

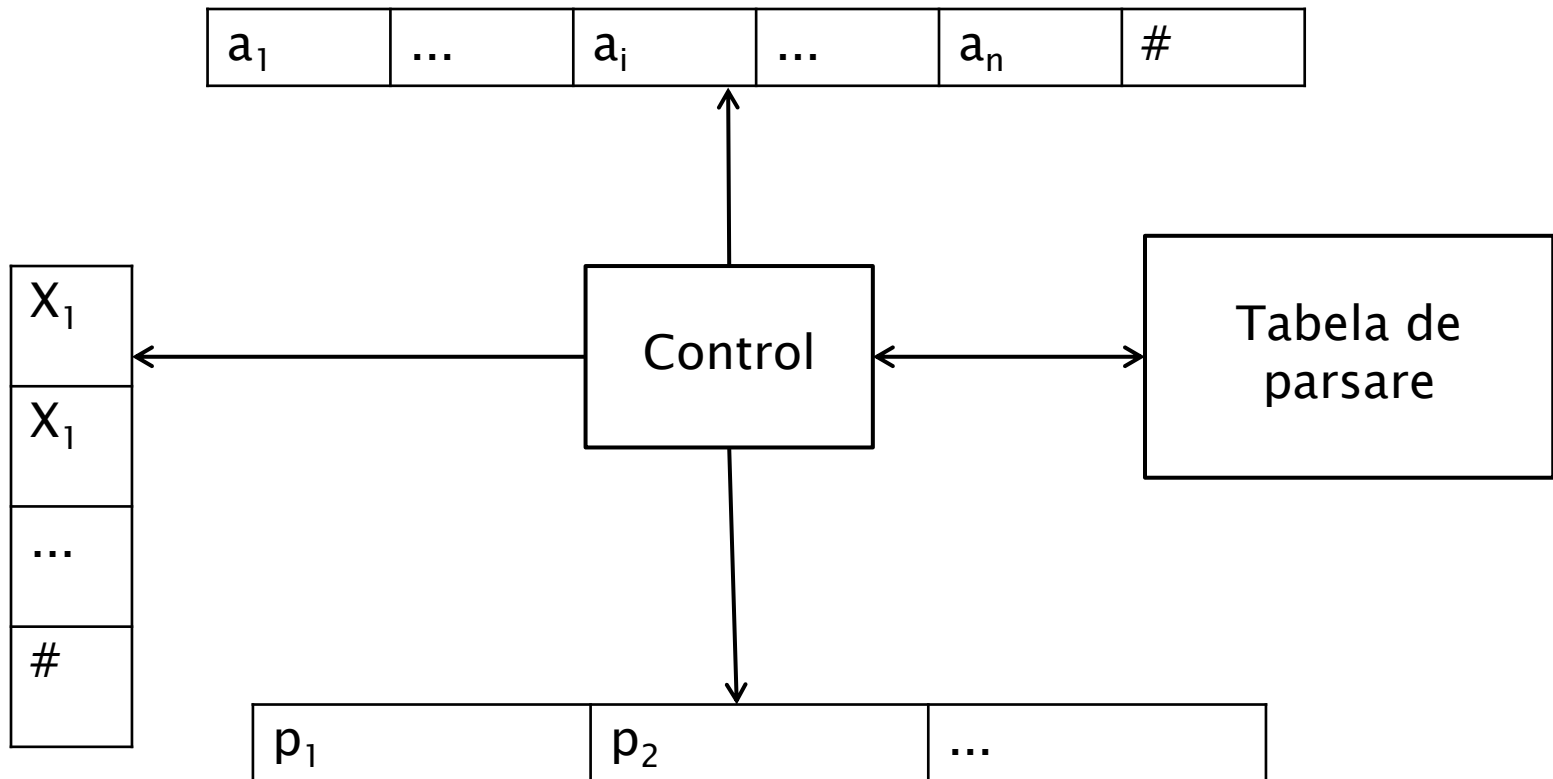
Limbaje formale, automate și compilatoare

