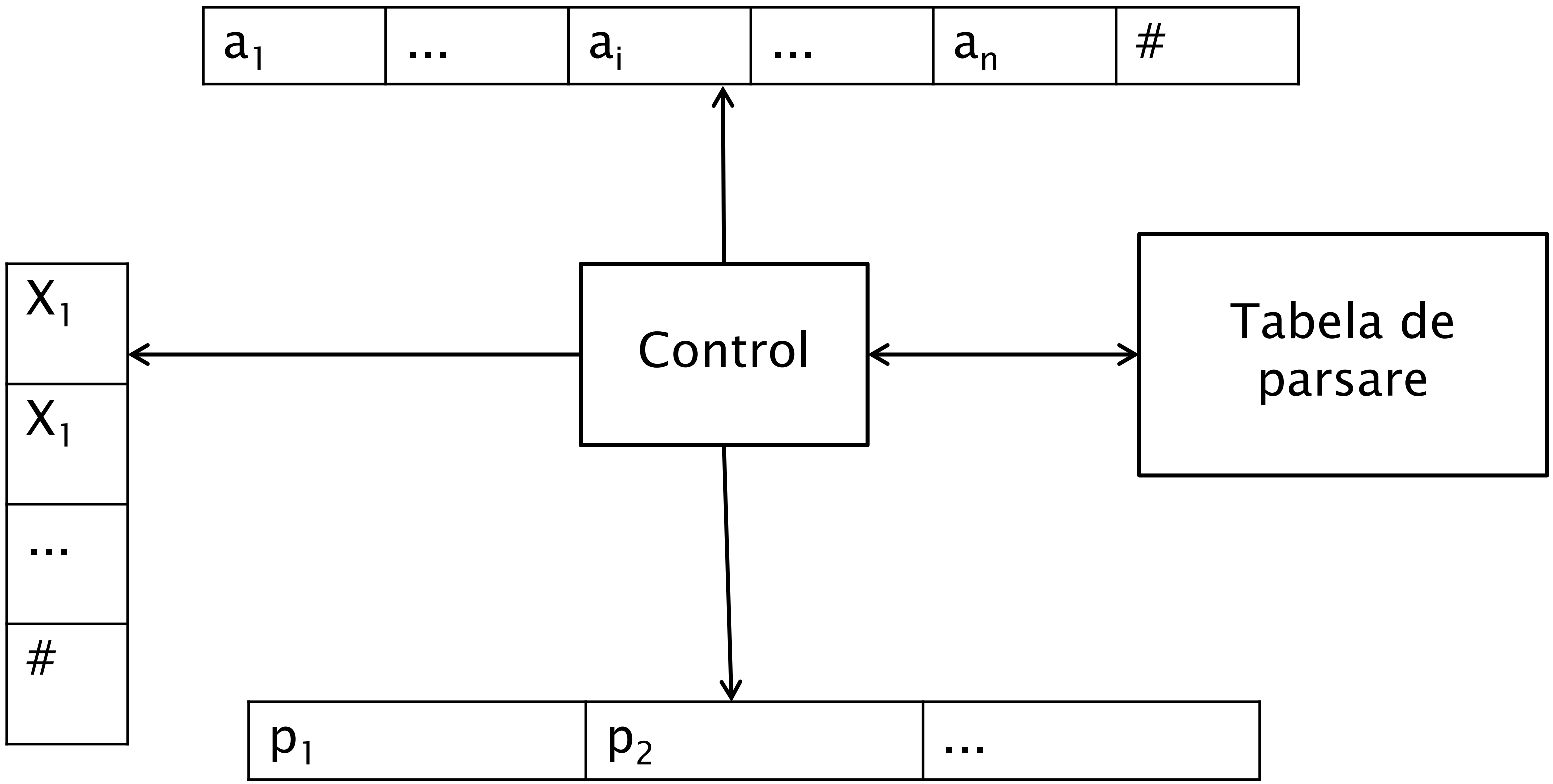
Limbaje formale, automate şi compilatoare

