Formális nyelvek és fordítóprogramok

10. Felülről-lefelé haladó elemzés. $LL(1)$ elemzések

Bodó Zalán

Babeș–Bolyai Tudományegyetem
Matematika és Informatika Kar
Magyar Matematika és Informatika Intézet

$LL(k)$ nyelvtanok

- balról-jobbra haladunk a terminális szimbólumok olvasásával
- fentről-lefelé haladó elemzés: a kezdőszimbólumból kiindulva próbáljuk meg elérni a szintaxisfa leveleit (a programot)
- ★ $LL =$ „left to right, tracing a leftmost derivation” (Knuth, 1971)
- ★ $k = k$ db. szimbólumot olvasunk előre

Def. Legbaloldalibb helyettesítés

Ha $(A \to \alpha) \in P$, akkor az $xA\beta$ mondatforma,
$x \in T^*, \alpha, \beta \in (N \cup T)^*$ **legbaloldalibb helyettesítése** $x\alpha\beta$, azaz

$$xA\beta \underset{\text{legbal}}{\Rightarrow} x\alpha\beta$$

Def. Legbaloldalibb levezetés

Ha az $S \overset{*}{\Rightarrow} x$, $x \in T^*$, levezetésben minden helyettesítés legbalodalibb, akkor ezt a levezetést **legbaloldalibb levezetésnek** nevezzük.

- a felülről-lefelé levezetés a legbaloldalibb levezetésnek felel meg
- szintaktikai elemzés: megpróbálunk legbaloldalibb helyettesítéseket végezve eljutni az elemzendő szöveghez
- a köv. triviális tételt használjuk

Tétel

Ha $S \overset{*}{\Rightarrow} x\alpha \overset{*}{\Rightarrow} yz$, $\alpha \in (N \cup T)^*, x, y, z \in T^*$, és $|x| = |y|$, akkor $x = y$.

- $=$ egy mondatforma bal oldalán levő terminálisok sorozatát egy környezetfüggetlen grammatika szabályai nem változtatják meg
- ha egy levezetésben a bal oldali terminálisok nem egyeznek meg az elemzendő szövegben levő terminálisokkal $\Rightarrow$ rossz irányba halad az elemzés
- legegyszerűbb de nagyon erőforrásigényes megoldás: **visszalépéses elemzés**
- $LL(k)$ nyelvtanok használata: a helyettesítés meghatározható ha ismerjük a köv. k darab terminálist

Tulajdonság

Ha az $S \overset{*}{\Rightarrow} wx$, $w, x \in T^*$, legbaloldalibb helyettesítés építésekor eljutunk az $S \overset{*}{\Rightarrow} wA\beta$ mondatformáig, $A \in N$, $\beta \in (N \cup T)^*$, és az $A\beta \overset{*}{\Rightarrow} x$-hez akarunk jutni, akkor az $A \to \alpha$ helyettesítés egyértelműen meghatározható ha ismerjük x első k darab szimbólumát.

Def.

Legyen $FIRST_k(\alpha)$, $k \geq 0, \alpha \in (N \cup T)^*$, az α-ból levezethető szimbólumsorozatok k hosszúságú terminális sorozatainak halmaza:

$$FIRST_k(\alpha) = \{x \mid \alpha \overset{*}{\Rightarrow} x\beta, |x| = k\} \cup \{x \mid \alpha \overset{*}{\Rightarrow} x, |x| < k\}, \quad x \in T^*, \beta \in (N \cup T)^*$$

Def.