Curs 8

Recapitulare

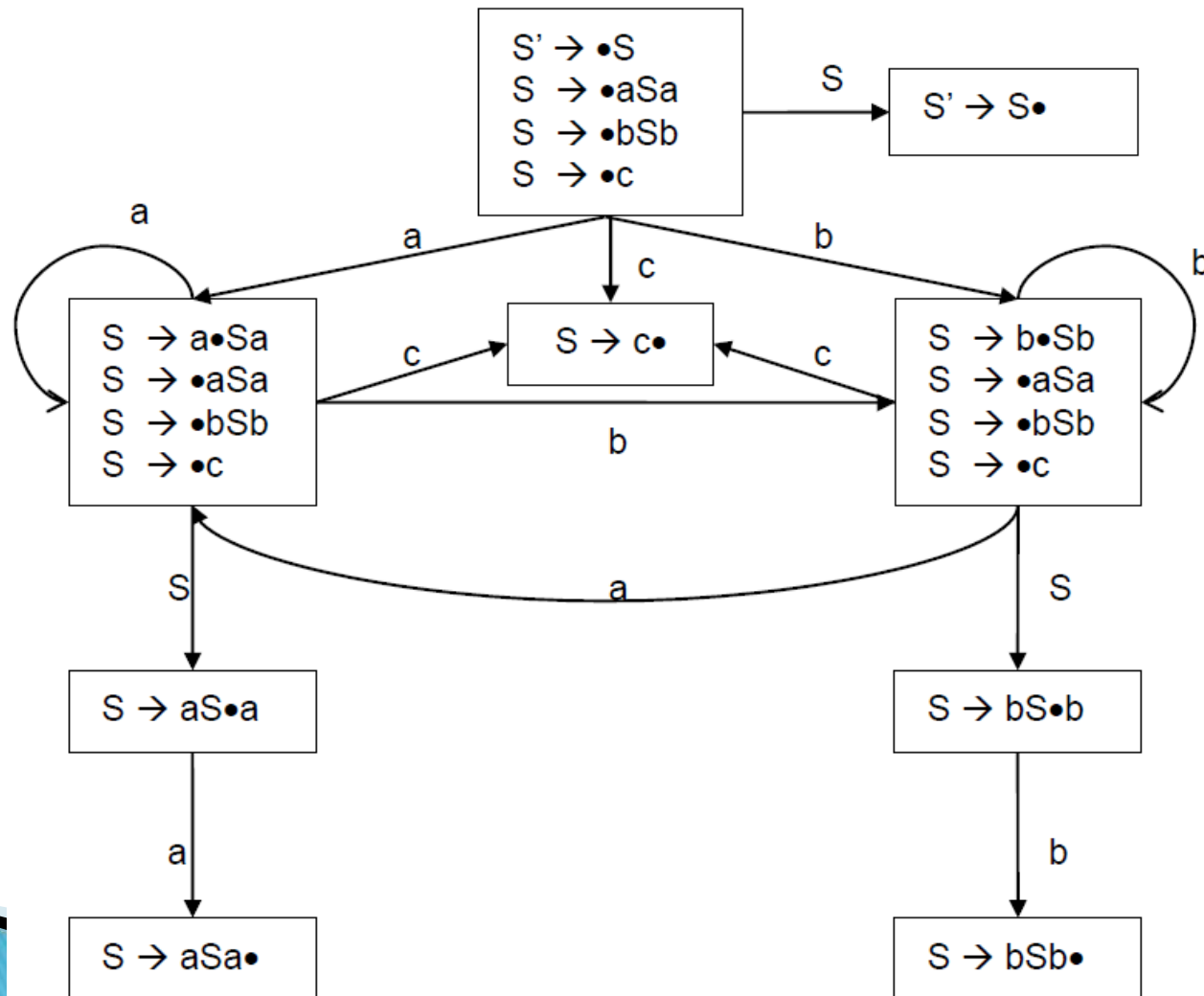
- ▶ Analiza sintactică ascendentă
 - Parser ascendent general
- ▶ Gramatici LR(k)
 - Definiție
 - Proprietăți
- ▶ Gramatici LR(0)
 - Teorema de caracterizare LR(0)
 - Automatul LR(0)
 - Parserul LR(0)

Parser ascendente general



Automatul LR(0) – Exemplu

- ▶ $S' \rightarrow S, S \rightarrow aSa \mid bSb \mid c$



Algoritmul de analiză LR(0)

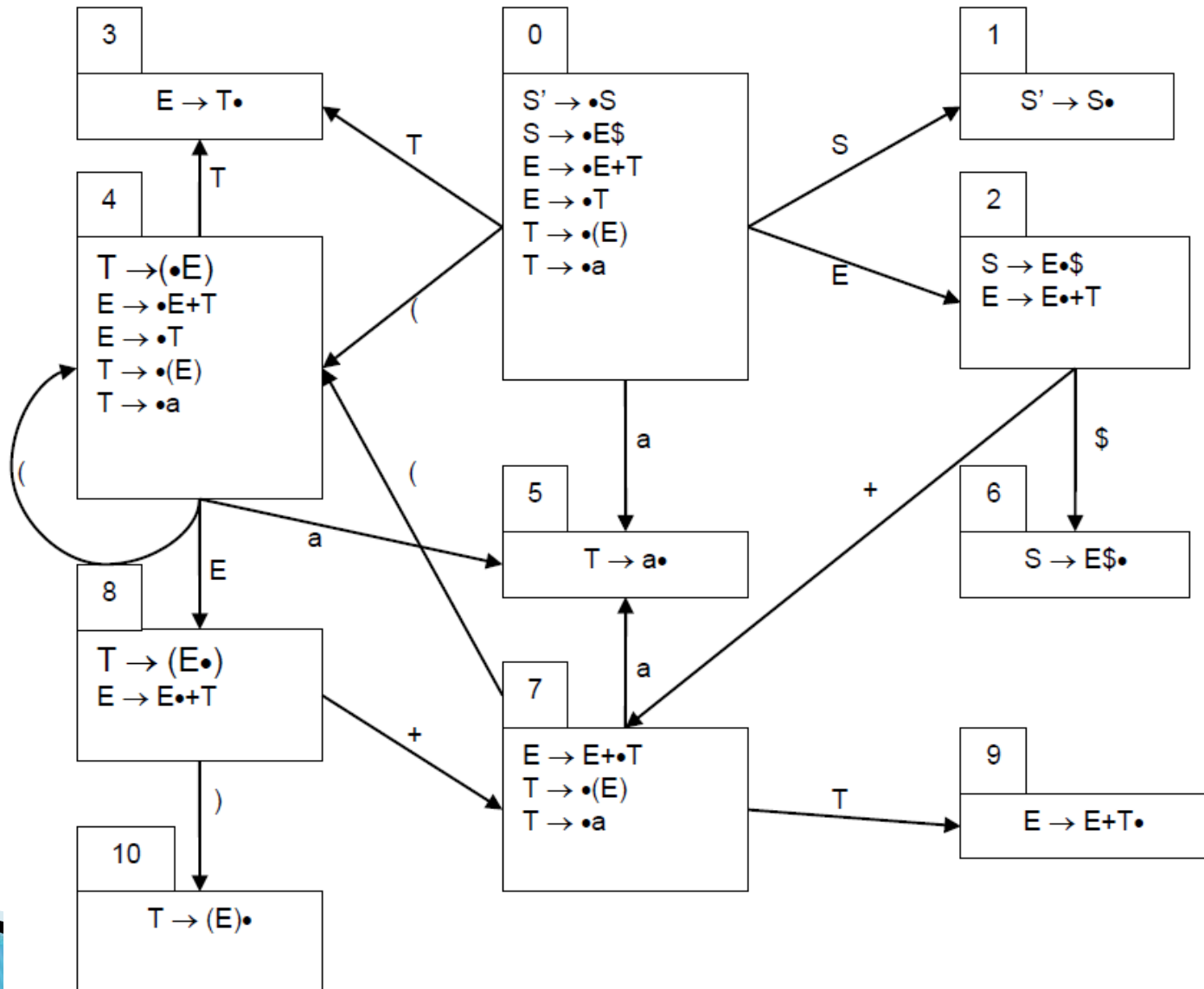
- ▶ Tabela de parsare coincide cu automatul LR(0), M.
- ▶ Configurație: $(\sigma, u\#, \pi)$ unde $\sigma \in t_0 T^*$, $u \in T^*$, $\pi \in P^*$.
- ▶ Configurația inițială este $(t_0, w\#, \varepsilon)$,
- ▶ Tranzițiile:
 - **Deplasare**: $(\sigma t, au\#, \pi) \vdash (\sigma t t', u\#, \pi)$ dacă $g(t, a) = t'$.
 - **Reducere**: $(\sigma t \sigma' t', u\#, \pi) \vdash (\sigma t t'', u\#, \pi r)$ dacă $A \rightarrow \beta \bullet \in t'$, $r = A \rightarrow \beta$, $|\sigma' t'| = |\beta|$ și $t'' = g(t, A)$.
 - **Acceptare**: $(t_0 t_1, \#, \pi)$ este configurația de acceptare dacă $S' \rightarrow S \bullet \in t_1$, π este parsarea acestuia.
 - **Eroare**: o configurație căreia nu i se poate aplica nici o tranziție

