LABORATOR NR 4

Gramatici LL(k). Analiza sintactică LL(k)

Fie $G = (N, \Sigma, P, S)$ o gramatică neambiguă şi $w = a_1 a_2 ... a_n \in L(G)$. Atunci există o unică derivare la stânga $p_0 p_1 ... p_{m-1}$ astfel incât $S = \alpha_0, \alpha_i \Rightarrow \alpha_{i+1}; 0 \le i < m$ şi $\alpha_m = w$. Dacă $\alpha_i = a_1 ... a_j A \beta$ dorim ca secvenţa α_{i+1} să poată fi determinată cunoscând primele j simboluri (partea din cuvântul de intrare citită până în acel moment), următoarele k simboluri $a_{j+1} ... a_{j+k}$ (pentru un anumit k) şi neterminalul A. Dacă aceste trei cantităţi determină în mod unic producţia folosită pentru expandarea lui A, vom spune că gramatica G este *LL(k)*. Vom considera că neterminalul care derivează este cel mai din stânga.

Definiţie. Fie $G = (N, \Sigma, P, S)$ o gramatică independentă de context, k un număr natural şi $\alpha \in (N \cup \Sigma)^*$. Definim funcţia $PRIM_k^G(\alpha)$ astfel:

$$PRIM_k^G(\alpha) = \{w \in \Sigma^* \mid |w| < k \quad si\ \alpha \overset{*}{\Rightarrow} w \quad sau\ |w| = k\ si\ \alpha \overset{*}{\Rightarrow} wx\ pentru\ un\ anumit\ x\}$$

Definiţie. Fie $G = (N, \Sigma, P, S)$ o gramatică independentă de context. Spunem că G este *LL(K)* dacă:

(1) $S \overset{*}{\Rightarrow} wA\alpha \Rightarrow w\beta\alpha \overset{*}{\Rightarrow} wx$

(2) $S \overset{*}{\Rightarrow} wA\alpha \Rightarrow w\gamma\alpha \overset{*}{\Rightarrow} wy$

(3) $PRIM_k(x) = PRIM_k(y)$

implică $\beta = \gamma$.

O gramatică este de tip *LL* dacă ea este *LL(k)* pentru un anumit k.

Teoremă. Fie $G = (N, \Sigma, P, S)$ o gramatică independentă de context. Atunci G este *LL(k)* dacă şi numai dacă este adevarată condiţia : $\forall A \in N$ astfel încât există

derivarea $S \overset{*}{\Rightarrow} wA\alpha$ şi pentru orice $A \to \beta \; si \; A \to \gamma$ producţii distincte, rezultă $PRIM_k(\beta\alpha) \cap PRIM_k(\gamma\alpha) = \phi$.

Definiţie. Fie $G = (N, \Sigma, P, S)$ o gramatică independentă de context. Ea se numeste *LL(k)* în sens tare dacă este satisfacută condiţia:dacă $A \to \beta \; si \; A \to \gamma$ sunt două *A*-producţii distincte, atunci

$PRIM_k(\beta URM_k(A)) \cap PRIM_k(\gamma URM_k(A)) = \phi$

Proprietate. Orice gramatică *LL(1)* este *LL(1)* în sens tare. Pentru *k>1* această proprietate nu mai este adevarată.

Analiza sintactică de tip LL

Există clase de gramatici independente de context, pentru care se pot construi analizori sintactici care efectuează analiza unui cuvânt de lungime *n* în $c_1 n$ operaţii şi $c_2 n$ paşi, unde c_1 şi c_2 sunt constante . Algoritmii se caracterizează prin faptul că procesul de analiză este complet determinist iar cuvântul de analizat se parcurge o singură dată de la stânga la dreapta. Clasele de gramatici sunt:

a) gramatici LL(k), care permit construirea arborelui de derivare stângă privind k simboluri spre dreapta din poziţia curentă;
b) gramatici LR(k), care permit construirea arborelui de derivare la dreapta privind k simboluri spre dreapta după simbolul curent;
c) gramatici de precedenţă, care permit construirea arborelui de derivare la dreapta determinând partea reductibilă pe baza unor relaţii (de precedenţă) între simbolurile gramaticii.

