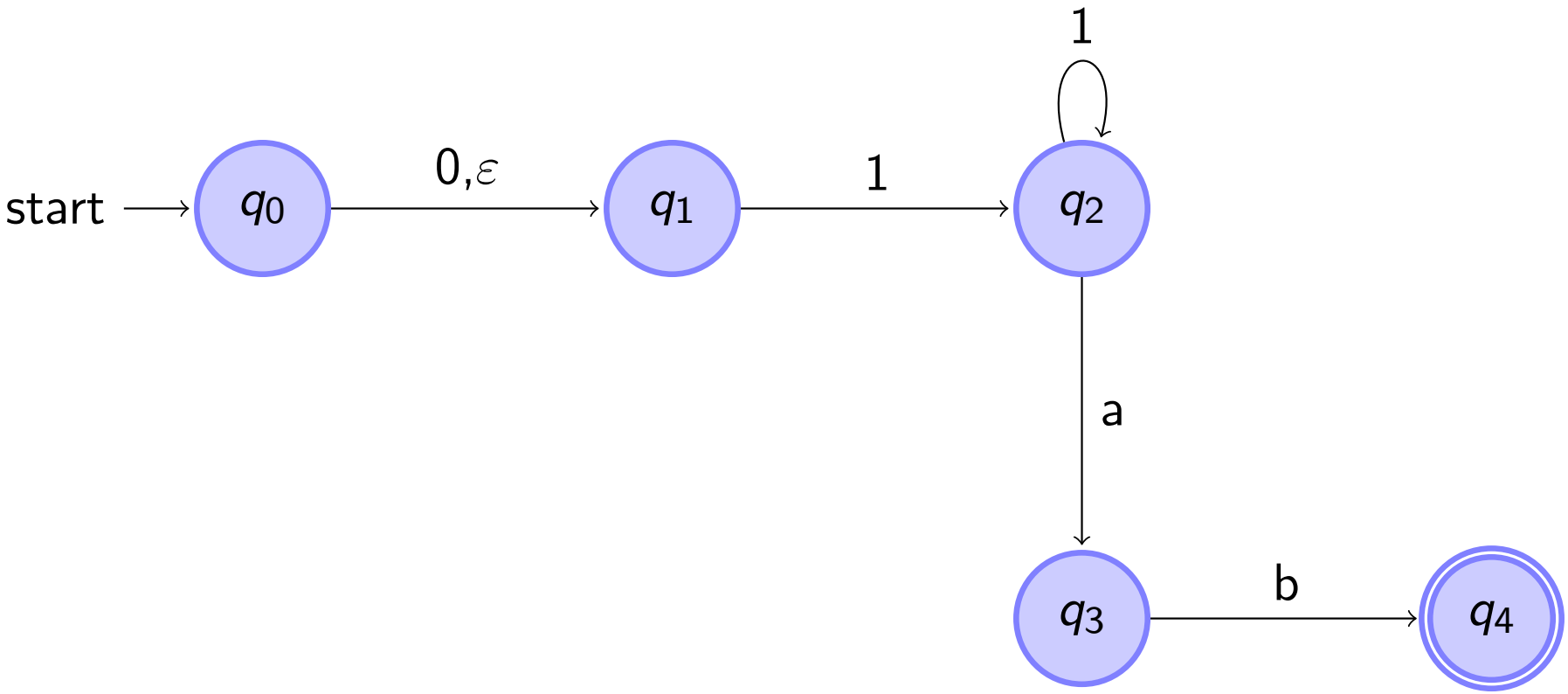
Exemple expresii regulate

- $01 = \{01\}$
- $01 + 0 = \{01, 0\}$ - in lex |
- $0(1 + 0) = \{01, 00\}$
- $0^* = \{\varepsilon, 0, 00, 000, ...\}$
- $(0 + 10)^*(\varepsilon + 1)$ Toate stringurile de 0 si 1 fara doua consecutive 1

Exemplu $(00)^*1$



Exemplu $0?1^{+}ab$



Operatori aditionali: ? +

Nu permit definirea unor limbaje aditionale, dar permit exprimarea mai usoara a expresiilor regulate

- Operatorul optional: ?
  Daca R este o expresie regulata, $R? = \varepsilon + R$
- Operatorul $+$
  Daca R este o expresie regulata $R^+ = RR^*$:

$$L(R^+) = L(R) \cup L(RR) \cup L(RRR) \cup ...$$

Proprietati algebrice ale expresiilor regulate

$X + Y = Y + X$	*comutativitate*
$(X + Y) + Z = Z + (Y + Z)$	*asociativitate*
$X(YZ) = (XY)Z$	
$X(Y + Z) = XY + XZ$	*distributivitate*
$(X + Y)Z = XZ + YZ$	
$X + \emptyset = \emptyset + X = X$	*identitate*
$X\varepsilon = \varepsilon X$	
$X\emptyset = \emptyset X = X$	*zero*
$X + X = X$	*idempotenta*
$(X^*)^* = X^*$	
$X^* = \varepsilon + XX^*$	
$X^* = X + X^*$	
$\varepsilon^* = \varepsilon$	
$\emptyset^* = \varepsilon$	

Echivalenta expresii regulate - automate finite

Fie R o expresie regulata care descrie un subset $S \subseteq T^*$.
Exista un automat finit determinist $A = (T, Q, P, q_0, F)$ a.i.
$L(A) = S$.

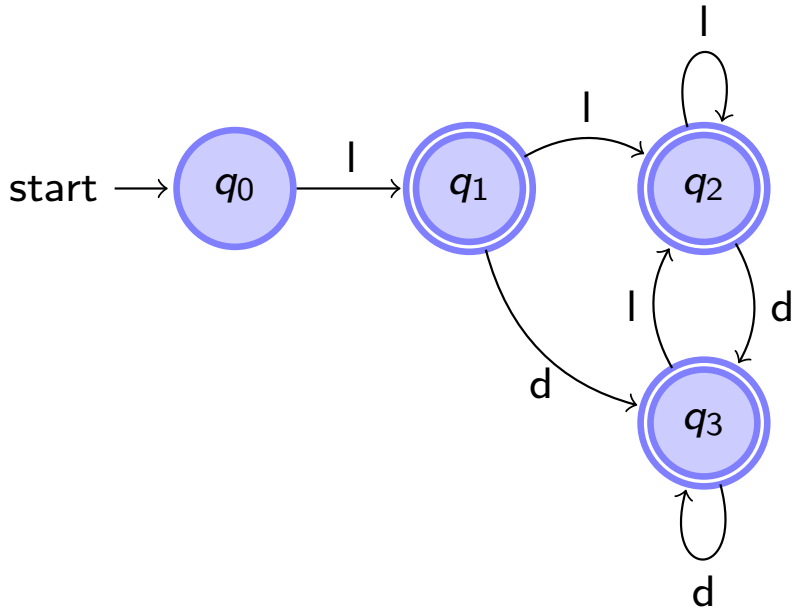
Construire automat

1. $R = I(I + d)^*$
2. $R' = 1(2 + 3)^*$ - o noua expresie in care se inlocuiesc elementele lui T din R cu simboluri distincte
   Aparitii multiple ale aceluiasi element - simboluri diferite
3. $R' = 01(2 + 3)^*$ - se adauga un prefix (un simbol distinct)
   Daca $R = E$ atunci R' este doar simbolul de start
4. starile automatului corespund submultimilor setului de simboluri.
5. Se inspecteaza pe rand starile lui lui Q si daca e necesar, se adauga stari noi:
   pentru $\forall q \in Q$ *si* $\forall t \in T$, *fie* q' corespondentul setului de simboluri din R':
   - care inlocuiesc pe t si
   - urmeaza unui simbol din setul corespunzator lui q

   Daca setul corespounzator lui q' nu e vid, se adauga $qt \to q'$ la P si se include q' in Q.
6. setul F de starile finale = toate starile care includ un simbol final posibil al lui R'

$l(l+d)^*$. $R' = 01(2+3)^*$

	l	d	
q_0	q_1		$\{0\}$
q_1	q_2	q_3	$\{1\}$
q_2	q_2	q_3	$\{2\}$
q_3	q_2	q_3	$\{3\}$



q_0 l: $\{1, 2\}$, dar numai 1 urmeaza lui 0 — deci $q_0 l \to q_1$

q_0 d: $\{3\}$, dar nu urmeaza lui 0 — deci $q_1 d$ nu face parte din

q_1 l: $\{1, 2\}$, dar numai 2 urmeaza lui 1 — deci $q_1 l \to q_2$

q_1 d: $\{3\}$ si 3 urmeaza lui 1 — deci $q_1 d \to q_3$

q_2 l: $\{1, 2\}$, dar numai 2 urmeaza lui 2 — deci $q_2 l \to q_2$

q_2 d: $\{3\}$, si 3 urmeaza lui 2 — deci $q_2 d \to q_3$

q_3 l: $\{1, 2\}$, dar numai 2 urmeaza lui 3 - — deci $q_3 l \to q_2$

q_3 d: $\{3\}$, si 3 urmeaza lui 3 - — deci $q_3 d \to q_3$

Stari finale: q_1, q_2 si q_3

Conversie expresie regulata - automat finit determinist - continuare

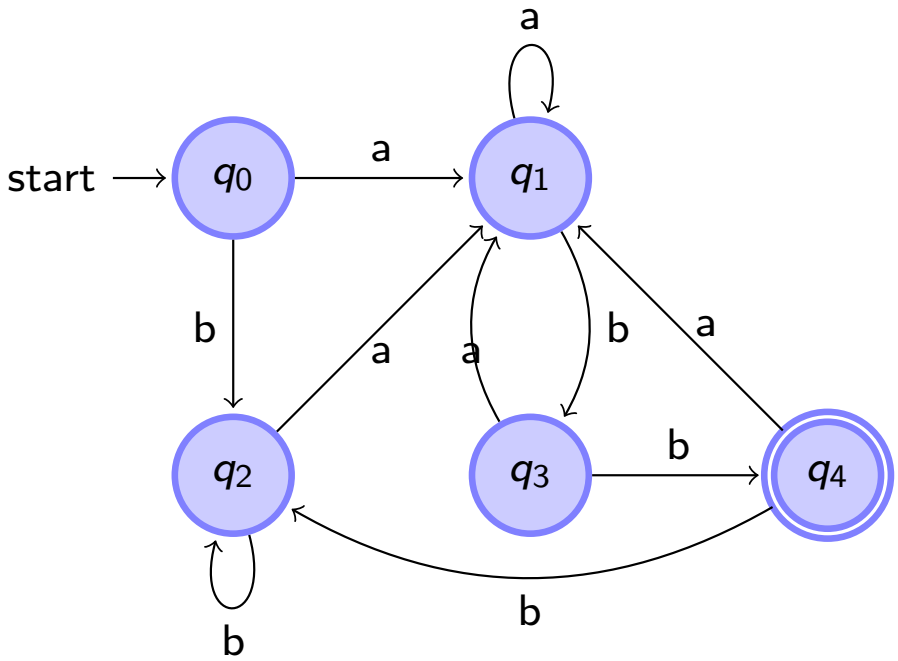
$(a + b) * abb$ becomes $0(1 + 2)^*345$

	a	b		a	follow	b	follow
q_0	q_1	q_2	0	1,3	1,3	2,4,5	2
q_1	q_1	q_3	1,3	1,3	1,3	2,4,5	2,4
q_2	q_1	q_2	2	1,3	1,3	2,4,5	2
q_3	q_1	q_4	2,4	1,3	1,3	2,4,5	2,5
q_4	q_1	q_2	2,5	1,3	1,3	2,4,5	2

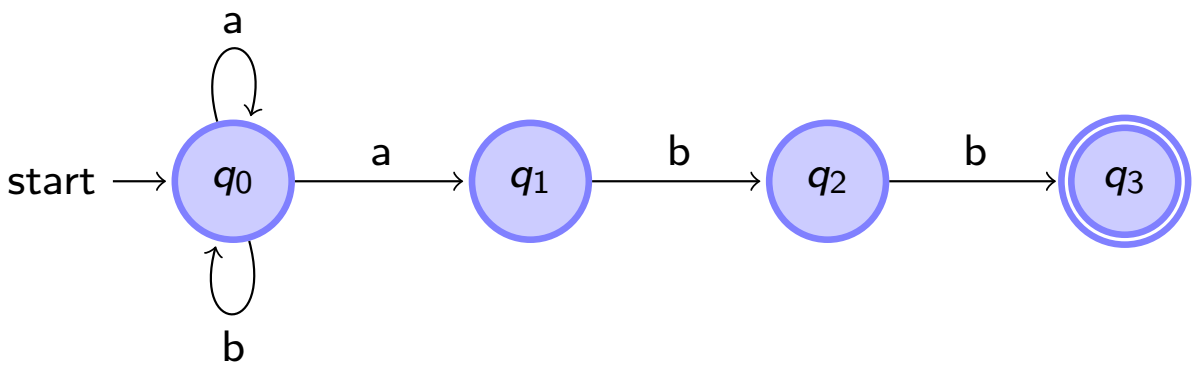
Test online: dfa_toolbox, dfa_nfa_pda

DFA NFA: (a+b)*abb

DFA

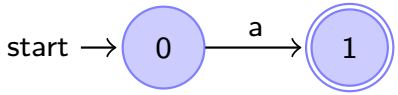


NFA

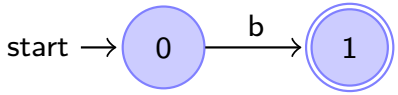


Alternativa

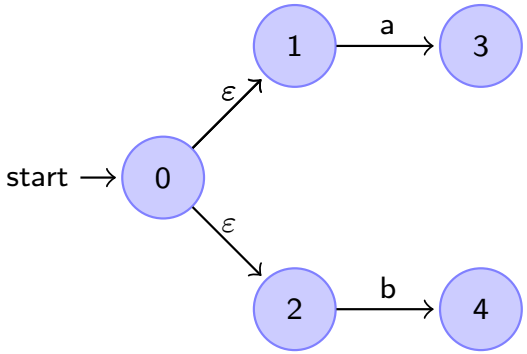
1. a



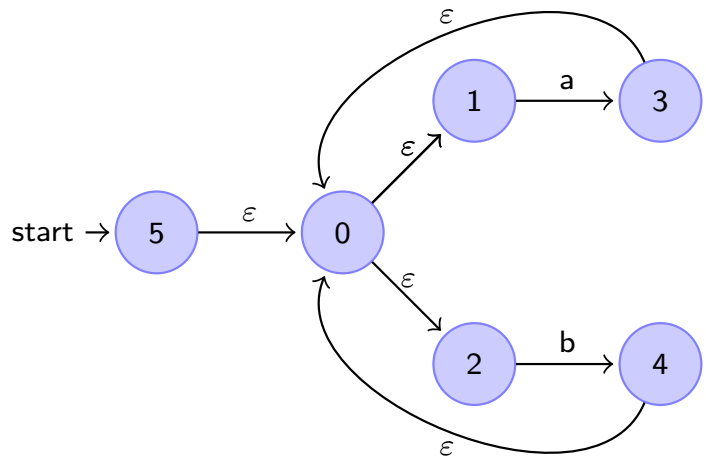
2. b



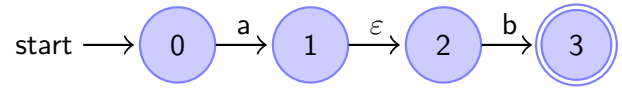
3. $a + b$



4. $(a + b)^*$



5. ab



Expresii regulate - backtracking recursiv vs automate finite deterministe

fast vs simple

Test regex in python vs lex for a string of length n

$$(a?)^n a\{n\}$$

```
Lex:
%%
(a?){100}a{100}          printf("Easy with DFA");
.                       ;

Python:
import re
xx='a'*30
print(re.match('(a?){30}a{30}',xx) )

test for a^30 for python and 100 for lex
```

why? Concluzii

- Assuming that the language has been described by a grammar, we are interested in techniques for automatically generating a **recognizer from that grammar**. There are two reasons for this requirement:
  - It provides a guarantee that the language recognized by the compiler is identical to that defined by the grammar.
  - It simplies the task of the compiler writer.

Expresiile regulate = un mod algebric de a descrie limbajele
Descriu limbajele regulate

$a^n b^n$

NU e un limbaj regulat: nu exista niciun automat finit care sa-l aiba ca limbaj
Dar limbajul *aaabbb*?

LEX - analiza lexicala

Caractere operator

$$'' \setminus [\,] \wedge - \ ? \ . \ * \ + \ | \ (\ ) \ \$ \ / \ \{\} \% <>$$

Folosirea lor drept caractere text: precedate de $\setminus$ sau intre $''$.

$$xyz'' + +''$$

$$''xyz + +''$$

$$xyz \setminus + \setminus +$$

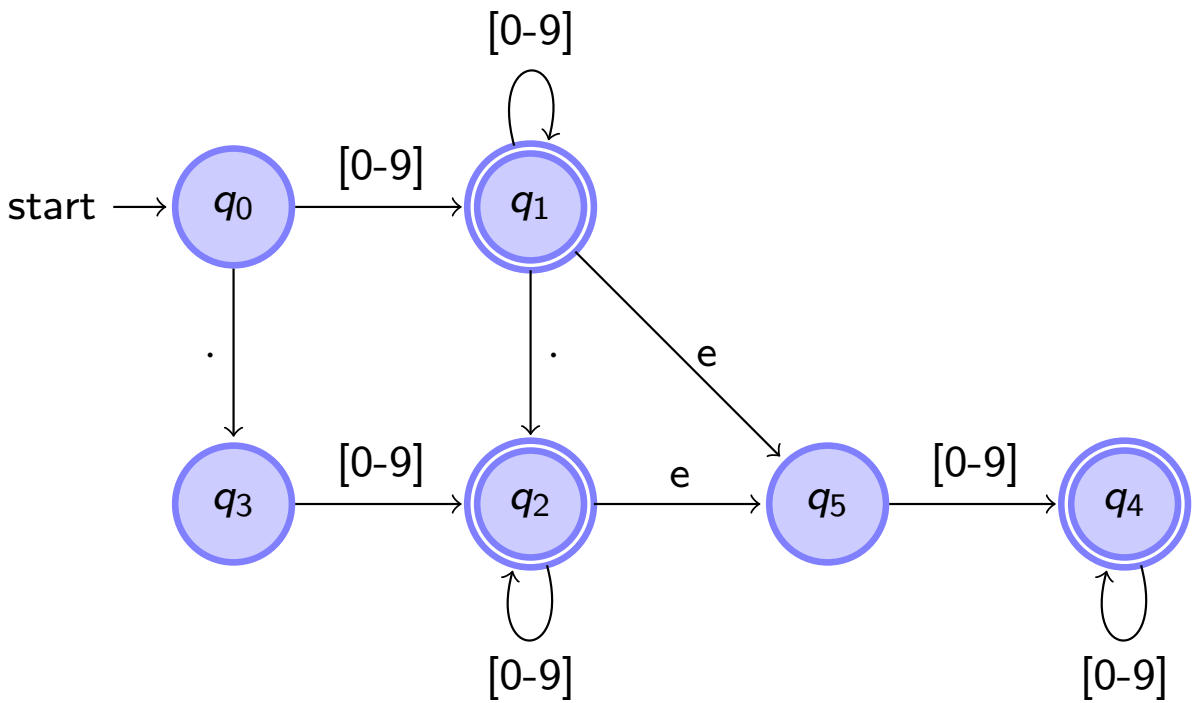
Expresii regulate in LEX

1. Clase de caractere $[a-z0-9<>]$, $[-0-9]$, ^abc
2. Caracter arbitrar .
3. Element optional $ab?c$
4. Repetitii $a*$, $a+$, $[a-z]+$
5. Alternare $ab|cd$ $a(b|d)$
6. Doar la inceput/final de linie ^abc, abc\$
7. Context ab/cd (ab daca e urmat de cd)
8. Operator {}: $a\{1,5\}$, $a\{2,\}$

Analizor lexical controlat prin automat finit 3.6.2

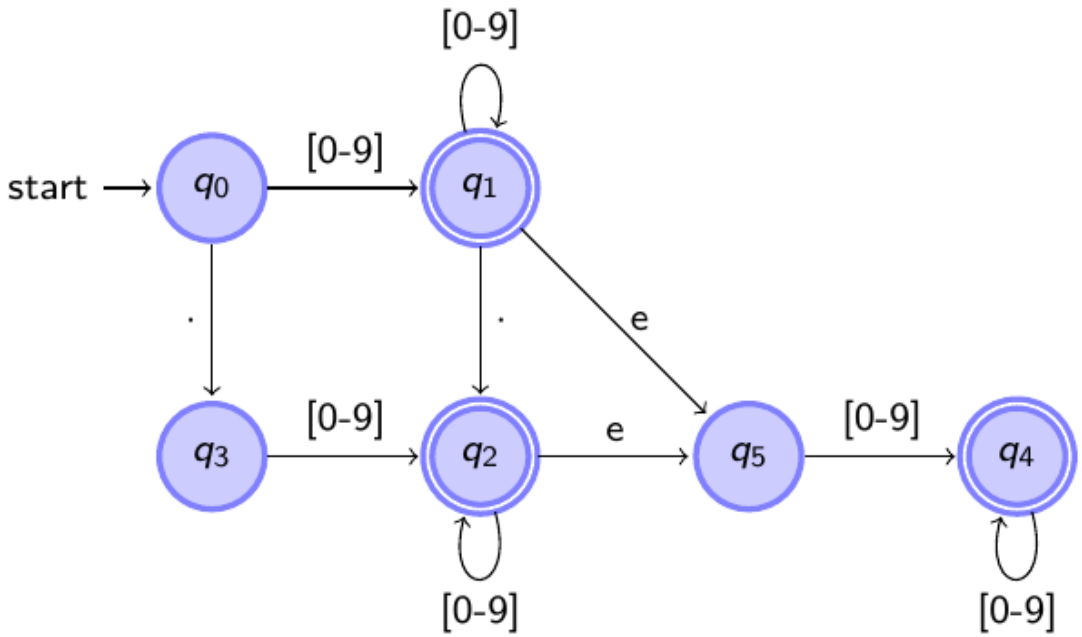
Fie:

```
D   [0-9]
%%
{D}+                                    return ICON;
({D}*|{D}*\.{D}+|{D}+\.{D}*)(e{D}+)?    return FCON;
```



Tabel de tranzitie

Stare curenta	caracter examinat inainte (lookahead) caracter de intrare			actiune la acceptare
	.	0-9	e	
0	3	1	-	-
1	2	1	5	return ICON;
2	-	2	5	return FCON;
3	-	2	-	-
4	-	4	-	return FCON;
5	-	4	-	-



Algoritmul utilizat de LEX (greedy) 3.6.3

```
stare_curenta = 0;
stare_acceptoare_vazuta_anterior = nimic_vazut;
if (caracter lookahead este end_of_input)
   return 0;
while(caracter lookahead nu este end_of_input){
  if (exista tranzitie din starea curenta cu caracterul
      lookahead curent){
     stare_curenta = acea stare;
     avanseaza in intrare;
     if (starea curenta este o stare acceptoare){
        memoreaza pozitia curenta in intrare
        si actiunea asociata starii curente;
     }  }
  else{
    if (nu a fost vazuta nicio stare acceptoare){
       exista o eroare:
          descarca lexemul curent si caracterul de intrare
              curent;
          stare_curenta = 0;
       }
    else {
      salveaza intrarea in pozitia in care se afla cand a
      vazut ultima stare acceptoare; realizeaza actiunea
      asociate acelei stari acceptoare;
      }
 }
```

Exemplu: *stare_urmatoare* = *array*[*stare_curenta*][*intrare*]

Stare curenta	caracter examinat inainte (lookahead) caracter de intrare			actiune la acceptare
	.	0-9	e	
0	3	1	-	-
1	2	1	5	return ICON;
2	-	2	5	return FCON;
3	-	2	-	-
4	-	4	-	return FCON;
5	-	4	-	-

Stare	Intrare	Ultima stare acceptoare	actiune	pozitie in intrare
0	1.2e4			
1	.2e4	1	ICON	.
2	2e4	2	FCON	2
2	e4	2	FCON	e
5	4			
4		4	FCON	