A $G = (N, T, P, S)$ nyelvtan $LL(k)$ nyelvtan, $k \geq 0$, ha bármely két

$$S \overset{*}{\Rightarrow} wA\beta \Rightarrow w\alpha_1\beta \overset{*}{\Rightarrow} wx,$$
$$S \overset{*}{\Rightarrow} wA\beta \Rightarrow w\alpha_2\beta \overset{*}{\Rightarrow} wy,$$

$A \in N, x, y, w \in T^*, \alpha_1, \alpha_2 \in (N \cup T)^*$, levezetésre

$$FIRST_k(x) = FIRST_k(y)$$

esetén

$$\alpha_1 = \alpha_2.$$

- ha $LL(k_0)$, akkor $\forall k > k_0 : LL(k)$
- a legkisebb k-t értjük, amelyre az értelmezésben leírtak igazak

1. Példa

Legyen a köv. helyettesítési szabályok által értelmezett G_1 grammatika:

$$\begin{aligned} S &\rightarrow aS \mid bA \\ A &\rightarrow aA \mid \varepsilon \end{aligned}$$

$(L(G_1) = \{a^m b a^n \mid m, n \geq 0\})$
Könnyen észrevehető, hogy 1 szimbólum előreolvasásával meghatározható, hogy melyik szabályt kell alkalmazni; ezért a grammatika $LL(1)$-es. (Ha a köv. szimbólum a, akkor az $S \rightarrow aS$-t, ha b, akkor az $S \rightarrow bA$-t; ha már átmentünk A-ba, akkor ha a köv. szimbólum a, akkor az $A \rightarrow aA$-t, ha a szimbólumsorozat végén vagyunk, akkor az $A \rightarrow \varepsilon$-t.)

2. Példa

Legyen a köv. helyettesítési szabályok által értelmezett G_2 grammatika:

$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow aAb \mid ab \\ B &\rightarrow aBc \mid ac \end{aligned}$$

$(L(G_2) = \{a^n b^n, a^n c^n \mid n \geq 1\})$
Nem eldönthető, hogy $S \rightarrow A$ vagy $S \rightarrow B$-t kell alkalmazni; pl. $a^{k+1}b^{k+1}$ esetében. Ezért a grammatika nem $LL(k)$ grammatika.

Tétel

A G nyelvtan akkor és csak akkor $LL(k)$ nyelvtan, ha minden

$$S \overset{*}{\Rightarrow} wA\beta \text{ és } A \to \gamma|\delta,$$
$$\gamma \neq \delta, w \in T^*, A \in N, \beta, \gamma, \delta \in (N \cup T)^*$$

esetén

$$FIRST_k(\gamma\beta) \cap FIRST_k(\delta\beta) = \emptyset.$$

- ha a grammatika nem erős $LL(k)$ (lásd a köv. slideokon!), akkor a **kontextus** is számít, azaz, a nemterminális előtt (és után) megjelenő szimbólumok is számítanak
- **jó**, azaz **hatékony** vizsgálati módszer csak az $LL(1)$ grammatikákhoz adható
  (illetve az erős $LL(k)$ grammatikákhoz; lásd a 8–9. példákat)

Def.

$$FOLLOW_k(\beta) = \{x \mid S \overset{*}{\Rightarrow} \alpha\beta\gamma, x \in FIRST_k(\gamma)\}.$$

Ha $\varepsilon \in FOLLOW_k(\beta)$, akkor
$FOLLOW_k(\beta) = FOLLOW_k(\beta) \setminus \{\varepsilon\} \cup \{\#\}$, $\alpha, \beta, \gamma \in (N \cup T)^*$, $x \in T^*$.

- $\#$ a program végét jelenti.
- $FOLLOW_1(A)$: $S \overset{*}{\Rightarrow} \alpha A\gamma \overset{*}{\Rightarrow} \alpha Aw$ az A után közvetlenül álló terminálisok

Tétel

A G nyelvtan akkor és csak akkor $LL(1)$-es, ha $\forall A \in N$ minden $A \to \gamma \mid \delta$ helyettesítési szabályára fennáll

$$FIRST_1(\gamma FOLLOW_1(A)) \cap FIRST_1(\delta FOLLOW_1(A)) = \emptyset$$

- $FIRST_1(\cdot) =: FIRST(\cdot)$, $FOLLOW_1(\cdot) =: FOLLOW(\cdot)$

3. Példa

Legyen a köv. helyettesítési szabályok által értelmezett G_1 grammatika (1. Példa):

$$\begin{aligned} S &\rightarrow aS \mid bA \\ A &\rightarrow aA \mid \varepsilon \end{aligned}$$

