

Curs 3

Ex Familile L_2, L_3 sunt închise la translatări finite.

T translator finit

$$L \in \mathcal{L}_2(\mathcal{L}_3) \Rightarrow T(L) \in \mathcal{L}_2(\mathcal{L}_3)$$

ANALIZA SINTACTICĂ

Sintaxa unui lb. de programare poate fi specificată ca o gram.
indep. de context.

Def: $G = (N, \Sigma, S, P)$ gram. undep. de context (CFG)

N : *terminalii*

Σ : nimb. terminale

SGN: vtb de start

$\Phi: A \rightarrow X \quad A \in N, \Phi \in (N \cup \Sigma)^*$

Dennote: $\pi \Rightarrow y$ ddac $\pi = \pi_1 \wedge \pi_2, y = \pi_1 \neq \pi_2 \quad A \rightarrow \exists \in P$

Diferențe în 0 sau mai mulți pași \Rightarrow
anul cel puțin un pas \Rightarrow

Limbajul generat de G:

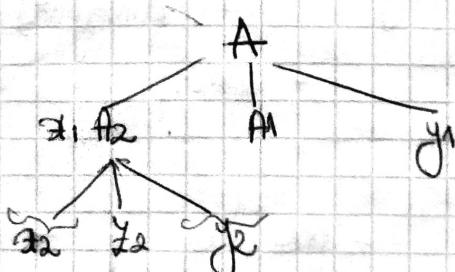
$$L(G) = \{w \in \Sigma^* \mid S \xrightarrow{*} w\}$$

Din varie astenii: la fiecare pas al derului este urmărit cel mai din astenii al formei ventriculare curiente.

Dérivat de dropt : — u — din dropt — u —

Arborele de derivare asociat unei derivații:

$$A \Rightarrow x_1 A_1 y_1 \Rightarrow x_1 z_1 y_1 = x_2 A_2 y_2 \Rightarrow x_2 z_2 y_2 = \dots \stackrel{*}{\Rightarrow}$$



Unei derivări. Măngi și corești un singur arbore de derivare.

Unui arbore să poată corespunde mai multe derivări.

Este $G = (N, \Sigma, S, P)$ CFG

$w \in \Sigma^*$, să analizăm după sintactică w :

- să decidă algoritmic dacă $w \in L(G)$

- dacă $w \in L(G)$, să furnizeze o derivare stângă (dreaptă) a lui w

Pentru un limbaj de programare s-

$G = (N, \Sigma, S, P)$ CFG care modelizează sintaxa limbajului respectiv?

N : categorii sintactice

Σ : categorii de tokene

P : reguli sintactice (pt. declarații, instr., expresii)

2 Automate stăru

Gramatica ambiguă:

$G = (N, \Sigma, S, P)$ CFG care este ambiguă dacă există $w \in L(G)$ astfel încât

a) w are 2 derivări stângi distincte

b) $w \xrightarrow{\text{ sau }} 2 \xrightarrow{\text{ sau }} \dots$ drepte - w

c) w are 2 arb. de deriv. distincte

Exemplu ① $E \xrightarrow{} E + E$

$E \xrightarrow{} a$

$a + a + a$

$E \xrightarrow{\text{ s }} E + E \xrightarrow{\text{ s }} E + E + E \Rightarrow a + E + E$

$E \xrightarrow{\text{ s }} E + E \xrightarrow{\text{ s }} a + E \Rightarrow a + E + E$

(gramatica echivalentă, care nu mai e ambiguă)

- anoc. stânga +

$E \xrightarrow{} E' + E$

$E' \xrightarrow{} E'$

$E' \xrightarrow{} a$

- anoc. dr. pt. + în parțial LR

$$② E \rightarrow E+E \mid E-E \mid E*E \mid E/E \mid m$$

$$E \rightarrow E+E'$$

$$\rightarrow E-E'$$

$$\rightarrow E'$$

$$E' \rightarrow E'*E''$$

$$E' \rightarrow E' \setminus E''$$

$$E' \rightarrow E''$$

$$E'' \rightarrow m$$

Sintaxă în forma Backus Naur

$\langle \text{nume_neterminál} \rangle$

$\gg = \longrightarrow$

| alternativă

()

{ }
" "

$\langle \text{expresie} \rangle ::= \langle \text{termen} \rangle + \langle \text{expresie} \rangle \mid \langle \text{termen} \rangle$

$\langle \text{termen} \rangle ::= \langle \text{factor} \rangle \times \langle \text{termen} \rangle \mid \langle \text{factor} \rangle$

$\langle \text{factor} \rangle ::= \langle \text{constanță} \rangle \mid \langle \text{identificator} \rangle \mid "(\text{expresie})"$

$\langle \text{constanță} \rangle ::= \langle \text{digit} \rangle \mid \langle \text{digit} \rangle \langle \text{constanță} \rangle$

$\langle \text{digit} \rangle = "0" \mid "1" \mid \dots \mid "9"$

} yacc
bișon - limbaj

Metode de analiză sintactică:

1) TOP-DOWN

- se pornește de la simbolul de start

- se aplică diferite producții, până când eventual se obține final analizat

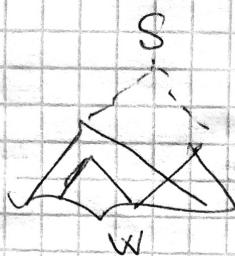
S



W - virul analizat (un program cursa)

2) BOTTOM-UP

- de la poartea de la virul analizat
- se fac reduceri urm. după un anumit alg. până când se obține o singură clăcătură de start.