Gramatici LL(k)

Fie $G = (N, \Sigma, P, S)$ o gramatică neambiguă şi $w = a_1 a_2 ... a_n \in L(G)$.

Atunci există o unică derivare la stânga $p_0 p_1 \cdots p_{m-1}$ astfel încât $S = \alpha_0, \alpha_i \overset{p_i}{\Rightarrow} \alpha_{i+1}; 0 \le i < m$ şi $\alpha_m = w$. Dacă $\alpha_i = a_1 \ldots a_j A\beta$, dorim ca secvența α_{i+1} să poată fi determinată cunoscând primele j simboluri (partea din cuvântul de intrare citită până în acel moment), următoarele k simboluri $a_{j+1} \ldots a_{j+k}$ (pentru un anumit k) şi neterminalul A. Dacă aceste trei cantități determină în mod unic producția folosită pentru expandarea lui A, vom spune că gramatica G este LL(k).

Definiție. Fie $G = (N, \Sigma, P, S)$ o GIC , *k* un număr întreg şi $\alpha \in (N \cup \Sigma)^*$.

Definim

funcția $PRIM_k^G(\alpha)$ astfel :

$$PRIM_k^G(\alpha) = \left\{ w \in \Sigma^* / |w| < k \text{ si } \alpha \overset{*}{\Rightarrow} w \quad \text{sau} \quad |w| = k \text{ si } \alpha \overset{*}{\Rightarrow} wx \quad \text{pentru un anumit } x \right\}$$

Am notat cu $|x|$ lungimea secvenței x.

Definiție. Fie $G = (N, \Sigma, P, S)$ o GIC. G este LL(k) dacă:

(1) $S \overset{*}{\Rightarrow} wA\alpha \Rightarrow w\beta\alpha \overset{*}{\Rightarrow} wx$

(2) $S \overset{*}{\Rightarrow} wA\alpha \Rightarrow w\gamma\alpha \overset{*}{\Rightarrow} wy$

(3) $PRIM_k(x) = PRIM_k(y)$

implică $\beta = \gamma$.

O gramatică este de tip LL dacă ea este LL(k) pentru un anumit k.

Teoremă. Fie $G = (N, \Sigma, P, S)$ o GIC . *Atunci G este LL(k) dacă şi numai dacă este adevărată condiția :* $\forall A \in N$ *astfel încât există derivarea* $S \overset{*}{\Rightarrow} wA\alpha$ *şi pentru orice* $A \to \beta$ *şi* $A \to \gamma$ *producții distincte, rezultă* $PRIM_k(\beta\alpha) \cap PRIM_k(\gamma\alpha) = \varnothing$

Definiție. Fie $G=(N,\Sigma,P,S)$ o GIC ; definim funcția $URM_k(\beta)$ unde k este un întreg pozitiv iar $\beta\in(N\cup\Sigma)^*$ astfel:

$$URM_k(\beta)=\left\{w \mid S\overset{*}{\Rightarrow}\alpha\beta\gamma \; si \quad w\in PRIM_k(\gamma)\right\} .$$

Funcțiile $PRIM_k$ şi URM_k pot fi extinse la mulțimi de cuvinte formate peste alfabetul $N\cup\Sigma$:dacă $L\subseteq(N\cup\Sigma)^*$, atunci

$$PRIM_k(L)=\{w / w=PRIM_k(\alpha)\, si\, \alpha\in L\}$$

$$URM_k(L)=\{w / w=URM_k(\alpha)\, si\, \alpha\in L\}$$

Definiție. Fie G o GIC; ea se numeşte LL(k) în sens tare dacă este satisfăcută condiția: dacă $A\to\beta$ şi $A\to\gamma$ sunt două A-producții distincte atunci

$$PRIM_k(\beta URM_k(A))\cap PRIM_k(\gamma URM_k(A))=\varnothing$$

Definiție. Fie Σ un alfabet şi $L_1,L_2\subseteq\Sigma^*$. Definim $L_1\oplus L_2=\{w / \exists x\in L_1\; si\; \exists y\in L_2\}$ astfel încât

a) $w=xy$ dacă $|xy|\le k$ şi

b) w este format din primele k simboluri din xy,în cax contrar }

Algoritm. –testarea condiției LL(k)

Intrare : o GIC $G=(N,\Sigma,P,S)$ şi un număr întreg pozitiv k

Ieşire : "da" dacă gramatica este LL(k) şi "nu" în caz contrar

Metoda :

Se consideră că toate neterminalele sunt nemarcate.

