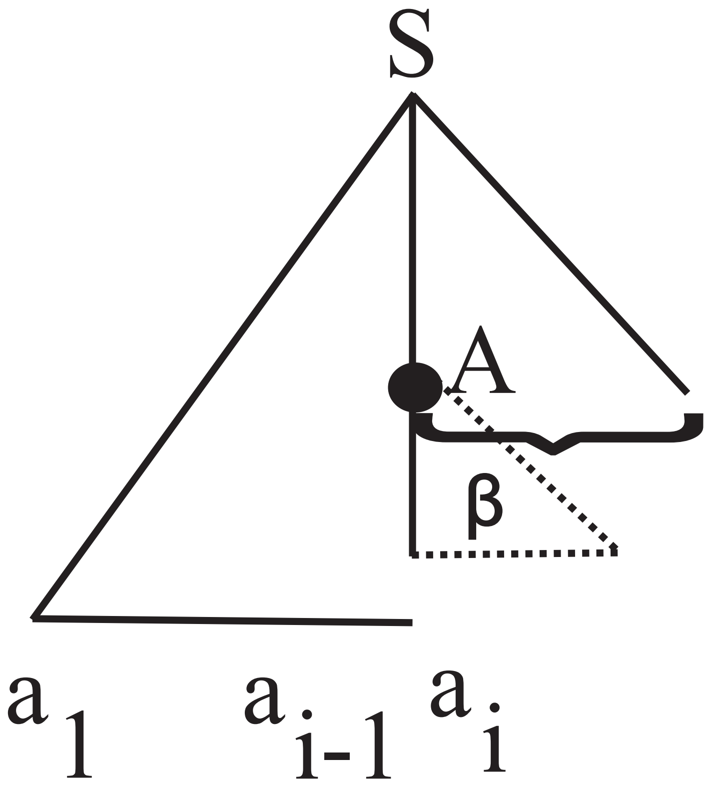
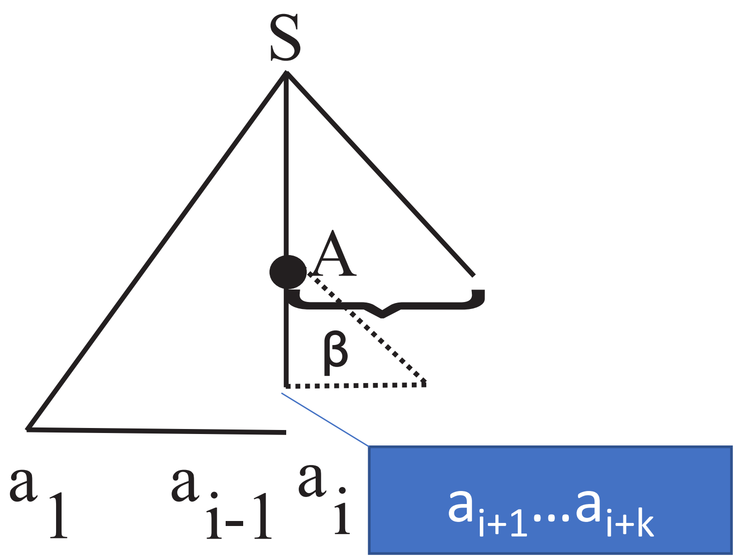
Analizor sintactic LL(1)



Algoritm liniar

LL(k)

- L = left (secvența este parcursă de la stânga la dreapta
- L = left (se folosesc derivări de stânga)
- Predicția are lungimea **k**



Principiu LL(k)

- In orice moment al analizei, <u>acțiunea este unic determinată</u> de:
- Partea închisă ($a_1 \ldots a_i$)
- Simbolul curent A
- Predicția $a_{i+1} \ldots a_{i+k}$ (lungime k)

FIRST_k

- ≈ primele k simboluri terminale care se pot genera din α
- Definiție:

$$FIRST_k : (N \cup \Sigma)^* \rightarrow \mathcal{P}(\Sigma^k)$$
$$FIRST_k(\alpha) = \{u | u \in \Sigma^k, \alpha \stackrel{*}{\Rightarrow} ux, |u| = k \text{ sau } \alpha \stackrel{*}{\Rightarrow} u, |u| \leq k\}$$

Definiție

- *O gic este de tip LL(k) dacă pentru oricare două derivări de stânga:*

$$1.\ S \overset{*}{\Rightarrow}_{st} wA\alpha \Rightarrow_{st} w\beta\alpha \overset{*}{\Rightarrow}_{st} wx;$$

$$2.\ S \overset{*}{\Rightarrow}_{st} wA\alpha \Rightarrow_{st} w\gamma\alpha \overset{*}{\Rightarrow}_{st} wy;$$

astfel încât $FIRST_k(x) = FIRST_k(y)$ avem că: $\beta = \gamma$.