Exemplu: *stare_urmatoare* = *array*[*stare_curenta*][*intrare*]

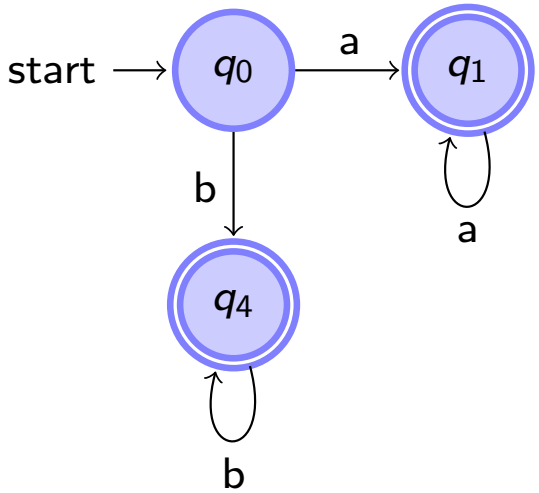
Stare curenta	caracter examinat inainte (lookahead) caracter de intrare			actiune la acceptare
	.	0-9	e	
0	3	1	-	-
1	2	1	5	return ICON;
2	-	2	5	return FCON;
3	-	2	-	-
4	-	4	-	return FCON;
5	-	4	-	-

Stare	Intrare	Ultima stare acceptoare	actiune	pozitie in intrare
0	1.2e			
1	.2e	1	ICON	.
2	2e	2	FCON	2
2	e	2	FCON	e
5				
	e			

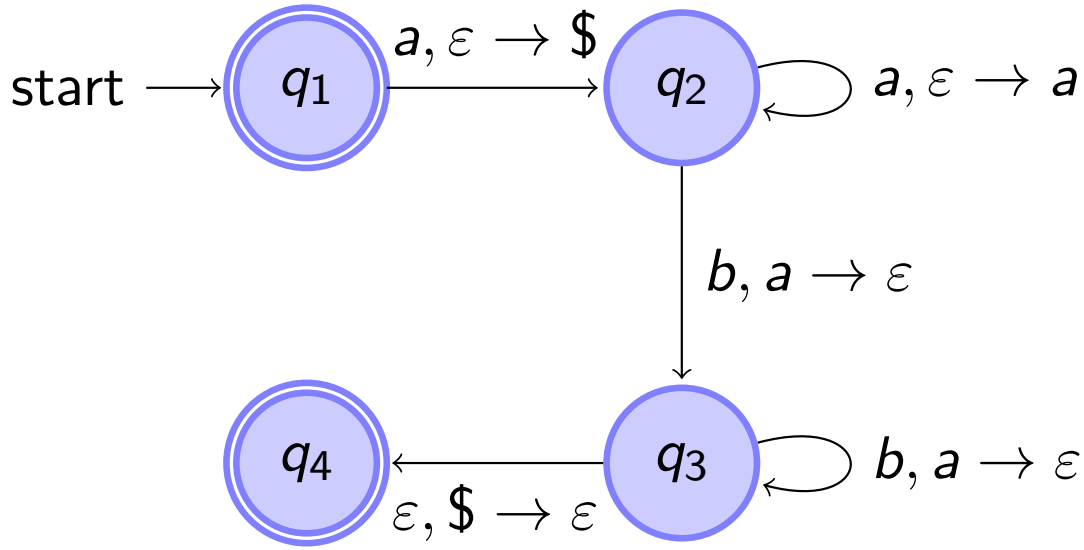
Rezumat

Gramatici regulate 2. Expresii regulate

Automat A



Automat B



Context free grammars - Gramatici independente de context. Automate stiva

March 31, 2024

$\{a^n b^n\}$ e regulat?

- Pt a fi limbaj regulat, ar trebui sa existe un automat finit care sa-l recunoasca.
- ar trebui sa tina minte cati a a citit, dar n nu este limitat
- dupa ce a citit a^m ar trebui sa fie intr-o stare ce specifica o multime de simboluri nonterminale din care sa fie derivate exact b^m. $\Rightarrow$ pt fiecare m ar trebui sa fie o stare distinctica
- deci automatul ar trebui sa aiba evidenta unui numar nelimitat de posibilitati

acest lucru nu se poate face cu un numar finit de stari

- dar, nu tot ce pare a avea nevoie de memorie nelimitata, chiar are:
  - $C = \{w | w$ are un numar egal de 0 si 1$\}$
  - $D = \{w | w$ are un numar egal de aparitii 01 si 10 $\}$

$\{a^n b^n\}$ e regulat?

- Pt a fi limbaj regulat, ar trebui sa existe un automat finit care sa-l recunoasca.
- ar trebui sa tina minte cati a a citit, dar n nu este limitat
- dupa ce a citit a^m ar trebui sa fie intr-o stare ce specifica o multime de simboluri nonterminale din care sa fie derivate exact b^m. $\Rightarrow$ pt fiecare m ar trebui sa fie o stare distinctica
- deci automatul ar trebui sa aiba evidenta unui numar nelimitat de posibilitati

acest lucru nu se poate face cu un numar finit de stari

- dar, nu tot ce pare a avea nevoie de memorie nelimitata, chiar are:
  - $C = \{w | w$ are un numar egal de 0 si 1$\}$
  - $D = \{w | w$ are un numar egal de aparitii 01 si 10 $\}$

D este limbaj regulat

$D = \{w | w$ are un numar egal de aparitii 01 si 10 ca substringuri$\}$

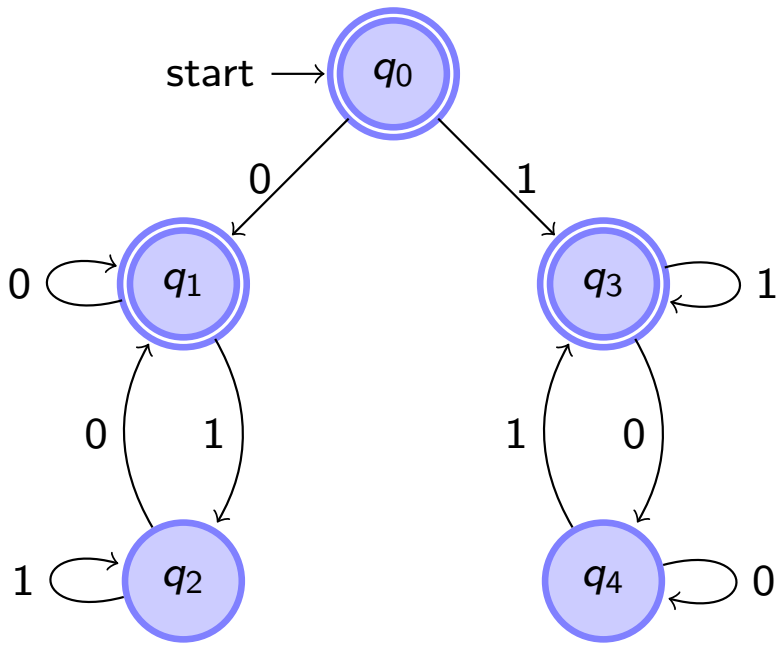
$D = \{0, 1, \varepsilon$

w daca incepe cu 0 se termina cu 0

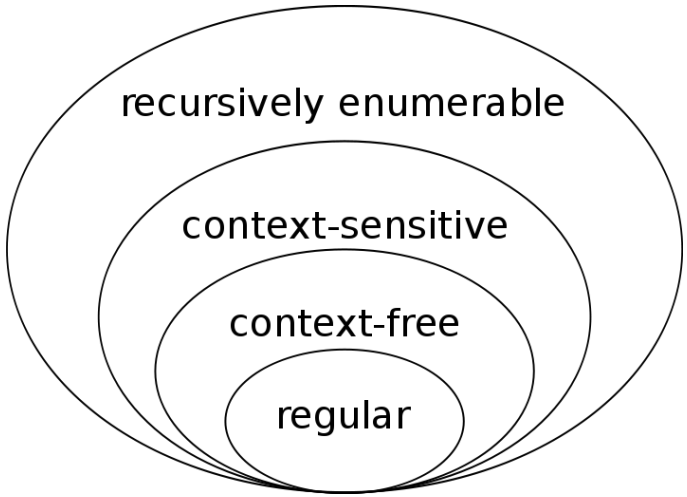
w daca incepe cu 1 se termina cu 1$\}$

? $101, 1010, 0110$

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



Gramatici independente de context. Context-free grammars



- $G = (T, N, Z, P)$ e independenta de context daca
- fiecare productie are forma

$$X \to \chi, X \in N, \chi \in V^*$$

Un limbaj care e definit de o gramatica independenta de context este limbaj independent de context.

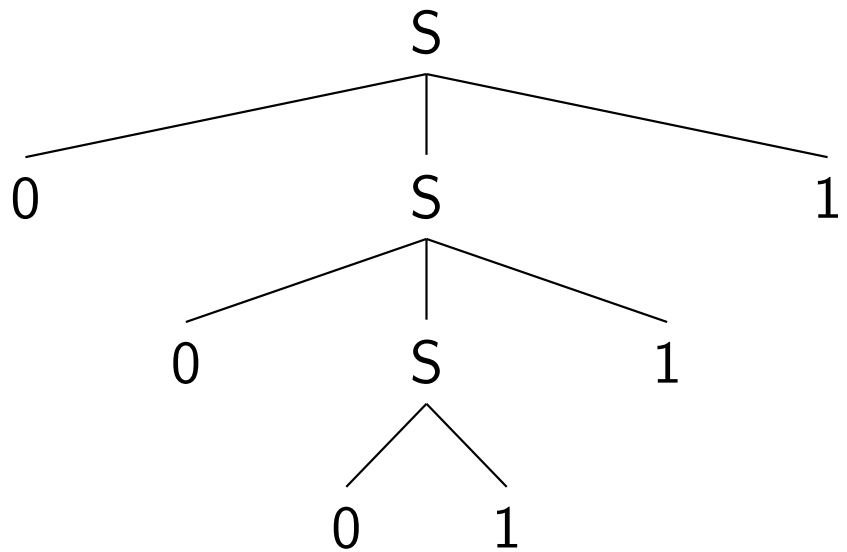
Exista CFL care nu sunt Regular languages

0^n1^n

Fie $G = (\{0,1\}, \{S\}, \{S \to 01|0S1\}, S)$

- $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000111$

Arbore de parsare pentru 000111:



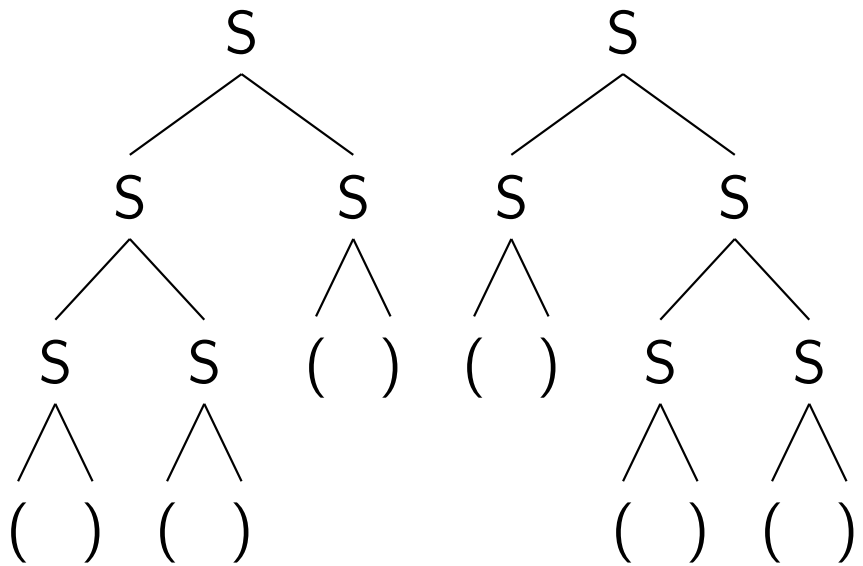
Limbajul parantezelor

Fie $G = (\{(,)\}, \{S\}, \{S \to SS|(S)|()\}, S)$

- left-most derivation: $S \Rightarrow \mathbf{S}S \Rightarrow (\mathbf{S})S \Rightarrow (())S \Rightarrow (())()$
  Deci $S \Rightarrow^L (())()$
- right-most derivation: $S \Rightarrow S\mathbf{S} \Rightarrow \mathbf{S}() \Rightarrow (S)() \Rightarrow (())()$
  Deci $S \Rightarrow^R (())()$

Gramatica ambigua - reamintire

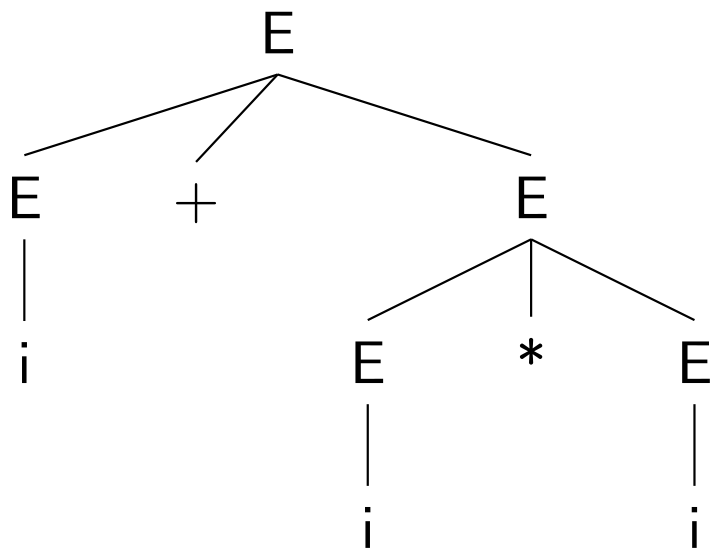
O gramatica e ambigua daca exista un string in limbaj care e parsat in doi arbori de derivare.
Pentru $G = (\{(,)\}, \{S\}, \{S \to SS|(S)|()\}, S)$
Derivarea $()()()$:

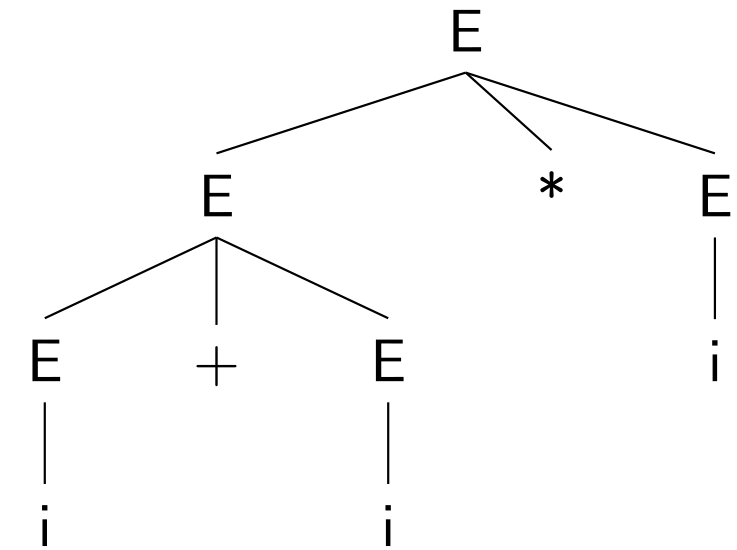


Ambiguitate

Fie $G_4 = (\{+, *, i\}, \{E\}, E, P)$ Doua derivari distincte stanga, doua derivari distincte dreapta

- $E \to E + E$
- $E \to E * E$
- $E \to i$





2+3*4??

CFG pentru Engleza

- $T = \{eats, saw, man, woman, telescope, the, with, at\}$
- $N = \{S, NP, VP, PP, DT, Vi, Vt, NN, IN\}$
- P=

S	→	NP VP
VP	→	Vi
VP	→	Vt NP
VP	→	VP PP
NP	→	DT N
NP	→	NP PP
PP	→	IN NP

Vi	→	eats
Vt	→	saw
N	→	man
N	→	woman
N	→	telescope
DT	→	the
IN	→	with
IN	→	at

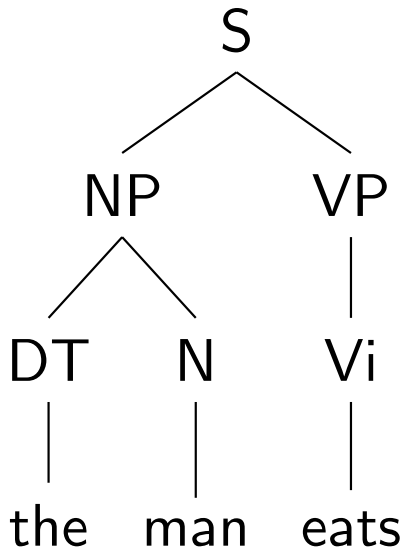
S = sentence, VP = verb phrase, NP = noun phrase, PP = prepositional phrase, DT = determiner, Vi = intransitive verb, Vt = transitive verb, N = noun, IN = preposition

Arbore de derivare

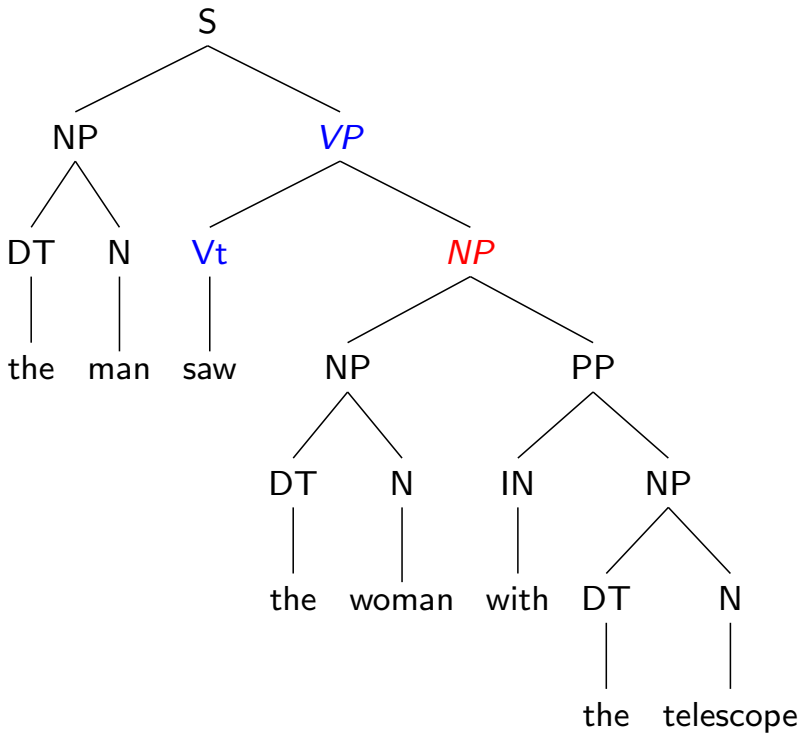
Derivarea stanga:

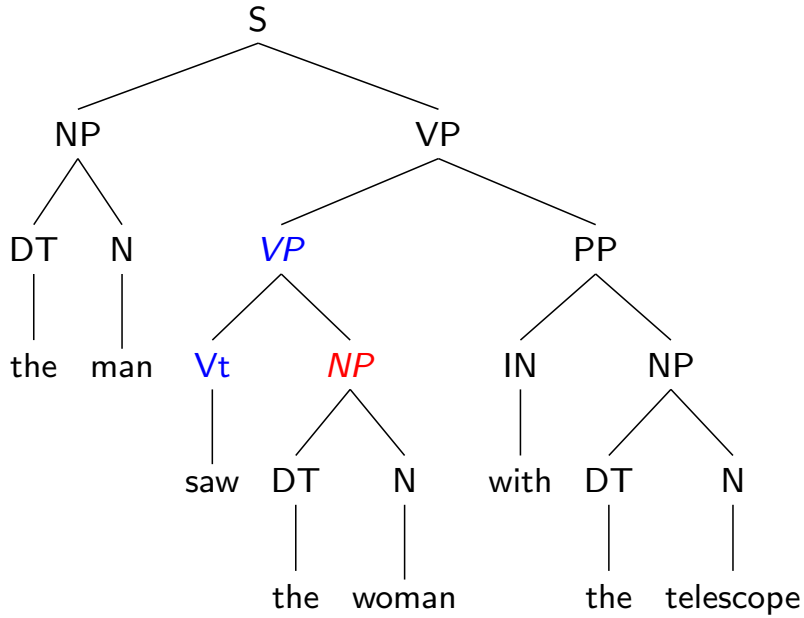
$S \Rightarrow \mathbf{NP}\ VP \Rightarrow \mathbf{DT}\ N\ VP \Rightarrow the\ \mathbf{N}\ VP \Rightarrow the\ man\ \mathbf{VP}$

$\Rightarrow the\ man\ \mathbf{Vi} \Rightarrow the\ man\ eats$



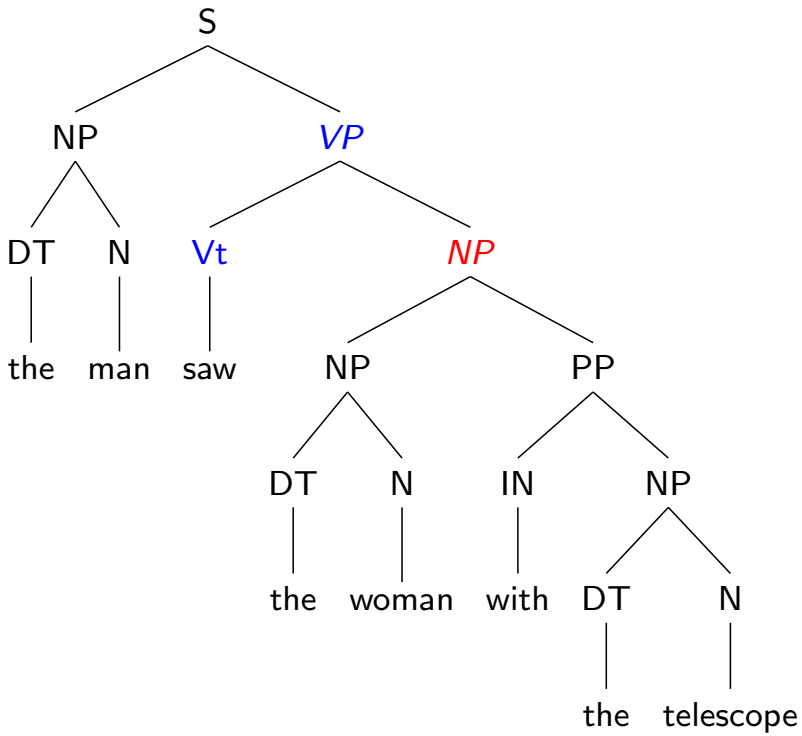
The man saw the woman with the telescope.

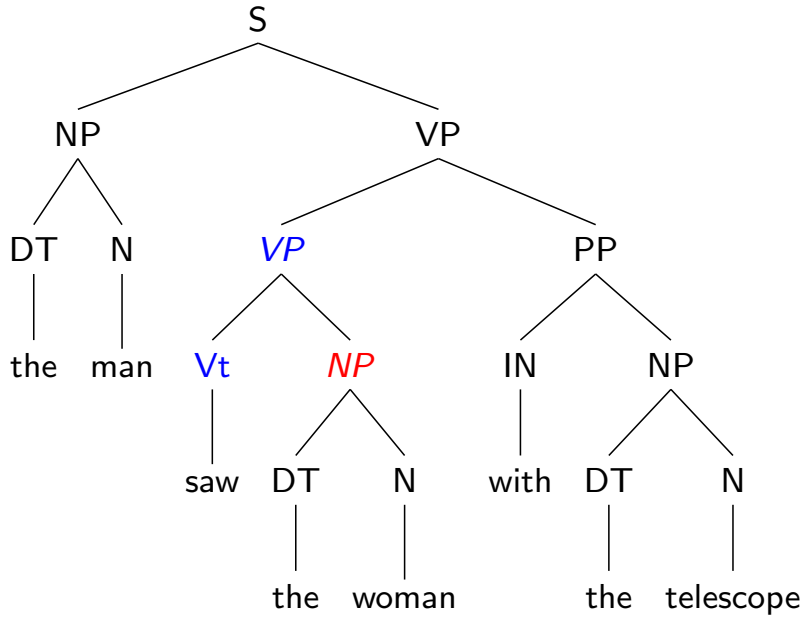




Ce a vazut “the man”?