$(L(G_1) = \{a^m b a^n \mid m, n \geq 0\})$

$FOLLOW(S) = \{\#\}$
$FOLLOW(A) = \{\#\}$

$FIRST(aS\#) \cap FIRST(bA\#) = \{a\} \cap \{b\} = \emptyset$ (OK)
$FIRST(aA\#) \cap FIRST(\#) = \{a\} \cap \{\#\} = \emptyset$ (OK)
$\Rightarrow LL(1)$-es

- a tétel nem érvényes általános ($k \geq 0$) esetben, csak $k = 1$-re
- azokat a grammatikákat, amelyekre igaz a tétel általánosított alakja, erős $LL(k)$-es grammatikáknak nevezzük

Def.

Egy G grammatikát **erős $LL(k)$ grammatikának** nevezünk ($k \geq 0$), ha tetszőleges

$$S \overset{*}{\Rightarrow} wA\beta \Rightarrow w\alpha_1\beta \overset{*}{\Rightarrow} wx,$$
$$S \overset{*}{\Rightarrow} vA\theta \Rightarrow v\alpha_2\theta \overset{*}{\Rightarrow} vy,$$

levezetésekre $FIRST_k(x) = FIRST_k(y)$ esetén $\alpha_1 = \alpha_2$.

Megjegyzés: erős $LL(k) \Rightarrow LL(k)$

Tétel

Egy G grammatika akkor és csak akkor erős $LL(k)$ grammatika, ha minden $A \to \gamma \mid \delta$ helyettesítési szabályra fennál, hogy

$$FIRST_k(\gamma FOLLOW_k(A)) \cap FIRST_k(\delta FOLLOW_k(A)) = \emptyset$$

4. Példa

Tekintsük a következő (szabályhalmazzal rendelkező) grammatikát:

$$\begin{aligned} S &\rightarrow aAaa \mid bAba \\ A &\rightarrow b \mid \varepsilon \end{aligned}$$

Ez $LL(2)$ grammatika, mivel:

$a\underline{b}aa$ $a\bar{A}aa$, vagyis $A \rightarrow b$	$a\underline{a}a$ $a\bar{A}aa$, vagyis $A \rightarrow \varepsilon$	$b\underline{b}ba$ $b\bar{A}ba$, vagyis nem tudjuk	$b\underline{b}a$ $b\bar{A}ba$, vagyis nem tudjuk

$\Rightarrow$ nem tudjuk 1 elem előreolvasásával eldönteni; 2-vel már igen, mivel ezek különbözőek a két úton: ba és aa, illetve bb és ba. Ezt beláthatjuk a definíció segítségével is:

$$\begin{aligned} S &\Rightarrow aAaa \Rightarrow a\underline{b}aa \\ S &\Rightarrow aAaa \Rightarrow a\underline{a}a \end{aligned}$$

azaz ebben az esetben elegendő egy szimbólum előreolvasása. Viszont: ▷

$$S \Rightarrow bAba \Rightarrow b\underline{bb}a$$
$$S \Rightarrow bAba \Rightarrow b\underline{ba}$$

vagyis itt már 2 szimbólumot kell ismernünk.
Az erős $LL(k)$ grammatika definíciója értelmében a grammatika nem erős $LL(2)$ grammatika (csak $k = 3$-ra lesz erős), mivel:

$$S \Rightarrow aAaa \Rightarrow a\underline{ba}a$$
$$S \Rightarrow bAba \Rightarrow b\underline{ba}$$

Ennek eldöntéséhez használhatjuk a tételt is: látható, hogy $FIRST_2(bFOLLOW_2(A)) = \{ba, bb\}$ és $FOLLOW_2(A) = \{aa, ba\}$, metszetük nem üreshalmaz, vagyis a grammatika nem erős $LL(2)$.

Megjegyzés

Az $LL(k)$ grammatikáknál a szabály alkalmazása függ a kontextustól (= a bal oldalon szereplő terminálisoktól) **is**, az erős $LL(k)$ nyelvtanok esetén viszont **csak** a következő k terminálistól.