Pasul 1. Pentru orice $A\in N$, nemarcat, pentru care există cel puțin două A-producții distincte calculează $\sigma(A)=\left\{L\subseteq\Sigma^{*k} \mid S\overset{*}{\Rightarrow}wA\alpha \; si\; L=PRIM_k(\alpha)\right\}$

Pasul 2. Dacă $A \to \beta$ şi $A \to \gamma$ sunt două A-producţii distincte, calculează $f(L) = (PRIM_k(\beta) \oplus_k L) \cap (PRIM_k(\gamma) \oplus_k L)$ pentru orice $L \in \sigma(A)$. Dacă $f(L) = \varnothing$, răspunde "nu" şi opreşte algoritmul. Dacă $f(L) = \varnothing$ pentru orice $L \in \sigma(A)$, repetă pasul 2 pentru toate perechile distincte de A-producţii şi la sfârsit marchează pe A.

Pasul 3. Repetă paşii 1 şi 2 pentru toate neterminalele nemarcate din N.

Pasul 4. Răspunde "da" dacă toate neterminalele au fost marcate; în caz contrar se merge la pasul 1.

Pentru a putea aplica acest algoritm trebuie calculate funcţiile $PRIM_k$ şi σ.

Algoritm. –calculează funcţia $PRIM_k$

Intrare : o GIC $G = (N, \Sigma, P, S)$ k un număr întreg pozitiv şi $\alpha = x_1 ... x_n \in (N \cup \Sigma)^*, n \geq 1$

Ieşire : $PRIM_k(\alpha)$

Metoda : Ţinând seama de descompunerea

$PRIM_k(\alpha) = PRIM_k(X_1) \oplus_k PRIM_k(X_2) \oplus_k ... \oplus_k PRIM_k(X_n)$ rezultă că este suficient să calculăm $PRIM_k(x)$ cu $x \in N$; dacă $x \in \Sigma \cup \{\varepsilon\}$ atunci $PRIM_k(x) = \{x\}$. Calculăm recursiv mulţimile $F_i(x)$ pentru $x \in N \cup \Sigma$ şi $i \geq 0$

Pasul 1. $F_i(a) = \{a\} \quad : \forall a \in \Sigma \cup \{\varepsilon\}, i \geq 0$

Pasul 2. $F_0(A) = \{x \in \Sigma^{*k} / (A \to x\alpha) \in P \, unde \, fie \, |x| = k \, fie \, |x| < k \, si \, \alpha = \varepsilon\}$

Pasul 3. Presupunem că mulţimile $F_0, F_1, ..., F_{i-1}$ au fost calculate pentru orice $A \in N$. Atunci $F_i(A) = F_{i-1}(A) \cup \{x / x \in F_{i-1}(y_1) \oplus_k ... \oplus_k F_{i-1}(y_n), A \to y_1 ... y_n \in P\}$

Pasul 4. Dacă pentru $\forall A \in N$ avem $F_{i-1}(A) = F_i(A)$ luăm $PRIM_k(A) = F_i(A)$ şi oprim algoritmul. În caz contrar se merge la pasul 3.

Algoritm – calculează funcția σ

Intrare : o GIC $G = (N, \Sigma, P, S)$ şi *k* un număr întreg nenegativ

Ieşire : $\sigma(A)$ pentru $\forall A \in N$

Metoda : Pentru $\forall A, B \in N$ calculăm $\sigma(A,B) = \left\{ L \mid L \subset \Sigma^{*k}, \exists \left(A \overset{+}{\Rightarrow} wB\alpha \right) si\ L = PRIM_k(\alpha) \right\}$ Pentru aceasta construim mulțimile $\sigma_i(A,B)$ pentru $\forall A, B \in N$ şi $i = 0,1,...$ astfel:

Pasul 1. Fie $\sigma_0(A,B) = \left\{ L \subseteq \Sigma^k \ / (A \to \beta B \alpha) \in P\ si\ L = PRIM_k(\alpha) \right\}$.