The man saw the woman with the telescope.





Ce a vazut "the man"?

The telescope at the man saw the woman $\in ?\ L(G)$

if then else grammar

- $T = \{if, then, else, E1, E2, S1, S2, S3\}$, $N = \{stmt, expr\}$
- P=

$$
\begin{array}{lcl}
stmt & \rightarrow & \text{if } expr \text{ then } stmt \\
stmt & \rightarrow & \text{if } expr \text{ then } stmt \text{ else } stmt \\
stmt & \rightarrow & \text{S1} \mid \text{S2} \mid \text{S3} \\
expr & \rightarrow & \text{E1} \mid \text{E2}
\end{array}
$$

if then else grammar

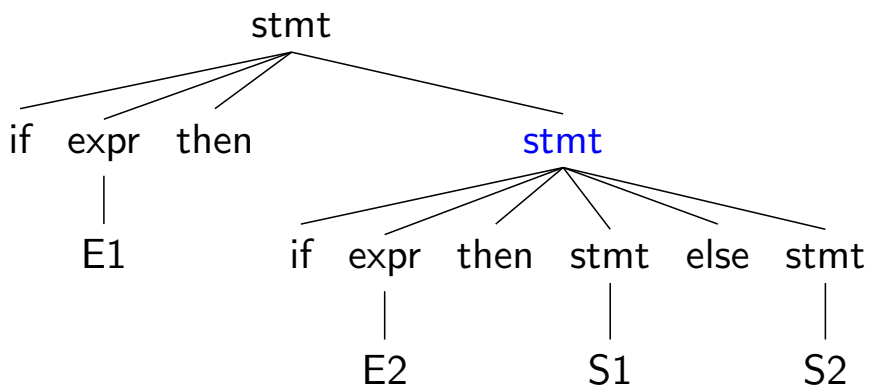
- $T = \{if, then, else, E1, E2, S1, S2, S3\}$, $N = \{stmt, expr\}$
- P=

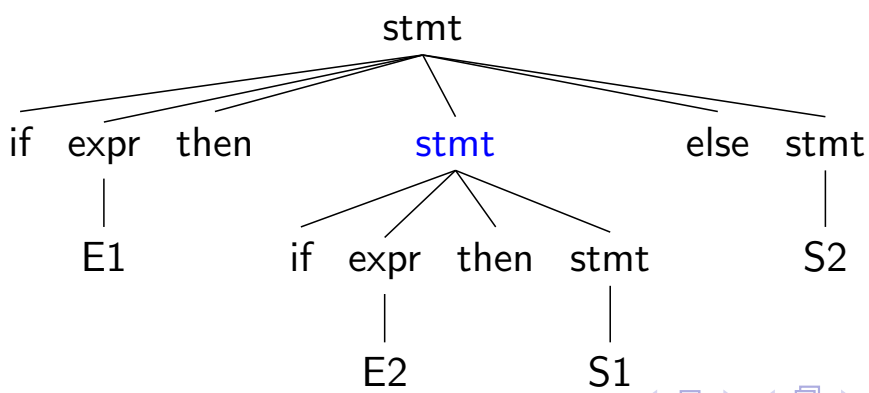
$$
\begin{array}{lcl}
stmt & \rightarrow & \text{if } expr \text{ then } stmt \\
stmt & \rightarrow & \text{if } expr \text{ then } stmt \text{ else } stmt \\
stmt & \rightarrow & S1 \mid S2 \mid S3 \\
expr & \rightarrow & E1 \mid E2
\end{array}
$$

```
if E1
then    if E2
        then S1
        else S2
```

sau?

```
if E1
then        if E2
            then S1
else S2
```





if then else - rezolvare ambiguitate

- $T = \{if, then, else, E1, E2, S1, S2, S3\}$,
  $N = \{stmt, matched\_stmt, unmatched\_stmt, expr\}$
- P=

stmt	$\rightarrow$	*m_stmt*
	|	*um_stmt*
m_stmt	$\rightarrow$	if *expr* then *m_stmt* else *um_stmt*
	|	*smtm1*
um_stmt	$\rightarrow$	if *expr* then *stmt*
	|	if *expr* then *m_stmt* else *um_stmt*
stmt1	$\rightarrow$	S1 | S2 | S3
expr	$\rightarrow$	E1 | E2

Intre un *then* si un *else* e permis doar *matched_stmt*.
m_stmt=*matched_stmt* (if cu ambele *then* si *else*),
um_stmt=*unmatched_stmt*

Exemple de gramatici: Liste de *stmt*

- Recursivitate dreapta

  $$\begin{array}{lcl} \textit{stmt\_list} & \rightarrow & \textit{stmt}\ \textit{stmt\_list} \\ & | & \text{stmt} \\ \text{stmt} & \rightarrow & \text{A}|\text{B}|\text{C} \end{array}$$

  Arbore pt A B C?

- Recursivitate stanga

  $$\begin{array}{lcl} \textit{stmt\_list} & \rightarrow & \textit{stmt\_list}\ \textit{stmt} \\ & | & \text{stmt} \\ \text{stmt} & \rightarrow & \text{A}|\text{B}|\text{C} \end{array}$$

Exemple de gramatici: Liste de elemente cu separator/marcaj de final

- Separator intre elemente

  $$\begin{array}{lcl} \textit{elem\_list} & \rightarrow & \textit{elem\_list}\ \ \textit{','}\ \ \textit{elem} \\ & | & \text{elem} \\ \text{elem} & \rightarrow & \text{A}|\text{B}|\text{C} \end{array}$$

  Arbore pt A, B, C?

- Semn de punctuatie la final

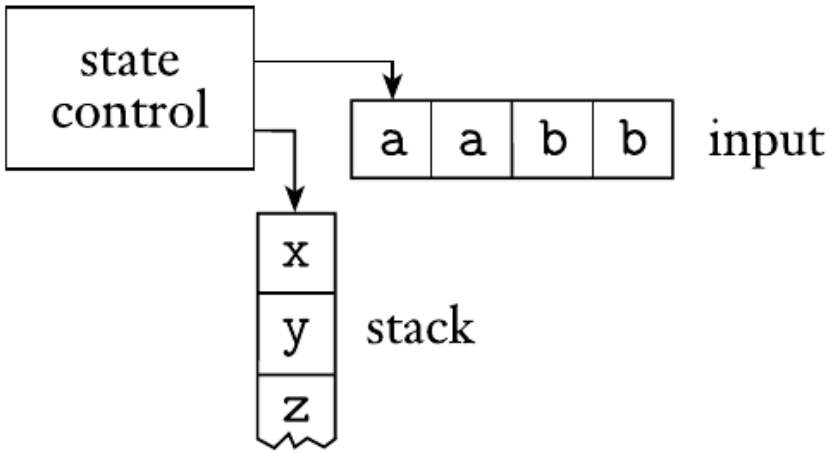
  $$\begin{array}{lcl} \textit{stmt\_list} & \rightarrow & \textit{stmt\_list}\ \textit{stmt}\ \ \textit{';'} \\ & | & \text{stmt ';'} \\ \text{stmt} & \rightarrow & \text{A}|\text{B}|\text{C} \end{array}$$

  Arbore pt A; B; C;?

Letia and Chifu. 2.3, 2.3.1 Sipser - 2.1,2.2

Automat stiva. Push down automaton (PDA)

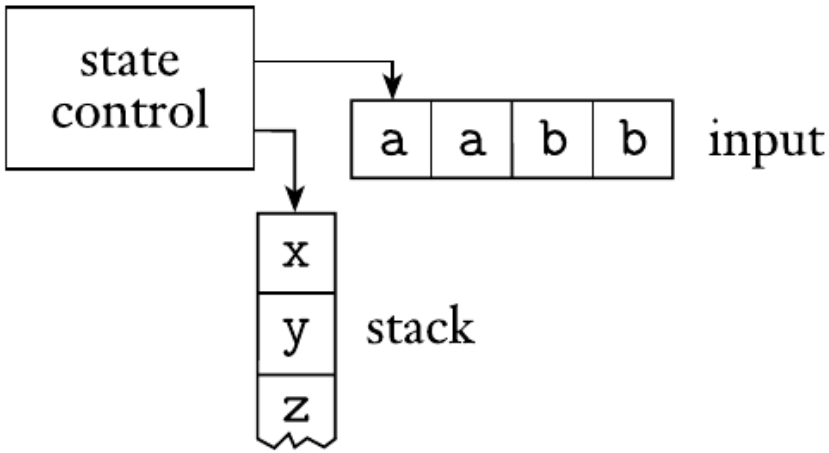
- Niciun automat finit nu poate fi construit pt a recunoaste $a^n b^n$ sau limbajul parantezelor - structuri imbricate
- Se creste puterea automatelor finite prin adaugarea unei stive drept structura aditionala de memorie



- Daca gramaticile regulate sunt o subclasa a gramaticilor independente de context, de ce se dezvolta metode specifice gramaticilor regulate si nu se aplica pt acestea cele de la gramaticile independente?

Automat stiva. Push down automaton (PDA)

- Niciun automat finit nu poate fi construit pt a recunoaste $a^n b^n$ sau limbajul parantezelor - structuri imbricate
- Se creste puterea automatelor finite prin adaugarea unei stive drept structura aditionala de memorie



- Daca gramaticile regulate sunt o subclasa a gramaticilor independente de context, de ce se dezvolta metode specifice gramaticilor regulate si nu se aplica pt acestea cele de la gramaticile independente?
- Datorita complexitatii analizei gramaticilor independente de context: gramaticile regulate sunt mai simplu de analizat

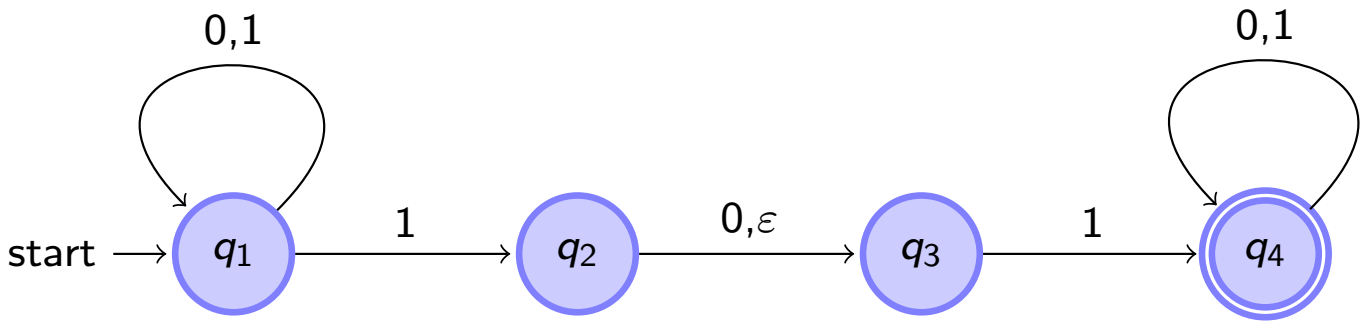
Idee: Push down automata: 00001111

1. citeste simboluri de la intrare
2. la fiecare 0 citit, impinge-l pe stiva
3. la fiecare 1 citit, scoate de pe stiva un 0
4. daca citirea stringului se termina cand stiva se goleste, accepta stringul. Daca stiva devine goala cand mai sunt 1 de citit sau s-a terminat sirul si in stiva inca mai sunt 0-uri, respinge stringul

NFA- reamintire

Un automat finit nedeterminist este $(Q, \Sigma, \delta, q_0, F)$, unde:

1. Q este setul de stari
2. Σ un alfabet finint de intrare
3. $\delta : Q \times \Sigma_\varepsilon \to P(Q)$ este o functie de tranzitie
4. $q_0 \in Q$ este starea de start
5. $F \subseteq Q$ setul de stari finale



$\delta(q_1, 1) = \{q_1, q_2\}$

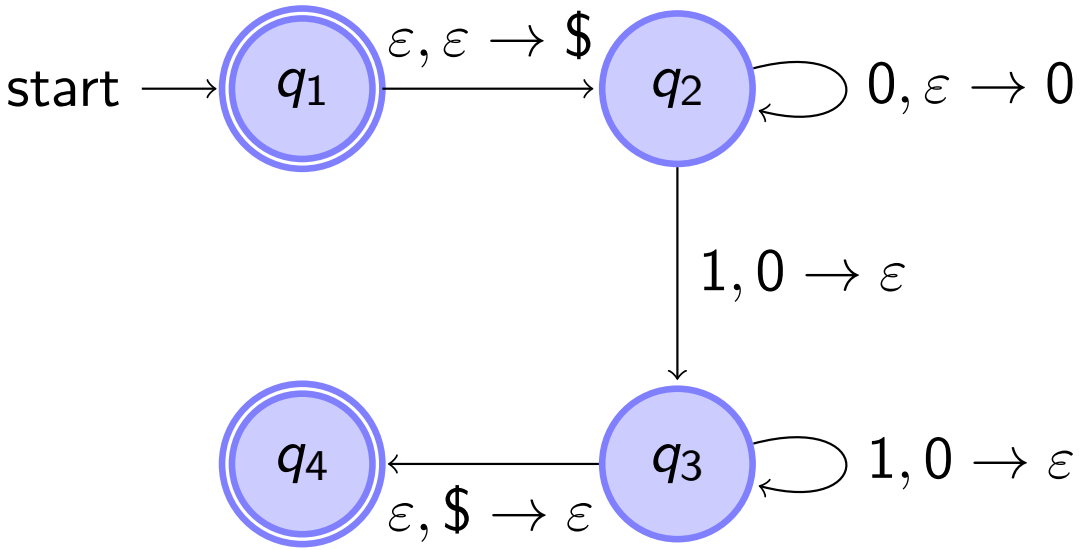
Definitie formala 1 a automatului stiva (Sipser)

Un automat stiva este $(Q, \Sigma, \Gamma, \delta, q_0, F)$, unde Q, Σ, Γ, F sunt seturi finite:

1. Q este setul de stari
2. Σ un alfabet de intrare
3. Γ este alfabetul stivei
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$ este o functie de tranzitie
5. $q_0 \in Q$ este starea de start
6. $F \subseteq Q$ setul de stari finale

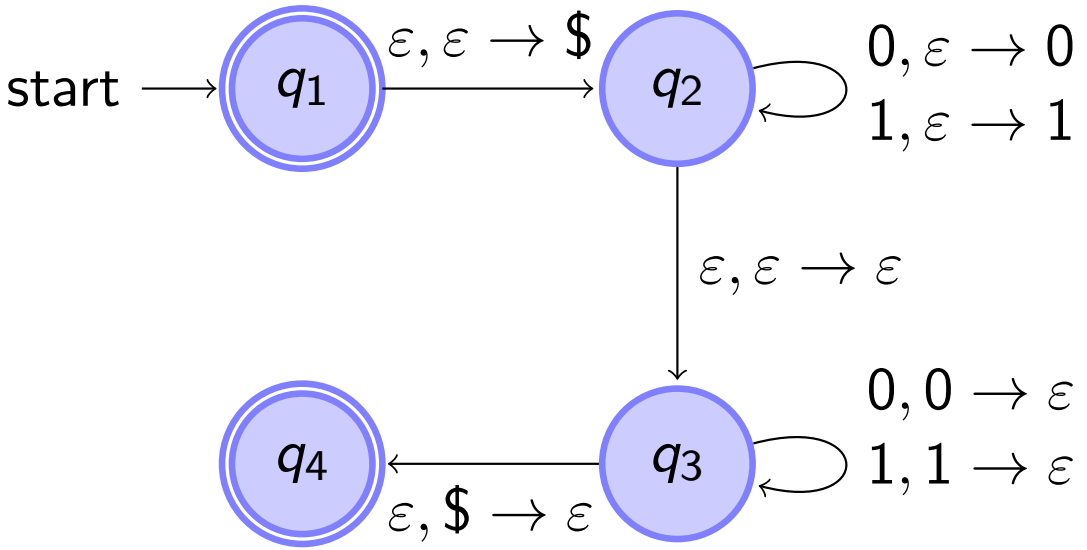
PDA for 0^n1^n

Fie $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, F)$



PDA for ?

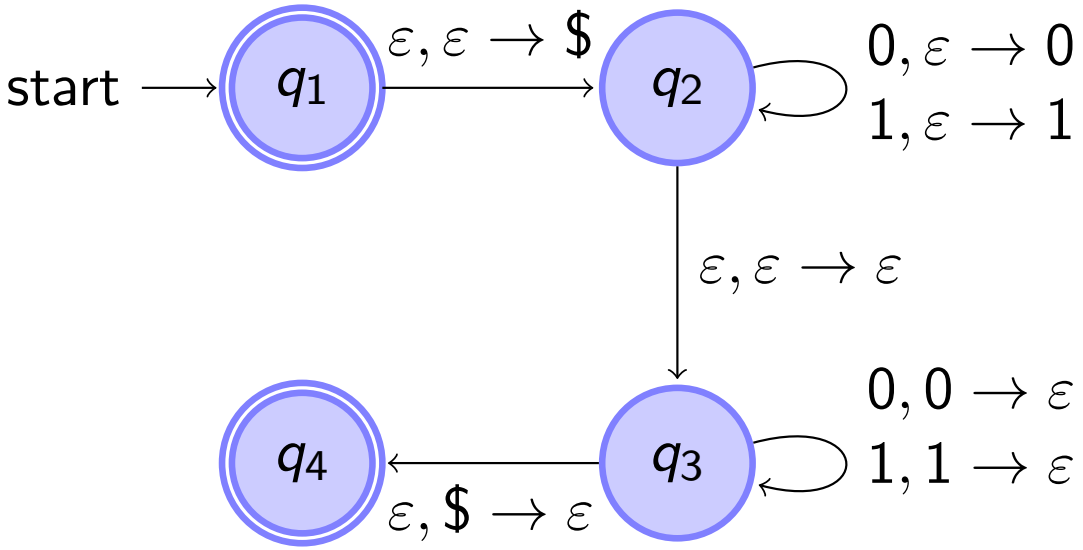
Fie $M_1 = (Q, \{0,1\}, \{\$, 0, 1\}, \delta, q_1, \{q_1, q_4\})$



PDA for $\{ww^R | w \in \{0,1\}^*\}$

$w^R = w$ scris invers

Fie $M_1 = (Q, \{0,1\}, \{\$, 0, 1\}, \delta, q_1, \{q_1, q_4\})$



la fiecare pas, ghiceste daca a ajuns la mijlocul stringului sau nu

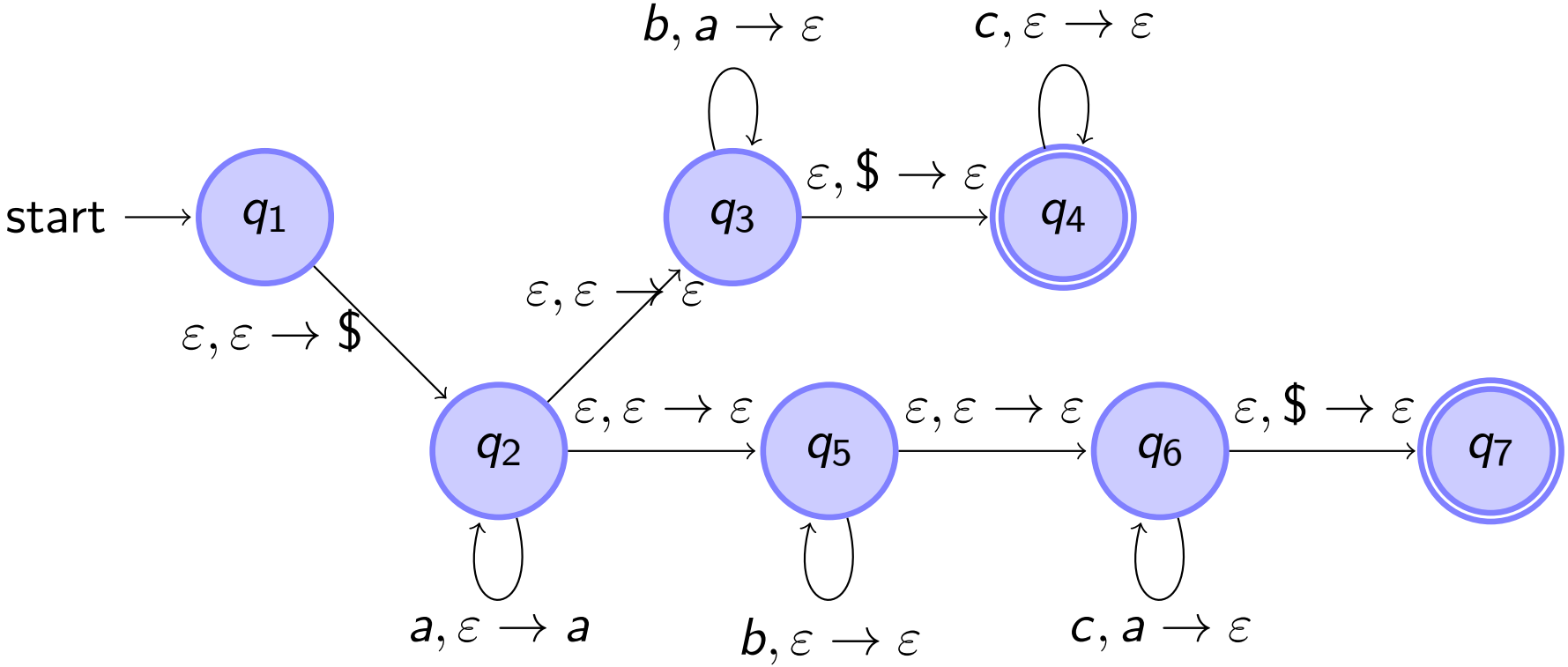
Gramatica palindrom par

- Palindrom: (T, N, P, A), $P =$

$$\{A \to 0A0|1A1$$
$$A \to \varepsilon\}$$

PDA for $\{a^i b^j c^k | i, j, k \geq 0, i = j \text{ sau } i = k\}$

Fie $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, F)$



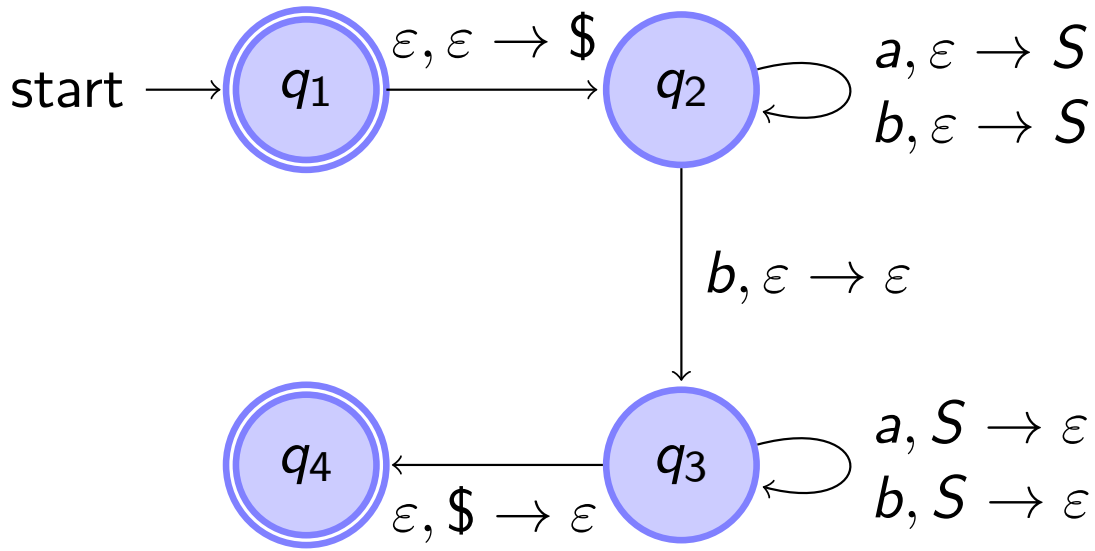
ghiceste daca e acelasi numar de a si b sau a si c

$\{vbw | v, w \in \{a, b\}^*, |v| = |w|\}$

$M = (Q, \{a, b\}, \{\$, S\}, \delta, q_1, \{q_4\})$

$\{vbw | v, w \in \{a, b\}^*, |v| = |w|\}$

$M = (Q, \{a, b\}, \{\$, S\}, \delta, q_1, \{q_4\})$



Exemplu Gramatica independenta de context

- Palindrom: (T, N, P, A), $P =$

$$\{A \to 0A0|1A1$$
$$A \to \varepsilon\}$$

- Acelasi numar de 0 si 1: $(\{0,1\}, \{A\}, P, A)$, $P =$

$$\{A \to 0A1A|1A0A$$
$$A \to \varepsilon\}$$

Automat finit - reamintire

Automat finit (finite automaton, finite state acceptor):

$$A = (T, Q, R, q_0, F)$$

- Q set nevid - setul starilor interne
- $(T \cup Q, R)$ sistem de rescriere; $T \cap Q = \emptyset$
- $q_0 \in Q$ - starea initiala
- $F \subseteq Q$ - stari finale
- fiecare element din R are forma $qt \to q'$, $q, q' \in Q, t \in T$

$$L(A) = \{\tau \in T^* | q_0\tau \Rightarrow^* q, q \in F\}$$

Automat stiva - definitie sistem de rescriere

Automat stiva

$$A = (T, Q, R, q_0, F, S, s_0)$$

, unde:

- Q set nevid - setul starilor interne
- $(T \cup Q \cup S, R)$ sistem de rescriere; $T \cap Q = \emptyset$
- $q_0 \in Q$ - starea initiala
- $s_0 \in S \cup \{\varepsilon\}$ - simboluri stiva, s_0 continutul initial al stivei
- $F \subseteq Q$ - stari finale
- fiecare element din R are forma $\sigma q t \tau \rightarrow \sigma' q' \tau$, $\sigma, \sigma' \in S^*,\ q, q' \in Q, t \in T \cup \varepsilon, \tau \in T^*$

Daca automatul e la configuratia $s_1 ... s_n q \tau$ intr-o derivare, automatul e in starea q, τ este partea necitita din input, $s_1, ..., s_n$ este continutul pe stiva, s_n in varf.