Algoritmul de analiză LR(0)

```
▶ char ps[] = "w#"; //ps este sirul de intrare w
▶ i = 0; // pozitia in sirul de intrare
▶ STIVA.push(t0); // se initializeaza stiva cu t0
▶ while(true) { // se repeta pana la succes sau eroare
    ◦ t = STIVA.top();
    ◦ a = ps[i] // a este simbolul curent din intrare
    ◦ if( g(t, a) ≠ ∅ { //deplasare
        • STIVA.push(g(t, a));
        • i++; //se inainteaza in intrare
        • }
    ◦ else {
    ◦ if(A → X1X2...Xm • ∈ t){
        • if(A == „S”)
            • if(a == „#”)exit( "acceptare");
            • else exit("eroare");
        • else // reducere
            • for( i = 1; i <= m; i++) STIVA.pop();
            • STIVA.push(g(top(STIVA), A));
        • } //endif
    ◦ else exit("eroare");
    ◦ }//endelse
▶ }//endwhile
```

Exemplu

► $S' \rightarrow S \quad S \rightarrow E\$ \quad E \rightarrow E+T \quad T \rightarrow (E) \quad E \rightarrow TT \rightarrow a$



Cuprins

- ▶ Mulțimile FIRST, FOLLOW
- ▶ Gramatici SLR(1)
 - Tabela de parsare SLR(1)
 - Analiza sintactică SLR(1)
- ▶ Gramatici LR(1)

Gramatici SLR(1)

► Definiție

- Fie G o gramatică pentru care automatul $LR(0)$ conține stări inconsistente (deci G nu este $LR(0)$). Gramatica G este gramatică $SLR(1)$ dacă oricare ar fi starea t a automatului $LR(0)$ sunt îndeplinite condițiile:
 - –Dacă $A \rightarrow \alpha \bullet$, $B \rightarrow \beta \bullet \in t$ atunci $FOLLOW(A) \cap FOLLOW(B) = \emptyset$;
 - –Dacă $A \rightarrow \alpha \bullet$, $B \rightarrow \beta \bullet a \gamma \in t$ atunci $a \notin FOLLOW(A)$.

Muhtimile FIRST ŝi FOLLOW

- ▶ $\text{FIRST}(\alpha) = \{a \mid a \in T, \alpha_{st} \Rightarrow^* au\} \cup$
if $(\alpha_{st} \Rightarrow^* \epsilon)$ then $\{\epsilon\}$ else \emptyset .
- ▶ $\text{FOLLOW}(A) = \{a \mid a \in T \cup \{\epsilon\}, S_{st} \Rightarrow^* uA\gamma,$
 $a \in \text{FIRST}(\gamma)\}$

