

# Limbaje formale, automate și compilatoare

Curs 10

- ▶ Analiza sintactică descendentă
  - Parser (Analizor de sintaxă) descendent general
- ▶ Gramatici LL(1)
  - Definiție
  - Caracterizare
  - FIRST, FOLLOW
  - Tabela de parsare
  - Algoritmul de analiză sintactică LL(1)

# Analizor sintactic descendent.

## Configurații

- ▶ O configurație  $(u\#, \gamma\#, \pi)$  este interpretată în felul următor:
  - $-\gamma\#$  este conținutul stivei cu simbolul  $\#$  la bază.
  - $-u\#$  este conținutul intrării.
  - $-\pi$  este conținutul ieșirii.

# Analizor sintactic descendent.

## Tranziții

- ▶ Dacă  $C$  este mulțimea configurațiilor atunci  $\vdash \subseteq C \times C$  este relația de tranziție definită astfel:
  - $(u\#, A\gamma\#, \pi) \vdash (u\#, \beta\gamma\#, \pi r)$ , unde  $r = A \rightarrow \beta \in P$ . (aplicare regulă, **expandare**)
  - $(uv\#, u\gamma\#, \pi) \vdash (v\#, \gamma\#, \pi)$ . (**potrivire**)
  - $-(\#, \#, \pi)$  este configurație de acceptare dacă  $\pi \neq \varepsilon$ .
  - –O configurație  $c$  pentru care nu există  $c'$  astfel ca  $c \vdash c'$  produce **eroare**.
  - **Configurații inițiale**:  $(w\#, S\#, \varepsilon)$  unde  $w \in T^*$

# Exemplu

- 1.  $E \rightarrow E + T$ , 2.  $E \rightarrow T$ , 3.  $T \rightarrow T * F$ , 4.  $T \rightarrow F$ , 5.  $F \rightarrow (E)$ , 6.  $F \rightarrow a$

Intrare	Stivă	Ieșire
$a + a * a \#$	$E \#$	
$a + a * a \#$	$E + T \#$	1
$a + a * a \#$	$T + T \#$	1 2
$a + a * a \#$	$F + T \#$	1 2 4
$a + a * a \#$	$a + T \#$	1 2 4 6
$a * a \#$	$T \#$	1 2 4 6
$a * a \#$	$T * F \#$	1 2 4 6 3
$a * a \#$	$F * F \#$	1 2 4 6 3 4
$a * a \#$	$a * F \#$	1 2 4 6 3 4 6
$a \#$	$F \#$	1 2 4 6 3 4 6
$a \#$	$a \#$	1 2 4 6 3 4 6 6
$\#$	$\#$	1 2 4 6 3 4 6 6

# Corectitudinea analizorului sintactic

## ▶ Teorema (de corectitudine a analizorului)

- Fie gramatica redusă  $G=(N,T,S,P)$  și  $w \in T^*$ . Atunci, are loc  $(w\#,S\#, \varepsilon) \vdash^+ (\#, \#, \pi)$  (acceptare) dacă și numai dacă  $w \in L(G)$  și  $\pi$  este o derivare extrem stângă a cuvântului  $w$ .

## ▶ Lema 1

- Dacă în analizorul sintactic descendent atașat gramaticii  $G=(N,T,S,P)$  are loc calculul  $(uv\#,u\gamma\#, \varepsilon) \vdash^+ (v\#, \psi\#, \pi)$ , atunci în gramatica  $G$  are loc derivarea  $\gamma^\pi \Rightarrow_{st} u\psi$ , oricare ar fi  $u,v \in T^*$ ,  $\gamma, \psi \in \Sigma^*$ ,  $\pi \in P^*$ .

## ▶ Lema 2

- Dacă în gramatica  $G$  are loc derivarea  $\gamma^\pi \Rightarrow_{st} u\psi$  și  $1:\psi \in N \cup \{\varepsilon\}$  atunci în parserul descendent are loc calculul:  
 $(uv\#,u\gamma\#, \varepsilon) \vdash^+ (v\#, \psi\#, \pi)$ ,  $\forall v \in T^*$ .