Pasul 2. Presupunem că mulțimile $\sigma_i(A,B)$ astfel:

(a) dacă $L \subset \sigma_{i-1}(A,B)$ atunci $L \in \sigma_i(A,B)$

(b) dacă există producția $A \to X_1 ... X_n$ şi pentru un *j*, $1 \le j \le n$, există o mulțime $L' \in \sigma_{i-1}(X_j, B)$, atunci include în $\sigma_i(A,B)$ pe $L = L' \oplus_k (X_{j+1} ... X_n)$.

Pasul 3. Dacă pentru orice $A, B \in N$ există un *i* astfel încât $\sigma_i(A,B) = \sigma_{i-1}(A,B)$ luăm $\sigma(A,B) = \sigma_i(A,B)$; în caz contrar se merge la pasul (2).

Pasul 4. Pentru orice $A \in N$ se ia $\sigma(A) = \sigma(S, A)$.

Definiție. Fie $G = (N, \Sigma, P, S)$ o GIC. Pentru fiecare $A \in N$ şi $L \subseteq \Sigma^{*k}$ definim funcțiile $T_{A,L}$,numite tabele LL(k) asociate cu A şi L,astfel:

(1) $T_{A,L}(u) =$ eroare dacă nu există nici-o producție $A \to \alpha$ astffel încât $u \in PRIM_k(\alpha) \oplus_k L$.

(2) $T_{A,L}(u) = (A \to \alpha, < Y_1, Y_2, ..., Y_m >)$ dacă există o unică producție $A \to \alpha$ astfel încât $u \in PRIM_k(\alpha) \oplus_k L$.

Dacă $\alpha = x_0 B_1 x_1 B_2 ... B_m x_m, m \geq 0, B_i \in N$ şi $x_i \in \Sigma^*$, atunci $Y_i = PRIM_k(x_i B_{i+1} x_{i+1} ... B_m x_m) \oplus_k L$. Y_i se numeşte mulţimea local următoare a lui B_i. Dacă $m = 0$, atunci $T_{A,L}(u) = (A \to \alpha, \Phi)$.

(3) $T_{A,L}(u)$ este nedefinită dacă există cel puţin două producţii $A \to \alpha_1 / \alpha_2 / ... / \alpha_n$ astfel încât $u \in PRIM_k(\alpha_i) \oplus_k L, \quad 1 \leq i \leq n, \quad n \geq 2$. Această situaţie nu apare dacă G este LL(k).

Intuitiv, $T_{A,L}(u) =$ *eroare* spune că nu este posibilă nici-o derivaţie de forma $Ax \overset{*}{\Rightarrow} uv$ pentru nici-un $x \in L$ şi $v \in \Sigma^*$. Când $T_{A,L}(u) = (A \to \alpha, < Y_1, Y_2, ..., Y_m >)$ există exact o producţie $A \to \alpha$ care poate fi utilizată la primul pas al derivaţiei $Ax \overset{*}{\Rightarrow} uv$ pentru orice $x \in L$ şi $v \in \Sigma^*$. Fiecare mulţime Y_i dă toate prefixele posibile de lungime cel mult k, formate din terminale, care pot urma un şir derivat din B_i când utilizăm producţia $A \to \alpha$, unde $\alpha = x_0 B_1 x_1 B_2 ... B_m x_m$ în orice derivaţie de forma $Ax \overset{+}{\Rightarrow} \alpha x \overset{*}{\Rightarrow} uv$, cu $x \in L$.

Algoritm –construcţia tabelelor LL(k)

Intrare : o GIC $G = (N, \Sigma, P, S)$ de tip LL(k)

Ieşire : mulţimea P a tabelelor LL(k)

Metoda :

Pasul 1. Se construieşte $T_0 = T_{S,\varepsilon}$ şi se iniţiază $\mathsf{P} = \{T_0\}$.

