Curs 9

Analizor sintactic SLR

- SLR = Simple LR

- Observație:

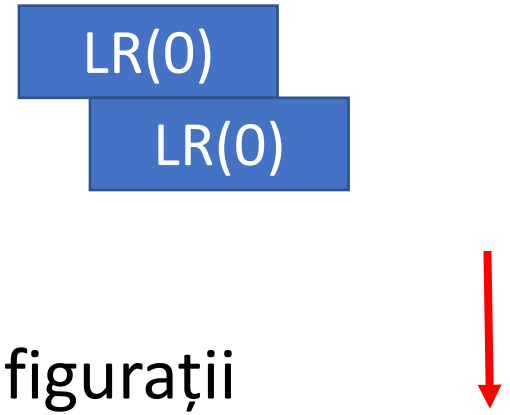
LR(0) – multe conflicte – dacă se ține cont de predicție atunci se elimină conflictul

=>

1. Colecție canonică de stări LR(0) – predicție de lungime 0
2. Tabel și analiza secvenței – predicție de lungime 1

Analiză sintactică LR(k): LR(0), SLR, LR(1), LALR

- definirea elementului de analiză
- construirea mulțimii de stări
- construirea tabelului de analiză
- Analiza secvenței de baza tranzițiilor între configurații



Construcția tabelului SLR

Obsevații:

1. Predicția = următorul simbol din șirul de intrare => FOLLOW

   - vezi LL(1)

2. Structura – LR(k):
   - Linii - stări
   - parte de acțiune + parte de goto

acțiune – câte o coloană pentru fiecare predicție $\in \Sigma$

goto – câte o coloană pentru fiecare simbol X $\in N \cup \Sigma$

Obs. (tabel LR(0)):

- Dacă într-o anumită stare ***s*** acțiunea este de acceptare, goto(s, X) = $\varnothing$, $\forall X \in N \cup \Sigma$.
- Dacă într-o anumită stare ***s*** acțiunea este de reducere, atunci goto(s, X) = $\varnothing$, $\forall X \in N \cup \Sigma$.

Tabel SLR

	Acțiune			GOTO		
	a_1	…	a_n	B_1	…	B_m
s_0						
s_1						
…						
s_k						

$a_1,\dots,a_n \in \Sigma$
$B_1,\dots,B_m \in N$
$s_0,\dots,s_k$ - stări

Reguli tabel SLR

1. dacă $[A \rightarrow \alpha.\beta] \in s_i$ și $goto(s_i,a) = s_j$ atunci acțiune(s_i,a)=**shift s_j**
2. dacă $[A \rightarrow \beta.] \in s_i$ și $A \neq S'$ atunci acțiune(s_i,u)=**reduce l**, unde l - numărul producției $A \rightarrow \beta$, $\forall u \in FOLLOW(A)$
3. dacă $[S' \rightarrow S.] \in s_i$ atunci acțiune$(s_i,\$)$=**acc**
4. dacă $goto(s_i, X) = s_j$ atunci **$goto(s_i, X) = s_j$** , $\forall X \in N$
5. toate celelalte valori = **eroare**

Observații

1. Similaritate cu LR(0)

2. O gramatică este de tip SLR dacă tabelul SLR **NU** conține conflicte

Analiza secvenței de baza tranzițiilor între configurații

- INPUT:
  - Gramatica limbajului G’ = (NU{S’}, **Σ**, P U {S’->S},S’)
  - Tabel de analiză SLR
  - Secvența de analizat w =a_1…a_n
- OUTPUT:

  Dacă (w ∈L(G)) *atunci* **șir de producții**

  altfel **locația erorii**

Configurații SLR ≈ LR(0)

$$(\alpha, \beta, \pi)$$

Unde:

- α = stiva de lucru
- β = stiva de intrare
- π = banda de ieșire (rezultat)

Configurația inițială:
($s$$s_0$,w$$,ε)

Configurația finală:
($s$$s_{acc}$, $$, π)

Tranziții

head(β) = predicția

1. **Deplasare**

dacă actiune(s_m,a_i)= shift s_j **atunci**

($s_0x_1 \ldots x_m s_m, a_i \ldots a_n\$, \pi) \vdash (\$s_0x_1 \ldots x_m s_m a_i s_j, a_{i+1} \ldots a_n\$, \pi)$