Curs 8

Recapitulare

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

Parser ascendent general



Gramatici LR(0)

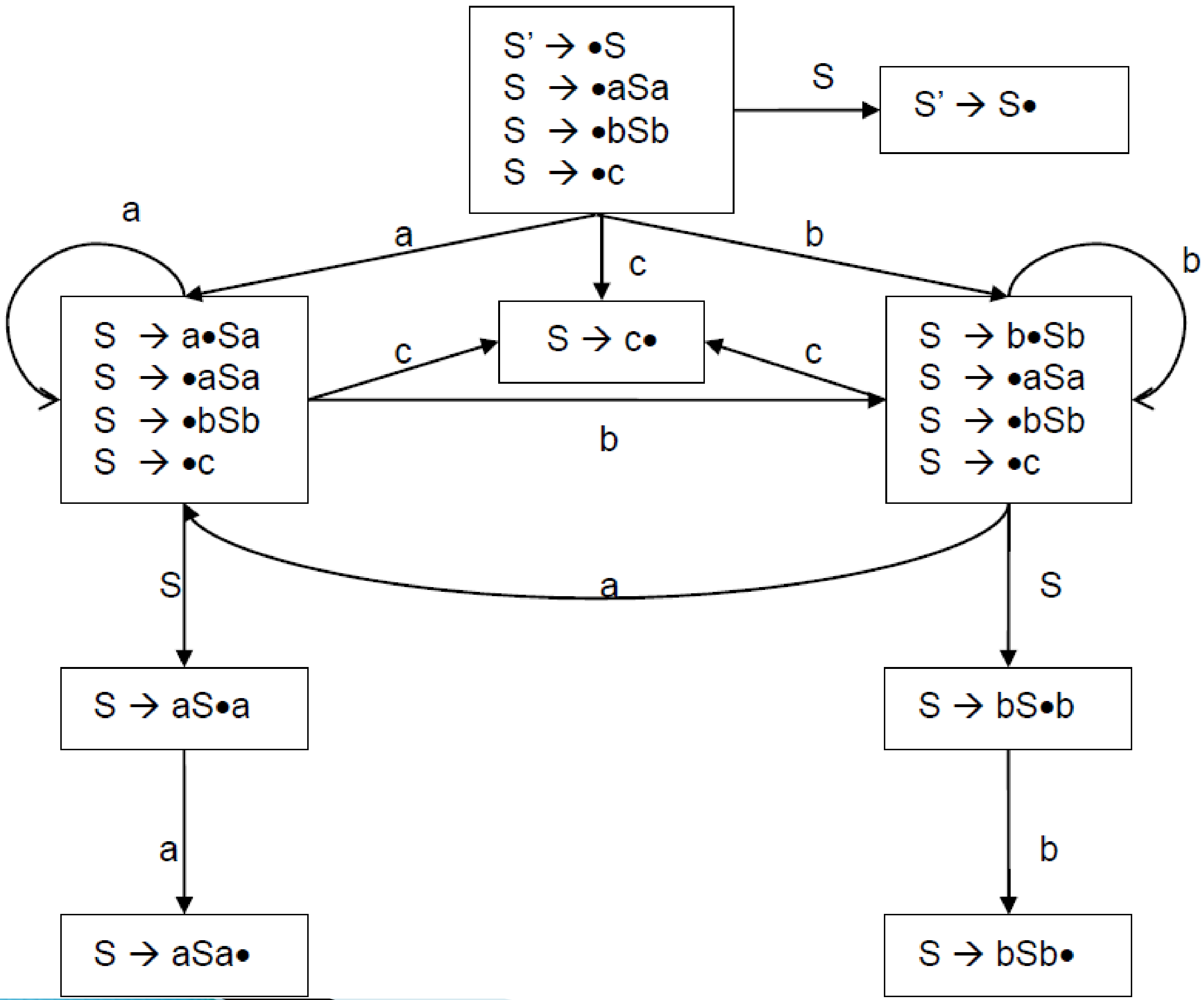
- **Definiţie**
  - Fie $G = (V, T, S, P)$ o gramatică independentă de context redusă. Să presupunem că simbolul • nu este în Σ. Un **articol** pentru gramatica G este o producţie $A \rightarrow \gamma$ în care s-a adăugat simbolul • într-o anume poziţie din γ. Notăm un articol prin $A \rightarrow \alpha \bullet \beta$ dacă $\gamma = \alpha\beta$. Un articol în care • este pe ultima poziţie se numeşte **articol complet**.
- **Definiţie**
  - Un **prefix viabil** pentru gramatica G este orice prefix al unui cuvânt $\alpha\beta$ dacă $S \ {}_{dr}\Rightarrow^* \alpha A u \ {}_{dr}\Rightarrow \alpha\beta u$. Dacă $\beta = \beta_1\beta_2$ şi $\varphi = \alpha\beta_1$ spunem că articolul $A \rightarrow \beta_1 \bullet \beta_2$ este **valid** pentru **prefixul viabil** φ.