Determinare FIRST

- ▶ 1. **for** ($X \in \Sigma$)
 - 2. **if** ($X \in T$) $\text{FIRST}(X) = \{X\}$ else $\text{FIRST}(X) = \emptyset$;
- ▶ 3. **for** ($A \rightarrow a\beta \in P$)
 - 4. $\text{FIRST}(A) = \text{FIRST}(A) \cup \{a\}$;
- ▶ 5. $\text{FLAG} = \text{true}$;
- ▶ 6. **while** (FLAG) {
 - 7. $\text{FLAG} = \text{false}$;
 - 8. **for** ($A \rightarrow X_1 X_2 \dots X_n \in P$) {
 - 9. $i = 1$;
 - 10. **if** ($(\text{FIRST}(X_1) \not\subseteq \text{FIRST}(A))$) {
 - 11. $\text{FIRST}(A) = \text{FIRST}(A) \cup (\text{FIRST}(X_1))$;
 - 12. $\text{FLAG} = \text{true}$;
 - 13. } //endif
 - 14. **while** ($i < n \ \&\& \ X_{i \text{ st}} \Rightarrow^* \epsilon$)
 - 15. **if** ($(\text{FIRST}(X_{i+1}) \not\subseteq \text{FIRST}(A))$) {
 - 16. $\text{FIRST}(A) = \text{FIRST}(A) \cup \text{FIRST}(X_{i+1})$;
 - 17. $\text{FLAG} = \text{true}; i++$;
 - } //endif
 - } //endwhile
 - } //endfor
- ▶ } //endwhile
- ▶ **for** ($A \in N$)
 - **if** ($A_{\text{st}} \Rightarrow^* \epsilon$) $\text{FIRST}(A) = \text{FIRST}(A) \cup \{\epsilon\}$;

Determinare FIRST

- ▶ **Intrare:** Gramatica $G=(N, T, S, P)$.
 - ▶ Multimiile $FIRST(X), X \in \Sigma$.
 - ▶ $\alpha = X_1X_2...X_n, X_i \in \Sigma, 1 \leq i \leq n$.
 - ▶ **Ieșire:** $FIRST(\alpha)$.
-
- ▶ 1. $FIRST(\alpha) = FIRST(X_1) - \{\epsilon\}; i=1;$
 - ▶ 2. **while** ($i < n \ \&\& \ X_i \Rightarrow^+ \epsilon$) {
 - 3. $FIRST(\alpha) = FIRST(\alpha) \cup (FIRST(X_{i+1}) - \{\epsilon\});$
 - 4. $i=i+1;$}
 - ▶ }//endwhile
 - ▶ 5. **if** ($i==n \ \&\& \ X_n \Rightarrow^+ \epsilon$)
 - 6. $FIRST(\alpha) = FIRST(\alpha) \cup \{\epsilon\};$

Exemplu

- ▶ Fie gramatica:
- ▶ $S \rightarrow E \mid B$, $E \rightarrow \varepsilon$, $B \rightarrow a \mid \text{begin SC end}$,
 $C \rightarrow \varepsilon \mid ; SC$
- ▶ $\text{FIRST}(S) = \{a, \text{begin}, \varepsilon\}$ $\text{FIRST}(E) = \{\varepsilon\}$
- ▶ $\text{FIRST}(B) = \{a, \text{begin}\}$ $\text{FIRST}(C) = \{;, \varepsilon\}$.
- ▶ $\text{FIRST}(\text{SEC}) = \{a, \text{begin}, ;, \varepsilon\}$,
- ▶ $\text{FIRST}(\text{SB}) = \{a, \text{begin}\}$,
- ▶ $\text{FIRST} (;SC) = \{;\}$.

Determinarea FOLLOW

- ▶ $\varepsilon \in \text{FOLLOW}(S)$.
- ▶ Dacă $A \rightarrow \alpha B \beta X \gamma \in P$ și $\beta \Rightarrow^+ \varepsilon$, atunci $\text{FIRST}(X) - \{\varepsilon\} \subseteq \text{FOLLOW}(B)$.
 - $S \Rightarrow^* \alpha_1 A \beta_1 \Rightarrow \alpha_1 \alpha B \beta X \gamma \beta_1 \Rightarrow^* \alpha_1 \alpha B X \gamma \beta_1$ și atunci rezultă $\text{FIRST}(X) - \{\varepsilon\} \subseteq \text{FOLLOW}(B)$.
- ▶ Dacă $A \rightarrow \alpha B \beta \in P$ atunci $\text{FIRST}(\beta) - \{\varepsilon\} \subseteq \text{FOLLOW}(B)$.
- ▶ Dacă $A \rightarrow \alpha B \beta \in P$ și $\beta \Rightarrow^+ \varepsilon$, atunci $\text{FOLLOW}(A) \subseteq \text{FOLLOW}(B)$.

Determinarea FOLLOW

- ▶ 1. **for** ($A \in \Sigma$) FOLLOW (A) = \emptyset ;
- ▶ 2. FOLLOW (S) = $\{\epsilon\}$;
- ▶ 3. **for** ($A \rightarrow X_1X_2...X_n$) {
- ▶ 4. $i=1$;
- 5. **while** ($i < n$) {
- 6. **while** ($X_i \notin N$) ++ i ;
- 7. **if** ($i < n$) {
- 8. FOLLOW (X_i) = FOLLOW (X_i) \cup
(FIRST ($X_{i+1}X_{i+2}...X_n$) - $\{\epsilon\}$) ;
- 9. ++ i ;
- }//endif
- }//endwhile
- ▶ }//endfor

Determinarea FOLLOW

```
▶ 10.FLAG=true;
▶ 11.while (FLAG) {
  ◦ 12.FLAG=false;
  ◦ 13.for (A → X1X2...Xn) {
    • 14.i=n;
    • 15.while (i>0 && Xi ∈ N) {
      • 16.if (FOLLOW(A) ⊄ FOLLOW(Xi)) {
        • 17.FOLLOW(Xi)=FOLLOW(Xi) ∪ FOLLOW(A);
        • 18.FLAG=true;
      • 19.}//endif
      • 20.if (Xi ⇒+ ε) --i;
      • 21.else continue;
    • 22.}//endwhile
  ◦ 23.}//endfor
▶ 24.}//endwhile
```