Tétel

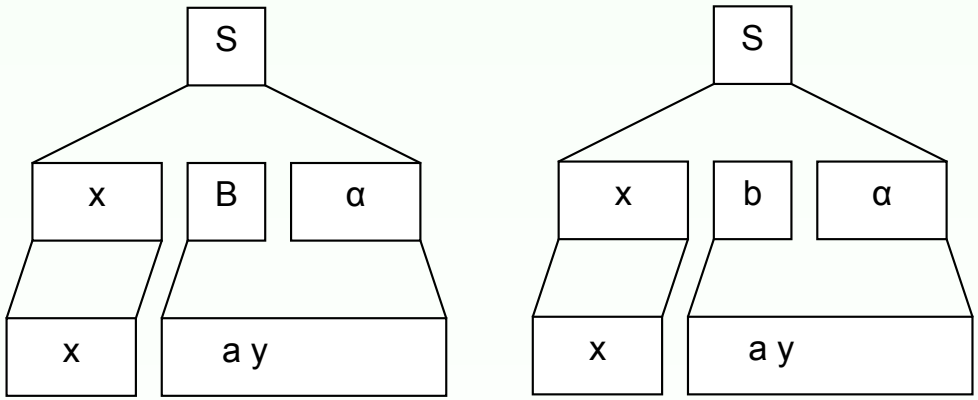
Erős $LL(1) \Leftrightarrow LL(1)$

Tétel (Rosenkrantz & Stearns, 1970)

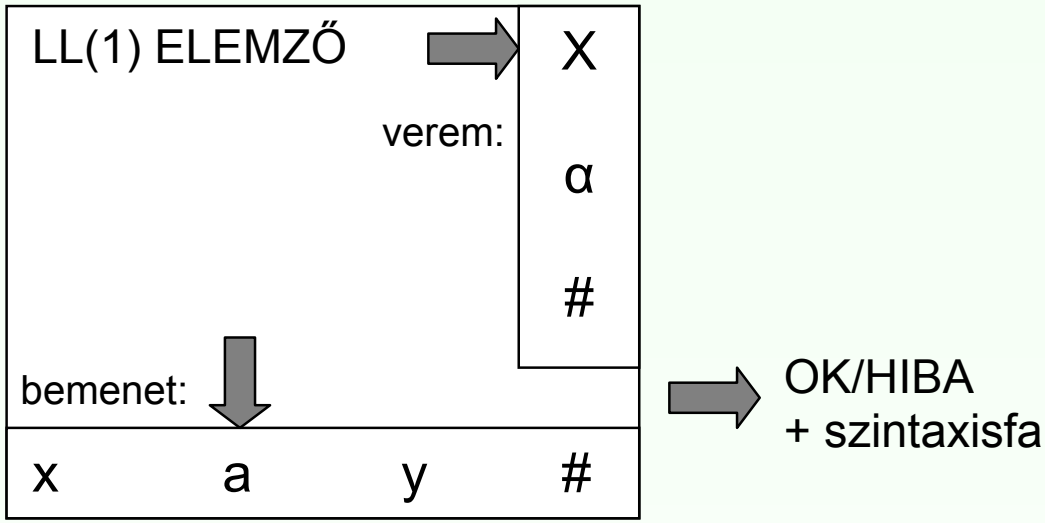
Minden $LL(k)$ grammatikához létezik egy vele ekvivalens erős $LL(k)$ grammatika.