Gramatici LR(0)

- **Lema**
  - Fie G o gramatică şi $A \to \beta_1 \bullet B\beta_2$ un articol valid pentru prefixul viabil γ. Atunci, oricare ar fi producţia $B \to \beta$, articolul $B \to \bullet\beta$ este valid pentru γ.
- **Teorema** (caracterizare LR(0))
  - Gramatica G este gramatică LR(0) dacă şi numai dacă, oricare ar fi prefixul viabil γ, sunt îndeplinite condiţiile:
    - 1.nu există două articole complete valide pentru γ.
    - 2.dacă articolul $A \to \beta\bullet$ este valid pentru γ, nu există nici un articol $B \to \beta_1 \bullet a\beta_2$, $a \in T$, valid pentru γ.

Automatul LR(0) – Exemplu

- S’ → S, S → aSa | bSb | c



Algoritmul de analiză LR(0)

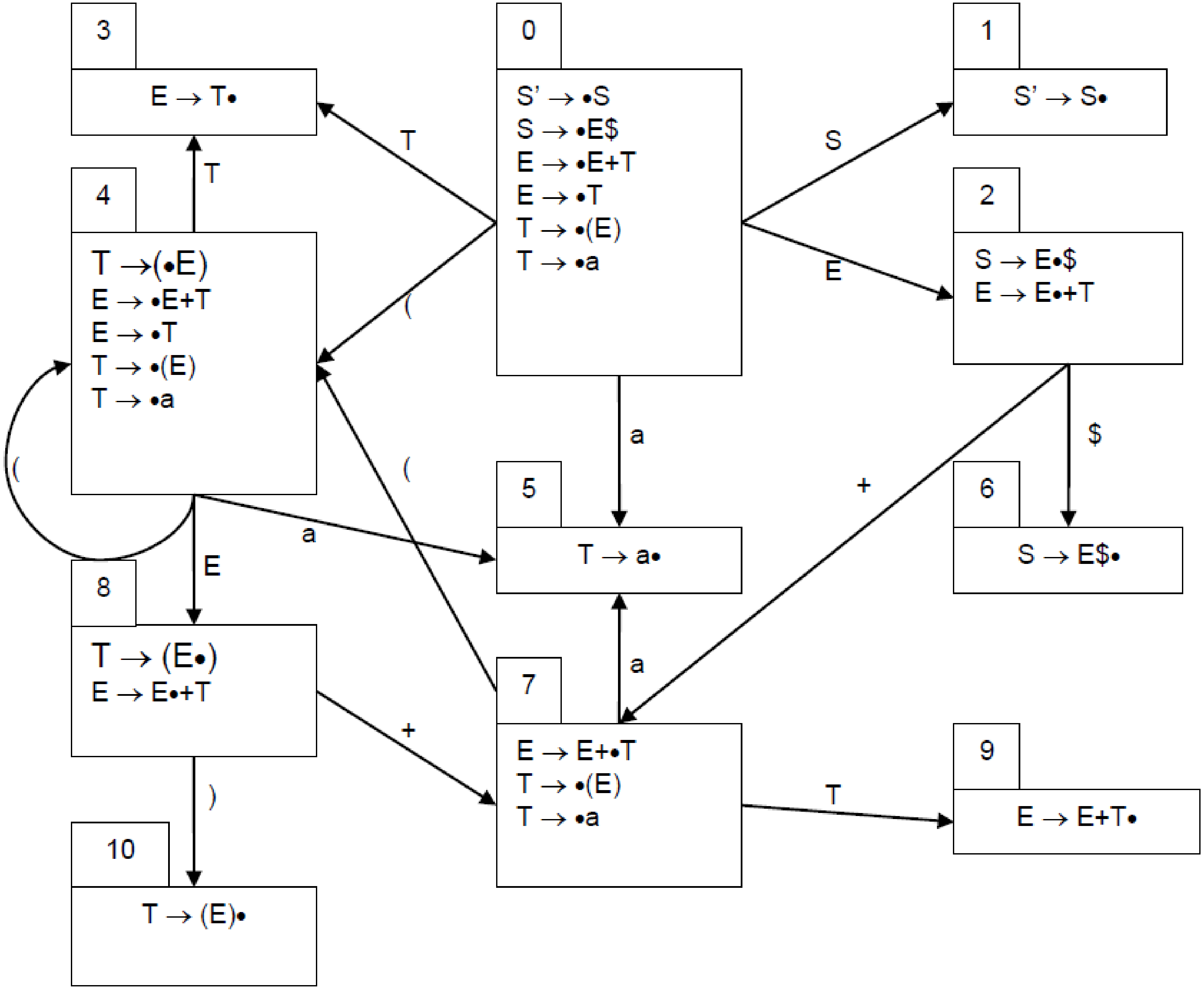
- Tabela de parsare coincide cu automatul LR(0), M.
- Configuraţie: $(\sigma, u\#, \pi)$ unde $\sigma \epsilon t_0 T^*$, $u \epsilon T^*$, $\pi \epsilon P^*$.
- Configuraţia iniţială este $(t_0, w\#, \varepsilon)$,
- Tranziţiile:
  - Deplasare: $(\sigma t, au\#, \pi) \vdash (\sigma t t', u\#, \pi)$ dacă $g(t, a) = t'$.
  - Reducere: $(\sigma t \sigma' t', u\#, \pi) \vdash (\sigma t t'', u\#, \pi r)$ dacă $A \rightarrow \beta \bullet \in t'$, $r = A \rightarrow \beta$, $|\sigma' t'| = |\beta|$ şi $t'' = g(t, A)$.