2. **Reducere**

dacă actiune(s_m,a_i) = reduce t AND (t) A → $x_{m-p+1} \ldots x_m$ AND goto(s_{m-p},A) = s_j

atunci

($s_0 \ldots x_m s_m, a_i \ldots a_n\$, \pi) \vdash (\$s_0 \ldots x_{m-p} s_{m-p} A s_j, a_i \ldots a_n\$, t\ \pi)$

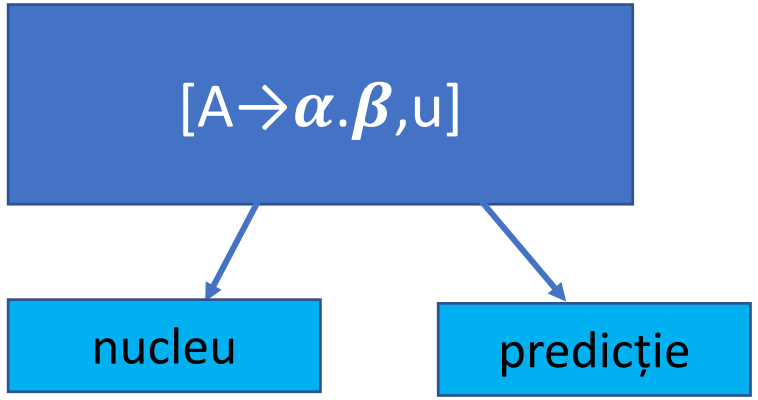
3. **Acceptare**

dacă actiune(s_m,$) = accept **atunci** ($s_m$,$, π)=acc

3. **Eroare** - altfel

Analizor sintactic LR(1)

1. definirea elementului de analiză
2. construirea mulțimii de stări
3. construirea tabelului de analiză
4. analiza secvenței de baza tranzițiilor între configurații



Construirea mulțimii de stări LR(1)

- Alg *ColCan_LR1*
- Funcția *goto1*
- Alg *Closure1*

Algoritm *ColCan_LR(1)*

INPUT: G'- gramatica îmbogăţită
OUTPUT: C - colecţia canonică de stări
$\mathcal{C}_1 := \emptyset$;
$s_0 := closure(\{[S' \rightarrow .S, \$]\})$
$\mathcal{C}_1 := \mathcal{C}_1 \cup \{s_0\}$;
repeat
  for $\forall s \in \mathcal{C}_1$ **do**
    for $\forall X \in N \cup \Sigma$ **do**
      T:=goto(s, X);
      if $T \neq \emptyset$ and $T \notin \mathcal{C}_1$ **then**
        $\mathcal{C}_1 = \mathcal{C}_1 \cup T$
      end if
    end for
  end for
until $\mathcal{C}_1$ nu se mai modifică

Funcția *goto1*

goto1 : $P(\mathscr{E}_0) \times (N \cup \Sigma) \rightarrow P(\mathscr{E}_0)$

unde $\mathscr{E}_0$ = mulțimea de elemente LR(0)

goto1(s, X) = closure({[A → αX.β,u] | [A → α.Xβ,u] $\in$ s})

Algoritm *Closure*

- [A → α.Bβ,u] valabil pentru prefixul viabil γα =>

$$S \overset{*}{\Rightarrow}_{dr} \gamma A w \Rightarrow_{dr} \gamma \alpha B \beta w$$

$$u = FIRST_k(w)$$

- [B → .δ, *ceva*] ∈ P => $S \overset{*}{\Rightarrow} \gamma A w \Rightarrow_{dr} \gamma \alpha B \beta w \Rightarrow_{dr} \gamma \alpha \delta \beta w.$

=> [B → .δ,b] valabil pentru prefixul viabil γα,

$\forall$ b ∈ FIRST(βu)

Algoritm *Closure1*

INPUT: I-element de analiză; G'- gramatica îmbogăţită;
$FIRST(X), \forall X \in N \cup \Sigma$;
OUTPUT: $C_1 =$ closure(I);
$C_1 := \{I\}$;
repeat
  for $\forall [A \to \alpha.B\beta, a] \in C_1$ **do**
    for $\forall B \to \gamma \in P$ **do**
      for $\forall b \in FIRST(\beta a)$ **do**
        if $[B \to .\gamma, b] \notin C_1$ **then**
          $C_1 = C_1 \cup [B \to .\gamma, b]$
        end if
      end for
    end for
  end for
until C_1 nu se mai modifică