Pasul 2. Pentru fiecare tabel LL(k) T cu intrarea $T(u) = (A \to x_0 B_1 x_1 ... B_m x_m, < Y_1, Y_2, ..., Y_m >)$ se adaugă la P tabelul T_{B_i, Y_i} pentru $1 \leq i \leq m$ dacă el nu există deja în P.

Pasul 3. Se repetă pasul (2) până când nu se mai poate adăuga nici un tabel .
Analiza sintactică, folosind mulțimea de tabele LL(k) este dată de algoritmul următor.

Algoritm

Intrare : gramatica $G=(N,\Sigma,P,S)$ de tip LL(k) şi P

Ieşire : tabelul M de analiză sintactică

Metoda : M este definit pe $(I\cup\Sigma\cup\{\$\})\times\partial\Sigma^{*k}$ astfel:

Pasul 1. Dacă $A\to x_0B_1x_1B_2...B_mx_m$ este producția cu numărul *i* , $T_{A,L}\in$ şi $T_{A,L}(u)=(A\to x_0B_1x_1B_2x_2...B_mx_m,<Y_1,Y_2,...,Y_m>)$ atunci M se defineşte astfel $M(T_{A,L},u)=(x_0T_{B_1,Y_1}x_1...T_{B_m,Y_m}x_m,i)$.

Pasul 2. $M(a,av)=$ reducere pentru orice $v\in\Sigma^{*(k-1)}$.

Pasul 3. $M(\$,\varepsilon)=$ acceptare.

Pasul 4. $M(X,u)=$ eroare,în alte cazuri.

Configurația inițială este $(T_0\$,w,\varepsilon)$ iar cea finală este $(\$,\varepsilon,\pi)$ unde *w* este cuvântul de analizat iar π este analiza sintactică la stânga a lui *w*.

Gramatici LL(1)

Analiza sintactică a gramaticilor LL(k) se simplifică în cazul $k=1$.

Gramatici LL(1) simple

O GIC $G=(N,\Sigma,P,S)$ fără ε-producții cu proprietatea că pentru orice $A\in N$, toate *A*-producțiile încep cu un terminal diferit se numeşte gramatica LL(1) simplă sau *s*-gramatică. Deci o s-gramatică are producțiile de forma $A\to a_1\alpha_1/a_2\alpha_2/.../a_m\alpha_m, a_i\neq a_j$ pentru $i\neq j, a_i\in\Sigma,\ \ 1\le i\le m$.

Tabelul de analiză în cazul s-gramaticilor se defineşte astfel: $M : N \cup \Sigma \cup \{\$\} \times (\Sigma \cup \{\varepsilon\} \to \{(\beta, i), reducere, acceptare, eroare\})$, unde β este membrul drept al producției i .

$$M(A,a) = \begin{cases} reducere \quad dacă\ A = a \in \Sigma \\ acceptare \quad dacă\ A = \$ \text{ şi } a = \varepsilon \\ (a\alpha, i) \quad dacă\ A \to a\alpha \text{ este productia } i \\ eroare \quad altfel \end{cases}$$

Trecerea de la o configurație la alta se face astfel:

$$(A\alpha, az, p) |- \begin{cases} (\alpha, z, p) \quad dacă\ M(A,a) = reducere \\ ter \min are \quad dacă\ M(A,a) = acceptare \\ (\beta\alpha, az, pi) \quad dacă\ M(A,a) = (\beta, i) \\ eroare \quad dacă\ M(A,a) = eroare \end{cases}$$

Configurația inițială este $(S\$, w, \varepsilon)$.

Gramatici LL(1) fără ε -producții

O gramatică fără ε -producții este LL(1) dacă pentru orice regulă de forma $A \to \alpha_1 / \alpha_2 / ... / \alpha_n$,mulțimile $PRIM_1(\alpha_1), PRIM_1(\alpha_2), ..., PRIM_1(\alpha_n)$ sunt distincte două câte două. Tabelul de analiză sintactică se defineşte cu ajutorul funcției $M : N \cup \Sigma \cup \{\$\} \times (\Sigma \cup \{\$\}) \to \{(\beta, i), reducere, acceptare, eroare\}$,unde β este membrul drept al producției i *:*

$$M(A,a) = \begin{cases} reducere \quad dacă\ A = a \in \Sigma \\ acceptare \quad dacă\ A = \$ \text{ şi } a = \varepsilon \\ (\beta, i) \quad dacă\ a \in PRIM_1(\beta \quad este\ producțro\ i) \\ eroare \quad în\ alte\ cazuri \end{cases}$$