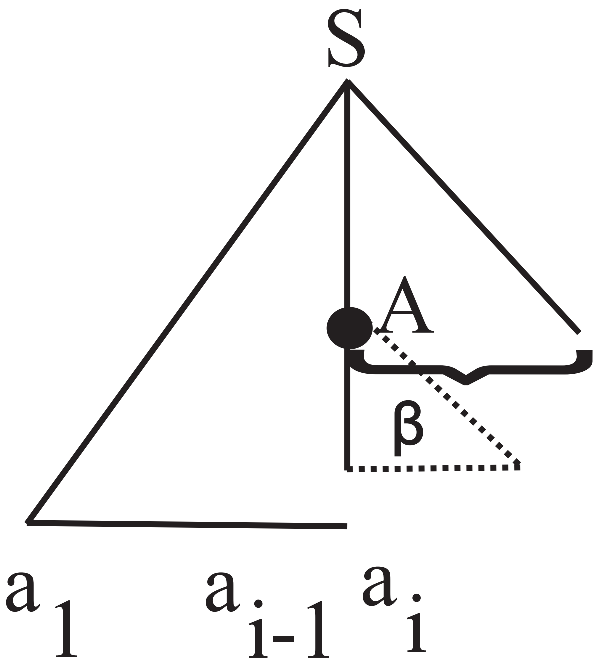
Teoremă

- *Condiția necesară și suficientă pentru ca o gramatică să fie de tip LL(k) este ca pentru orice pereche de producții distincte ale aceluiași neterminal ($A\rightarrow \beta$, $A\rightarrow\gamma, \beta\neq\gamma$) să fie verificată condiția:*

$$\mathrm{FIRST}_k(\beta\alpha) \cap \mathrm{FIRST}_k(\gamma\alpha) = \Phi, \forall\, \alpha \text{ astfel încât } S \overset{*}{\Rightarrow} uA\alpha$$

FOLLOW

$A \rightarrow \boldsymbol{\varepsilon}$



- FOLLOW$_k$(A)≈ următoarele k simboluri care se generează / urmează după A

$$FOLLOW : (N \cup \Sigma)^* \to \mathcal{P}(\Sigma)$$
$$FOLLOW(\beta) = \{w \in \Sigma | S \stackrel{*}{\Rightarrow} \alpha\beta\gamma, w \in FIRST(\gamma)\}$$

Analizor sintactic LL(1)

- Predicția de lungime 1

- Pași:
  1) construire FIRST, FOLLOW
  2) Construire tabel de analiză LL(1)
  3) Analiza secvenței de baza tranzițiilor între configurații

Se execută 1 dată

Teorema *O gramatică este de tip LL(1) dacă şi numai dacă pentru fiecare neterminal A cu producţiile* $A \rightarrow \alpha_1 | \alpha_2 | \ldots | \alpha_n$, $FIRST_k(\alpha_i) \cap FIRST_k(\alpha_j) = \emptyset$ *şi dacă* $\alpha_i \stackrel{*}{\Rightarrow} \epsilon$, $FIRST(\alpha_i) \cap FOLLOW(A) = \emptyset, \forall i, j = \overline{1, n}, i \neq j$.

Construcție FIRST

Concatenare de lungime 1

- $FIRST_1$ notat FIRST
- Observații:
  - Dacă L_1, L_2 sunt două limbaje peste alfabetul Σ, atunci: $L_1 \oplus L_2 = \{w | x \in L_1, y \in L_2, xy = w, |w| \leq 1 \text{ sau } xy = wz, |w| = 1\}$ şi
  - $FIRST(\alpha\beta) = FIRST(\alpha) \oplus FIRST(\beta)$

    $FIRST(X_1 \ldots X_n) = FIRST(X_1) \oplus \ldots \oplus FIRST(X_n)$

Algoritmul 3.3 FIRST

INPUT: G

OUTPUT: $FIRST(X), \forall X \in N \cup \Sigma$

for $\forall a \in \Sigma$ **do**

  $F_i(a) = \{a\}, \forall i \geq 0$

end for

i := 0;