Limbaj acceptat

Daca automatul e la configuratia $s_1 \ldots s_n q \tau$ intr-o derivare, automatul e in starea q, τ este partea necitita din input, $s_1, \ldots, s_n$ este continutul pe stiva, s_n in varf.

$$L(A) = \{\tau | s_0 q_0 \tau \# \Rightarrow^* q\#, q \in F, \tau \in T^*\}$$

0^n1^n

$M_1 = (\{0,1\}, \{q_2, q_3\}, R, q_2, \{q_3\}, \{0,1\}, \varepsilon)$, $R = \{$

1. $\varepsilon q_2 0 \rightarrow 0 q_2$
2. $0 q_2 1 \rightarrow \varepsilon q_3$
3. $0 q_3 1 \rightarrow \varepsilon q_3\}$

$$??\varepsilon q_2 0011 \Rightarrow^* q_3$$

Pe stiva pot fi alte simboluri decat cele din alfabetul de intrare.

CFG - PDA

Pentru fiecare gramatica independenta de context G exista un automat stiva A a.i. L(A)=L(G).

exemplu 2

Fie $G_1 = (T, N, E, P)$

- $T = \{+, *, (,), i\}$, $N = \{E, T, F\}$
- cu productiile P
  - $(1, 2) E \to T | E + T$
  - $(3, 4) T \to F | T * F$
  - $(5, 6) F \to i | (E)$

Automatul stiva construit pentru analiza descendenta:

- $T = \{+, *, (,), i\}$, $Q = \{q\}$,
  $q_0 = q, F = \{q\}, S = \{+, -, *, (,), i, E, T, F\}, s_0 = E$
- cu productiile R
  1. $Eq \to Tq, Eq \to T + Eq$,
  2. $Tq \to Fq, Tq \to F * Tq$,
  3. $Fq \to iq, Fq \to)E(q$,
  4. $+q+ \to q, *q* \to q$, $(q(\to q,)q) \to q, iqi \to q\}$

Derivarea gasita: i+i*i

stiva	stare	intrare	derivarea cea mai din stanga
E	q	$i + i * i$	E
T+E	q	$i + i * i$	E+T
T+T	q	$i + i * i$	T+T
T+F	q	$i + i * i$	F+T
T+i	q	$i + i * i$	i+T
T+	q	$+i * i$	
T	q	$i * i$	
F*T	q	$i * i$	i+T*F
F*F	q	$i * i$	i+F*F
F*i	q	$i * i$	i+i*F
F*	q	$*i$	
F	q	i	
i	q	i	i+i*i
	q		

Exemplu

Fie $G_1 = (T, N, E, P)$

- $T = \{+, *, (,), i\}$, $N = \{E, T, F\}$
- cu productiile P
  - $(1, 2)E \to T | E + T$
  - $(3, 4)T \to F | T * F$
  - $(5, 6)F \to i | (E)$

Automatul stiva:

- $T = \{+, *, (,), i\}$, $Q = \{q\}$,
  $q_0 = q, F = \{q\}, S = \{+, -, *, (,), i, E, T, F\}, s_0 = E$
- cu productiile R
  1. $Tq \to Eq, E + Tq \to Eq$,
  2. $Fq \to Tq, T * Fq \to Tq$,
  3. $iq \to Fq, (E)q \to Fq$,
  4. $q+ \to +q, q* \to *q$, $q(\to (q, q) \to q), qi \to iq\}$
  5. $Eq \to q\}$

Derivarea gasita: i+i*i

stiva	stare	intrare	derivarea cea mai din dreapta
	q	$i + i * i$	i+i*i
i	q	$+i * i$	
F	q	$+i * i$	F+i*i
T	q	$+i * i$	T+i*i
E	q	$+i * i$	E+i*i
E+	q	$i * i$	
E+i	q	$*i$	
E+F	q	$*i$	E+F*i
E+T	q	$*i$	E+T*i
E+T*	q	i	
E+T*i	q	i	
E+T*F	q		
E+T	q		E+T*F
E	q		E+T
	q		E

Analiza descendenta - prima parte

April 14, 2024

Automat finit - reamintire

Automat finit (finite automaton, finite state acceptor):

$$A = (T, Q, R, q_0, F)$$

- Q set nevid - setul starilor interne
- $(T \cup Q, R)$ sistem de rescriere; $T \cap Q = \emptyset$
- $q_0 \in Q$ - starea initiala
- $F \subseteq Q$ - stari finale
- fiecare element din R are forma $qt \to q'$, $q, q' \in Q, t \in T$

$$L(A) = \{\tau \in T^* | q_0\tau \Rightarrow^* q, q \in F\}$$

Automat stiva - definitie sistem de rescriere -reamintire

Automat stiva

$$A = (T, Q, R, q_0, F, S, s_0)$$

, unde:

- Q set nevid - setul starilor interne
- $(T \cup Q \cup S, R)$ sistem de rescriere; $T \cap Q = \emptyset$
- $q_0 \in Q$ - starea initiala
- $s_0 \in S \cup \{\varepsilon\}$ - simboluri stiva, s_0 continutul initial al stivei
- $F \subseteq Q$ - stari finale
- fiecare element din R are forma $\sigma q t \tau \rightarrow \sigma' q' \tau$, $\sigma, \sigma' \in S^*,\ q, q' \in Q, t \in T \cup \varepsilon, \tau \in T^*$

Daca automatul e la configuratia $s_1 ... s_n q \tau$ intr-o derivare, automatul e in starea q, τ este partea necitita din input, $s_1, ..., s_n$ este continutul pe stiva, s_n in varf.

Limbaj acceptat

Daca automatul e la configuratia $s_1 \ldots s_n q \tau$ intr-o derivare, automatul e in starea q, τ este partea necitita din input, $s_1, \ldots, s_n$ este continutul pe stiva, s_n in varf.

$$L(A) = \{\tau | s_0 q_0 \tau \Rightarrow^* q, q \in F, \tau \in T^*\}$$

CFG - PDA

Pentru fiecare gramatica independenta de context G exista un automat stiva A a.i. L(A)=L(G).

Parsing

- Rolul parsarii: reconstruirea derivarii prin care o CFG poate genera un input string dat.
- Echivalent cu construirea arborelui de parsare care reprezinat derivarea
- Directii:
  - Top-Down - constructia incepe de la radacina; derivarea stanga
  - Bottom-up - constructia incepe de la frunze; mai greu de construit manual, dar pot fi generate

Rezumat

Recursive descent parsing

Predictive parsing

Structuri ajutatoare: FIRST, FOLLOW

Definire Gramatici LL(k)

Algoritmul LL(K)

- Exemplu aplicare LL(k)
- Exemplu aplicare LL(3)

1. Recursive descent - top down parse

Arborele de parsare e construit:

- de la simbolul de start
- de la stanga la dreapta
- se incearca regulile in ordinea in care apar
- revenire si incercare alternative

$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

Parse tree pt:

(int)

↑

E

|

T

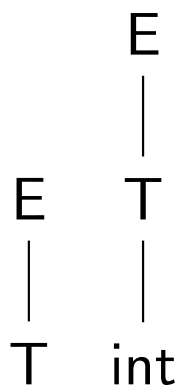
$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

Parse tree pt:

(int)

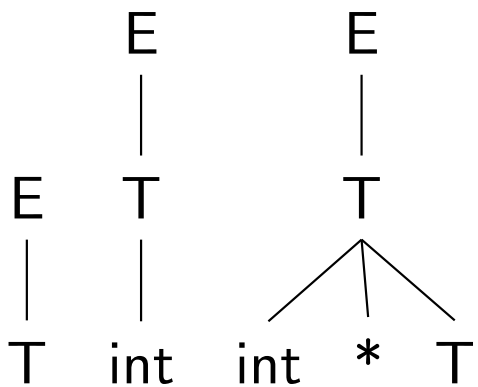
↑



$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

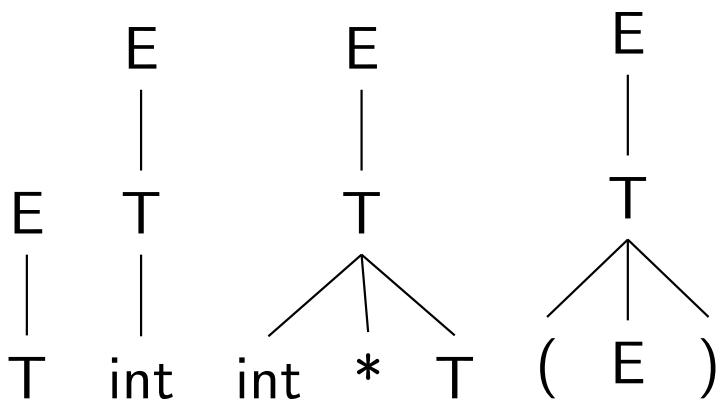
Parse tree pt: (int)



$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

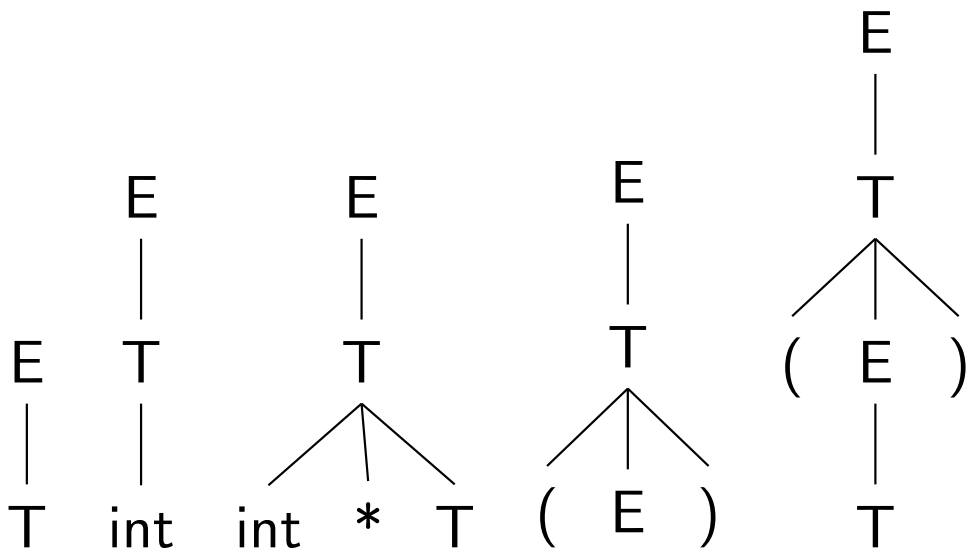
Parse tree pt: (int)



$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

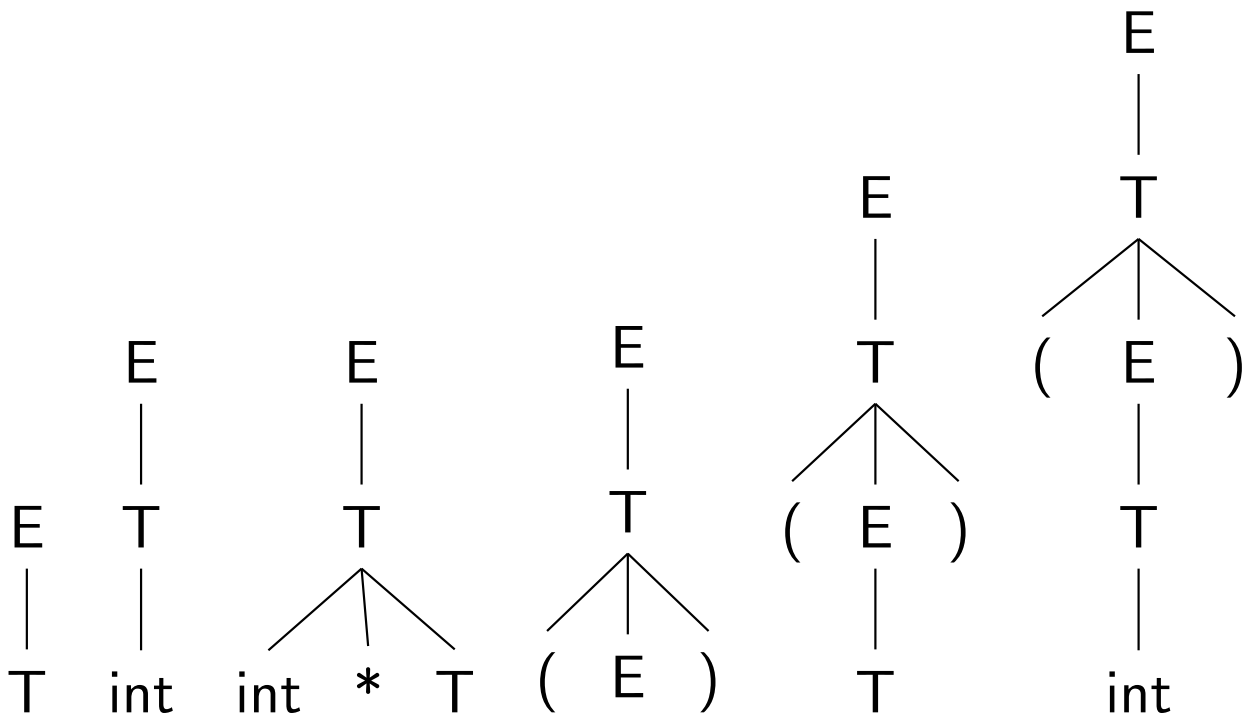
Parse tree pt: (int) ↑



$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

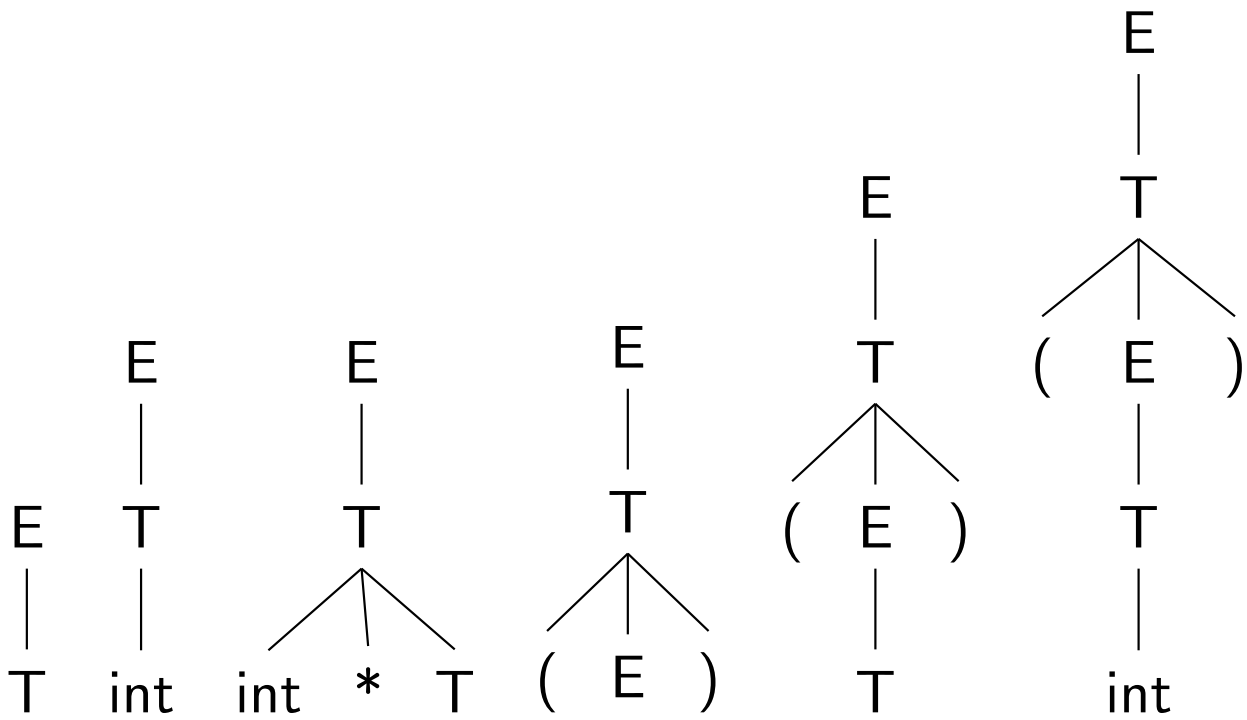
Parse tree pt: (int) ↑



$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

Parse tree pt: (int)



Recursive-descent parser - nestiind care dintre productiile alternative pt un nonterminal trebuie aplicata, exista posibilitatea de esec

- Predictive parser: dat fiind sirul de intrare a (primul din sirul ramas) si nonterminalul A care trebuie expandat, am putea determina care productie alternativa e cea care deriveaza stringul ramas dupa a
  idee: alternativa corecta trebuie detectata uitandu-ne inainte la k simboluri din stringul care trebuie derivate: $LL(1)$ si $LR(1)$

2. Parser predictiv[1]

Fie $G = (T, N, P, Z)$ o CFG si automatul stiva
$A = (T, \{q\}, R, q, \{q\}, V, Z)$ cu $V = T \cup N$ si R:
(alfabet, stari, productii, stare initiala, stari finale, alfabet stiva, continut initial stiva)

$$\{tqt \to q | t \in T\} \cup$$
$$\{Xq \to x_n....x_1 q | X \to x_1 x_2 ... x_n \in P, n \geq 0, X \in N, X_i \in V\}$$

Automatul accepta un sir din L(G) prin

- construirea unei derivari cea mai din stanga a acelui sir si
- compararea simbolurilor generate (de la stanga la dreapta) si incarcate pe stiva cu simbolurile care apar in sir.

[1] Letia& Chifu 4.2

exemplu 1

Fie $G_1 = (T, N, S, P)$

- $T = \{+, (,), i\}$, $N = \{S, F\}$
- cu productiile P
  - $S \to F$
  - $S \to (S + F)$
  - $F \to i$

Care e automatul pentru analiza descendenta?
Care e derivarea stanga pentru $(i + i)$?
Automatul accepta $(i + i)$?

exemplu 2

Fie $G_1 = (T, N, E, P)$

- $T = \{+, *, (,), i\}$, $N = \{E, T, F\}$
- cu productiile P
  - $(1, 2) E \to T | E + T$
  - $(3, 4) T \to F | T * F$
  - $(5, 6) F \to i | (E)$

Automatul stiva construit pentru analiza descendenta:

- $T = \{+, *, (,), i\}$, $Q = \{q\}$,
  $q_0 = q, F = \{q\}, S = \{+, -, *, (,), i, E, T, F\}, s_0 = E$
- cu productiile R
  1. $Eq \to Tq, Eq \to T + Eq$,
  2. $Tq \to Fq, Tq \to F * Tq$,
  3. $Fq \to iq, Fq \to)E(q$,
  4. $+q+ \to q, *q* \to q$, $(q(\to q,)q) \to q, iqi \to q\}$

Derivarea gasita: i+i*i

stiva	stare	intrare	derivarea cea mai din stanga
E	q	$i + i * i$	E
T+E	q	$i + i * i$	E+T
T+T	q	$i + i * i$	T+T
T+F	q	$i + i * i$	F+T
T+i	q	$i + i * i$	i+T
T+	q	$+i * i$	
T	q	$i * i$	
F*T	q	$i * i$	i+T*F
F*F	q	$i * i$	i+F*F
F*i	q	$i * i$	i+i*F
F*	q	$*i$	
F	q	i	
i	q	i	i+i*i
	q		

Exemplul 3

Fie $G_1 = (T, N, E, P)$

- $T = \{a, b, c\}$, $N = \{Z, X, Y\}$
- cu productiile P
  - (1) $Z \to X$
  - (2,3) $X \to Y|bYa$
  - (4,5) $Y \to c|ca$

Automatul $(\{a, b, c\}, \{q\}, R, q, \{q\}, \{a, b, c, X, Y, Z\}, Z)$:

- $aqa \to q$
- $bqb \to q$
- $cqc \to q$
- $Zq \to Xq$
- $Xq \to Yq$
- $Xq \to aYbq$
- $Yq \to cq$
- $Yq \to acq$

bcaa?

3. No backtracking

Analiza descendenta sau predictiva - traseaza derivarea de la simbolul de start la propozitie, prezicand simbolurile care trebuie sa fie prezente.