  - Acceptare: $(t_0 t_1, \#, \pi)$ este configuraţia de acceptare dacă $S' \rightarrow S \bullet \in t1$, π este parsarea acestuia.
  - Eroare: o configuraţie căreia nu i se poate aplica nici o tranziţie

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)≠Ø { //deplasare
    - STIVA.push(g(t, a));
    - i++; //se inainteaza in intrare
    - }
  - else {
  - if(A → $X_1X_2…X_m$ • ∈ 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’ → S S → E$ E → E+T T →(E) E → T T → a



Cuprins

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

Gramatici SLR(1)

- **Definiţie**
  - Fie G o gramatică pentru care automatul LR(0) conţine 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 condiţiile:

  - –Dacă $A\to\alpha\bullet$, $B \to \beta \bullet \in t$ atunci $FOLLOW(A) \cap FOLLOW(B) = \emptyset$;
  - –Dacă $A\to\alpha\bullet$, $B \to \beta \bullet a\gamma \in t$ atunci $a \notin FOLLOW(A)$.

Mulţimile FIRST şi FOLLOW

- FIRST(α) = $\{a | a \in T, \alpha \ {}_{st}\Rightarrow^* au \} \cup$ if ($\alpha \ {}_{st}\Rightarrow^* \varepsilon$) then $\{\varepsilon\}$ else $\emptyset$.