$LL(1)$ táblázatos elemzés

- balról jobbara olvasunk, felülről lefelé haladva elemzünk, legbaloldalibb helyettesítéseket alkalmazunk
- az elemzendő mondatforma: $xB\alpha$ vagy $xb\alpha$ alakú, ahol $x \in T^*$, $B \in N$, $b \in T$, $\alpha \in (N \cup T)^*$
- köv. lépés: B helyettesítése vagy b ellenőrzése
  1. ha az elemzendő szöveg következő karaktere a, akkor $B \to \beta$ alkalmazása, ahol $a \in FIRST(\beta FOLLOW(B))$ (ha a nyelvtan $LL(1)$-es, akkor vagy egy db. ilyen van vagy egy sem); ha van ilyen szabály, akkor alkalmazzuk, ha nincs, akkor ez **szintaktikai hiba**
  2. ha az elemzendő szöveg következő karaktere a és $b = a$, akkor helyes és továbblépünk; ellenkező esetben **szintaktikai hibát** jelzünk



Az elemző



Az $LL(1)$-es táblázatos elemző sematikus rajza

- az **elemzés** egy **állapota**: $(ay\#, X\alpha\#, v)$, ahol:
  - $ay\#$ az elemzendő szöveg még nem elemzett része
  - $X\alpha\#$ az elemzés mondatformájának még nem elemzett része (a verem tartalma)
  - v az alkalmazott szabályok sorszámait tartalmazó lista (ebből épül majd a szintaxisfa)
  - a az aktuális szimbólum

- a kezdőállapot: $(x\#, S\#, \varepsilon)$
- a végállapot: $(\#, \#, w)$
- T **elemző táblázat** definiálása:

$$T[X, a] = \begin{cases} (\beta, i), & \text{ha } X \to \beta \text{ az } i\text{-edik szabály,} \\ & a \in \mathit{FIRST}(\beta) \text{ vagy} \\ & (\varepsilon \in \mathit{FIRST}(\beta) \text{ és } a \in \mathit{FOLLOW}(X)) \\ \mathit{pop}, & \text{ha } X = a, \\ \mathit{elfogad}, & \text{ha } X = \# \text{ és } a = \#, \\ \mathit{hiba}, & \text{különben.} \end{cases}$$

- az állapotátmenetek:

$$(ay\#, X\alpha\#, v) \to \begin{cases} (ay\#, \beta\alpha\#, vi), & \text{ha } T[X, a] = (\beta, i), \\ (y\#, \alpha\#, v), & \text{ha } T[X, a] = \mathit{pop}, \\ \mathit{OK}, & \text{ha } T[X, a] = \mathit{elfogad}, \\ \mathit{HIBA}, & \text{ha } T[X, a] = \mathit{hiba}. \end{cases}$$

ALG 1 $LL(1)$ táblázatos elemzés

1: $s = (xay\#, S\#, \varepsilon)$, $s' = elemez$
2: **repeat**
3: **if** $s = (ay\#, A\alpha\#, v)$ és $T[A, a] = (\beta, i)$ **then**
4: $s = (ay\#, \beta\alpha\#, vi)$
5: **else if** $s = (ay\#, a\alpha\#, v)$ **then**
6: $s = (y\#, \alpha\#, v)$
7: **else if** $s = (\#, \#, v)$ **then**
8: $s' = OK$
9: **else**
10: $s' = HIBA$
11: **end if**
12: **until** $s' = OK$ vagy $s' = HIBA$

5. Példa

Tekintsük a következő egyszerű grammatikát:

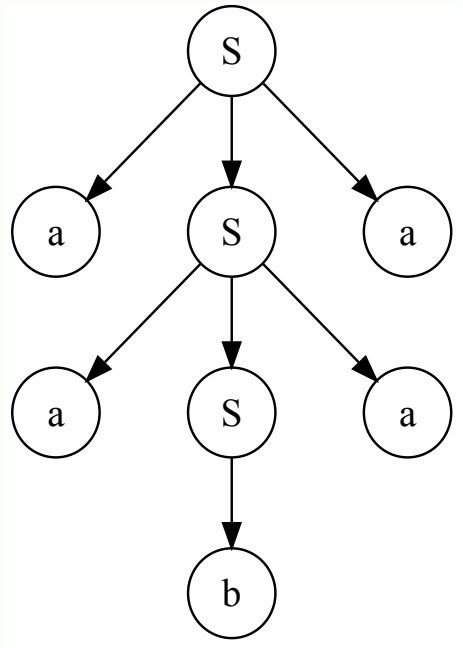
$$S \quad \rightarrow \quad b \mid aSa$$