- stiva precizeaza sirul din V^* utilizat pentru derivarea restului sirului de la intrare
- automat stiva determinist: pentru gramatici $LL(k)$

Asumptii si structuri ajutatoare: CFG

Presupunem ca CFG (T, N, P, Z) contin

- $Z \to S$ singura in care apare Z - daca nu exista o introducem
- fiecare propozitie se termina cu $\#$ - indica finalul propozitiei
- productia i are forma

$$X_i \to \chi_i, \ \textit{unde} \ \chi_i = x_{i,1}x_{i,2}...x_{i,m}$$

- $k:\omega$ primele $min(k, |\omega|+1)$ simboluri din $\omega\#$

$$k:\omega = \begin{cases} \omega\#, & daca\ |\omega| < k \\ \alpha, & daca\ \omega = \alpha\gamma\ si\ |\alpha| = k \end{cases}$$

- $FIRST_k(\omega)$ setul tuturor capetelor $k:\omega$ terminale ale sirurilor derivabile din ω

$$FIRST_k(\omega) = \{\tau | \exists \nu \in T^*\ a.i.\ \omega \Rightarrow^* \nu, \tau = k:\nu\}$$

- $EFF_k(\omega)$ ($\varepsilon - free\ first$, primul fara ε) - toate sirurile din $FIRST_k(\omega)$ pentru care nu s-a aplicat nicio productie ε in ultimul pas din derivarea cea mai din dreapta

$$EFF_k(\omega) = \{\tau \in FIRST_k(\omega) | \nexists A \in N, \nu \in T^*\ a.i.\ \omega \Rightarrow^R A\tau\nu \Rightarrow \tau\nu\}$$

- $FOLLOW(\omega)$ captele k terminale care ar putea urma lui ω; $FOLLOW_k(Z) = \{\#\}$

$$FOLLOW_k(\omega) = \{\tau | \exists \nu \in T^*\ a.i.\ Z \Rightarrow^* \mu\omega\nu, \tau \in FIRST_k(\nu)\}$$

Exemplu de valori FIRST, FOLLOW pt $k = 1$

- $T = \{id, *, +, (,)\}$, $N = \{E, E', T, T', F\}$
- cu productiile P
  - $Z \to E$
  - $E \to TE'$
  - $E' \to +TE'|\varepsilon$
  - $T \to FT'$
  - $T' \to *FT'|\varepsilon$
  - $F \to (E)|id$

simbol	$FIRST_1(X)$	$FOLLOW_1(X)$
E	$\{(, id\}$	$\{), \#\}$
E'	$\{+, \varepsilon\}$	$\{), \#\}$
T	$\{(, id\}$	$\{+, \#,)\}$
T'	$\{*, \varepsilon\}$	$\{+, \#,)\}$
F	$\{(, id\}$	$\{*, +, \#,)\}$

Exemplu

$$E \Rightarrow TE' \Rightarrow FT'E' \Rightarrow (E)T'E' \Rightarrow^{+} (id) * FT'E' \Rightarrow$$

$$(id) * F * T' + TE' \Rightarrow (id) * id * id + id$$

Exemplu de valori FIRST, FOLLOW pt $k = 1$

- $T = \{id, *, +, (,)\}$, $N = \{E, E', T, T', F\}$
- cu productiile P
  - $Z \rightarrow E$
  - $E \rightarrow TE'$
  - $E' \rightarrow +TE'|\varepsilon$
  - $T \rightarrow FT'$
  - $T' \rightarrow *FT'|\varepsilon$
  - $F \rightarrow (E)|id$

simbol	$FIRST_1(X)$	$FOLLOW_1(X)$
E	$\{(, id\}$	$\{), \#\}$
E'	$\{+, \varepsilon\}$	$\{), \#\}$
T	$\{(, id\}$	$\{+, \#,)\}$
T'	$\{*, \varepsilon\}$	$\{+, \#,)\}$
F	$\{(, id\}$	$\{*, +, \#,)\}$

Exemplu

$$E \Rightarrow TE' \Rightarrow FT'E' \Rightarrow (E)T'E' \Rightarrow^{+} (id) * FT'E' \Rightarrow$$

$$(id) * F * T' + TE' \Rightarrow (id) * id * id + id$$

Gramatici $LL(k)$

O gramatica independenta de context $G = (T, N, P, Z)$ este $LL(k)$ pentru un $k \geq 0$ daca pentru derivari arbitrare

$$Z \Rightarrow^L \mu X \chi \Rightarrow \mu \nu \chi \Rightarrow^* \mu \gamma$$

$$Z \Rightarrow^L \mu X \chi \Rightarrow \mu \omega \chi \Rightarrow^* \mu \gamma'$$

$$\textit{unde } \mu, \gamma, \gamma' \in T^*, \nu, \chi, \omega \in V^*, X \in N$$

avem urmatoarea proprietate: $k : \gamma = k : \gamma'$ implica $\nu = \omega$

Fie gramatica G=$(\{i,(,+,)\},\{Z,E,F\},P,Z)$ cu productiile

- $Z \to E$
- $E \to F$
- $E \to (E+F)$
- $F \to i$

$Z \Rightarrow E \Rightarrow \quad (E{+}F) \quad \Rightarrow (F{+}F) \Rightarrow^* \quad (i+i)$

$Z \Rightarrow E \Rightarrow \quad (E{+}F) \quad \Rightarrow ((E+F){+}F) \Rightarrow^* \quad ((i+i)+i)$

$Z \Rightarrow^L \quad \mu X \chi \quad \Rightarrow \mu\nu\chi \Rightarrow^* \quad \mu\gamma$

$Z \Rightarrow^L \quad \mu X \chi \quad \Rightarrow \mu\omega\chi \Rightarrow^* \quad \mu\gamma'$

unde $\mu,\gamma,\gamma' \in T^*, \nu,\chi,\omega \in V^*, X \in N$

Exemplul 3 - reluat

Fie $G_1 = (T, N, E, P)$

- $T = \{a, b, c\}$, $N = \{Z, X, Y\}$
- cu productiile P
  - (1) $Z \to X$
  - (2,3) $X \to Y|bYa$
  - (4,5) $Y \to c|ca$

```
1)   Z=> X =>Y=> c
2)   Z=> X =>Y=> ca
3)   Z=> X =>bYa=> bca
4)   Z=> X =>bYa=> bcaa
```

Pentru 1 si 2, s-au aplicat $Y \to c$ si $Y \to ca$ dar 1:c=1:ca.
Pentru 3 si 4, s-au aplicat $Y \to c$ si $Y \to ca$ dar 1:ca=1:caa, la fel si 2:ca = 2:caa.

Situatie

$$[X_p \to \mu.\nu; \Omega]$$

$$\mu = x_{p,1}...x_{p,j}, \nu = x_{p,j+1}...x_{p,n_p},$$

$$|\mu| = j, |\nu| = n_p - j$$

Punctul nu face parte din vocabular. Marcheaza pozitia curenta a analizei in partea dreapta a productiei
ex: $q_7 = [X \to b.Ya; \#]$

Algoritmul LL(k)

Fie $G = (T, N, P, Z)$. Pt automatul stiva se determina Q si tranzitiile R:

1. $Q = \{q_0\}$ si $R = \emptyset$ cu $q_0 = [Z \to .S, \{\#\}]$
   Obs: $FOLLOW_k(Z) = \{\#\}$. q_0 starea initiala si a stivei. Automatul se opreste daca aceasta stare se intalneste din nou, stiva este vida, simbolul de intrare urmator este $\#$.

Algoritmul LL(k)

Fie $G = (T, N, P, Z)$. Pt automatul stiva se determina Q si tranzitiile R:

1. $Q = \{q_0\}$ si $R = \emptyset$ cu $q_0 = [Z \to .S, \{\#\}]$
   Obs: $FOLLOW_k(Z) = \{\#\}$. q_0 starea initiala si a stivei. Automatul se opreste daca aceasta stare se intalneste din nou, stiva este vida, simbolul de intrare urmator este $\#$.
2. fie $q = [X \to \mu.\nu; \Omega]$ un element al lui Q care inca nu a fost tratat

Algoritmul LL(k)

Fie $G = (T, N, P, Z)$. Pt automatul stiva se determina Q si tranzitiile R:

1. $Q = \{q_0\}$ si $R = \emptyset$ cu $q_0 = [Z \to .S, \{\#\}]$
   Obs: $FOLLOW_k(Z) = \{\#\}$. q_0 starea initiala si a stivei. Automatul se opreste daca aceasta stare se intalneste din nou, stiva este vida, simbolul de intrare urmator este $\#$.
2. fie $q = [X \to \mu.\nu; \Omega]$ un element al lui Q care inca nu a fost tratat
3. Daca $\nu = \varepsilon$ atunci se include $q\varepsilon \to \varepsilon$ in R.

Algoritmul LL(k)

Fie $G = (T, N, P, Z)$. Pt automatul stiva se determina Q si tranzitiile R:

1. $Q = \{q_0\}$ si $R = \emptyset$ cu $q_0 = [Z \to .S, \{\#\}]$
   Obs: $FOLLOW_k(Z) = \{\#\}$. q_0 starea initiala si a stivei. Automatul se opreste daca aceasta stare se intalneste din nou, stiva este vida, simbolul de intrare urmator este $\#$.
2. fie $q = [X \to \mu.\nu; \Omega]$ un element al lui Q care inca nu a fost tratat
3. Daca $\nu = \varepsilon$ atunci se include $q\varepsilon \to \varepsilon$ in R.
4. Daca $\nu = t\gamma$, $t \in T$ si $\gamma \in V^*$, fie $q' = [X \to \mu t.\gamma; \Omega]$. Adauga q' in Q si $qt \to q'$ in R.

Algoritmul LL(k)

Fie $G = (T, N, P, Z)$. Pt automatul stiva se determina Q si tranzitiile R:

1. $Q = \{q_0\}$ si $R = \emptyset$ cu $q_0 = [Z \to .S, \{\#\}]$
   Obs: $FOLLOW_k(Z) = \{\#\}$. q_0 starea initiala si a stivei. Automatul se opreste daca aceasta stare se intalneste din nou, stiva este vida, simbolul de intrare urmator este $\#$.
2. fie $q = [X \to \mu.\nu; \Omega]$ un element al lui Q care inca nu a fost tratat
3. Daca $\nu = \varepsilon$ atunci se include $q\varepsilon \to \varepsilon$ in R.
4. Daca $\nu = t\gamma$, $t \in T$ si $\gamma \in V^*$, fie $q' = [X \to \mu t.\gamma; \Omega]$. Adauga q' in Q si $qt \to q'$ in R.
5. Daca $\nu = Y\gamma$, $Y \in N$ si $\gamma \in V^*$,
   - fie $q' = [X \to \mu Y.\gamma; \Omega]$
   - si $H = \{[Y \to .\beta_i; FIRST_k(\gamma\Omega)] | Y \to \beta_i \in P\}$.
   - actualizeaza $Q = Q \cup \{q'\} \cup H$
   - si $R = R \cup \{q\tau_i \to q' h_i \tau_i | h_i \in H, \tau_i \in FIRST_k(\beta_i\gamma\Omega)\}$
6. daca toate starile din q au fost analizate, stop. Altfel continua cu 2.

Construirea automatului se termina datorita numarului finit de situatii.

Automatul rezultat este determinist daca si numai daca G este o gramatica $LL(k)$.

Exemplu de construire (gramatica e CFG dar si regulata)

- $Z \to S$
- $S \to 0S$
- $S \to 1$

k=1

		stari noi	tranzitii noi
			$q_0 = [Z \to .S; \{\#\}]$
q_0	5	$q' = [Z \to S.; \#] = q_1$ $H = \{[S \to .0S; \#] = q_2,$ $[S \to .1; \#] = q_3\}$	$q_0\tau? \to q_1 h?\tau?$ $q_0 0 \to q_1 q_2 0$ $q_0 1 \to q_1 q_3 1$
q_1	3	-	$q_1\varepsilon \to \varepsilon$
q_2	4	$q' = [S \to 0.S; \#] = q_4$	$q_2 0 \to q_4$
q_3	4	$q' = [S \to 1.; \#] = q_5$	$q_3 1 \to q_5$
q_4	5	$q' = [S \to 0S.; \#] = q_6$ H = la fel cu analiza pt q_0	? $q_4 0 \to q_6 q_2 0$ $q_4 1 \to q_6 q_3 1$
q_5	3	-	$q_5\varepsilon \to \varepsilon$
q_6	3	-	$q_6\varepsilon \to \varepsilon$

derivare

Care e derivarea pt 001#?

$$q_0 q_0 001\# \Rightarrow ?$$

Incercare cu $k = 1$; $Z \to S$, $S \to 0S1$, $S \to 01$

		stari noi	No	tranzitii noi
		$q_0 = [Z \to .S; \{\#\}]$		
q_0	5	$q' = [Z \to S.; \#] = q_1$ $H = \{[S \to .0S1; \#] = q_2,$ $[S \to .01; \#] = q_3\}$	 1 2	$q_0\tau? \to q_1 h?\tau?$ $\tau \in FIRST_1(0S1\#);$ $q_0 0 \to q_1 q_2 0$ $\tau \in FIRST_1(01\#);$ $q_0 0 \to q_1 q_3 0$
q_1	3	-	3	$q_1\varepsilon \to \varepsilon$
q_2	4	$q' = [S \to 0.S1; \#] = q_4$	4	$q_2 0 \to q_4$
q_3	4	$q' = [S \to 0.1; \#] = q_5$	5	$q_3 0 \to q_5$
q_4	5	$q' = [S \to 0S.1; \#] = q_6$ $H = \{[S \to .0S1; FIRST_1(1\#)]$ $= q_7,$ $[S \to .01; FIRST_1(1\#)] = q_8\}$	 6 7	 $\tau \in FIRST_1(0S11\#);$ $q_4 0 \to q_6 q_7 0$ $\tau \in FIRST_1(011\#);$ $q_4 0 \to q_6 q_8 0$
q_5	4	$[S \to 01.; \#] = q_9$	8	$q_5 1 \to q_9$
q_6	4	$[S \to 0S1.; \#] = q_{10}$	9	$q_6 1 \to q_{10}$
	...			
q_9	3		10	$q_9\varepsilon \to \varepsilon$
	....			

k=1 : automat nedeterminist: $q_0 0 \to q_1 q_2 0$ si $q_0 0 \to q_1 q_3 0$
Derivare 01: cu un lookahead de 1 nu stim pe care productie sa o aplicam

$$q_0 q_0 01\# \overset{1}{\Rightarrow} q_0 q_1 q_2 01\# \overset{4}{\Rightarrow} q_0 q_1 q_4 1\# \quad \textit{deadend}$$

$$q_0 q_0 01\# \overset{2}{\Rightarrow} q_0 q_1 q_3 01\# \overset{5}{\Rightarrow} q_0 q_1 q_5 1\# \overset{8}{\Rightarrow} q_0 q_1 q_9\# \overset{10}{\Rightarrow} q_0 q_1 \varepsilon\# \Rightarrow q_0\#$$

Incercare cu $k = 2$; $Z \to S$, $S \to 0S1$, $S \to 01$

		stari noi	No	tranzitii noi
		$q_0 = [Z \to .S; \{\#\}]$		
q_0	5	$q' = [Z \to S.; \#] = q_1$ $H = \{[S \to .0S1; \#] = q_2,$ $[S \to .01; \#] = q_3\}$	 1 2	$q_0\tau? \to q_1 h?\tau?$ $\tau \in FIRST_2(0S1\#);$ $q_0 00 \to q_1 q_2 00$ $\tau \in FIRST_2(01\#);$ $q_0 01 \to q_1 q_3 01$
q_1	3	-	3	$q_1\varepsilon \to \varepsilon$
q_2	4	$q' = [S \to 0.S1; \#] = q_4$	4	$q_2 0 \to q_4$
q_3	4	$q' = [S \to 0.1; \#] = q_5$	5	$q_3 0 \to q_5$
q_4	5	$q' = [S \to 0S.1; \#] = q_6$ $H = \{[S \to .0S1; FIRST_2(1\#)]$ $= q_7,$ $[S \to .01; FIRST_2(1\#)] = q_8\}$	 6 7	 $\tau \in FIRST_2(0S11\#);$ $q_4 00 \to q_6 q_7 00$ $\tau \in FIRST_2(011\#);$ $q_4 01 \to q_6 q_8 01$
q_5	4	$[S \to 01.; \#] = q_9$	8	$q_5 1 \to q_9$
q_6	4	$[S \to 0S1.; \#] = q_{10}$	9	$q_6 1 \to q_{10}$
q_7	4	$[S \to 0.S1; 1\#] = q_{11}$	10	$q_7 0 \to q_{11}$
q_8	4	$[S \to 0.1; 1\#] = q_{12}$	11	$q_8 0 \to q_{12}$
q_9	3		10	$q_9\varepsilon \to \varepsilon$
q_{10}	3		10	$q_{10}\varepsilon \to \varepsilon$
q_{11}	5	$q' = [S \to 0S.1; \{1\#\}] = q_{13}$ $H = \{[S \to .0S1; FIRST_2(11\#)]$ $= q_{14},$ $[S \to .01; FIRST_2(11\#)] = q_{15}\}$	 6 7	 $\tau \in FIRST_2(0S111\#);$ $q_{11} 00 \to q_{13} q_{14} 00$ $\tau \in FIRST_2(0111\#);$ $q_{11} 01 \to q_{13} q_{15} 01$

Derivare $Z \Rightarrow 0011$

Stiva	Stare	Intrare	Derivarea cea mai din stanga
q_0	q_0	$0011\#$	Z
$q_0 q_1$	q_2	$0011\#$	S
$q_0 q_1$	q_4	$011\#$	0S1
$q_0 q_1 q_6$	q_8	$011\#$	0011
$q_0 q_1 q_6$	q_{12}	$11\#$	
....			

Gramatica $LL(3)$

- $Z \to X$
- $X \to Y | bYa$
- $Y \to c | ca$

		stari noi		tranzitii noi
		$q_0 = [Z \to .X; \#]$		
q_0	5	$q' = [Z \to X.; \#] = q_1$		$\tau \in FIRST_3(Y\#) = \{c\#, ca\#$
		$H = \{[X \to .Y; \#] = q_2,$	1	$q_0c\# \to q_1q_2c\#$
			2	$q_0ca\# \to q_1q_2ca\#$
		$[X \to .bYa; \#] = q_3\}$		$\tau \in FIRST_3(bYa) = \{bca\}$
			3	$q_0bca \to q_1q_3bca$
q_2	5	$q' = [X \to Y.; \#] = q_4$		$\tau \in FIRST_3(c\#) = \{c\#\}$
		$H = \{[Y \to .c; \#] = q_5,$	4	$q_2c\# \to q_4q_5c\#$
		$Y \to .ca; \#] = q_6\}$	5	$q_2ca\# \to q_4q_6ca\#$
		.....		

Gramatica $LL(3)$

- $Z \to X$
- $X \to Y | bYa$
- $Y \to c | ca$

$$
\begin{aligned}
q_0 &= [Z \to \bullet X; \#] & q_9 &= [Y \to c \bullet a; \#] \\
q_1 &= [Z \to X \bullet; \#] & q_{10} &= [X \to bY \bullet a; \#] \\
q_2 &= [X \to \bullet Y; \#] & q_{11} &= [Y \to \bullet c; a\#] \\
q_3 &= [X \to \bullet bY a; \#] & q_{12} &= [Y \to \bullet ca; a\#] \\
q_4 &= [X \to Y \bullet; \#] & q_{13} &= [Y \to ca \bullet; \#] \\
q_5 &= [Y \to \bullet c; \#] & q_{14} &= [X \to bY a \bullet; \#] \\
q_6 &= [Y \to \bullet ca; \#] & q_{15} &= [Y \to c \bullet; a\#] \\
q_7 &= [X \to b \bullet Y a; \#] & q_{16} &= [Y \to c \bullet a; a\#] \\
q_8 &= [Y \to c \bullet; \#] & q_{17} &= [Y \to ca \bullet; a\#]
\end{aligned}
$$

$$
\begin{array}{ll}
q_0 = [Z \to \bullet X; \#] & q_9 = [Y \to c \bullet a; \#] \\
q_1 = [Z \to X \bullet; \#] & q_{10} = [X \to bY \bullet a; \#] \\
q_2 = [X \to \bullet Y; \#] & q_{11} = [Y \to \bullet c; a\#] \\
q_3 = [X \to \bullet bYa; \#] & q_{12} = [Y \to \bullet ca; a\#] \\
q_4 = [X \to Y \bullet; \#] & q_{13} = [Y \to ca \bullet; \#] \\
q_5 = [Y \to \bullet c; \#] & q_{14} = [X \to bYa \bullet; \#] \\
q_6 = [Y \to \bullet ca; \#] & q_{15} = [Y \to c \bullet; a\#] \\
q_7 = [X \to b \bullet Ya; \#] & q_{16} = [Y \to c \bullet a; a\#] \\
q_8 = [Y \to c \bullet; \#] & q_{17} = [Y \to ca \bullet; a\#]
\end{array}
$$

$$
\begin{aligned}
R = \{ & q_0c\# \to q_1q_2c\#, & & q_7ca\# \to q_{10}q_{11}ca\# \\
& q_0ca\# \to q_1q_2ca\#, & & q_7caa \to q_{10}q_{12}caa, \\
& q_0bca \to q_1q_3bca, & & q_8 \to \epsilon, \\
& q_1 \to \epsilon, & & q_9a \to q_{13}, \\
& q_2c\# \to q_4q_5c\#, & & q_{10}a \to q_{14}, \\
& q_2ca\# \to q_4q_6ca\#, & & q_{11}c \to q_{15}, \\
& q_3b \to q_7, & & q_{12}c \to q_{16},\ q_{13} \to \epsilon, \\
& q_4 \to \epsilon, & & q_{14} \to \epsilon, \\
& q_5c \to q_8, & & q_{15} \to \epsilon, \\
& q_6c \to q_9, & & q_{16}a \to q_{17},\ q_{17} \to \epsilon \ \}
\end{aligned}
$$

aceeasi gramatica dar cu $k = 2$

$$q_7 ca \rightarrow q_{10} q_{11} ca$$

$$q_7 ca \rightarrow q_{10} q_{12} ca$$

Cu $k = 3$

$$q_7 ca\# \rightarrow q_{10} q_{11} ca\#$$

$$q_7 caa \rightarrow q_{10} q_{12} caa$$

unde pt $k = 3$

- $q_7 = [X \rightarrow b.Ya; \#]$
- $q_{10} = [X \rightarrow bY.a; \#]$
- $q_{11} = [Y \rightarrow .c; a\#]$
- $q_{12} = [Y \rightarrow .ca; a\#]$

Derivare $Z \Rightarrow X \Rightarrow bYa \Rightarrow bcaa$

Stiva	Stare	Intrare	Derivarea cea mai din stanga
q_0	q_0	$bcaa\#$	Z
$q_0 q_1$	q_3	$bcaa\#$	X
$q_0 q_1$	q_7	$caa\#$	bYa
$q_0 q_1 q_{10}$	q_{12}	$caa\#$	bcaa
$q_0 q_1 q_{10}$	q_{16}	$aa\#$	
$q_0 q_1 q_{10}$	q_{17}	$a\#$	
$q_0 q_1$	q_{10}	$a\#$	
$q_0 q_1$	q_{14}	$\#$	
q_0	q_1	$\#$	
	q_0	$\#$	

- La tranzitiile de stivuire sunt examinate simbolurile dinainte (lookaheads symbols).
- Aceste tranzitii corespund intrarii intr-o productie noua

- Citirea simbolurilor terminale si decizia de terminare a productiei printr-o tranzitie de destivuire se realizaeaza fara inspectarea simbolurilor dinainte

Rezumat

Recursive descent parsing

Predictive parsing

Structuri ajutatoare: FIRST, FOLLOW

Definire Gramatici LL(k)

Algoritmul LL(K)

- Exemplu aplicare LL(k)
- Exemplu aplicare LL(3)

$$Z => X => ca$$

$$Z => X => bca$$

Analiza descendenta. Gramatici LL(k) - eliminare recursivitate stanga. factorizare.

Table of Contents

Teorema

4.2.2, 4.2.3 Teorema. O gramatica LL(k) nu poate avea simbol nonterminal recursiv stanga.

Daca $X \Rightarrow X\omega, \omega \neq \varepsilon$ - X nonterminal recursiv stanga

- $E \rightarrow E + T|T$
- $T \rightarrow T * F|F$
- $F \rightarrow (E)|id$

are doua productii cu recursivitate stanga

Teorema

Teorema. Pentru orice gramatica CFG $G = (T, N, P, Z)$ cu simboluri nonterminale recursive stanga exista o gramatica echivalenta $G' = (T, N', P', Z)$ fara nonterminale recursive stanga.

Idee

$$X \to X\alpha|\beta \text{ devine } \begin{cases} X \to \beta X' \\ X' \to \alpha X'|\varepsilon \end{cases}$$

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

- $E \to TE'$
- $E' \to +TE'|\varepsilon$
- $T \to FT'$
- $T' \to *FT'|\varepsilon$
- $F \to (E)|id$

Dar...

NU intra la examen
vezi Testare online

- $S \to Aa|b$
- $A \to Ac|Sd|\varepsilon$

$$S \Rightarrow Aa \Rightarrow Sda$$

ne trebuie un algoritm care sa elimine toate nonterminalele cu recursivitate stanga

- Consideram ca $N = \{X_1, X_2, ...X_n\}$ - simbolurile nonterminale sunt numerotate consecutiv.
- Daca putem alege indicii a.i. indicii sa respecte $i < j$ pentru toate productiile $X_i \to X_j\omega$ atunci G nu are recursivitate stanga.
- Daca o astfel de numerotare nu este posibila pentru G, atunci se genereaza G'.