- FOLLOW(A) = $\{a | a \in T \cup \{\varepsilon\}, S \ {}_{st}\Rightarrow^* uA\gamma,$ $a \in$ FIRST $(\gamma) \}$

Determinare FIRST

- 1.for(X ∈ Σ)
  - 2.if(X ∈ T)FIRST(X)={X} else FIRST(X)=∅;
- 3.for(A→aβ ∈ P)
  - 4.FIRST(A)=FIRST(A)∪{a};
- 5.FLAG=true;
- 6.while(FLAG){
  - 7.FLAG=false;
  - 8.for(A → $X_1X_2…X_n$ ∈ P){
    - 9.i=1;
    - 10.if((FIRST(X1) ⊈ FIRST(A)){
      - 11.FIRST(A)=FIRST(A) ∪(FIRST(X1)-{ε});
      - 12.FLAG=true;
    - 13.}//endif
    - 14.while(i<n && X_i st ⇒* ε)
      - 15.if((FIRST(X_{i+1}) ⊈ FIRST(A)){
        - 16.FIRST(A)=FIRST(A)∪ FIRST(X_{i+1});
        - 17.FLAG=true;i++;
      - }//endif
    - }//endwhile
  - }//endfor
- }//endwhile
- for(A ∈ N)
  - if(A st ⇒* ε)FIRST(A)=FIRST(A) ∪{ε};

Determinare FIRST

- **Intrare:** Gramatica $G=(N,T,S,P)$.
- Mulţimile $FIRST(X), X \in \Sigma$.
- $\alpha = X_1X_2 \ldots X_n,\ X_i \in \Sigma, 1 \le i \le n$.
- **Ieşire:** $FIRST(\alpha)$.

- 1.FIRST(α)=FIRST(X_1)-{ε};i=1;
- 2.while(i<n && $X_i \Rightarrow^+ \varepsilon$){
  - 3.FIRST(α)=FIRST(α)∪(FIRST(X_{i+1})-{ε});
  - 4.i=i+1;
- }//endwhile
- 5.if(i==n && $X_n \Rightarrow^+ \varepsilon$)
  - 6.FIRST(α)=FIRST(α) ∪ {ε};

Exemplu

- Fie gramatica:
- $S \to E \mid B$, $E \to \varepsilon$, $B \to a \mid begin\ SC\ end$, $C \to \varepsilon \mid ;\ SC$

- FIRST(S) = {a, begin, ε} FIRST(E) = {ε}
- FIRST(B) = {a, begin} FIRST(C) = {;, ε}.
- FIRST(SEC) = {a, begin, ;, ε},
- FIRST(SB)= {a, begin},
- FIRST(;SC)= {;}.

Determinarea FOLLOW

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

- Dacă $A \rightarrow \alpha B\beta \in P$ atunci FIRST(β)–$\{\varepsilon\} \subseteq$ FOLLOW (B).

- Dacă $A \rightarrow \alpha B\beta \in P$ şi $\beta \Rightarrow^{+} \varepsilon$, atunci FOLLOW(A) $\subseteq$ FOLLOW(B).

Determinarea FOLLOW

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

Determinarea FOLLOW

- 10.FLAG=true;
- 11.while(FLAG){
  - 12.FLAG=false;
  - 13.for(A → $X_1X_2…X_n$){
    - 14.i=n;
    - 15.while(i>0 && $X_i \in N$){
      - 16.if(FOLLOW(A) $\not\subset$ FOLLOW(X_i)){
        - 17.FOLLOW(Xi)=FOLLOW(X_i) $\cup$ FOLLOW(A);
        - 18.FLAG=true;
      - 19.}//endif
      - 20.if($X_i \Rightarrow^{+} \varepsilon$)--i;
      - 21.else continue;
    - 22.}//endwhile
  - 23.}//endfor
- 24.}//endwhile

Exemplu

- Fie gramatica:
- $S \to E \mid B$, $E \to \varepsilon$, $B \to a \mid$ begin SC end,
  $C \to \varepsilon \mid ;\ SC$

- FOLLOW(S)=FOLLOW(E)=FOLLOW(B) ={ε, ; , end}
- FOLLOW(C) = {end}.