## ▶ Demonstrații: inducție după lungimea lui $\pi$

# Observații

- ▶ Parserul este nedeterminist
- ▶ Există tipuri de gramatici pentru care este determinist?
- ▶ Construirea unei tabele de parsare

# Gramatici LL(k)

- ▶ LL(k): Parsing from Left to right using Leftmost derivation and k symbols lookahead.
- ▶ Informal, o gramatică este LL(k) dacă tranziția de tip “aplicare producție” din parser se face cu o unică regulă  $A \rightarrow \beta$  determinată prin următoarele k simboluri de la intrare.
- ▶ Definiție
  - $k:\alpha$  reprezintă primele k simboluri din  $\alpha$  (sau  $\alpha$  dacă  $|\alpha| < k$ )
  - $\alpha:k$  reprezintă ultimele k simboluri din  $\alpha$  (sau  $\alpha$  dacă  $|\alpha| < k$ )



# Gramatici LL(k)

## ► Definiție

- O gramatică independentă de context redusă este gramatică LL(k),  $k \geq 1$ , dacă pentru orice două derivări de forma:
  - $S \Rightarrow_{st}^* uA\gamma \Rightarrow_{st} u\beta_1\gamma \Rightarrow_{st}^* ux$
  - $S \Rightarrow_{st}^* uA\gamma \Rightarrow_{st} u\beta_2\gamma \Rightarrow_{st}^* uy$
- unde  $u, x, y \in T^*$ , pentru care  $k:x = k:y$ , are loc  $\beta_1 = \beta_2$ .

# Gramatici LL(k)

## ▶ Teorema

- Orice gramatică LL(k) este neambiguă.

## ▶ Teorema

- Dacă G este o gramatică stâng recursivă, atunci nu există nici un număr k astfel încât G să fie LL(k).

## ▶ Teorema

- Clasele de limbaje LL(k) formează o ierarhie infinită:
  - $\mathcal{LL}(0) \subset \mathcal{LL}(1) \subset \mathcal{LL}(2) \subset \dots \subset \mathcal{LL}(k) \subset \mathcal{LL}(k+1) \subset \dots$

## ▶ Lema

- Există limbaje care nu sunt LL(k) pentru nici o valoare  $k \in \mathbb{N}$ .

# Gramatici LL(1). Caracterizare

## ► Teoremă

- O gramatică  $G = (N, T, S, P)$  este gramatică LL(1) dacă și numai dacă pentru orice  $A \in N$  și pentru orice două producții  $A \rightarrow \beta_1 | \beta_2$  are loc:
  - $\text{FIRST}(\beta_1 \text{ FOLLOW}(A)) \cap \text{FIRST}(\beta_2 \text{ FOLLOW}(A)) = \emptyset$

# Tabela de parsare LL(1)

- ▶ 1.for( $A \in N$ )
  - 2.for( $a \in T \cup \{\#\}$ )
    - 3. $M(A,a) = \emptyset$ ;
- ▶ 4.for( $p = A \rightarrow \beta \in P$ ) {
  - 5.for( $a \in \text{FIRST}(\beta) - \{\epsilon\}$ )
    - 6. $M(A,a) = M(A,a) \cup \{(\beta,p)\}$ ;
  - 7.if( $\epsilon \in \text{FIRST}(\beta)$ ) {
    - 8.for( $b \in \text{FOLLOW}(A)$ ) {
      - 9.if( $b == \epsilon$ )  $M(A,\#) = M(A,\#) \cup \{(\beta,p)\}$ ;
      - 10.else  $M(A,b) = M(A,b) \cup \{(\beta,p)\}$ ;
    - } //endfor
  - } //endif
- ▶ } //endfor
- ▶ 11.for( $A \in N$ )
  - 12.for( $a \in T \cup \{\#\}$ )
    - 13.if( $M(A,a) = \emptyset$ )  $M(A,a) = \{\text{eroare}\}$ ;