Exemple:

- $S \to Aa|b$
- $A \to Ac|Sd|\varepsilon$
- Daca S e 1, A e 2, prima productie respecta $i < j$ dar nu si a doua
- $E \to E + T$ nu respecta $i < j$

Algoritm de eliminare recursivitate stanga

NU intra la examen Testare online

1. Fie $N' = N, P' = P$. Se executa pasii 2,3 pentru $i = 1, ...n$
2. Pentru $j = 1, ...i-1$
   $X_i \to X_j\omega \in P'$ se inlocuiesc cu $\{X_i \to \chi_j\omega | X_j \to \chi_j \in P'\}$.
   In consecinta, $X_i \Rightarrow^+ X_j\gamma$ implica $i \leq j$.
3. Se inlocuiesc $X_i \to X_i\omega \in P'$ cu $\{Y_i \to \omega Y_i\} \cup \{Y_i \to \varepsilon\}$
   adaugand un nou simbol Y_i la N'.
   $+$ se inlocuiesc $X_i \to \chi, \chi \neq X_i\gamma$ cu $X_i \to \chi Y_i$.
   Simbolurile noi se numeroteaza cu n+1, n+2,..

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

presupunem ordinea $E(1) < T(2) < F(3)$

i	pasul2	pasul 3	variabila noua
1	nu se executa	$E \to E + T \mid T$ devin $E' \to +TE' \mid \varepsilon$ si $E \to TE'$;	$E'(4)$
2	$j = 1$ $T \to E\omega$ nu exista	$T \to T * F \mid F$ devin $T' \to *FT' \mid \varepsilon$ $T \to FT'$	$T'(5)$
3	$j = 1, 2$ $F \to E\omega$ sau $F \to T\omega$ nu exista	$F \to F\omega$ nu exista	
4,5	nu se modifica nimic		

Rezultat:

- $E \rightarrow E + T | T$
- $T \rightarrow T * F | F$
- $F \rightarrow (E) | id$

- $E \rightarrow TE'$
- $E' \rightarrow +TE' | \varepsilon$
- $T \rightarrow FT'$
- $T' \rightarrow *FT' | \varepsilon$
- $F \rightarrow (E) | id$

- $S \to Aa|b$
- $A \to Ac|Sd|\varepsilon$

...pasul 2 al algoritmului: Pentru $j = 1, ...i - 1$ $X_i \to X_j\omega \in P'$ se inlocuiesc cu $\{X_i \to \chi_j\omega | X_j \to \chi_j \in P'\}$.

Daca $S(1) < A(2)$

- $i = 1$ nimic
- $i = 2$ la pasul 2 $A \to Sd$ se inlocuieste cu $\{A \to Aad|bd\}$

- $S \to Aa|b$
- $A \to Ac|Sd|\varepsilon$

...pasul 3 al algoritmului: Se inlocuiesc $X_i \to X_i\omega \in P'$ cu $\{Y_i \to \omega Y_i\} \cup \{Y_i \to \varepsilon\}$ adaugand un nou simbol Y_i la N'.
$+$ se inlocuiesc $X_i \to \chi, \chi \neq X_i\gamma$ cu $X_i \to \chi Y_i$.

Daca $S(1) < A(2)$

- $i = 1$ nimic
- $i = 2$ la pasul 2 $A \to Sd$ se inlocuieste cu $\{A \to Aad|bd\}$
- $i = 2$ la pasul 3 $A \to Ac|Aad|bd|\varepsilon$ se inlocuieste cu $A' \to cA', A' \to adA', A' \to \varepsilon$ si $A \to bdA', A \to A'$

Teorema. Daca sirul ω din $X_i \to X_i\omega$ nu incepe cu $X_j, j \leq i$ atunci $X_i \to X_i\omega$ se poate inlocui cu $\{Y_i \to \omega, Y_i \to \omega Y_i\}$ si $X_i \to \chi$ cu $\{X_i \to \chi, X_i \to \chi Y_i\}$ la pasul 3.

pasul 3 anterior ...se inlocuiesc $X_i \to X_i\omega \in P'$ cu $\{Y_i \to \omega Y_i\} \cup \{Y_i \to \varepsilon\}$ adaugand un nou simbol Y_i la N'.
$+$ se inlocuiesc $X_i \to \chi, \chi \neq X_i\gamma$ cu $X_i \to \chi Y_i$.

Se evita introducerea productiilor ε.

Table of Contents

Intra la examen

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

Cu productii ε

- $E \to TE'$
- $E' \to +TE' | \varepsilon$
- $T \to FT'$
- $T' \to *FT' | \varepsilon$
- $F \to (E) | id$

Fara productii ε

- $E \to TE' | T$
- $E' \to +T | +TE'$
- $T \to FT' | F$
- $T' \to *F | *FT'$
- $F \to (E) | id$

Observatii

- Recursivitatea stanga precum $E \to T|E + T$ - utilizata pentru a reflecta asociativitatea stanga a operatorilor.
- Aceeasi proprietate avem si in $E \to TE', E' \to +TE', E' \to \varepsilon$
- Insa asociativitate dreapta $E \to T, E \to T + E$.

Productii ε

Productiile ε se pot elimina intotdeauna dintr-o gramatica LL(k), dar aceasta poate mari valoare lui k.
4.2.3

Teorema

TEOREMA. Pentru orice gramatica LL(k) cu productii ε exista o gramatica LL(k+1) fara productii ε care genereaza limbajul $L(G) - \varepsilon$.

Prin introducerea productiilor ε se poate reduce k.

Teorema

TEOREMA. Pentru orice gramatica LL(k+1), $k > 0$ fara productii ε exista o gramatica LL(k) echivalenta cu productii ε.

Table of Contents

Factorizare stanga

$$\text{Fie } P = \{ \quad Z \to X$$
$$X \to Yc|Yd$$
$$Y \to a|bY\}$$

Productiile $X \to Yc$ si $X \to Yd$ nu pot fi distinse chiar prin examinarea oricarui numar fix de simboluri din sirul de intrare deoarece din Y se poate deriva un sir de lungime si mai mare.

Solutie: evitarea problemei prin amanarea deciziei. Ambele incep cu Y, nu trebuie facauta distinctie intre ele decat dupa ce Y a fost recunoscut.

Factorizare stanga

$$\begin{aligned} \textit{Fie } P = \{ \quad & Z \to X \\ & X \to Yc|Yd \\ & Y \to a|bY \} \end{aligned}$$

devine

$$\begin{aligned} \textit{Fie } P = \{ \quad & Z \to X \\ & X \to YX' \\ & X' \to c|d \\ & Y \to a|bY \} \end{aligned}$$

Se poate examina un singur caracter inainte pt a face diferente intre cele doua variante c sau d .

Factorizare stanga

$$
\begin{aligned}
Fie\ P = \{ \quad & Z \to X \\
& X \to if\ E\ then\ S | if\ E\ then\ S\ else\ S | a \\
& E \to b \}
\end{aligned}
$$

devine

$$
\begin{aligned}
Fie\ P = \{ \quad & Z \to X \\
& X \to if\ E\ then\ S\ S' | a \\
& S' \to else\ S | \varepsilon \\
& E \to b \}
\end{aligned}
$$

Gramatici LL(k) tari. Derivare descendent recursiva

Ce e gramatica LL(k)? - reaminitire

O gramatica independenta de context $G = (T, N, P, Z)$ este $LL(k)$ pentru un $k \geq 0$ daca pentru derivari arbitrare

$$Z \Rightarrow^L \mu X \chi \Rightarrow \mu \nu \chi \Rightarrow^* \mu \gamma$$

$$Z \Rightarrow^L \mu X \chi \Rightarrow \mu \omega \chi \Rightarrow^* \mu \gamma'$$

$$\textit{unde } \mu, \gamma, \gamma' \in T^*, \nu, \chi, \omega \in V^*, X \in N$$

avem urmatoarea proprietate: $k : \gamma = k : \gamma'$ implica $\nu = \omega$
Observatie: Dependenta de μ obliga pastrarea in situatiile $[X \to \alpha.\beta; \omega]$ a contextului dreapta. Daca se elimina aceasta dependenta: gramatici **LL(k) tari**

Gramatici LL(k) tari

O gramatica independenta de context $G = (T, N, P, Z)$ este o gramatica $LL(k)$ tare pentru un $k > 0$ daca pentru derivari arbitrare

$$Z \Rightarrow^L \mu X \chi \Rightarrow \mu \nu \chi \Rightarrow^* \mu \gamma$$

$$Z \Rightarrow^L \mu' X \chi' \Rightarrow \mu' \omega \chi' \Rightarrow^* \mu' \gamma'$$

$$\textit{unde } \mu, \mu', \gamma, \gamma' \in T^*, \nu, \chi, \omega \in V^*, X \in N$$

avem urmatoarea proprietate: $k : \gamma = k : \gamma'$ implica $\nu = \omega$

$$\text{Fie } G \text{ cu } P = \{ \quad Z \to X$$
$$X \to aAab|bAbb$$
$$A \to a|\varepsilon\}$$

$Z \Rightarrow X \Rightarrow aAab \overset{A\to\varepsilon}{\Rightarrow} aab$

$Z \Rightarrow X \Rightarrow aAab \Rightarrow aaab$

$Z \Rightarrow X \Rightarrow bAbb \Rightarrow bbb$

$Z \Rightarrow X \Rightarrow bAbb \overset{A\to a}{\Rightarrow} babb$

Este LL(1)? Este LL(2)? Este strong LL(2)?

$$\text{Fie } G \text{ cu } P = \{ \quad Z \to X \quad X \to aAab|bAbb \quad A \to a|\varepsilon \}$$

$Z \Rightarrow X \Rightarrow aAab \overset{A \to \varepsilon}{\Rightarrow} aab$

$Z \Rightarrow X \Rightarrow aAab \Rightarrow aaab$

$Z \Rightarrow X \Rightarrow bAbb \Rightarrow bbb$

$Z \Rightarrow X \Rightarrow bAbb \overset{A \to a}{\Rightarrow} babb$

Este LL(1)? Este LL(2)? Este strong LL(2)?

$Z \Rightarrow X \Rightarrow aAab \Rightarrow aab$

$Z \Rightarrow X \Rightarrow bAbb \Rightarrow babb$

pt LL(k) tare: $k : \gamma = k : \gamma \Rightarrow$ aceeasi productie pt A; dar aici contextul stanga conteaza

Conditia strong LL(k)

O gramatica independenta de context G este strong LL(k) daca pentru orice pereche de productii $X \to \chi$, $X \to \chi', \chi \neq \chi'$ urmatoarea conditie este adevarata:

$$FIRST_k(\chi FOLLOW_k(X)) \cap FIRST_k(\chi' FOLLOW_k(X)) = \emptyset$$

exemplu

$$Fie\ G\ cu\ P = \{ \quad Z \to X \quad X \to aAab|bAbb \quad A \to a|\varepsilon \}$$

pt A : $FIRST_2(a\{ab, bb\}) \cap FIRST_2(\varepsilon\{ab, bb\}) = \{ab\}$

Strong LL(k)

NU e necesar niciun context pt a decide productia pentru nonterminalul X. Nu trebuie tinuti minte pasii anteriori din derivarea stanga, cei care au condus la nonterminalul X.

Algoritmul LL(k) - reamintire

Fie $G = (T, N, P, Z)$. Pt automatul stiva se determina Q si tranzitiile R:

1. $Q = \{q_0\}$ si $R = \emptyset$ cu $q_0 = [Z \to .S, \{\#\}]$
   Obs: $FOLLOW_k(Z) = \{\#\}$. q_0 starea initiala si a stivei. Automatul se opreste daca aceasta stare se intalneste din nou, stiva este vida, simbolul de intrare urmator este $\#$.
2. fie $q = [X \to \mu.\nu; \Omega]$ un element al lui Q care inca nu a fost tratat
3. Daca $\nu = \varepsilon$ atunci se include $q\varepsilon \to \varepsilon$ in R.
4. Daca $\nu = t\gamma$, $t \in T$ si $\gamma \in V^*$, fie $q' = [X \to \mu t.\gamma; \Omega]$. Adauga q' in Q si $qt \to q'$ in R.
5. Daca $\nu = Y\gamma$, $Y \in N$ si $\gamma \in V^*$,
   - fie $q' = [X \to \mu Y.\gamma; \Omega]$
   - si $H = \{[Y \to .\beta_i; FIRST_k(\gamma\Omega)] | Y \to \beta_i \in P\}$.
   - actualizeaza $Q = Q \cup \{q'\} \cup H$
   - si $R = R \cup \{q\tau_i \to q' h_i \tau_i | h_i \in H, \tau_i \in FIRST_k(\beta_i\gamma\Omega)\}$
6. daca toate starile din q au fost analizate, stop. Altfel continua cu 2.

Algoritm LL(k) tare

Daca $\nu = Y\gamma,\ Y \in N$ si $\gamma \in V^*$ in loc de pasul 5 din LL(k)

- fie $q' = [X \to \mu Y.\gamma; \Omega]$
- si $H = \{[Y \to .\beta_i; FIRST_k(\gamma\Omega)] | Y \to \beta_i \in P\}$.
- actualizeaza $Q = Q \cup \{q'\} \cup H$ si
- $R = R \cup \{q\tau_i \to q' h_i \tau_i | h_i \in H, \tau_i \in FIRST_k(\beta_i\gamma\Omega)\}$

se poate folosi pentru strong LL(k)

- fie $q' = [X \to \mu Y.\gamma; \Omega]$
- si $H = \{[Y \to .\beta_i; FOLLOW_k(Y)] | Y \to \beta_i \in P\}$.
- actualizeaza $Q = Q \cup \{q'\} \cup H$ si
- $R = R \cup \{q\tau_i \to q' h_i \tau_i | h_i \in H, \tau_i \in FIRST_k(\beta_i FOLLOW_k(Y))\}$

Toate situatiile distincte anterior doar prin context dreapta apartin intotdeauna aceleiasi stari.

LL(1) tare

Fie $Z \to E,\ E \to E + F | F,\ F \to i | (E)$

Prin eliminarea recursivitatii stanga:

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | + FE_1,\ F \to i | (E)$

simbol	$FIRST_1(X)$	$FOLLOW_1(X)$
E	$\{(, i\}$	$\{), \#\}$
E_1	$\{+, \varepsilon\}$	$\{), \#\}$
F	$\{(, i\}$	$\{+, \#,)\}$

Conditie LL(1) tare:

pt E_1:

$$FIRST_1(\varepsilon FOLLOW(E_1)) \cap FIRST_1(+FE_1 FOLLOW(E_1)) = \emptyset$$

pt F:

$$FIRST_1(iFOLLOW(F)) \cap FIRST_1((E)FOLLOW(F)) = \emptyset$$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | + FE_1,\ F \to i | (E)$

stari noi	tranzitii noi
$q_0 = [Z \to .E; \#]$	

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($

$$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | + FE_1,\ F \to i | (E)$$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0(\to q_1 q_2($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2(\to q_3 q_5($

fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0(\to q_1 q_2($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2(\to q_3 q_5($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$
q_5	$[F \to (.E)] = q_{10}$	$q_5 (\to q_{10}$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$
q_5	$[F \to (.E)] = q_{10}$	$q_5 (\to q_{10}$
q_6		$q_6 \varepsilon \to \varepsilon$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$
q_5	$[F \to (.E)] = q_{10}$	$q_5 (\to q_{10}$
q_6		$q_6 \varepsilon \to \varepsilon$
q_7		$q_7 \varepsilon \to \varepsilon$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | + FE_1,\ F \to i | (E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$
q_5	$[F \to (.E)] = q_{10}$	$q_5 (\to q_{10}$
q_6		$q_6 \varepsilon \to \varepsilon$
q_7		$q_7 \varepsilon \to \varepsilon$
q_8	$[E_1 \to +.FE_1] = q_{11}$	$q_8 + \to q_{11}$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$
q_5	$[F \to (.E)] = q_{10}$	$q_5 (\to q_{10}$
q_6		$q_6 \varepsilon \to \varepsilon$
q_7		$q_7 \varepsilon \to \varepsilon$
q_8	$[E_1 \to +.FE_1] = q_{11}$	$q_8 + \to q_{11}$
q_9		$q_9 \varepsilon \to \varepsilon$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$
q_5	$[F \to (.E)] = q_{10}$	$q_5 (\to q_{10}$
q_6		$q_6 \varepsilon \to \varepsilon$
q_7		$q_7 \varepsilon \to \varepsilon$
q_8	$[E_1 \to +.FE_1] = q_{11}$	$q_8 + \to q_{11}$
q_9		$q_9 \varepsilon \to \varepsilon$
q_{10}	$[F \to (E.)] = q_{12}$ $H = \{[E \to .FE_1] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW(E))$ $q_{10} (\to q_{12} q_2 ($ $q_{10} i \to q_{12} q_2 i$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$
q_5	$[F \to (.E)] = q_{10}$	$q_5 (\to q_{10}$
q_6		$q_6 \varepsilon \to \varepsilon$
q_7		$q_7 \varepsilon \to \varepsilon$
q_8	$[E_1 \to +.FE_1] = q_{11}$	$q_8 + \to q_{11}$
q_9		$q_9 \varepsilon \to \varepsilon$
q_{10}	$[F \to (E.)] = q_{12}$ $H = \{[E \to .FE_1] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW(E))$ $q_{10} (\to q_{12} q_2 ($ $q_{10} i \to q_{12} q_2 i$
q_{11}	$[E_1 \to +F.E_1] = q_{13}$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_{11} i \to q_{13} q_4 i$ $q_{11} (\to q_{13} q_5 ($

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$
q_5	$[F \to (.E)] = q_{10}$	$q_5 (\to q_{10}$
q_6		$q_6 \varepsilon \to \varepsilon$
q_7		$q_7 \varepsilon \to \varepsilon$
q_8	$[E_1 \to +.FE_1] = q_{11}$	$q_8 + \to q_{11}$
q_9		$q_9 \varepsilon \to \varepsilon$
q_{10}	$[F \to (E.)] = q_{12}$ $H = \{[E \to .FE_1] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW(E))$ $q_{10} (\to q_{12} q_2 ($ $q_{10} i \to q_{12} q_2 i$
q_{11}	$[E_1 \to +F.E_1] = q_{13}$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_{11} i \to q_{13} q_4 i$ $q_{11} (\to q_{13} q_5 ($
q_{12}	$[F \to (E).] = q_{14}$	$q_{12}) \to q_{14}$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$
q_5	$[F \to (.E)] = q_{10}$	$q_5 (\to q_{10}$
q_6		$q_6 \varepsilon \to \varepsilon$
q_7		$q_7 \varepsilon \to \varepsilon$
q_8	$[E_1 \to +.FE_1] = q_{11}$	$q_8 + \to q_{11}$
q_9		$q_9 \varepsilon \to \varepsilon$
q_{10}	$[F \to (E.)] = q_{12}$ $H = \{[E \to .FE_1] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW(E))$ $q_{10} (\to q_{12} q_2 ($ $q_{10} i \to q_{12} q_2 i$
q_{11}	$[E_1 \to +F.E_1] = q_{13}$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_{11} i \to q_{13} q_4 i$ $q_{11} (\to q_{13} q_5 ($
q_{12}	$[F \to (E).] = q_{14}$	$q_{12}) \to q_{14}$
q_{13}	$[E_1 \to +FE_1.] = q_{15}$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_{13}) \to q_{15} q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_{13} + \to q_{15} q_8 +$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | +FE_1,\ F \to i|(E)$

	stari noi	tranzitii noi
	$q_0 = [Z \to .E; \#]$	
q_0	$q' = [Z \to E.; \#] = q_1$ $H = \{[E \to .FE_1; \#] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW_1(E)) = \{i, (\}$ $q_0 i \to q_1 q_2 i$ $q_0 (\to q_1 q_2 ($
q_1		$q_1 \varepsilon \to \varepsilon$
q_2	$[E \to F.E_1] = q_3$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_2 i \to q_3 q_4 i$ $q_2 (\to q_3 q_5 ($
fiind LL(1) strong, capetele din situatii nu le mai pastram (se pot deduce din situatie)		
q_3	$[E \to FE_1.] = q_6$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_3) \to q_6 q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_3 + \to q_6 q_8 +$
q_4	$[F \to i.] = q_9$	$q_4 i \to q_9$
q_5	$[F \to (.E)] = q_{10}$	$q_5 (\to q_{10}$
q_6		$q_6 \varepsilon \to \varepsilon$
q_7		$q_7 \varepsilon \to \varepsilon$
q_8	$[E_1 \to +.FE_1] = q_{11}$	$q_8 + \to q_{11}$
q_9		$q_9 \varepsilon \to \varepsilon$
q_{10}	$[F \to (E.)] = q_{12}$ $H = \{[E \to .FE_1] = q_2\}$	$\tau \in FIRST_1(FE_1 FOLLOW(E))$ $q_{10} (\to q_{12} q_2 ($ $q_{10} i \to q_{12} q_2 i$
q_{11}	$[E_1 \to +F.E_1] = q_{13}$ $H = \{[F \to .i, FOLLOW_1(F)] = q_4$ $[F \to .(E); FOLLOW_1(F)] = q_5\}$	$\tau \in FIRST_1(iFOLLOW_1(F))$ $q_{11} i \to q_{13} q_4 i$ $q_{11} (\to q_{13} q_5 ($
q_{12}	$[F \to (E).] = q_{14}$	$q_{12}) \to q_{14}$
q_{13}	$[E_1 \to +FE_1.] = q_{15}$ $H = \{[E_1 \to .\varepsilon] = q_7$ $[E_1 \to . + FE_1] = q_8\}$	$\tau \in FIRST_1(\varepsilon FOLLOW(E_1))$ $q_{13}) \to q_{15} q_7)$ $q_3 \# \to q_6 q_7 \#$ $q_{13} + \to q_{15} q_8 +$
q_{14}		$q_{14} \varepsilon \to \varepsilon$
q_{15}		$q_{15} \varepsilon \to \varepsilon$

q_0 : $[Z \to \bullet E]$
q_1 : $[Z \to E\bullet]$
q_2 : $[E \to \bullet F E_1]$
q_3 : $[E \to F \bullet E_1]$
q_4 : $[F \to \bullet i]$
q_5 : $[F \to \bullet(E)]$
q_6 : $[E \to F E_1 \bullet]$
q_7 : $[E_1 \to \bullet\epsilon]$
q_8 : $[E_1 \to \bullet + F E_1]$
q_9 : $[F \to i\bullet]$
q_{10} : $[F \to (\bullet E)]$
q_{11} : $[E_1 \to + \bullet F E_1]$
q_{12} : $[F \to (E\bullet)]$
q_{13} : $[E_1 \to +F \bullet E_1]$
q_{14} : $[F \to (E)\bullet]$
q_{15} : $[E_1 \to +F E_1 \bullet]$

$q_0 i \to q_1 q_2 i$, $q_0(\to q_1 q_2($,
$q_1 \to \epsilon$,
$q_2 i \to q_3 q_4 i$, $q_2(\to q_3 q_5($,
$q_3 \# \to q_6 q_7 \#$, $q_3) \to q_6 q_7)$, $q_3 + \to q_6 q_8 +$,
$q_4 i \to q_9$,
$q_5(\to q_{10}$,
$q_6 \to \epsilon$,
$q_7 \to \epsilon$,
$q_8 + \to q_{11}$,
$q_9 \to \epsilon$,
$q_{10} i \to q_{12} q_2 i$, $q_{10}(\to q_{12} q_2($,
$q_{11} i \to q_{13} q_4 i$, $q_{11}(\to q_{13} q_5($,
$q_{12}) \to q_{14}$,
$q_{13} \# \to q_{15} q_7 \#$, $q_{13}) \to q_{15} q_7)$, $q_{13} + \to q_{15} q_8 +$,
$q_{14} \to \epsilon$,
$q_{15} \to \epsilon$