Gramatici SLR(1)

- **Definiţie**
  - Fie G o gramatică pentru care automatul LR(0) conţine 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 condiţiile:
  - –Dacă $A\to\alpha\bullet$, $B \to \beta \bullet \in t$ atunci $FOLLOW(A) \cap FOLLOW(B) = \emptyset$;
  - –Dacă $A\to\alpha\bullet$, $B \to \beta \bullet a\gamma \in t$ atunci $a \notin FOLLOW(A)$.

- Analiza sintactică SLR(1) este similară cu cea LR(0); tabela de analiză sintactică are două componente:
  - –Prima, numită ACŢIUNE, determină dacă parserul va face deplasare respectiv reducere, în funcţie de starea ce se află în topul stivei şi de simbolul următor din intrare
  - –Cea de a doua, numită GOTO, determină starea ce se va adăuga în stivă în urma unei reduceri.

Construcţia tabelei de parsare SLR(1)

- Intrare:
  - Gramatica G = (N, T, S, P) augmentată cu S' → S;
  - Automatul M = (Q, Σ, g, t_0, Q);
  - Mulţimile FOLLOW(A), A∈V
- Ieşire:
  - Tabela de analiză SLR(1) compusă din două părţi:
  - ACŢIUNE(t, a), t ∈ Q, a ∈ T ∪ { # },
  - GOTO(t, A), t ∈ Q, A ∈ N.

Construcţia tabelei de parsare SLR(1)

- for(t ∈ Q)
  - for (a ∈ T) ACTIUNE(t, a) = "eroare";
  - for (A ∈ V) GOTO(t, A) = "eroare";
- for(t ∈ Q}{
  - for(A→ α•aβ ∈ t)
    - ACTIUNE(t,a)="D g(t, a)";//deplasare in g(t, a)
  - for(B→ γ• ∈ t){// acceptare sau reducere
    - if(B == 'S') ACTIUNE(t, #) = "acceptare";
    - else
      - for(a ∈ FOLLOW(B)) ACTIUNE(t,a)="R B→γ";
  - }// endfor
  - for (A ∈ N) GOTO(t, A) = g(t, A);
- }//endfor

Parsarea SLR(1)

- **Deplasare**: $(\sigma t, au\#, \pi) \vdash (\sigma t t', u\#, \pi)$ dacă ACTIUNE(t, a)=Dt';

- **Reducere**: $(\sigma t \sigma' t', u\#, \pi) \vdash (\sigma t t'', u\#, \pi r)$ ACTIUNE(t, a) = Rp unde $p = A \rightarrow \beta$, $|\sigma' t'| = |\beta|$ şi t''= GOTO(t, A);

- **Acceptare**: $(t_0 t, \#, \pi)$ dacă ACTIUNE(t,a) = "acceptare"; Analizorul se opreşte cu acceptarea cuvântului de analizat iar π este parsarea acestuia (şirul de reguli care s-a aplicat, în ordine inversă, în derivarea extrem dreaptă a lui w).

- **Eroare**: $(\sigma t, au\#, \pi) \vdash$ eroare dacă ACTIUNE(t,a) = "eroare"; Analizorul se opreşte 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 → $X_1X_2 \ldots X_m$”){
      - for(i = 1; i ≤m; i++) St.pop();
      - St.push(GOTO(St.top, A));
    - } //endif
    - else exit(“eroare”);
  - }//endelse
- }//endwhile