Mivel nincs benne ε-os szabály, ezért elegendő csak a $FIRST$-halmazokat vizsgálni:
$FIRST(b) \cap FIRST(aSa) = \{b\} \cap \{a\} = \emptyset \Rightarrow LL(1)$
Az elemző táblázat:

	a	b	$\#$
S	$(aSa, 2)$	$(b, 1)$	
a	pop		
b		pop	
$\#$			$elfogad$

Próba: $aabaa$
$(aabaa\#, S\#, \varepsilon) \Rightarrow (aabaa\#, aSa\#, 2) \overset{pop}{\Rightarrow} (abaa\#, Sa\#, 2) \Rightarrow$
$(abaa\#, aSaa\#, 22) \overset{pop}{\Rightarrow} (baa\#, Saa\#, 22) \Rightarrow (baa\#, baa\#, 221)$
$\overset{pop}{\Rightarrow} (aa\#, aa\#, 221) \overset{pop}{\Rightarrow} (a\#, a\#, 221) \overset{pop}{\Rightarrow} (\#, \#, 221) \overset{elfogad}{\Rightarrow}$ OK

A szintaxisfa:



Próba: *ba*

$(ba\#, S\#, \varepsilon) \Rightarrow (ba\#, b\#, 1) \overset{pop}{\Rightarrow} (a\#, \#, 1) \overset{hiba}{\Rightarrow}$ HIBA

6. Példa

Tekintsük a következő grammatikát:

$$\begin{aligned} S &\rightarrow aSb \mid \varepsilon \mid cA \\ A &\rightarrow cA \mid \varepsilon \end{aligned}$$

Vizsgáljuk meg, hogy a grammatika $LL(1)$-es-e?
Ha igen, építsük meg az elemző táblázatot, majd próbáljuk ki az elemzőt az *aabb* szóra (építsünk szintaxisfát is).

7. Példa

$$\begin{aligned} S &\rightarrow aS \mid bA \\ A &\rightarrow aA \mid \varepsilon \end{aligned}$$

8. Példa

Tekintsük a köv. szabályhalmazzal rendelkező grammatikát:

$$\begin{aligned} S &\rightarrow A + S \mid A \\ A &\rightarrow 0 \mid 1 \end{aligned}$$

A grammatika $LL(2)$ és erős $LL(2)$ típusú, mivel:

$$FIRST_2(A + S\ FOLLOW_2(S)) \cap FIRST_2(A\ FOLLOW_2(S)) = \{0+, 1+\} \cap \{0\#, 1\#\} = \emptyset$$

Így felírható a következő elemző táblázat:

	$0+$	$1+$	$0\#$	$1\#$
S	$(A + S, 1)$	$(A + S, 1)$	$(A, 2)$	$(A, 2)$
A	$(0, 3)$	$(1, 4)$	$(0, 3)$	$(1, 4)$

Végezzük el $0 + 0 + 1$ elemzését.

9. Példa

Tekintsük a köv. szabályhalmazzal rendelkező grammatikát (4. Példa):

$$\begin{aligned} S &\rightarrow aAaa \mid bAba \\ A &\rightarrow b \mid \varepsilon \end{aligned}$$

Láttuk, hogy a grammatika $LL(2)$, de nem erős $LL(2)$ típusú:

$$FIRST_2(aAaa\ FOLLOW_2(S)) \cap FIRST_2(bAba\ FOLLOW_2(S)) = \{ab, aa\} \cap \{bb, bb\} = \emptyset$$

de

$$FIRST_2(b\ FOLLOW_2(A)) \cap FIRST_2(\ FOLLOW_2(A)) = \{ba, bb\} \cap \{aa, ba\} = \{ba\}$$

▷

Ha felírjuk az elemző táblázatot, konfliktushoz jutunk:

	ab	aa	bb	ba
S	$(aAaa, 1)$	$(aAaa, 1)$	$(bAba, 2)$	$(bAba, 2)$
A			$(b, 3)$	$(\mathbf{b}, \mathbf{3})/(\varepsilon, \mathbf{4})$

Mivel,

- ha az $a\mathbf{baa}$ mondatot akarjuk elemezni, és elérünk az $a\mathbf{Aaa}$ mondatformáig, akkor az $A \to b$ helyettesítést kell elvégezzük;
- ha viszont a $b\mathbf{ba}$ mondatot akarjuk elemezni, és elérünk a $b\mathbf{Aba}$ mondatformáig, akkor az $A \to \varepsilon$ helyettesítést kell elvégezzük.