$q_0 q_0$	$(i+i)\#$	$q_0(\to q_1 q_2($
$q_0 q_1 q_2$	$(i+i)\#$	$q_2(\to q_3 q_5($
$q_0 q_1 q_3 q_5$	$(i+i)\#$	$q_5(\to q_{10}$
$q_0 q_1 q_3 q_{10}$	$i+i)\#$	$q_{10} i \to q_{12} q_2 i$
$q_0 q_1 q_3 q_{12} q_2$	$i+i)\#$	$q_2 i \to q_3 q_4 i$
$q_0 q_1 q_3 q_{12} q_3 q_4$	$i+i)\#$	$q_4 i \to q_9$
$q_0 q_1 q_3 q_{12} q_3 q_9$	$+i)\#$	$q_9 \to \varepsilon$
$q_0 q_1 q_3 q_{12} q_3$	$+i)\#$	$q_3 + \to q_6 q_8 +$
$q_0 q_1 q_3 q_{12} q_6 q_8$	$+i)\#$	$q_8 + \to q_{11}$
$q_0 q_1 q_3 q_{12} q_6 q_{11}$	$i)\#$	$q_{11} i + \to q_{13} q_4 i$
$q_0 q_1 q_3 q_{12} q_6 q_{13} q_4$	$i)\#$	$q_4 i \to q_9$
$q_0 q_1 q_3 q_{12} q_6 q_{13} q_9$	$)\#$	$q_9 \to \varepsilon$
$q_0 q_1 q_3 q_{12} q_6 q_{13}$	$)\#$	$q_{13}) \to q_{15} q_7)$
$q_0 q_1 q_3 q_{12} q_6 q_{15} q_7$	$)\#$	$q_7 \to \varepsilon$
$q_0 q_1 q_3 q_{12} q_6 q_{15}$	$)\#$	$q_{15} \to \varepsilon$
$q_0 q_1 q_3 q_{12} q_6$	$)\#$	$q_6 \to \varepsilon$
$q_0 q_1 q_3 q_{12}$	$)\#$	$q_{12}) \to q_{14}$
$q_0 q_1 q_3 q_{14}$	$\#$	$q_{14} \to \varepsilon$
$q_0 q_1 q_3$	$\#$	$q_3 \# \to q_6 q_7 \#$
$q_0 q_1 q_6 q_7$	$\#$	$q_7 \to \varepsilon$
$q_0 q_1 q_6$	$\#$	$q_6 \to \varepsilon$
$q_0 q_1$	$\#$	$q_1 \to \varepsilon$
q_0	$\#$	$q_1 \to \varepsilon$

$$q_0 = [Z \to .E]$$
$$q_1 = [Z \to E.],\ q_2 = [E \to .FE_1]$$
$$q_3 = [E \to F.E_1],\ q_5 = [F \to .(E)]$$
$$q_{10} = [F \to (.E)]$$
$$q_{12} = [F \to (E.)], q_2 = [E \to .FE_1]$$
$$q_3 = [E \to F.E_1], \ldots$$

Algoritm derivator LL(1)

Convertirea automatului LL(1) in proceduri recursive: Descendenta recursiva (Recursive descent)

- derivator descendent recursiv: starea automatului este o pozitie din derivator
- stiva - locatii de unde derivatorul poate relua executia
- daca starea e $[X \to \mu.B\nu; \omega]$, $B \in N$: se pune pe stiva informatia despre $[X \to \mu B.\nu; \omega]$ inainte de a lua in considerare $B \to \beta$.
- daca folosim limbaje de programare cu suport pt recursivitate: procedura pt fiecare nonterminal B + mecanismul standard de recursivitate pentru a implementa stiva automatului

Schema de program

$q \to \varepsilon$	q: end
$qt \to q'$	q: if symbol = t then next_symbol else error; q'
	q : X; q' :...
	....
	proc X:
$qt_1 \to q'q_1t_1$	*begin*
....	*case symbol of*
$qt_m \to q'q_mt_m$	t_1 : *begin* q_1 : ... *end*;
	....
	t_m: *begin* q_m : ... *end*;
unde	otherwise error
$q = [Y \to \mu.X\nu;]$	*end*
	end

Reguli de transformare

1. nonterminal X - procedura X; simbolul de start - programul principal
2. corpul functiei X:
   - ramificare case pt productiile cu X in partea stanga
   - fiecare nonterminal din partea dreapta a productiei - apel al procedurii corespunzatoare
   - fiecare terminal din partea dreapta a productiei - verificare a presentei terminalului, urmat de apel al *next_symbol*
3. daca niciunul dintre terminalele asteptate nu e prezent - apel functia de tratare a erorilor

- Pt tranzitii $qt_1 \rightarrow q'q_1t1...$
- schema program indica:
  $q : F(); q'$
  procedura $F()$ - case pt toate t_i
- $q_2 i \rightarrow q_3 q_4 i$, $q_2(\rightarrow q_3 q_5($
  $q_4 i \rightarrow q_9$, $q_9 \rightarrow \varepsilon$, $q_5(\rightarrow q_{10}$,
  $q_{10} i \rightarrow q_{12} q_2 i$, $q_{10}(\rightarrow q_{12} q_2($,
  $q_{12}) \rightarrow q_{14}$,
- $q_2 = [E \rightarrow .FE_1]$, $q_3 = [E \rightarrow F.E_1]$, $q_{10} = [F \rightarrow (.E)]$

```
q2: F(); q3

procedure F()
{ case symbol of
   'i' : { q4:  if (symbol == 'i') then next_symbol else
         error();
             q9:  ;}
   '(' : { q5:  if (symbol == '(') then next_symbol else
         error();
             q10: E();
             q12: if (symbol == ')') then next_symbol else
                  error();
             q14: ;}
    otherwise error();   }
```

```
derivator()
{ q0: E()
  q1: if (symbol != '#')
        error();
}
procedure E1()
{ case symbol of
    '#' , ')' : q7: ;
    '+' : {
          q8:  if (symbol == '+')  next_symbol(); else error
              ();
          q11: F();
          q13: E1;
          q15: ;
    otherwise   : error();
}
procedure F()
{ case symbol of
   'i' : { q4:  if (symbol == 'i') then next_symbol else
       error();
           q9:  ;}
   '(' : { q5:  if (symbol == '(') then next_symbol else
       error();
           q10: E();
           q12: if (symbol == ')') then next_symbol else
               error();
           q14: ;}
    otherwise error();   }
```

```
procedure E()
{ q2: F();
  q3: E1();
  q6: ;
 }
```

Parsing table - tabel de derivare

- Ullman 4.4 . Nonrecursive predictive parsing
- Table-driven predictive parsing: input, stiva, parsing table.
- Tabel de derivare: M[A,a] - A nonterminal, a - terminal sau #

Exemplu de tabel de derivare

	lookahead					
	i	+	*	(	)	#
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \varepsilon$	$T' \to *FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to i$			$F \to (E)$		

$$
\begin{aligned}
P = \{ & E \to TE' \\
& E' \to +TE' | \varepsilon \\
& T \to FT' \\
& T' \to *FT' | \varepsilon \\
& F \to (E) | id \}
\end{aligned}
$$

Algoritm de derivare predictiva cu tabel de derivare

```
  #S (simbol de start) pe stiva, string# la intrare
set ip  to point to the first symbol of input string
repeat
  let X be the top stack symbol and a the symbol pointed to
      by ip
  if X is a terminal or # then
     if X = a then
        pop X from the stack and advance ip
     else error()
  else
     if M[X,a] = X-> Y1 Y2 ...Yk then begin
        pop X fro the stack
        push Yk, Yk-1, ...Y1 onto the stack, with Y1 on top
        output the production X-> Y1 Y2 ...Yk

     else error()
until X=#
```

Algoritm de derivare predictiva cu tabel de derivare

```
  #S (simbol de start) pe stiva, string# la intrare
set ip  to point to the first symbol of input string
repeat
  let X be the top stack symbol and a the symbol pointed to
      by ip
  if X is a terminal or # then
     if X = a then
        pop X from the stack and advance ip
     else error()
  else
     if M[X,a] = X-> Y1 Y2 ...Yk then begin
        pop X fro the stack
        push Yk, Yk-1, ...Y1 onto the stack, with Y1 on top
        output the production X-> Y1 Y2 ...Yk

     else error()
until X=#
```

$\{tqt \to q | t \in T\} \cup$

$\{Xq \to x_n....x_1 q | X \to x_1 x_2 ... x_n \in P, n \geq 0, X \in N, X_i \in V\}$

Exemplu de tabel de derivare

	lookahead					
	id	+	*	(	)	#
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \varepsilon$	$T' \to *FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to id$			$F \to (E)$		

$$P = \{E \to TE'$$
$$E' \to +TE'|\varepsilon$$
$$T \to FT'$$
$$T' \to *FT'|\varepsilon$$
$$F \to (E)|id\}$$

simbol	$FIRST_1(X)$	$FOLLOW_1(X)$
E	$\{(, id\}$	$\{), \#\}$
E'	$\{+, \varepsilon\}$	$\{), \#\}$
T	$\{(, id\}$	$\{+, \#,)\}$
T'	$\{*, \varepsilon\}$	$\{+, \#,)\}$
F	$\{(, id\}$	$\{*, +, \#,)\}$

1. for each production $A \to \alpha$ do steps 2 and 3
2. for each terminal a in $FIRST(\alpha)$, add $A \to \alpha$ to $M[A, a]$
3. if $\varepsilon \in FIRST(\alpha)$, add$A \to \alpha$ to $M[A, b$ for each terminal $b \in FOLLOW(A)$. if $\varepsilon \in FIRST(\alpha)$ and $\# \in FOLLOW(A)$, add $A \to \alpha$ to $M[A, \#]$
4. Make each undefined entry of M be error

test it online

First, follow sets, predict set

Algoritm $First_1$ - gramatici fara recursivitate stanga

Nu intra la examen

Se aplica urmatoarele reguli pana cand nu se mai poate reduce nimic

- $FIRST(a) = \{a\}$
- $X \to \varepsilon$: $FIRST(X) = \varepsilon$
- $X \to Y_1 Y_2 Y_3$:
  - daca $\varepsilon \notin FIRST(Y_1)$ then $FIRST(X) = FIRST(Y_1)$
  - daca $\varepsilon \in FIRST(Y_1)$ then $FIRST(X) = (FIRST(Y_1) - \{\varepsilon\}) \cup FIRST(Y_2 Y_3)$

Algoritm $Follow_1$

Nu intra la examen

- penrtu simbolul de start: se adauga $\{\#\}$ in follow(Z)
- $X \to \alpha Y$: $FOLLOW(Y) = FOLLOW(X)$
- $X \to \alpha Y \beta$:
  - daca $\varepsilon \notin FIRST(\beta)$ then $FOLLOW(Y) = FIRST(\beta)$
  - daca $\varepsilon \in FIRST(\beta)$ then $FOLLOW(Y) = (FIRST(\beta) - \{\varepsilon\}) \cup FOLLOW(X)$

Parsare ascendenta (Bottom-up) 1.

Parsare ascendenta

- necesita luarea unei decizii dupa analiza sirului derivat dintr-o productie
- mai multa informatie pt decizie $\rightarrow$
  - clasa larga de gramatici
  - pret: cresterea complexitati procedurii de analiza si a automatului rezultat

Automat stiva - analiza ascendenta

Fie $G = (T, N, P, Z)$ o CFG si automatul stiva
$A = (T, \{q\}, R, q, \{q\}, V, \varepsilon)$ cu $V = T \cup N$ si R:
(alfabet, stari, productii, stare initiala, stari finale, alfabet stiva, continut initial stiva)

$$\{x_1 x_2 ... x_n q \to Xq | X \to x_1 x_2 ... x_n \in P, n \geq 0, X \in N, X_i \in V\} \cup$$
$$\{qt \to tq | t \in T\} \cup$$
$$\{Zq \to q\}$$

Automatul accepta un sir din L(G) lucrand inapoi printr-o derivare cea mai din dreapta a sirului.

Comparatie automat stiva in analiza descendenta vs ascendenta pt $G = (T, N, P, Z)$ o CFG

- descedenta - cursuri anterioare
  $A = (T, \{q\}, R, q, \{q\}, V, Z)$ cu $V = T \cup N$ si R:
  $\{tqt \to q | t \in T\} \cup$
  $\{Xq \to x_n....x_1 q | X \to x_1 x_2 ... x_n \in P, n \geq 0, X \in N, X_i \in V\}$

- ascendenta
  $A = (T, \{q\}, R, q, \{q\}, V, \varepsilon)$ cu $V = T \cup N$ si R:

  $\{x_1 x_2 ... x_n q \to Xq | X \to x_1 x_2 ... x_n \in P, n \geq 0, X \in N, X_i \in V\} \cup$
  $\{qt \to tq | t \in T\} \cup$
  $\{Zq \to q\}$

Exemplu

Fie $G_1 = (T, N, E, P)$

- $T = \{+, *, (,), i\}$, $N = \{E, T, F\}$
- cu productiile P
  - $(1, 2)E \to T | E + T$
  - $(3, 4)T \to F | T * F$
  - $(5, 6)F \to i | (E)$

Automatul stiva:

- $T = \{+, *, (,), i\}$, $Q = \{q\}$,
  $q_0 = q, F = \{q\}, S = \{+, -, *, (,), i, E, T, F\}, s_0 = \epsilon$
- cu productiile R
  1. $Tq \to Eq, E + Tq \to Eq$,
  2. $Fq \to Tq, T * Fq \to Tq$,
  3. $iq \to Fq, (E)q \to Fq$,
  4. $q+ \to +q, q* \to *q$, $q(\to (q, q) \to q), qi \to iq\}$
  5. $Eq \to q\}$

Derivarea gasita: i+i*i

stiva	stare	intrare	derivarea cea mai din dreapta
	q	$i + i * i$	i+i*i
i	q	$+i * i$	
F	q	$+i * i$	F+i*i
T	q	$+i * i$	T+i*i
E	q	$+i * i$	E+i*i
E+	q	$i * i$	
E+i	q	$*i$	
E+F	q	$*i$	E+F*i
E+T	q	$*i$	E+T*i
E+T*	q	i	
E+T*i	q	i	
E+T*F	q		
E+T	q		E+T*F
E	q		E+T
	q		E

Observatii

- coloana din dreapta este inversul derivarii celei mai din dreapta

- ascendenta - traseaza derivarea de jos la simbolul de start
- stiva contine la fiecare pas un sir din care se poate deriva portiunea de sir deja citita
- informatia semnificativa: perechea (ρ, σ), unde
  - $\rho \in V^*$ - continutul stivei,
  - $\sigma \in T^*$ - restul sirului de la intrare

LL	LR
Does a leftmost derivation.	Does a rightmost derivation in reverse.
Starts with the root nonterminal on the stack.	The last nonterminal on the stack is the root nonterminal.
Ends when the stack is empty.	Starts with an empty stack.
Uses the stack for designating what is still to be expected.	Uses the stack for designating what is already seen.
Builds the parse tree top-down.	Builds the parse tree bottom-up.
Continuously pops a nonterminal off the stack, and pushes the corresponding right hand side.	Tries to recognize a right hand side on the stack, pops it, and pushes the corresponding non-terminal.
Expands the non-terminals.	Reduces the non-terminals.
Reads the terminals when it pops one off the stack.	Reads the terminals while it pushes them on the stack.
Pre-order traversal of the parse tree.	Post-order traversal of the parse tree.

Clase de echivalenta pentru perechile (ρ, σ)

Pentru $p \in 1..n$, fie $X_p \to \chi_p$ productia a $p-a$ a gramaticii independente de context $G = (T, N, P, Z)$. Clasele de reducere $R_j, j \in 0, ..n$ sunt definite de

$$R_0 = \{(\rho, \sigma) | \rho = \mu\gamma, \sigma = \nu\omega \text{ a.i. } Z \Rightarrow^R \mu Y \omega, Y \Rightarrow^{R'} \gamma\nu, \nu \neq \varepsilon\}$$
$$R_p = \{(\rho, \sigma) | \rho = \mu\chi_p, Z \Rightarrow^R \mu X_p \sigma, X_p \Rightarrow \chi_p\}$$

unde $Y \Rightarrow^{R'} \alpha$ este $Y \Rightarrow^R \alpha$ si ultimul pas din derivare nu ia forma $Y_1\alpha \Rightarrow \alpha$

Clase de reducere - continuare

- clasele de reducere - perechile de siruri care ar putea sa apara in timpul analizei ascendente a unei propozitii din $L(G)$ de catre automatul stiva
- clasa de reducere careia ii apartine o pereche caracterizeaza tranzitia efectuata de catre automat cand acea pereche apare ca o configuratie
  1. $(\rho, \sigma) \in R_0$ - fraza simpla χ nu e complet in stiva; se aplica $qt \to tq$ cu $t = 1 : \sigma$ tranzitie de deplasare
  2. $(\rho, \sigma) \in R_p, p \in 1..n$ - fraza simpla χ_p e complet in stiva; se aplica $\chi_p q \to X_p q$ tranzitie de reducere
     Obs: pt $p = 1$ tranzitia $Zq \to q$ si automatul se opreste
  3. $(\rho, \sigma) \notin R_j, j \in 0..n$. nu mai sunt posibile alte tranzitii; sirul de intrare nu apartine $L(G)$

Clase stiva k

- Pentru un $k \geq 0$, multimile $R_{j,k}, k \in 0..n$ se numesc clase stiva k al gramaticii G daca

$$R_{j,k} = \{(\rho, \tau) | \exists (\rho, \sigma) \in R_j, \tau = k : \sigma\}$$

- Daca clasele stiva k sunt mutual disjuncte atuncti automatul stiva este determinist chiar si cand examinarea inainte este limitata la k simboluri

Gramatica LR(k)

O gramatica independenta de context $G = (T, N, P, Z)$ este $LR(k)$ pentru un $k \geq 0$ dat daca pentru derivari arbitrare

$$Z \Rightarrow^R \mu X \omega \Rightarrow \mu \chi \omega \quad \mu \in V^*, \omega \in T^*, X \to \chi \in P$$

$$Z \Rightarrow^R \mu' Y \omega' \Rightarrow \mu' \gamma \omega' \quad \mu' \in V^*, \omega' \in T^*, Y \to \gamma \in P$$

$(|\mu\chi| + k) : \mu\chi\omega = (|\mu'\gamma| + k) : \mu'\gamma\omega'$ implica
$\mu = \mu', X = Y, \chi = \gamma$

LR(k)

Automatul

- baleiaza sirul de intrare de la stanga la dreapta (Left to right)
- traversand inversa celei mai din dreapta derivari (Right)
- fara sa examinze mai mult de k simboluri de intrare intr-un pas

Teorema

O gramatica independenta de context este LR(k) daca si numai daca clasele sale stiva k sunt mutual disjuncte.

Verificarea proprietatii LR(k) prin intersectarea claselor stiva

NU intra la examen

- clasele stiva k contin o infinitate de perechi (ρ, τ), $\#\tau$ fiind finit datorita limitei de lungime, insa lungimea stivei nefiind limitata, $\#\rho$ este infinitate
- pentru fiecare clasa stiva k $R_{j,k}$ se poate preciza o gramatica regulata G_j a.i.

$$L(G_j) = \{(\rho \& \tau) | (\rho, \tau) \in R_{j,k}\}$$

- exista algoritmi pt a determina daca doua limbaje regulate sunt distincte

Situatii si inchidere nonterminal

Gramaticile regulate care genereaza clase stiva k:
Simbolurile nonterminale:

$$W = \{[X \to \mu.\nu; \omega] | X \to \mu\nu \in P, \omega \in FOLLOW_k(X)\}$$

Gramatici care genereaza clasele stiva k, fara a fi regulate

$$G'_j = (V \cup \{\&, \#\}, W, P' \cup P'' \cup P_j, [Z \to .S; \#])$$

$$P' = \{[X \to \mu.\nu\gamma; \omega] \to \nu[X \to \mu\nu.\gamma; \omega] \quad | \nu \in V\}$$
$$P'' = \{[X \to \mu.Y\gamma; \omega] \to [Y \to .\beta; \tau] \quad | Y \to \beta \in P, \tau \in EFF_k(\gamma\omega)\}$$
$$P_0 = \{[X \to \mu.\nu; \omega] \to \&\tau] \quad | \nu \neq \varepsilon, \tau \in EFF_k(\nu\omega)\}$$
$$P_p = \{[X_p \to \chi_p.; \omega] \to \&\omega\} \quad p \in 1..n$$

Care productii sunt permise in gramatica regulata?
Lungimile $\&\tau$, $\&\omega$ sunt finite datorita lui k;

Gramatici care genereaza clasele stiva k, fara a fi regulate

$$G'_j = (V \cup \{\&, \#\}, W, P' \cup P'' \cup P_j, [Z \to .S; \#])$$

$$P' = \{[X \to \mu.\nu\gamma; \omega] \to \nu[X \to \mu\nu.\gamma; \omega] \quad | \nu \in V\}$$
$$P'' = \{[X \to \mu.Y\gamma; \omega] \to [Y \to .\beta; \tau] \quad | Y \to \beta \in P, \tau \in EFF_k(\gamma\omega)\}$$
$$P_0 = \{[X \to \mu.\nu; \omega] \to \&\tau] \quad | \nu \neq \varepsilon, \tau \in EFF_k(\nu\omega)\}$$
$$P_p = \{[X_p \to \chi_p.; \omega] \to \&\omega\} \quad p \in 1..n$$

Care productii sunt permise in gramatica regulata?
P' *si* P_j *unde* $j \in 0..n$
Lungimile $\&\tau$, $\&\omega$ sunt finite datorita lui k; sunt considerate simboluri terminale

Inchiderea nonterminalului

o gramatica se poate rescrie a.i. sa nu contina productii precum cele din P''

- Inchiderea unui nonterminal

$$H(X) = \{X\} \cup \{Y | Y_l \to Y \in P, Y_l \in H(X)\}$$

Algoritm de rescriere a gramaticii

1. se selecteaza un $X \in N$ pentru care $H(X) \neq \{X\}$.
2. $P = P - \{X \to Y | Y \in N\}$
3. $P = P \cup \{X \to \beta | Y \to \beta \in P, Y \in H(X), \beta \notin N\}$
   Alg se termina cand nu se mai poate face nicio selectie la pasul 1

din G'_j rezulta G_j. Sirurile β sunt toate de forma $\nu[..], \&\tau$ sau $\&\omega$: deci gramatica regulata

Teorema - materie de examen

Pentru orice gramatica G de tipul LR(k) exista un automat stiva determinist A a.i. L(A) = L(G).