Exemplu

- 0.S → E, 1.E → E+T, 2.E → T, 3.T → T*F, 4.T → F, 5.F →(E), 6.F → a

0

S → •E
E → •E+T
E → •T
T → •T*F
T → •F
F → •(E)
F → •a

1

S → E•
E → E•+T

2

E → T•
T → T•*F

3

T → F•

4

F → (•E)
E → •E+T
E → •T
T → •T*F
T → •F
F → •(E)
F → •a

5

F → a•

6

E → E+•T
T → •T*F
T → •F
F → •(E)
F → •a

7

T → T*•F
F → •(E)
F → •a

8

F → (E•)
E → E•+T

9

E → E+T•
T → T•*F

10

T → T*F•

11

F → (E)•

Tabela de tranziţie 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 conţin conflict de deplasare/reducere

- FOLLOW(S)={#}, FOLLOW(E)={#,+,)}

- Gramatica este SLR(1) pentru că:
  - în starea 1: $+ \notin$ FOLLOW(S);
  - în starea 2: $* \notin$ FOLLOW(E);
  - în starea 9: $* \notin$ FOLLOW(E).

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

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

Gramatici LR(1)

- **Definiţie**
  - Fie G = (V, T, S, P) o gramatică redusă. Un articol LR(1) pentru gramatica G este o pereche ($A \to \alpha \bullet \beta$, a), unde $A \to \alpha\beta$ este un articol LR(0), iar a $\in$FOLLOW(A) (se pune # în loc de ε).
- **Definiţie**
  - Articolul ($A \to \beta 1 \bullet \beta 2$, a) este valid pentru prefixul viabil $\alpha\beta 1$dacă are loc derivarea
    - S dr$\Rightarrow$*αAu $\Rightarrow$ $\alpha\beta 1\beta 2$u
    - iar a = 1:u (a = # dacă u = ε).
- **Teorema**
  - O gramatică G = (V, T, S, P) este gramatică LR(1) dacă şi numai dacă oricare ar fi prefixul viabil φ, nu există două articole distincte, valide pentru φ, de forma($A \to \alpha\bullet$, a), ($B \to \beta\bullet\gamma$, b) unde a $\in$FIRST(γb).

Gramatici LR(1)

- Nu există conflict deplasare/reducere. Un astfel de conflict înseamnă două articole ($A \to \alpha\bullet$, a) şi ($B \to \beta\bullet a\beta'$, b) valide pentru acelaşi prefix.
- Nu există conflict reducere/reducere. Un astfel de conflict înseamnă două articole complete($A \to \alpha\bullet$, a) şi ($B \to \beta\bullet$, a) valide pentru acelaşi 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 ($B \to \beta\bullet A\beta'$, b) este valid pentru un prefix viabil atunci toate articolele de forma ($A \to \bullet\alpha$, a), unde $a \in$FIRTS(β'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 = IU{(B → •γ , b)};
          - flag= true;
        - }//endif
      - }//endforb
    - }//endforB
  - }//endforA
- }//endwhile
- return I;

Automatul LR(1)

- t0 = închidere((S′ → ●S,#));T={t_0};marcat(t_0)=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 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 şi L(M) coincide cu mulţimea prefixelor viabile ale lui G. Mai mult, pentru orice prefix viabil γ, $g(t_0,\gamma)$ reprezintă mulţimea articolelor LR(1) valide pentru γ.
- Automatul LR(1) pentru o gramatică G, se foloseşte pentru a verifica dacă G este LR(1)
  - Conflict reducere/reducere: Dacă în T există o stare ce conţine articole de forma $(A \rightarrow \alpha\bullet, a)$, $(B \rightarrow \beta\bullet, a)$ atunci gramatica nu este LR(1);
  - Conflict deplasare/reducere: Dacă în Texistă o stare ce conţine articole de forma $(A \rightarrow \alpha\bullet, a)$ şi $(B \rightarrow \beta_1\bullet a\beta_2, b)$, atunci G nu este LR(1).
  - O gramatică este LR(1) dacă orice stare $t \in T$este liberă de conflicte

Exemplu

- S → L=R|R, L → *R|a, R → L

0

(S' → •S, #)
(S → •L=R, #)
(S → •R, #)
(L → •*R, {=,#})
(L → •a, {=,#})
(R → •L, #)

6

(S → L=•R, #)
(R → •L, #)
(L → •*R, #)
(L → •a, #)