Construcție tabel LR(1)

- Structura – SLR
- Reguli:

1. dacă $[A \rightarrow \alpha.\beta,u] \in s_i$ și $goto(s_i,a) = s_j$ atunci acțiune(s_i,a)=**shift s_j**
2. dacă $[A \rightarrow \beta.,u] \in s_i$ și $A \neq S'$ atunci acțiune(s_i,u)=**reduce l**, unde l - numărul producției $A \rightarrow \beta$
3. dacă $[S' \rightarrow S.,\$] \in s_i$ atunci acțiune$(s_i,\$)$=**acc**
4. dacă $goto(s_i, X) = s_j$ atunci **$goto(s_i, X) = s_j$** , $\forall X \in N$
5. toate celelalte valori = **eroare**

Observații

1. O gramatică este de tip LR(1) dacă tabelul de analiză LR(1) nu conține conflicte
2. Număr de stări – crește semnificativ

Analiza secvenței de baza tranzițiilor între configurații

- INPUT:
  - Gramatica limbajului G’ = (NU{S’}, **Σ**, P U {S’->S},S’)
  - Tabel de analiză LR(1)
  - Secvența de analizat w =a_1…a_n
- OUTPUT:

  Dacă (w ∈L(G)) *atunci* **șir de producții**

  altfel **locația erorii**

Configurații LR(1)

$$(\alpha, \beta, \pi)$$

Unde:

- α = stiva de lucru
- β = stiva de intrare
- π = banda de ieșire (rezultat)

Configurația inițială:
($s$$s_0$,w$$,ε)

Configurația finală:
($s$$s_{acc}$, $$, π)

Tranziții

head(β) = predicția

1. **Deplasare**

dacă actiune(s_m,a_i)= shift s_j **atunci**

($s_0x_1 \ldots x_m s_m, a_i \ldots a_n\$, \pi) \vdash (\$s_0x_1 \ldots x_m s_m a_i s_j, a_{i+1} \ldots a_n\$, \pi)$

2. **Reducere**

dacă actiune(s_m,a_i) = reduce t AND (t) A → $x_{m-p+1} \ldots x_m$ AND goto(s_{m-p},A) = s_j

atunci

($s_0 \ldots x_m s_m, a_i \ldots a_n\$, \pi) \vdash (\$s_0 \ldots x_{m-p} s_{m-p} A s_j, a_i \ldots a_n\$, t\ \pi)$

3. **Acceptare**

dacă actiune(s_m,$) = accept **atunci** ($s_m$,$, π)=acc

3. **Eroare** - altfel

Analizor sintactic LALR

- LALR = Look Ahead LR(1)

- De ce?

Principiul LALR

- Unificarea stărilor care au același nucleu, cu conservarea tuturor predicțiilor, cu condiția să nu se creeze conflict

$[A \rightarrow \alpha.\beta,u] \in s_i$

$[A \rightarrow \alpha.\beta,v] \in s_j$

$\Rightarrow [A \rightarrow \alpha.\beta,u/v] \in s_{i,j}$

Analiză sintactică LALR

- Ca și LR(1)
- Număr de stări LALR = număr de stări SLR / LR(0)

Analizoare sintactice LR(k)

- LR(0):
  - elementele de analiză nu iau în considerare predicția
  - reducerea poate fi efectuată doar în stări singulare (care conțin un singur element de analiză)
  - se generează multe conflicte.
- SLR:
  - folosește aceleași elemente de analiză ca și LR(0)
  - la reducere se ia în considerare predicția
  - se elimina unele cazuri de conflict care apăreau la LR(0).
- LR(1):
  - algoritm performant de construire a stărilor
  - se generează conflicte puține,
  - se generează prea multe stări.
- LALR:
  - unifică stările LR(1) corespunzătoare aceluiași nucleu
  - este cel mai des folosit algoritm

Analiza sintactică - recapitulare

	Descendent	Ascendent
Recursiv	a.s. descendent cu reveniri	a.s. ascendent cu reveniri
Liniar	LL(1)	LR(0), SLR, LR(1), LALR

Analiza sintactică - recapitulare