Constructia automatului se bazeaza pe gramaticile G_j:

- automatul genereaza clasele stiva k
- si le verifica fata de inversa celei mai din dreapta derivari a sirului
- in functie de clasa stiva k particulara, automatul
  - stivuieste simbolul de intrare, sau
  - reduce un numar de simboluri stivuite la un nonterminal

..continuare construire automat LR

- alg de construire genereaza treptat situatiile necesare si utilizeaza operatia de inchidere pentru evitarea productiilor din P''.
- o stare - o multime de situatii:
  - fiecare situatie dintr-o stare poate fi utilizata pentru derivarea clasei stiva k curente
- o alta formulare a inchiderii direct in functie de o multime de situatii M:

$$H(M) = M \cup \{[Y \to .\beta; \tau] | \exists [X \to \mu.Y\gamma; \omega] \in H(M), Y \to \beta \in P, \tau \in FIRST_k(\gamma\omega)\}$$

Algoritm LR(k)-determinare Q si R :

1. $Q = \{q_0\}$ si $R = \emptyset$ cu $q_0 = H([Z \to .S; \#])$
2. pt orice $q \in Q$ se efectueaza pasii 3-5 pt fiecare $\nu \in V$
3. fie $basis(q, \nu) = \{[X \to \mu\nu.\gamma; \omega] | [X \to \mu.\nu\gamma; \omega] \in q\}$
4. daca $basis(q, \nu) \neq \emptyset$ atunci $next(q, \nu) = H(basis(q, \nu))$. Se include $q' = next(q, \nu)$ in Q
5. daca $basis(q, \nu) \neq \emptyset$ si $\nu \in T$ se actualizeaza
   $$R = R \cup \begin{cases} \{q\nu \to qq'\}, & k \leq 1 \\ \{q\nu\tau \to qq'\tau | [X \to \mu.\nu\gamma; \omega] \in q, \tau \in FIRST_{k-1}(\gamma\omega)\}, & k > 1 \end{cases}$$
6. daca toate elementele lui Q au fost tratate se executa pasul 7 pt fiecare $q \in Q$ si alg se termina; altfel se continua pasul 2
7. pentru fiecare $[X \to \chi.; \omega] \in q$, unde $\chi = x_1..x_n$ se face

$$R = R \cup \{q_1..q_n q\omega \to q_1 q'\omega | [X \to .\chi; \omega] \in q_1,$$
$$q_{i+1} = next(q_i, x_i)(i \in 1..n-1),$$
$$q = next(q_n, x_n),$$
$$q' = next(q_1, X)\}$$

Exemplu LR(k) cu k=2

$$T = \{a, b, c\}, N = \{Z.X, Y\}$$
$$P = \{(1)Z \to X,$$
$$(2, 3)X \to Y|bYa,$$
$$(4, 5)Y \to c|ca\}$$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X; \#])$ $[Z \to .X; \#]$, $[X \to .Y; \#]$, $[X \to .bYa; \#]$, $[Y \to .c; \#]$, $[Y \to .ca; \#]$	$[Z \to X.; \#]$	$[X \to Y.; \#]$	-	$[X \to b.Ya; \#]$	$[Y \to c.; \#]$ $[Y \to c.a; \#]$
$next(q_0, ...)$	q_1	q_2		q_3	q_4
R				$q_0bc \to q_0q_3c$ $FIRST_1(Ya\#)$	$q_0c\# \to q_0q_4\#$ $FIRST_1(\#)$ $q_0ca \to q_0q_4a$ $FIRST_1(a\#)$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X; \#])$ $[Z \to .X; \#]$, $[X \to .Y; \#]$, $[X \to .bYa; \#]$, $[Y \to .c; \#]$, $[Y \to .ca; \#]$	$[Z \to X.; \#]$	$[X \to Y.; \#]$	-	$[X \to b.Ya; \#]$	$[Y \to c.; \#]$ $[Y \to c.a; \#]$
$next(q_0, ...)$	q_1	q_2		q_3	q_4
R				$q_0bc \to q_0q_3c$ $FIRST_1(Ya\#)$	$q_0c\# \to q_0q_4\#$ $FIRST_1(\#)$ $q_0ca \to q_0q_4a$ $FIRST_1(a\#)$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X; \#])$ $[Z \to .X; \#]$, $[X \to .Y; \#]$, $[X \to .bYa; \#]$, $[Y \to .c; \#]$, $[Y \to .ca; \#]$	$[Z \to X.; \#]$	$[X \to Y.; \#]$	-	$[X \to b.Ya; \#]$	$[Y \to c.; \#]$ $[Y \to c.a; \#]$
$next(q_0, ...)$	q_1	q_2		q_3	q_4
R				$q_0bc \to q_0q_3c$ $FIRST_1(Ya\#)$	$q_0c\# \to q_0q_4\#$ $FIRST_1(\#)$ $q_0ca \to q_0q_4a$ $FIRST_1(a\#)$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X; \#])$ $[Z \to .X; \#]$, $[X \to .Y; \#]$, $[X \to .bYa; \#]$, $[Y \to .c; \#]$, $[Y \to .ca; \#]$	$[Z \to X.; \#]$	$[X \to Y.; \#]$	-	$[X \to b.Ya; \#]$	$[Y \to c.; \#]$ $[Y \to c.a; \#]$
$next(q_0, ...)$	q_1	q_2		q_3	q_4
R				$q_0bc \to q_0q_3c$ $FIRST_1(Ya\#)$	$q_0c\# \to q_0q_4\#$ $FIRST_1(\#)$ $q_0ca \to q_0q_4a$ $FIRST_1(a\#)$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X; \#])$ $[Z \to .X; \#]$, $[X \to .Y; \#]$, $[X \to .bYa; \#]$, $[Y \to .c; \#]$, $[Y \to .ca; \#]$	$[Z \to X.; \#]$	$[X \to Y.; \#]$	-	$[X \to b.Ya; \#]$	$[Y \to c.; \#]$ $[Y \to c.a; \#]$
$next(q_0, ...)$	q_1	q_2		q_3	q_4
R				$q_0bc \to q_0q_3c$ $FIRST_1(Ya\#)$	$q_0c\# \to q_0q_4\#$ $FIRST_1(\#)$ $q_0ca \to q_0q_4a$ $FIRST_1(a\#)$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X; \#])$ $[Z \to .X; \#]$, $[X \to .Y; \#]$, $[X \to .bYa; \#]$, $[Y \to .c; \#]$, $[Y \to .ca; \#]$	$[Z \to X.; \#]$	$[X \to Y.; \#]$	-	$[X \to b.Ya; \#]$	$[Y \to c.; \#]$ $[Y \to c.a; \#]$
$next(q_0, ...)$	q_1	q_2		q_3	q_4
R				$q_0bc \to q_0q_3c$ $FIRST_1(Ya\#)$	$q_0c\# \to q_0q_4\#$ $FIRST_1(\#)$ $q_0ca \to q_0q_4a$ $FIRST_1(a\#)$
q_1 $H([Z \to X.; \#])$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X; \#])$ $[Z \to .X; \#]$, $[X \to .Y; \#]$, $[X \to .bYa; \#]$, $[Y \to .c; \#]$, $[Y \to .ca; \#]$	$[Z \to X.; \#]$	$[X \to Y.; \#]$	-	$[X \to b.Ya; \#]$	$[Y \to c.; \#]$ $[Y \to c.a; \#]$
$next(q_0, ...)$	q_1	q_2		q_3	q_4
R				$q_0bc \to q_0q_3c$ $FIRST_1(Ya\#)$	$q_0c\# \to q_0q_4\#$ $FIRST_1(\#)$ $q_0ca \to q_0q_4a$ $FIRST_1(a\#)$
q_1 $H([Z \to X.; \#])$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
q_2 $H([X \to Y.; \#])$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X; \#])$ $[Z \to .X; \#]$, $[X \to .Y; \#]$, $[X \to .bYa; \#]$, $[Y \to .c; \#]$, $[Y \to .ca; \#]$	$[Z \to X.; \#]$	$[X \to Y.; \#]$	-	$[X \to b.Ya; \#]$	$[Y \to c.; \#]$ $[Y \to c.a; \#]$
$next(q_0, ...)$	q_1	q_2		q_3	q_4
R				$q_0bc \to q_0q_3c$ $FIRST_1(Ya\#)$	$q_0c\# \to q_0q_4\#$ $FIRST_1(\#)$ $q_0ca \to q_0q_4a$ $FIRST_1(a\#)$
q_1 $H([Z \to X.; \#])$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
q_2 $H([X \to Y.; \#])$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$q_3 : H([X \to b.Ya; \#])$ $[X \to b.Ya; \#]$ $[Y \to .c; a\#]$ $[Y \to .ca; a\#]$	$\emptyset$	$[X \to bY.a.; \#]$	$\emptyset$	$\emptyset$	$[Y \to c.; a\#]$ $[Y \to c.a; a\#]$
$next(q_3, ...)$		q_5			q_6
R					$q_3ca \to q_3q_6a$

Stare	X	Y	a	b	c
q_4 $H([Y \to c.; \#],)$ $[Y \to c.a; \#])$	$\emptyset$	$\emptyset$	$[Y \to ca.; \#]$	$\emptyset$	$\emptyset$
$next(q_5 ...)$			q_7		
R			$q_4 a\# \to q_4 q_7 \#$		
q_5 $H([X \to bY.a; \#],)$	$\emptyset$	$\emptyset$	$[X \to bYa.; \#]$	$\emptyset$	$\emptyset$
$next(q_5, ...)$			q_8		
R			$q_5 a\# \to q_5 q_8 \#$		
q_6 $H([Y \to c.; a\#],)$ $[Y \to c.a; a\#]$	$\emptyset$	$\emptyset$	$[Y \to ca.; a\#]$	$\emptyset$	$\emptyset$
$next(q_6, ...)$			q_9		
R			$q_6 aa \to q_6 q_9 a$		
q_7 $H([Y \to ca.; \#])$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
q_8 $H([X \to bYa.; \#])$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
q_9 $H([Y \to ca.; a\#])$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

pas 7. Pentru fiecare $[X \to \chi.;\omega] \in q$, unde $\chi = x_1..x_n$ se face

$$R = R \cup \{q_1..q_n q\omega \to q_1 q'\omega | [X \to .\chi;\omega] \in q_1,$$
$$q_{i+1} = next(q_i, x_i)(i \in 1..n-1),$$
$$q = next(q_n, x_n),$$
$$q' = next(q_1, X)\}$$

Pt q_2: $H([X \to Y.;\#])$

rol de q_1 $[X \to .Y;\#] \in ?$

rol de q $? = next(?, Y)$

rol de q' $? = next(?, X)$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X;\#])$ $[Z \to .X;\#]$, $[X \to .Y;\#]$, $[X \to .bYa;\#]$, $[Y \to .c;\#]$, $[Y \to .ca;\#]$	$[Z \to X.;\#]$	$[X \to Y.;\#]$	-	$[X \to b.Ya;\#]$	$[Y \to c.;\#]$ $[Y \to c.a;\#$
$next(q_0, ...)$	q_1	q_2		q_3	q_4

pas 7. Pentru fiecare $[X \to \chi.;\omega] \in q$, unde $\chi = x_1..x_n$ se face

$$R = R \cup \{q_1..q_n q\omega \to q_1 q'\omega | [X \to .\chi;\omega] \in q_1,$$
$$q_{i+1} = next(q_i, x_i)(i \in 1..n-1),$$
$$q = next(q_n, x_n),$$
$$q' = next(q_1, X)\}$$

Pt q_2: $H([X \to Y.;\#])$

rol de q_1 $[X \to .Y;\#] \in q_0$

rol de q $q_2 = next(q_0, Y)$

rol de q' $q_1 = next(q_0, X)$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X;\#])$ $[Z \to .X;\#]$, $[X \to .Y;\#]$, $[X \to .bYa;\#]$, $[Y \to .c;\#]$, $[Y \to .ca;\#]$	$[Z \to X.;\#]$	$[X \to Y.;\#]$	-	$[X \to b.Ya;\#]$	$[Y \to c.;\#]$ $[Y \to c.a;\#$
$next(q_0, ...)$	q_1	q_2		q_3	q_4

pas 7. Pentru fiecare $[X \to \chi.;\omega] \in q$, unde $\chi = x_1..x_n$ se face

$$R = R \cup \{q_1..q_n q\omega \to q_1 q'\omega | [X \to .\chi;\omega] \in q_1,$$
$$q_{i+1} = next(q_i, x_i)(i \in 1..n-1),$$
$$q = next(q_n, x_n),$$
$$q' = next(q_1, X)\}$$

Pt q_2: $H([X \to Y.;\#])$

rol de q_1 $[X \to .Y;\#] \in q_0$

rol de q $q_2 = next(q_0, Y)$

rol de q' $q_1 = next(q_0, X)$ $\quad q_0 q_2 \# \to q_0 q_1 \#$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X;\#])$ $[Z \to .X;\#]$, $[X \to .Y;\#]$, $[X \to .bYa;\#]$, $[Y \to .c;\#]$, $[Y \to .ca;\#]$	$[Z \to X.;\#]$	$[X \to Y.;\#]$	-	$[X \to b.Ya;\#]$	$[Y \to c.;\#]$ $[Y \to c.a;\#$
$next(q_0, ...)$	q_1	q_2		q_3	q_4

pas 7. Pentru fiecare $[X \to \chi.;\omega] \in q$, unde $\chi = x_1..x_n$ se face

$$R = R \cup \{q_1..q_n q\omega \to q_1 q'\omega | [X \to .\chi;\omega] \in q_1,$$
$$q_{i+1} = next(q_i, x_i)(i \in 1..n-1),$$
$$q = next(q_n, x_n),$$
$$q' = next(q_1, X)\}$$

Pt q_4: $H([Y \to c.;\#], [Y \to c.a;\#])$

q_1 $[Y \to .c;\#] \in q_0$

q $q_4 = next(q_0, c)$

q' $q_2 = next(q_0, Y)$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X;\#])$ $[Z \to .X;\#]$, $[X \to .Y;\#]$, $[X \to .bYa;\#]$, $[Y \to .c;\#]$, $[Y \to .ca;\#]$	$[Z \to X.;\#]$	$[X \to Y.;\#]$	-	$[X \to b.Ya;\#]$	$[Y \to c.;\#]$ $[Y \to c.a;\#$
$next(q_0, ...)$	q_1	q_2		q_3	q_4

pas 7. Pentru fiecare $[X \to \chi.;\omega] \in q$, unde $\chi = x_1..x_n$ se face

$$R = R \cup \{q_1..q_n q\omega \to q_1 q'\omega | [X \to .\chi;\omega] \in q_1,$$
$$q_{i+1} = next(q_i, x_i)(i \in 1..n-1),$$
$$q = next(q_n, x_n),$$
$$q' = next(q_1, X)\}$$

Pt q_4: $H([Y \to c.;\#], [Y \to c.a;\#])$

q_1 $[Y \to .c;\#] \in q_0$

q $q_4 = next(q_0, c)$

q' $q_2 = next(q_0, Y)$ $\quad q_0 q_4 \# \to q_0 q_2 \#$

Stare	X	Y	a	b	c
q_0 $H([Z \to .X;\#])$ $[Z \to .X;\#]$, $[X \to .Y;\#]$, $[X \to .bYa;\#]$, $[Y \to .c;\#]$, $[Y \to .ca;\#]$	$[Z \to X.;\#]$	$[X \to Y.;\#]$	-	$[X \to b.Ya;\#]$	$[Y \to c.;\#]$ $[Y \to c.a;\#$
$next(q_0, ...)$	q_1	q_2		q_3	q_4

Pt q_6: $H([Y \to c.; a\#], [Y \to c.a; a\#])$

q_1 $[Y \to .c; a\#] \in q_3$

q $q_6 = next(q_3, c)$

q' $q_5 = next(q_3, Y)$ $\qquad q_3 q_6 a\# \to q_3 q_5 a\#$

Pt q_7: $H([Y \to ca.; \#])$

q_1 $[Y \to .ca; \#] \in q_0$

q_2 $q_4 = next(q_0, c)$

q $q_7 = next(q_4, a)$

q' $q_2 = next(q_0, Y)$ $\qquad q_0 q_4 q_7 \# \to q_0 q_2 \#$

Pt q_8: $H([X \to bYa.; \#]$

q_1 $[X \to .bYa; \#] \in q_0$

q_2 $q_3 = next(q_0, b)$

q_3 $q_5 = next(q_3, Y)$

q $q_8 = next(q_5, a)$

q' $q_1 = next(q_0, X)$ $\qquad q_0 q_3 q_5 q_8 \# \to q_0 q_1 \#$

Pt q_9: $H([Y \to ca.; a\#])$

q_1 $[Y \to .ca; a\#] \in q_3$

q_2 $q_6 = next(q_3, c)$

q $q_9 = next(q_6, a)$

q' $q_5 = next(q_3, Y)$ $\qquad q_3 q_6 q_9 a\# \to q_3 q_5 a\#$

q_0: $[Z \to \bullet X; \#]$
$[X \to \bullet Y; \#]$
$[X \to \bullet bY\,a; \#]$
$[Y \to \bullet c; \#]$
$[Y \to \bullet ca; \#]$

q_1: $[Z \to X \bullet; \#]$

q_2: $[X \to Y \bullet; \#]$

q_3: $[X \to b \bullet Y\,a; \#]$
$[Y \to \bullet c; a\#]$
$[Y \to \bullet ca; a\#]$

q_4: $[Y \to c \bullet; \#]$
$[Y \to c \bullet a; \#]$

q_5: $[X \to bY \bullet a; \#]$

q_6: $[Y \to c \bullet; a\#]$
$[Y \to c \bullet a; a\#]$

q_7: $[Y \to ca \bullet; \#]$

q_8: $[X \to bY\,a \bullet; \#]$

q_9: $[Y \to ca \bullet; a\#]$

$$
\begin{aligned}
R = \{ & q_0bc \to q_0q_3c, \\
& q_0c\# \to q_0q_4\#, \\
& q_0ca \to q_0q_4a, \\
& q_3ca \to q_3q_6a, \\
& q_4a\# \to q_4q_7\#, \\
& q_5a\# \to q_5q_8\#, \\
& q_6aa \to q_6q_9a, \\
& q_0q_2\# \to q_0q_1\#, \\
& q_0q_4\# \to q_0q_2\#, \\
& q_3q_6a\# \to q_3q_5a\#, \\
& q_0q_4q_7\# \to q_0q_2\#, \\
& q_0q_3q_5q_8\# \to q_0q_1\#, \\
& q_3q_6q_9a\# \to q_3q_5a\# \}
\end{aligned}
$$

q_0: $[Z \to \bullet X; \#]$
$[X \to \bullet Y; \#]$
$[X \to \bullet bYa; \#]$
$[Y \to \bullet c; \#]$
$[Y \to \bullet ca; \#]$

q_1: $[Z \to X\bullet; \#]$

q_2: $[X \to Y\bullet; \#]$

q_3: $[X \to b \bullet Ya; \#]$
$[Y \to \bullet c; a\#]$
$[Y \to \bullet ca; a\#]$

q_4: $[Y \to c \bullet ; \#]$
$[Y \to c \bullet a; \#]$

q_5: $[X \to bY \bullet a; \#]$

q_6: $[Y \to c\bullet; a\#]$
$[Y \to c \bullet a; a\#]$

q_7: $[Y \to ca\bullet; \#]$

q_8: $[X \to bYa\bullet; \#]$

q_9: $[Y \to ca\bullet; a\#]$

$$
\begin{aligned}
R = \{ & q_0bc \to q_0q_3c, \\
& q_0c\# \to q_0q_4\#, \\
& q_0ca \to q_0q_4a, \\
& q_3ca \to q_3q_6a, \\
& q_4a\# \to q_4q_7\#, \\
& q_5a\# \to q_5q_8\#, \\
& q_6aa \to q_6q_9a, \\
& q_0q_2\# \to q_0q_1\#, \\
& q_0q_4\# \to q_0q_2\#, \\
& q_3q_6a\# \to q_3q_5a\#, \\
& q_0q_4q_7\# \to q_0q_2\#, \\
& q_0q_3q_5q_8\# \to q_0q_1\#, \\
& q_3q_6q_9a\# \to q_3q_5a\# \}
\end{aligned}
$$

Cu $k = 1$: aceleasi stari; $k = 0$ ar fuziona q_4, q_6, respectiv q_7, q_9. Dar: un singur simbol inainte nu face distinctie intre tranzitiile de deplasare (shift) si reducere (reduce) din starea 6.

$Z \Rightarrow X \rightarrow bYa \Rightarrow bcaa$

Stiva	Stare	intrare	$\Rightarrow^R$	*tranzitie*
$\#$	q_0	bcaa$\#$	bcaa	1
$\#q_0$	q_3	caa$\#$		3
$\#q_0q_3$	q_6	aa$\#$		6
$\#q_0q_3q_6$	q_9	a$\#$	bYa	12
$\#q_0q_3$	q_5	a$\#$		5
$\#q_0q_3q_5$	q_8	$\#$	X	11
$\#q_0$	q_1	$\#$	Z	

Derivatoare LR

Algoritmul LR(k) - poate fi folosit atat pt a verifica daca o gramatica este LR(1) cat si pt construirea derivatorului sau
Algoritmul LR(k): numarul de stari este foarte mare

- similar cazului strong LL(k), multe tranzitii din LR(1) sunt independente de simbolul de lookahead
- $\Rightarrow$ putem construi un parser cu mai putine stari care implementeaza analiza LR(1) dar cu tranzitii mai putine, foosind lookahead doar cand este necesar

- LR(k)
- *simple* LR(k) : SLR(k)
- *lookahead* LR(k): LALR(k)

- se porneste cu LR(0): nu examineaza deloc simbolurile dinainte
- si se folosete lookahead doar la nevoie - simple LR(1) (SLR(1))
- Obs: nu toate LR(1) sunt SLR(1)
- LALR(1) - lookahead aplicat la SLR(1)

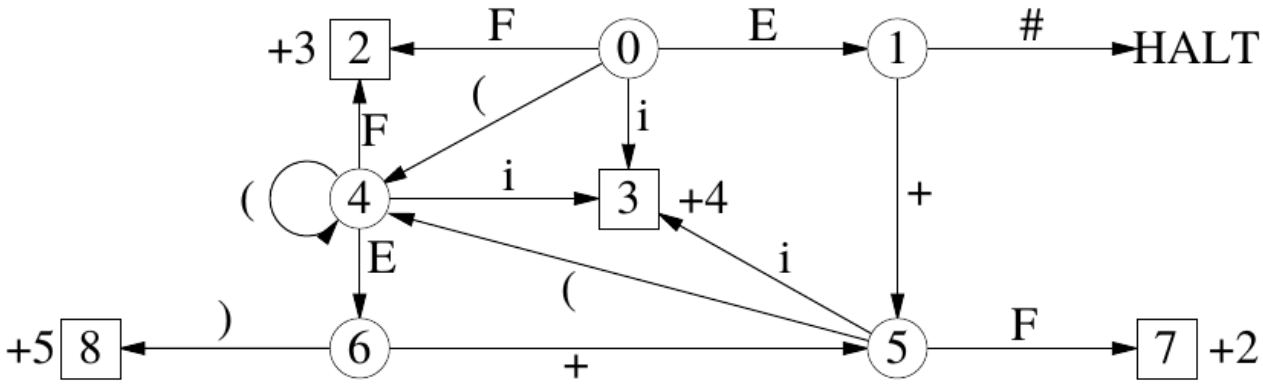
Tabel de tranzitii

$(1)Z \to E \quad (2)E \to E + F \quad (3)E \to F (4)F \to i \quad (5)F \to (E)$

	i	$($	$)$	$+$	$\#$	E	F
0	3	4	.	.	.	1	2
1	.	.	.	5	*		
2	+3	+3	+3	+3	+3		
3	+4	+4	+4	+4	+4		
4	3	4	.	.	.	6	2
5	3	4	.	.	.		7
6	.	.	8	5	.		
7	+2	+2	+2	+2	+2		
8	+5	+5	+5	+5	+5		

Diagrama de tranzitii

- suprapunerea gramaticilor regulate care corespund claselor stiva k
- starile prin care trece sunt inarcate pe stiva pana cand se ajunge intr-o stare finala
- in stare finala se face reducerea pe baza productiei $X \to \chi$, se elimina $|\chi|$ stari de pe stiva si se continua ca si cand s-ar fi citit simbolul X



$(1)Z \to E \quad (2)E \to E + F \quad (3)E \to F \quad (4)F \to i \quad (5)F \to (E)$

Stiva	Derivare dreapta	simbol urmator	reducere cu productia	stare urmatoare
0	.i+(i+i)#	i		3
0 3	i.+(i+i)#		4	2
0 2	F.+(i+i)#		3	1
0 1	E.+(i+i)#	+		5
0 1 5	E+.(i+i)#	(		4
0 1 5 4	E+(.i+i)#	i		3
0 1 5 4 3	E+(i.+i)#		4	2
0 1 5 4 2	E+(F.+i)#		3	6
0 1 5 4 6	E+(E.+i)#	+		5
0 1 5 4 6 5	E+(E+.i)#	i		3
0 1 5 4 6 5 3	E+(E+i.)#		4	7
0 1 5 4 6 5 7	E+(E+F.)#		2	6
0 1 5 4 6	E+(E.)#	)		8
0 1 5 4 6 8	E+(E).#		5	7
0 1 5 7	E+F.#		2	1
0 1	E.#			

- Instrumente pentru reprezentare
  - Siruri de rescriere
  - Gramatici - ierarhia lui Chomsky
  - Derivari si arbori de derivare
- Gramatici regulate si automate finite
  - Automate finite
  - Diagrame de stare si expresii regulate
- Gramatici independente de context si automate stiva: Automate stiva
- Analiza sintatica descendenta:
  - LL(k)
  - eliminare recursivitate stanga
  - Factorizare stanga
  - gramatici LL(k) tari
  - Derivator LL(1) - segmente de program
- Analiza sintatica ascendenta
  - LR(k)
  - Derivator LR(0) - functia de tranzitie
  - SLR(1)