Exemplu

- ▶ Fie gramatica:
- ▶ $S \rightarrow E \mid B, E \rightarrow \varepsilon, B \rightarrow a \mid \text{begin } SC \text{ end},$
 $C \rightarrow \varepsilon \mid ; SC$
- ▶ $\text{FOLLOW}(S) = \text{FOLLOW}(E) = \text{FOLLOW}(B) = \{\varepsilon, ;, \text{end}\}$
- ▶ $\text{FOLLOW}(C) = \{\text{end}\}.$

Gramatici SLR(1)

► Definiție

- Fie G o gramatică pentru care automatul $LR(0)$ conține stări inconsistente (deci G nu este $LR(0)$). Gramatica G este gramatică $SLR(1)$ dacă oricare ar fi starea t a automatului $LR(0)$ sunt îndeplinite condițiile:
 - -Dacă $A \rightarrow \alpha \bullet$, $B \rightarrow \beta \bullet \in t$ atunci $FOLLOW(A) \cap FOLLOW(B) = \emptyset$;
 - -Dacă $A \rightarrow \alpha \bullet$, $B \rightarrow \beta \bullet a \gamma \in t$ atunci $a \notin FOLLOW(A)$.
- Analiza sintactică $SLR(1)$ este similară cu cea $LR(0)$; tabela de analiză sintactică are două componente:
 - -Prima, numită ACȚIUNE, determină dacă parserul va face deplasare respectiv reducere, în funcție de starea ce se află în topul stivei și de simbolul următor din intrare
 - -Cea de a doua, numită GOTO, determină starea ce se va adăuga în stivă în urma unei reduceri.

Construcția tablei de parsare SLR(1)

▶ Intrare:

- Gramatica $G = (N, T, S, P)$ augmentată cu $S' \rightarrow S$;
- Automatul $M = (Q, \Sigma, g, t_0, Q)$;
- Mulțimile $FOLLOW(A)$, $A \in V$

▶ Ieșire:

- Tabela de analiză SLR(1) compusă din două părți:
- $ACȚIUNE(t, a)$, $t \in Q$, $a \in T \cup \{ \# \}$,
- $GOTO(t, A)$, $t \in Q$, $A \in N$.

Construcția tabelii de parsare SLR(1)

- ▶ `for` ($t \in Q$)
 - `for` ($a \in T$) `ACTIUNE` (t, a) = "eroare";
 - `for` ($A \in V$) `GOTO` (t, A) = "eroare";
- ▶ `for` ($t \in Q$) {
 - `for` ($A \rightarrow \alpha \bullet a \beta \in t$)
 - `ACTIUNE` (t, a) = "D $g(t, a)$ "; //deplasare in $g(t, a)$
 - `for` ($B \rightarrow \gamma \bullet \in t$) { // acceptare sau reducere
 - `if` ($B == 'S'$) `ACTIUNE` (t, a) = "acceptare";
 - `else`
 - `for` ($a \in \text{FOLLOW}(B)$) `ACTIUNE` (t, a) = "R $B \rightarrow \gamma$ ";
 - } // endfor
 - `for` ($A \in N$) `GOTO` (t, A) = $g(t, A)$;
- ▶ } // endfor

Parsarea SLR(1)

- ▶ **Deplasare:** $(\sigma t, au\#, \pi) \vdash (\sigma tt', u\#, \pi)$ dacă $ACTIUNE(t, a) = Dt'$;
- ▶ **Reducere:** $(\sigma t \sigma' t', u\#, \pi) \vdash (\sigma tt'', u\#, \pi)$ $ACTIUNE(t, a) = Rp$ unde $p = A \rightarrow \beta$, $|\sigma' t'| = |\beta|$ și $t'' = GOTO(t, A)$;
- ▶ **Acceptare:** $(t_0 t, \#, \pi)$ dacă $ACTIUNE(t, a) = \text{“acceptare”}$;
Analizorul se oprește cu acceptarea cuvântului de analizat iar π este parsarea acestuia (șirul de reguli care s-a aplicat, în ordine inversă, în derivarea extrem dreaptă a lui w).
- ▶ **Eroare:** $(\sigma t, au\#, \pi) \vdash \text{eroare}$ dacă $ACTIUNE(t, a) = \text{“eroare”}$;
Analizorul se oprește cu respingerea cuvântului de analizat.

Parsarea SLR(1)

▶ Intrare:

- Gramatica $G = (N, T, S, P)$ care este SLR(1) ;
- Tabela de parsare SLR(1) (ACTIUNE, GOTO);
- Cuvântul de intrare $w \in T^*$.

▶ Ieşire:

- Analiza sintactică (parsarea) ascendentă a lui w dacă $w \in L(G)$;
 - eroare, în caz contrar.
- ▶ Se foloseşte stiva St pentru a implementa tranziţiile deplasare/reducere

Parsarea SLR(1)

- ▶ `char ps[] = "w#";` //ps este cuvantul de intrare w
- ▶ `int i = 0;` // pozitia curenta in cuvantul de intrare
- ▶ `St.push(t0);` // se initializeaza stiva cu t0
- ▶ `while(true) {` // se repeta pana la succes sau eroare
 - `t = St.top();`
 - `a = ps[i]` // a este simbolul curent din intrare
 - `if(ACTIUNE(t,a) == "acceptare") exit("acceptare");`
 - `if(ACTIUNE(t,a) == "Dt") {`
 - `St.push(t);`
 - `i++;` // se inainteaza in w
 - `} //endif`
 - `else {`
 - `if(ACTIUNE(t,a) == "R A → X1X2...Xm") {`
 - `for(i = 1; i ≤ m; i++) St.pop();`
 - `St.push(GOTO(St.top, A));`
 - `} //endif`
 - `else exit("eroare");`
 - `} //endelse`
- ▶ `} //endwhile`