$\Rightarrow$ számít a kontextus, figyelembe kell azt is venni!

Konfliktusok kezelése

Kétféle konfliktus:

1. *FIRST*/*FIRST* konfliktus – a *FIRST* halmazok metszete nem üres
2. *FIRST*/*FOLLOW* konfliktus – van ε-szabály és valamely *FIRST* és a *FOLLOW* halmaz metszete nem üres

Konfliktusok kiküszöbölése:

1. **balfaktorizálás**:

$$A \rightarrow X \mid XYZ \quad \Rightarrow \quad \begin{array}{lcl} A & \rightarrow & XB \\ B & \rightarrow & \varepsilon \mid YZ \end{array}$$

2. **balrekurzió kiküszöbölése**: általában olyan új/módosított szabályok bevezetésével, melyek ugyanazt a (rész)nyelvet generálják;
   A balrekurziót tartalmazó szabály – ha több jobb oldal tartozik ugyanahhoz a nemterminálishoz, azaz több alternatíva közül választhatunk – minden alternatívára konfliktushoz vezet:

$$E \rightarrow E + i \mid alt_1 \mid alt_2$$

Példa megoldásra:

$$E \rightarrow E + i \mid i \quad \Rightarrow \quad \begin{array}{lll} E & \rightarrow & iF \\ F & \rightarrow & \varepsilon \mid +iF \end{array}$$

3. **behelyettesítések**: a konfliktusok sokszor megoldhatók a szabályok egymásba való behelyettesítései által; sokszor viszont *FIRST*/*FIRST* konfliktusokhoz vezethetnek.
   Például:

$$\begin{array}{lll} S & \rightarrow & Aab \\ A & \rightarrow & a \mid \varepsilon \end{array} \quad \Rightarrow \quad S \rightarrow aab \mid ab \quad \Rightarrow \quad \begin{array}{lll} S & \rightarrow & aA \\ A & \rightarrow & ab \mid b \end{array}$$

A rekurzív leszállás módszere

- nem építünk táblázatot
- rekurzív függvényhívásokon keresztül valósítjuk meg az elemzést
- minden (bal oldalon szereplő) nemterminálishoz egy eljárást/függvényt írunk

A `Vizsgal` eljárás: a *pop* műveletet valósítja meg:

```
Procedure Vizsgal(a)
    if (aktualis_szimbolum == a)
        aktualis_szimbolum = kov_szimbolum();
    else
        Hibajelzes();
EndProcedure
```

ahol

- az `aktualis_szimbolum` az elemzendő szimbólumsorozat aktuális karakterét tartalmazza
- `kov_szimbolum()` visszadja a következő bemeneti szimbólumot
- `Hibajelzes()` valamilyen módon kijelzi a szintaktikai hibát

Minden A nemterminálishoz létrehozunk egy eljárást:

```
Procedure A()
    ...
EndProcedure
```

A pontok helyére a szabály jobb oldalának függvényében írjuk az utasításokat.