$F_0(A) = \{x | x \in \Sigma, A \to x\alpha$ sau $A \to x \in P\}$; {*inițializare*}

repeat

  i := i+1;

  for $\forall X \in N$ **do**

    if F_{i-1} au fost calculate $\forall X \in N \cup \Sigma$ **then**

      {*dacă* $\exists Y_j, F_{i-1}(Y_j) = \emptyset$ *atunci nu se poate aplica*}

      $F_i(A) = F_{i-1}(A) \cup$

      $\{x | A \to Y_1 \ldots Y_n \in P, x \in F_{i-1}(Y_1) \oplus \ldots \oplus F_{i-1}(Y_n)\}$

    end if

  end for

until $F_{i-1}(A) = F_i(A)$

$FIRST(X) := F_i(X), \forall X \in N \cup \Sigma$

Algoritmul 3.4 FOLLOW

INPUT: G, FIRST(X), $\forall X \in N \cup \Sigma$

OUTPUT: FOLLOW(X), $\forall X \in N \cup \Sigma$

$F(X) = \emptyset, \forall X \in N - \{S\}$; {*inițializare*}

$F(S) = \{\epsilon\}$; {*corespunzător simbolului \$ folosit în analiză*}

repeat

  for $B \in N$ **do**

    for $A \to \alpha B y \in P$ **do**

      if $\epsilon \in FIRST(y)$ **then**

        $F'(B) = F(B) \cup F(A)$;

        $F'(B) = F(B) \cup FIRST(y)$

      end if

    end for

  end for

until $F'(X) = F(X), \forall X \in N$

$FOLLOW(X) = F(X), \forall X \in N.$

Construcție tabel de analiză LL(1)

- Acțiuni posibile în funcție de:
  - Simbolul curent $\in \mathbf{N} \cup \boldsymbol{\Sigma}$
  - Predicția posibilă $\in \boldsymbol{\Sigma}$
- Se adaugă un caracter special “\$” ($\notin \mathbf{N} \cup \boldsymbol{\Sigma}$) – marcaj de “stivă vidă”

= > tabel:

- Câte o linie pentru fiecare simbol $\in \mathbf{N} \cup \boldsymbol{\Sigma} \cup \{\$\}$
- Câte o coloană pentru fiecare simbol $\in \boldsymbol{\Sigma} \cup \{\$\}$

Reguli tabel LL(1)

1. $M(A,a) = (\alpha, i), \forall a \in FIRST(\alpha), a \neq \epsilon, A \rightarrow \alpha$ producţie în P cu numărul i;
   $M(A,b) = (\alpha, i)$, dacă $\epsilon \in FIRST(\alpha), \forall b \in FOLLOW(A), A \rightarrow \alpha$ producţie în P cu numărul i;

2. $M(a,a) = pop, \forall a \in \Sigma;$

3. $M(\$,\$) = acc;$

4. M(x,a)=err (eroare) în celelalte cazuri.

Observație

O <u>gramatică este de tip LL(1)</u> dacă tabelul de analiză LL(1) **nu** conține conflicte (nu există mai mult de o valoare într-o celulă de tabel M(A,a)

Definire configurații și tranziții

- INPUT:
  - Gramatica limbajului G = (N, Σ, P,S)
  - Tabel de analiză LL(1)
  - Secvența de analizat w =$a_1 \ldots a_n$
- OUTPUT:

  Dacă (w $\in$ L(G)) *atunci* **șir de producții**
  altfel **locația erorii**

Configurații LL(1)

$$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\pi})$$

Unde:

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

Configurația inițială:
(w\$,S\$,ε)

Configurația finală:
(\$, \$, π)

Tranziții

1. Push – punere în stivă

$$(ux, A\alpha\$, \pi) \vdash (ux, \beta\alpha\$, \pi i), \text{ dacă } M(A,u) = (\beta, i);$$

(se scoate A și se pun simbolurile din β)

2. Pop – scoatere din stivă (din ambele stive)

$$(ux, a\alpha\$, \pi) \vdash (x, \alpha\$, \pi), \text{ dacă M(a,u)=pop}$$

3. Acceptare

$$(\$, \$, \pi) \vdash acc$$

4. Eroare - altfel

Algoritml de analiză sintactică LL(1)

- aici