Exemplu

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

0

$S \rightarrow \bullet E$
 $E \rightarrow \bullet E+T$
 $E \rightarrow \bullet T$
 $T \rightarrow \bullet T^*F$
 $T \rightarrow \bullet F$
 $F \rightarrow \bullet (E)$
 $F \rightarrow \bullet a$

1

$S \rightarrow E \bullet$
 $E \rightarrow E \bullet +T$

2

$E \rightarrow T \bullet$
 $T \rightarrow T \bullet ^*F$

3

$T \rightarrow F \bullet$

5

$F \rightarrow a \bullet$

8

$F \rightarrow (E \bullet)$
 $E \rightarrow E \bullet +T$

9

$E \rightarrow E+T \bullet$
 $T \rightarrow T \bullet ^*F$

4

$F \rightarrow (\bullet E)$
 $E \rightarrow \bullet E+T$
 $E \rightarrow \bullet T$
 $T \rightarrow \bullet T^*F$
 $T \rightarrow \bullet F$
 $F \rightarrow \bullet (E)$
 $F \rightarrow \bullet a$

10

$T \rightarrow T^*F \bullet$

11

$F \rightarrow (E) \bullet$

6

$E \rightarrow E+ \bullet T$
 $T \rightarrow \bullet T^*F$
 $T \rightarrow \bullet F$
 $F \rightarrow \bullet (E)$
 $F \rightarrow \bullet a$

7

$T \rightarrow T^* \bullet F$
 $F \rightarrow \bullet (E)$
 $F \rightarrow \bullet a$

Tabela de tranziție a automatului LR(0)

g	a	+	*	()	E	T	F
0	5			4		1	2	3
1		6						
2			7					
3								
4	5			4		8	2	3
5								
6	5			4			9	3
7	5			4				10
8					11			
9			7					
10								
11								

Tabela de analiză SLR(1)

STARE	ACȚIUNE						GOTO		
	a	+	*	()	#	E	T	F
0	D5			D4			1	2	3
1		D6				accepta			
2		R2	D7		R2	R2			
3		R4	R4		R4	R4			
4	D5			D4			8	2	3
5		R6	R6		R6	R6			
6	D5			D4				9	3
7	D5			D4					10
8		D6			D11				
9		R1	D7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

Test SLR(1)

- ▶ G nu este LR(0) stările 1, 2, 9 conțin conflict de deplasare/reducere
- ▶ $\text{FOLLOW}(S) = \{\#\}$, $\text{FOLLOW}(E) = \{\#, +,)\}$
- ▶ Gramatica este SLR(1) pentru că:
 - în starea 1: $+$ $\notin \text{FOLLOW}(S)$;
 - în starea 2: $*$ $\notin \text{FOLLOW}(E)$;
 - în starea 9: $*$ $\notin \text{FOLLOW}(E)$.

Stiva	Intrare	Actiune	Iesire
0	$a*(a+a)\#$	deplasare	
05	$*(a+a)\#$	reducere	6.F \rightarrow a
03	$*(a+a)\#$	reducere	4.T \rightarrow F
02	$*(a+a)\#$	deplasare	
027	$(a+a)\#$	deplasare	
0274	$a+a)\#$	deplasare	
02745	$+a)\#$	reducere	6.F \rightarrow a
02743	$+a)\#$	reducere	4.T \rightarrow F
02742	$+a)\#$	reducere	2.E \rightarrow T
02748	$+a)\#$	deplasare	

Stiva	Intrare	Actiune	Iesire
027486	a)#	deplasare	
0274865)#	reducere	6.F \rightarrow a
0274863)#	reducere	4.T \rightarrow F
0274869)#	reducere	1.E \rightarrow E+T
02748)#	deplasare	
02748(11)	#	reducere	5.F \rightarrow (E)
027(10)	#	reducere	3.T \rightarrow T*F
02	#	reducere	2.E \rightarrow T
01	#	acceptare	

Gramatici LR(1)

► Definiție

- Fie $G = (V, T, S, P)$ o gramatică redusă. Un articol LR(1) pentru gramatica G este o pereche $(A \rightarrow \alpha \bullet \beta, a)$, unde $A \rightarrow \alpha \beta$ este un articol LR(0), iar $a \in \text{FOLLOW}(A)$ (se pune $\#$ în loc de ϵ).

► Definiție

- Articolul $(A \rightarrow \beta_1 \bullet \beta_2, a)$ este valid pentru prefixul viabil $\alpha \beta_1$ dacă are loc derivarea
 - $S \xRightarrow{dr} \alpha A u \Rightarrow \alpha \beta_1 \beta_2 u$
 - iar $a = 1 : u$ ($a = \#$ dacă $u = \epsilon$).

► Teorema

- O gramatică $G = (V, T, S, P)$ este gramatică LR(1) dacă și numai dacă oricare ar fi prefixul viabil φ , nu există două articole distincte, valide pentru φ , de forma $(A \rightarrow \alpha \bullet, a)$, $(B \rightarrow \beta \bullet \gamma, b)$ unde $a \in \text{FIRST}(\gamma b)$.