# Exemplu

►  $S \rightarrow aSa \mid bSb \mid c$

M	a	b	c	#
S	(aSa, 1)	(bSb, 2)	(c, 3)	eroare

►  $S \rightarrow aSa \mid bSb \mid a \mid b \mid c \mid \varepsilon$

M	a	b	c	#
S	(aSa, 1) (a, 3) ( $\varepsilon$ , 6)	(bSb, 2) (b, 4) ( $\varepsilon$ , 6)	(c, 5)	( $\varepsilon$ , 6)

# Parser LL(1)

- ▶ Configurația inițială:  $(w\#, S\#, \varepsilon)$
- ▶ Tranziții
  - $(u\#, A\gamma\#, \pi) \vdash (u\#, \beta\gamma\#, \pi r)$ , dacă  $M(A, 1:u\#)=(\beta, r)$ , (**expandare**)
  - $(uv\#, u\gamma\#, \pi) \vdash (v\#, \gamma\#, \pi)$  (**potrivire**)
  - $(\#, \#, \pi) \vdash \text{acceptare}$ , dacă  $\pi \neq \varepsilon$  (**acceptare**)
  - $(au\#, b\gamma\#, \pi) \vdash \text{eroare}$  dacă  $a \neq b$
  - $(u\#, A\gamma\#, \pi) \vdash \text{eroare}$  dacă  $M(A, 1:u\#)=\text{eroare}$

# Exemplu

- 1.  $S \rightarrow E$ , 2.  $S \rightarrow B$ , 3.  $E \rightarrow \varepsilon$ , 4.  $B \rightarrow a$ , 5.  $B \rightarrow \text{begin SC end}$ , 6.  $C \rightarrow \varepsilon$ , 7.  $C \rightarrow ;SC$

X	FIRST(X)	FOLLOW(X)
S	a begin $\varepsilon$	end ; $\varepsilon$
E	$\varepsilon$	end ; $\varepsilon$
B	a begin	end ; $\varepsilon$
C	; $\varepsilon$	end

M	a	begin	end	;	#
S	(B, 2)	(B, 2)	(E, 1)	(E, 1)	(E, 1)
E	eroare	eroare	( $\varepsilon$ , 3)	( $\varepsilon$ , 3)	( $\varepsilon$ , 3)
B	(a, 4)	(begin SC end, 5)	eroare	eroare	eroare
C	eroare	eroare	( $\varepsilon$ , 6)	(;SC, 7)	eroare

Intrare	Stivă	Acțiune	Ieșire
begin a;;a end#	S#	expandare	2
begin a;;a end#	B#	expandare	5
begin a;;a end#	begin SC end#	potrivire	
a;;a end#	SC end#	expandare	2
a;;a end#	BC end#	expandare	4
a;;a end#	aC end#	potrivire	
;;a end#	C end#	expandare	7
;;a end#	;SC end#	potrivire	
;a end#	SC end#	expandare	1
;a end#	EC end#	expandare	3
;a end#	C end#	expandare	7
;a end#	;SC end#	potrivire	
a end#	SC end#	expandare	2
a end#	BC end#	expandare	4
a end#	aC end#	potrivire	
end#	C end#	expandare	6
end#	end#	potrivire	
#	#	acceptare	



# Eliminarea recursiei stângi

- ▶ Fie  $G = (N, T, S, P)$  o gramatică în formă redusă
- ▶ Fie  $A \in N$  imediat recursiv
- ▶ Fie  $A \rightarrow A\alpha_1 | A\alpha_2 | \dots A\alpha_k | \beta_1 | \beta_1 | \dots$  toate regulile care încep cu  $A$ . Fie  $P_A$  mulțimea acestor reguli.
- ▶ Gramatica  $G'$  unde  $A$  nu este recursiv imediat
  - $G' = (N \cup \{A'\}, T, S, P')$
  - $P' = P - P_A \cup \{A' \rightarrow \alpha_1 A' | \alpha_2 A' | \dots \alpha_k A' | \varepsilon, A \rightarrow \beta_1 A' | \beta_2 A' | \dots\}$