METODA GENERALĂ TOP DOWN

1) Parcurgerea arborelui BF

- alg. poate fi aplicat p. V grame
- este f. lent, exponential (temp, spațiu) pt. căzările de

2) BF în adunări ustăngi

- alg. nu funcț. pt. gram. recursive de tip. A \Rightarrow A⁺

3) Parcurgerea arb. în echilibru, DF

push \$; X \leftarrow S; a = getNextToken();

backtrack()

while (true)

}

if (X este terminal)

}

if (X == ω și nu mai am produse pt. reanalizate) error();

else if (alege o producție

$\Rightarrow Y_1 \dots Y_k$)

push $Y_k \dots Y_1$

```

X ← pop();
}
else if ( x = $ & a = EOF )
    accept();
else if ( x = a )
    {
        a = getNextToken();
        x = pop(); }
    else
        backtrack();
}

```

Gramatici și climăjile de tip LL

Die $G = (N, \Sigma, S, P)$ CFG, $\alpha \in (NU\Sigma)^*$ $k \geq 1$ da.

$\text{First}_K(\alpha) = \{ w \in \Sigma^* \mid \exists \alpha \xrightarrow{*} w \alpha' \rightarrow |w| = K \text{ and } \alpha \xrightarrow{*} w, |w| < K \}$

- $L \subseteq (N \cup \Sigma)^*$, $\text{First}_K(L) = \bigcup_{x \in L} \text{First}_K(x)$
 - $\alpha \stackrel{*}{\Rightarrow} \beta$, $\text{First}_K(\beta) \subseteq \text{First}_K(\alpha)$
 - $K = 0$; $\text{First}_0(\alpha) = \{\alpha\}$

$$\text{Follow}_K(\alpha) = \{ w \in \Sigma^* \mid \exists S \stackrel{*}{\Rightarrow} S \alpha p, w \in \text{First}_K(p) \}$$

Def: Spunem că $G = (N, \Sigma, S, P)$ este gram. de tip.

$LL(n)$, $n \geq 1$ dat, dacă pt. $\neq 2$ derit. mărgine

$$S \xrightarrow[s]{*} wAd \xrightarrow[s]{*} wpd \xrightarrow[s]{*} wx$$

$$\qquad\qquad\qquad \xrightarrow[s]{*} wpd \xrightarrow[s]{*} wy$$

a.i. $\text{First}_h(x) = \text{First}_k(y) \rightarrow$ adunca $p = y^e$

Prop 1 ~~Def~~: Once gram. de tip LL(λ) este nuambiguă.

Dem: P_p īā $G = (N, \Sigma, S, P)$ gram LL(k) ambiguă, at.

$\exists \neq \in L(G)$ cu 2 derivații ast. distințe;

$$(2) : S \xrightarrow{wA\alpha} \underset{\in S}{\text{WPD}} \xrightarrow{w\beta\alpha} w\gamma = w\gamma'$$

(D₂): $S \xrightarrow{*} wA\alpha \Rightarrow w \not\models \alpha \Rightarrow \exists = w\bar{z}^n$
 cu $\beta \neq \emptyset$. Atunci $\text{First}_K(\bar{z}^1) = \text{First}_K(\bar{z}^n)$ pt. că
 $\bar{z}^1 = \bar{z}^n$. Decoarece G LL(k) nu. $\beta = \emptyset$ că

Prop₂: Dacă $G = (N, \Sigma, S, P)$ CFG fără nimic inutilizabil este LL(k) dacă și doar dacă următoarele sunt adevărate:
 1. $S \xrightarrow{*} wA\alpha$ pt. orice 2 prod. $A \rightarrow \beta$, $A \rightarrow \gamma \in P$
 $\beta \neq \emptyset$, $\text{First}_K(\beta\alpha) \cap \text{First}_K(\gamma\alpha) = \emptyset$

Def: Spunem că $G = (N, \Sigma, S, P)$ este LL(k) dacă, pentru orice 2 produse $A \rightarrow \beta$, $A \rightarrow \gamma \in P$, $\beta \neq \emptyset$ avem $\text{First}_K(\beta \cdot \text{Follow}_K(A)) \cap \text{First}_K(\gamma \cdot \text{Follow}_K(A)) = \emptyset$

Prop₃: Orice gram. fără nimic inutilizabil de tip LL(k) este de tip LL(2).

Ex1: $S \rightarrow aAb \mid aBaa$

$$A \rightarrow b \mid \lambda$$

este LL(2), dar nu este LL(1) dacă

Ex2: $S \rightarrow aBaa \mid aBb \mid a$

$$A \rightarrow b \mid \lambda$$

$B \rightarrow c$ este LL(1) dacă

Prop₄: Orice CFG $G = (N, \Sigma, S, P)$ fără nimic inutilizabil este LL(1) dacă este LL(1) dacă.