Gramatici LR(1)

- ▶ Nu există conflict deplasare/reducere. Un astfel de conflict înseamnă două articole $(A \rightarrow \alpha \bullet, a)$ și $(B \rightarrow \beta \bullet a \beta', b)$ valide pentru același prefix.
- ▶ Nu există conflict reducere/reducere. Un astfel de conflict înseamnă două articole complete $(A \rightarrow \alpha \bullet, a)$ și $(B \rightarrow \beta \bullet, a)$ valide pentru același prefix
- ▶ Pentru a verifica dacă o gramatică este LR(1) se construiește automatul LR(1) în mod asemănător ca la LR(0):
 - Automatul are ca stări mulțimi de articole LR(1)
 - Tranzițiile se fac cu simboluri ce apar după punct
 - Închiderea unei mulțimi de articole se bazează pe faptul că dacă articolul $(B \rightarrow \beta \bullet A \beta', b)$ este valid pentru un prefix viabil atunci toate articolele de forma $(A \rightarrow \bullet \alpha, a)$, unde $a \in \text{FIRST}(\beta' b)$ sunt valide pentru același prefix.

Procedura de închidere LR(1)

```
▶ flag= true;
▶ while( flag) {
  ◦ flag= false;
  ◦ for ( (A→ α•Bβ, a) ∈ I) {
    • for B → γ ∈ P)
      • for( b ∈ FIRST(βa)) {
        • if( (B → •γ , b) ∉ I) {
          ▪ I = I ∪ { (B → •γ , b) };
          ▪ flag= true;
        • }//endif
      • }//endforb
    • }//endforB
  ◦ }//endforA
▶ }//endwhile
▶ return I;
```


Automatul LR(1)

```
▶ t0 = închidere((S' → •S, #)); T={t0}; marcat(t0)=false;
▶ while(∃ t∈T && !marcat(t)) { // marcat(t) = false
  ◦ for( X ∈ Σ ) {
    ◦ t' = Φ;
      • for( (A → α•Xβ, a) ∈ t )
        • t' = t' ∪ { (B → αX•β, a) | (B → α•Xβ, a) ∈ t };
        • if( t' ≠ Φ ) {
          • t' = închidere( t' );
          • if( t' ∈ T ) {
            ▪ T = T ∪ { t' };
            ▪ marcat( t' ) = false;
          } //endif
          • g(t, X) = t';
        • } //endif
    ◦ } //endfor
    ◦ marcat( t ) = true;
  ▶ } //endwhile
```

Automatul LR(1)

► Teorema

- Automatul M construit în algoritmul 2 este determinist și $L(M)$ coincide cu mulțimea prefixelor viabile ale lui G . Mai mult, pentru orice prefix viabil γ , $g(t_0, \gamma)$ reprezintă mulțimea articolelor LR(1) valide pentru γ .
- Automatul LR(1) pentru o gramatică G , se folosește pentru a verifica dacă G este LR(1)
 - Conflict reducere/reducere: Dacă în T există o stare ce conține articole de forma $(A \rightarrow \alpha \bullet, a)$, $(B \rightarrow \beta \bullet, a)$ atunci gramatica nu este LR(1);
 - Conflict deplasare/reducere: Dacă în T există o stare ce conține articole de forma $(A \rightarrow \alpha \bullet, a)$ și $(B \rightarrow \beta_1 \bullet a \beta_2, b)$, atunci G nu este LR(1).
 - O gramatică este LR(1) dacă orice stare $t \in T$ este liberă de conflicte

Exemplu

► $S \rightarrow L=R \mid R, L \rightarrow *R \mid a, R \rightarrow L$

0
 $(S' \rightarrow \bullet S, \#)$
 $(S \rightarrow \bullet L=R, \#)$
 $(S \rightarrow \bullet R, \#)$
 $(L \rightarrow \bullet *R, \{=, \#\})$
 $(L \rightarrow \bullet a, \{=, \#\})$
 $(R \rightarrow \bullet L, \#)$

6
 $(S \rightarrow L=\bullet R, \#)$
 $(R \rightarrow \bullet L, \#)$
 $(L \rightarrow \bullet *R, \#)$
 $(L \rightarrow \bullet a, \#)$

1
 $(S' \rightarrow S\bullet, \#)$

2
 $(S \rightarrow L\bullet=R, \#)$
 $(R \rightarrow L\bullet, \#)$

7
 $(L \rightarrow *R\bullet, \{=, \#\})$

8
 $(R \rightarrow L\bullet, \{=, \#\})$

12
 $(L \rightarrow a\bullet, \#)$

3
 $(S \rightarrow R\bullet, \#)$

5
 $(L \rightarrow a\bullet, \{=, \#\})$

9
 $(S \rightarrow L=R\bullet, \#)$

10
 $(R \rightarrow L\bullet, \#)$

13
 $(L \rightarrow *R\bullet, \#)$

4
 $(L \rightarrow *\bullet R, \{=, \#\})$
 $(R \rightarrow \bullet L, \{=, \#\})$
 $(L \rightarrow \bullet *R, \{=, \#\})$
 $(L \rightarrow \bullet a, \{=, \#\})$

11
 $(L \rightarrow *\bullet R, \#)$
 $(R \rightarrow \bullet L, \#)$
 $(L \rightarrow \bullet *R, \#)$
 $(L \rightarrow \bullet a, \#)$

Tabela de tranziție

g	a	=	*	S	L	R
0	5		4	1	2	3
1						
2		6				
3						
4	5		4		8	7
5						
6	12		11		10	9
7						
8						
9						
10						
11	12		11		10	13
12						
13						

Construcția tablei de analiză LR(1)