Trecerea de la o configurație la alta se face astfel:

1. $(A\alpha, z, p) |- (\beta\alpha, z, pi) \quad dacă\ M(A,u) = (\beta, i)$

2. $(a\alpha, z, p) \vdash (\alpha, z', p)$ *dacă* $M(A,u) = reducere$ *şi* $z = az'$

3. $(\$, \varepsilon, P) \vdash ter \min are$

4. $(X\alpha, z, p) \vdash eroare$ *dacă* $M(X,u) = eroare$

unde am notat $u = PRIM_1(z)$.

Configurația inițială este $(S\$, w, \varepsilon)$.

Gramatici LL(1) cu ε -producții

O gramatică $G = (N, \Sigma, P, S)$ este LL(1) dacă pentru orice neterminal A şi orice două A-producții distincte $A \to \alpha$ şi $A \to \beta$ este verificată condiția $PRIM_1(\alpha URM(A)) \cap PRIM_1(\beta URM_1(A)) = \varnothing$.

O formă mai simplă a condiției de mai sus este: pentru orice reguli $A \to \alpha_1 / \alpha_2 / ... / \alpha_n$:

1. $PRIM_1(\alpha_i) \cap PRIM_1(\alpha_j) = \varnothing$ pentru $i \neq j$

2. dacă $\alpha_i \overset{*}{\Rightarrow} \varepsilon$ atunci $PRIM_1(\alpha_i) \cap URM_1(A) = \varnothing$ pentru $i \neq j$

Analiza sintactică se face cu ajutorul funcției $M : N \cup \Sigma \cup \{\$\} \times (\Sigma \cup \{\$\}) \to \{(\beta, i), reducere, acceptare, eroare\}$, unde β este membrul drept al producției *i*.

$$M(A,a) = \begin{cases} reducere \quad dacă \quad A = a \in \Sigma \\ acceptare \quad dacă \quad A = \$ \quad şi \quad a = \varepsilon \\ (\alpha, i) \quad dacă \quad a \in PRIM_1(\alpha) \quad şi \quad A \to \alpha \ este \ productia \ i \\ (\alpha, i) \quad dacă \ a \in URM_1(A) şi \ A \to \alpha \quad este \ producţro i \ iar \ \varepsilon \in PRIM_1(\alpha) \\ eroare \quad în \ alte \ cazuri \end{cases}$$

Analiza sintactică se efectuează cu următorul algoritm:

$p \leftarrow 1$

$pd \leftarrow$ Start $\circ$'$'

$be \leftarrow$''

ind $\leftarrow$ false

while ind=false do

begin

t $\leftarrow$ Vârf (pd)

u $\leftarrow$ bi (p)

if $M(t,u)=(\alpha,i)$

then

begin

pd $\leftarrow \alpha \circ$ Rest (pd)

be $\leftarrow$ be $\circ$'i'

end

else if M(t,u)=reducere

then

begin

pd $\leftarrow$ Rest (pd)

p $\leftarrow$ p+1

end

else if M(t,u)=acceptare

then

begin

Scr (be)

ind $\leftarrow$ true

end

else

```
                begin
                  write  (‘eroare’)
                  ind ← true
                end
        end
```

S-au folosit următoarele notații:

bi = banda de intrare

be = banda de ieşire

pd = banda pushdown

Start = simbolul inițial

Vârf = procedură ce extrage simbolul din vârful benzii pd

Rest = procedură ce elimină simbolul din vârful benzii pd

Scr = procedură de scriere a conținutului benzii be

- = operația de concatenare

Teme propuse:

1. Implementați algoritmul PRIM k.
2. Implementați algoritmul σ.
3. Implementați algoritmul pentru testarea condiției LL k.
4. Implementați algoritmul pentru analiza sintactică LL k.
5. Implementați algoritmul de analiză sintactică LL 1.