1

(S' → S•, #)

2

(S → L•=R, #)
(R → L•, #)

7

(L → *R•, {=,#})

8

(R → L•, {=,#})

12

(L → a•, #)

3

(S → R•, #)

5

(L → a•, {=,#})

9

(S → L=R• , #)

10

(R → L•, #)

13

(L → *R•, #)

4

(L → *•R, {=, #})
(R → •L, {=, #})
(L → •*R, {=, #})
(L → •a, {=, #})

11

(L → *•R, #)
(R → •L, #)
(L → •*R, #)
(L → •a, #)

Tabela de tranziţie

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						

Construcţia tabelei de analiză LR(1)

- for(t ∈ T)
  - for (a ∈ T) ACTIUNE(t, a) = "eroare";
  - for (A ∈ V) GOTO(t, A) = "eroare";
- for(t ∈ T}{
  - for((A→ α•aβ, L) ∈ t)
    - ACTIUNE(t,a)="D g(t, a)";//deplasare in g(t, a)
  - for((B→ γ•, L) ∈ t){// acceptare sau reducere
    - for(c ∈ L) {
      - if(B == 'S') ACTIUNE(t, #) = "acceptare";
      - else ACTIUNE(t,c)="R B→γ";//reducere cu B→γ
      - }//endfor
  - }// endfor
  - for (A ∈ N) GOTO(t, A) = g(t, A);
- }//endfor

Exemplu

- 0:S'→S, 1:S →L=R, 2:S →R, 3:L →*R, 4:L →a, 5:R →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)

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

- **Defininiţie**
  - Două stări t_1 şi t_2 ale automatului LR(1) pentru gramatica G sunt echivalente dacă au acelaşi nucleu.

Gramatici LALR(1)

- Fiecare stare a automatului LR(1) este o mulţime de articole LR(1). Pornind de la două stări t_1 şi t_2 putem vorbi de starea t1 $\cup$ t2.
  - Fie $t_1 = \{(L \rightarrow *R., \{=, \#\})\}$, $t_2 = \{(L \rightarrow *R., \#)\}$, atunci $t_1 \cup t_2 = t_1$ pentru că $t_2 \subset t_1$.

- **Definiţie**
  - Fie G gramatică LR(1) şi M = (Q, Σ, g, t0, 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' → S;
- **Ieşire**: Tabela de analiză LALR(1) (ACŢIUNE şi GOTO).
- **Algoritm**:
  - 1. Se construieşte automatul LR(1), M = (Q, Σ, g, t_0, Q) Fie Q = {t_0, t_1,..., t_n}. Dacă toate stările din Q sunt libere de conflict, urmează 2, altfel algoritmul se opreşte deoarece gramatica nu este LR(1).
  - 2. Se determină stările echivalente din Q şi, prin reuniunea acestora, se obţine o nouă mulţime de stări Q' = {t'_0, t'_1,..., t'_m}
  - 3. Dacă în Q' există stări ce conţin conflicte, algoritmul se opreşte deoarece gramatica G nu este LALR(1).

Tabela de analiză LALR(1)

- **4.** Se construieşte 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 $t' = t_1 \cup t_2 \cup \ldots$, $t_1, t_2, \ldots \in Q$, atunci
    - **7.** $g'(t', X) = g(t1, X) \cup g(t2, X) \cup \ldots$.
- **8.** Se aplică algoritmul pentru construirea tabelei de parsare LR(1) pornind de la automatul M'. Tabela obţinută se numeşte tabela LALR(1) pentru gramatica G.

Exemplu

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

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	D5		D4			8	9
7		R3		R3			
8		R5		R5			
9				R1			

Exemplu

- Există gramatici LR(1) care nu sunt LALR(1).
  - S →aAb | bAd | aBd | bBb
  - A →e
  - B →e

Bibliografie

- Grigoraş Gh., *Construcţia compilatoarelor. Algoritmi fundamentali*, Editura Universităţii "Alexandru Ioan Cuza", Iaşi, 2005