- ▶ `for` ($t \in T$)
 - `for` ($a \in T$) `ACTIUNE` (t, a) = "eroare";
 - `for` ($A \in V$) `GOTO` (t, A) = "eroare";
- ▶ `for` ($t \in T$) {
 - `for` ($(A \rightarrow \alpha \bullet a \beta, L) \in t$)
 - `ACTIUNE` (t, a) = "D $g(t, a)$ "; //deplasare in $g(t, a)$
 - `for` ($(B \rightarrow \gamma \bullet, L) \in t$) { // acceptare sau reducere
 - `for` ($c \in L$) {
 - `if` ($B == 'S'$) `ACTIUNE` ($t, \#$) = "acceptare";
 - `else` `ACTIUNE` (t, c) = "R $B \rightarrow \gamma$ "; //reducere cu $B \rightarrow \gamma$
 - } //endfor
 - } // endfor
 - `for` ($A \in N$) `GOTO` (t, A) = $g(t, A)$;
- ▶ } //endfor

Exemplu

- 0: $S' \rightarrow S$, 1: $S \rightarrow L=R$, 2: $S \rightarrow R$, 3: $L \rightarrow *R$, 4: $L \rightarrow a$, 5: $R \rightarrow L$

STARE	ACȚIUNE				GOTO		
	a	=	*	#	S	L	R
0	D5		D4		1	2	3
1				acceptare			
2		D6		R5			
3				R2			
4	D5		D4			8	7
5		R4		R4			
6	D12		D11			10	9
7		R3		R3			
8		R5		R5			
9				R1			
10				R5			
11	D12		D11			10	13
12				R4			
13				R3			

Exemplu

- ▶ Fie cuvintele
 - $***a$
 - $a = **a$
 - $*a = **a$
- ▶ Analiza LR(1)?

Gramatici LALR(1)

► Definiție

- Fie t o stare în automatul LR(1) pentru G . Nucleul acestei stări este mulțimea articolelor LR(0) care apar ca prime componente în articolele LR(1) din t .

► Definiție

- Două stări t_1 și t_2 ale automatului LR(1) pentru gramatica G sunt echivalente dacă au același nucleu.

Gramatici LALR(1)

- ▶ Fiecare stare a automatului LR(1) este o mulțime de articole LR(1). Pornind de la două stări t_1 și t_2 putem vorbi de starea $t_1 \cup t_2$.
 - Fie $t_1 = \{(L \rightarrow *R., \{=, \# \})\}$, $t_2 = \{(L \rightarrow *R., \#)\}$, atunci $t_1 \cup t_2 = t_1$ pentru că $t_2 \subset t_1$.
- ▶ **Definiție**
 - Fie G gramatică LR(1) și $M = (Q, \Sigma, g, t_0, Q)$ automatul LR(1) corespunzător. Spunem că gramatica G este LALR(1) (Look Ahead LR(1)) dacă oricare ar fi perechea de stări echivalente t_1, t_2 din automatul LR(1), starea $t_1 \cup t_2$ este liberă de conflicte.

Tabela de analiză LALR(1)

- ▶ **Intrare:** Gramatica $G = (N, T, S, P)$ augmentată cu $S' \rightarrow S$;
- ▶ **Ieșire:** Tabela de analiză LALR(1) (ACȚIUNE și GOTO).
- ▶ **Algoritm:**
 - **1.** Se construiește automatul LR(1), $M = (Q, \Sigma, g, t_0, Q)$
Fie $Q = \{t_0, t_1, \dots, t_n\}$. Dacă toate stările din Q sunt libere de conflict, urmează 2, altfel algoritmul se oprește deoarece gramatica nu este LR(1).
 - **2.** Se determină stările echivalente din Q și, prin reuniunea acestora, se obține o nouă mulțime de stări $Q' = \{t'_0, t'_1, \dots, t'_m\}$
 - **3.** Dacă în Q' există stări ce conțin conflicte, algoritmul se oprește deoarece gramatica G nu este LALR(1).

Tabela de analiză LALR(1)

- **4.** Se construiește automatul $M' = (Q', \Sigma, g', t'_0, Q')$, unde $\forall t' \in Q'$:
 - **5.** Dacă $t' \in Q$ atunci $g'(t', X) = g(t, X)$, $\forall X \in \Sigma$;
 - **6.** Dacă $t' = t_1 \cup t_2 \cup \dots$, $t_1, t_2, \dots \in Q$, atunci
 - **7.** $g'(t', X) = g(t_1, X) \cup g(t_2, X) \cup \dots$
- **8.** Se aplică algoritmul pentru construirea tabelii de parsare LR(1) pornind de la automatul M' . Tabela obținută se numește tabela LALR(1) pentru gramatica G .

Exemplu

- Pentru gramatica discutată anterior avem $4 \cup 11 = 4$, $5 \cup 12 = 5$, $7 \cup 13 = 7$, $8 \cup 10 = 8$

STAR E	ACȚIUNE				GOTO		
	a	=	*	#	S	L	R
0	D5		D4		1	2	3
1				acceptare			
2		D6		R5			
3				R2			
4	D5		D4			8	7
5		R4		R4			
6	D5		D4			8	9
7		R3		R3			
8		R5		R5			
9				R1			

Exemplu

- ▶ Există gramatici LR(1) care nu sunt LALR(1).
 - $S \rightarrow aAb \mid bAd \mid aBd \mid bBb$
 - $A \rightarrow e$
 - $B \rightarrow e$

Bibliografie

- ▶ Grigoraș Gh., *Construcția compilatoarelor. Algoritmi fundamentali*, Editura Universității “Alexandru Ioan Cuza”, Iași, 2005