# Eliminarea recursiei stângi

- ▶  $E \rightarrow E+T \mid E-T \mid -T \mid T$

- ▶  $T \rightarrow T*F \mid T/ F \mid F$

- ▶  $F \rightarrow (E) \mid a$

- ▶  $E \rightarrow TE' \mid -TE'$

- ▶  $E' \rightarrow +T E' \mid -TE' \mid \varepsilon$

- ▶  $T \rightarrow FT'$

- ▶  $T' \rightarrow *FT' \mid /FT' \mid \varepsilon$

- ▶  $F \rightarrow (E) \mid a$

# Eliminarea recursiei stângi

X	FIRST(X)	FOLLOW(X)
E	( a -	$\epsilon$ )
E'	+ - $\epsilon$	$\epsilon$ )
T	( a	+ - $\epsilon$ )
T'	* / $\epsilon$	+ - $\epsilon$ )
F	( a	* / + - $\epsilon$ )

M	a	+	-	*	/	(	)	#
E	(TE', 1)	eroare	(-TE', 2)	eroare	eroare	(TE', 1)	eroare	eroare
E'	eroare	(+TE', 3)	(-TE', 4)	eroare	eroare	eroare	( $\epsilon$ , 5)	( $\epsilon$ , 5)
T	(FT', 6)	eroare	eroare	eroare	eroare	(FT', 6)	eroare	eroare
T'	eroare	( $\epsilon$ , 9)	( $\epsilon$ , 9)	(*FT', 7)	(/FT', 8)	eroare	( $\epsilon$ , 9)	( $\epsilon$ , 9)
F	(a, 11)	eroare	eroare	eroare	eroare	((E), 10)	eroare	eroare

# Factorizare la stânga

- ▶ Factorizarea este o transformare aplicată asupra unei gramatici pentru a obține o gramatică echivalentă, eventual LL(1)
- ▶ Dacă există producțiile
  - $A \rightarrow \alpha\beta_1, A \rightarrow \alpha\beta_2$  cu  $|\alpha| > 1$
- ▶ (deci gramatica nu este LL(1)), acestea se înlocuiesc cu  $A \rightarrow \alpha A'$ ,  $A'$  un nou neterminal, și producțiile  $A' \rightarrow \beta_1$  și  $A' \rightarrow \beta_2$ . Metoda de transformare se numește **factorizare la stânga**

# Factorizare la stânga: Exemplu

- ▶ Fie gramatica
  - $S \rightarrow \text{if } E \text{ then } S \mid \text{if } E \text{ then } S \text{ else } S \mid a$
  - $E \rightarrow b$
- ▶ factorizarea la stânga:  $\alpha = \text{if } E \text{ then } S$
- ▶ Gramatica echivalentă va fi
  - $S \rightarrow \text{if } E \text{ then } S S' \mid a$
  - $S' \rightarrow \text{else } S \mid \varepsilon$
  - $E \rightarrow b$

# Factorizare la stânga: Exemplu

M	a	b	if	then	else	#
S	(a, 2)	<i>eroare</i>	(if E then S S', 1)	<i>eroare</i>	<i>eroare</i>	<i>eroare</i>
S'	<i>eroare</i>	<i>eroare</i>	<i>eroare</i>	<i>eroare</i>	(else S,3)	(ε, 4)
					(ε, 4)	
E	<i>eroare</i>	(b, 5)	<i>eroare</i>	<i>eroare</i>	<i>eroare</i>	<i>eroare</i>

- ▶ Prin factorizare nu este sigur că obținem o gramatică LL(1).
- ▶ Putem rezolva ambiguitatea alegând regula **S' → else S** în **M(S', else)**. Această alegere ar corespunde asocierii lui else pentru acel *if* precedent cel mai apropiat de el, soluție adoptată de majoritatea limbajelor de programare.

# Corectitudinea parserului LL(1)

- ▶ Este dovedită pe baza
  - Teoremei de corectitudine a parserului descendent general,
  - Teoremei de caracterizare a gramaticilor LL(1)
  - Modulului în care a fost construită tabela de parsare