(1) Ha csak egy helyettesítési szabály van, akkor

1. az $A \to a$ szabályhoz rendelt blokk a `Vizsgal(a);`
2. az $A \to B$ szabályhoz rendelt blokk a `B;`
3. az $A \to X_1 X_2 \dots X_n$ szabályhoz rendelt blokk:

```
T(X_1);
T(X_2);
...
T(X_n);
```

ahol `T(X_i)` a kifejtett programblokkot jelenti

(2) Ha az A-ra több helyettesítési szabály van:
$A \to \alpha_1|\alpha_2|\ldots|\alpha_n|e$, akkor a hozzá rendelt blokk:

```
Case (aktualis_szimbolum)
    FIRST(alpha_1): T(alpha_1);
    FIRST(alpha_2): T(alpha_2);
    ...
    FIRST(alpha_n): T(alpha_n);
    FIRST(e): T(e);
    FOLLOW(A): nop; //e
EndCase
```

ahol a `T(alpha_i)` az `alpha_i`-hez tartozó kifejtett programblokkot jelenti (lásd az (1) pontot), e-re pedig fennáll, hogy $e \stackrel{*}{\Rightarrow} \varepsilon$, vagyis $\varepsilon \in \mathit{FIRST}(e)$ (a többi α_i-re nem);
a $\mathit{FIRST}(e)$-ben nincs benne az ε

10. Példa

Tekintsük a következő egyszerű grammatikát (5. Példa):

$$S \quad \rightarrow \quad b \mid aSa$$

Mivel csak egyetlen nemterminálisunk van, ezért a `Vizsgal` eljáráson kívül csak az `S` eljárást kell megírnunk. Ez a következőképpen fog kinézni:

```
Procedure S()
    Case (aktualis_szimbolum)
        {b}: Vizsgal(b);
        {a}: Vizsgal(a); S(); Vizsgal(a);
    EndCase
EndProcedure
```

A grammatika kiterjeszthető egy új S' kezdőszimbólum bevezetésével, illetve a # speciális terminális szimbólum bevezetésével, amely a bemenet végét fogja jelölni.
A grammatikába bevesszük az

$$S' \to S\#$$

szabályt. Így a kezdő eljárásunk a következő lesz:

```
Procedure S'()
        S();
        Vizsgal(#);
EndProcedure
```

Példa rekurzív leszállással megvalósított elemzésre:

```
/*
        S -> aS | A
        A -> bA | epsilon

  L = {a^{m}b^{n} | m,n >= 0}
*/

#include <iostream>
#include <cstdlib>
#include <string>
using namespace std;

char akt_szimb;
// char w[1024];
string w;
int i;

void S();
void A();

void Vizsgal(char a) {
  if (a == akt_szimb)
    akt_szimb = w[++i];
  else {
    cout << "HIBA!" << endl;
    exit(1);
  }
}

/* S -> aS | A */
void S() {
  switch (akt_szimb) {
    case 'a': Vizsgal('a'); S(); break;
    case 'b': A(); break;
    case '#': ;
  }
}

/* A -> bA | eps */
void A() {
  switch (akt_szimb) {
    case 'b': Vizsgal('b'); A(); break;
    case '#': ;
  }
}

int main() {
  cout << "w = ";
  cin >> w;
  w += "#";
  i = 0;
  akt_szimb = w[i];

  S();
  Vizsgal('#');

  return 0;
}
```

11. Példa

$$S \rightarrow aSA \mid A$$
$$A \rightarrow b \mid \varepsilon$$

```
...

Procedure S()
    Case (aktualis_szimbolum)
        {a}: Vizsgal(a); S(); A();
        {b}: A();
        {#}: nop;
    EndCase
EndProcedure

Procedure A()
    Case (aktualis_szimbolum)
        {b}: Vizsgal(b);
        {#}: nop;
    EndCase
EndProcedure

S(); Vizsgal(#);
```

bemenet: ab#

```
↓ aktualis_szimbolum = a
S(); Vizsgal(#); ↓ aktualis_szimbolum = a
Vizsgal(a); S(); A(); Vizsgal(#);
↓ aktualis_szimbolum = b
S(); A(); Vizsgal(#);
↓ aktualis_szimbolum = b
A(); A(); Vizsgal(#);
↓ aktualis_szimbolum = b
Vizsgal(b); A(); Vizsgal(#);
↓ aktualis_szimbolum = #
A(); Vizsgal(#);
↓ aktualis_szimbolum = #
Vizsgal(#);
```

✓