Szintaktikai elemzés

A környezetfüggetlen nyelvtanokkal leírható programnyelvi tulajdonságok vizsgálatát szintaktikai elemzésnek nevezzük. A programnyelv szintaktikájának azon követelményei, amelyek nem írhatók le környezetfüggetlen nyelvtannal, a statikus szemantikát alkotják.

Itt nyelvtan alatt mindig környezetfüggetlen nyelvtant, vagy más néven Chomsky 2-es típusú nyelvtant értünk, és feltesszük, hogy a környezetfüggetlen nyelvtan kiterjesztett nyelvtan, azaz helyettesítési szabályai

$$A \to \alpha, \qquad \text{ahol } A \in N,\ \alpha \in (N \cup T)^*$$

alakúak.

A szintaktikai elemzés alapfogalmai

Legyen $G = (N, T, P, S)$ egy nyelvtan. Ha $S \overset{*}{\Longrightarrow} \alpha$, ahol $\alpha \in (N \cup T)^*$, akkor az α-t mondatformának nevezzük. Ha $S \overset{*}{\Longrightarrow} x$, ahol $x \in T^*$, akkor az x a nyelvtan által generált nyelv egy mondata.

Az elemzés szempontjából a mondatnak kiemelt szerepe van, hiszen a programozó által írt program is terminális szimbólumok sorozata, de ez a szimbólumsorozat csak akkor lesz a nyelvnek egy mondata, ha a program szintaktikailag helyes.

Legyen a $G = (N, T, P, S)$ nyelvtannak $\alpha = \alpha_1\beta\alpha_2$ egy mondatformája $(\alpha, \alpha_1, \alpha_2, \beta \in (N \cup T)^*)$. A β-t az α egy részmondatának nevezzük, ha van olyan $A \in N$ szimbólum, amelyre $S \overset{*}{\Longrightarrow} \alpha_1 A \alpha_2$ és $A \overset{*}{\Longrightarrow} \beta$. Az α-nak β egy egyszerű részmondata, ha a fentiekben az $A \to \beta \in P$ teljesül.

Minden mondat egyben mondatforma is, és a mondatforma „részét” is részmondatnak, és nem részmondatformának nevezzük.

Egy mondatforma legbaloldalibb egyszerű részmondatát a mondatforma nyelének nevezzük.

A mondat szintaxisfájának levelei a nyelvtan terminális szimbólumai, a szintaxisfa többi pontja a nemterminális szimbólumokat reprezentálja, a gyökérelem pedig a nyelvtan kezdőszimbóluma.

Egy nemegyértelmű nyelvtanban van legalább egy olyan mondat, amelyhez több szintaxisfa tartozik. Ez az elemzés szempontjából azt jelenti, hogy ezt a mondatot többféleképpen is lehet elemezni, azaz a különböző elemzésekhez különböző tárgyprogramok is tartozhatnak. Ezért fordítóprogramokkal csak az egyértelmű nyelvtanok által generált nyelvek fordítását végezzük el.

A továbbiakban feltesszük azt is, hogy a G nyelvtanra a következő feltételek teljesülnek:

1. a nyelvtan ciklusmentes, azaz nem tartalmaz $A \overset{+}{\Longrightarrow} A$ ($A \in N$) helyettesítési szabálysorozatot,
2. a nyelvtan redukált, azaz nincs benne „felesleges” nemterminális szimbólum, minden nemterminális szimbólum előfordul legalább egy levezetésben, és minden nemterminális szimbólumból levezethető legalább egy mondatnak egy része, azaz minden $A \in N$-re fennáll, hogy $S \overset{*}{\Longrightarrow} \alpha A \beta \overset{*}{\Longrightarrow} \alpha y \beta \overset{*}{\Longrightarrow} xyz$, ahol $A \overset{*}{\Longrightarrow} y$ és $|y| > 0$ $(\alpha, \beta \in (N \cup T)^*,\ x, y, z \in T^*)$.

A szintaktikai elemzési módszerek

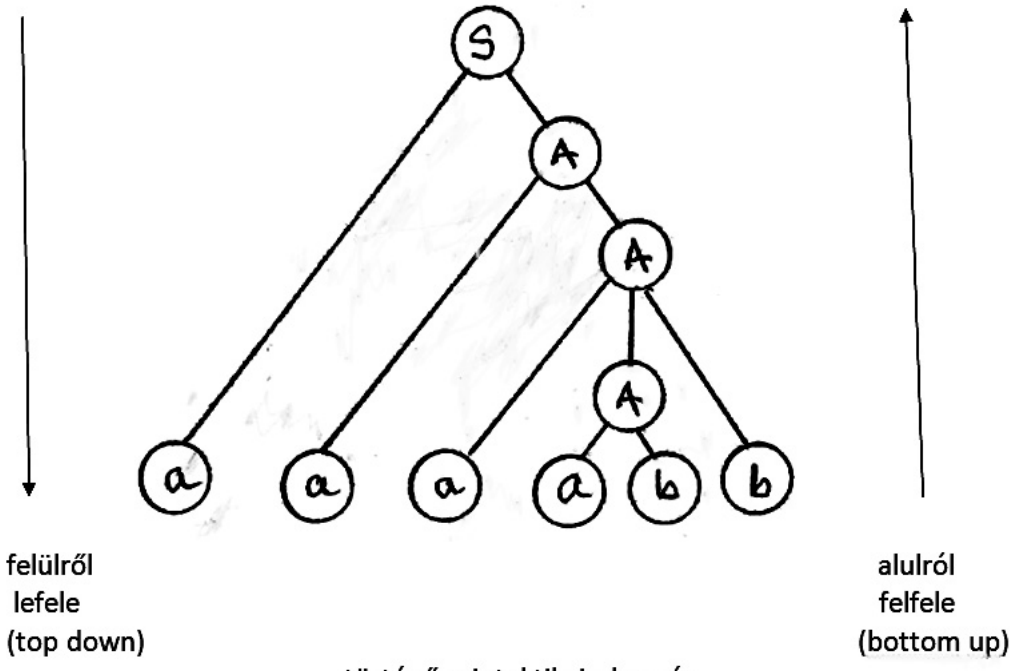
A programot a lexikális elemző egy terminális szimbólumokból álló sorozattá alakítja, ez a terminálisokból álló sorozat a szintaktikai elemzés bemenete.

A szintaktikai elemzés feladata eldönteni azt, hogy ez a szimbólumsorozat a nyelv egy mondata-e. A szintaktikai elemzőnek ehhez például meg kell határoznia a szimbólumsorozat szintaxisfáját, ismerve a szintaxisfa gyökérelemét és a leveleit, elő kell állítania a szintaxisfa többi pontját és élét, vagyis meg kell határoznia a program egy levezetését.

Ha ez sikerül, akkor azt mondjuk, hogy a program eleme a nyelvnek, azaz a program szintaktikailag helyes.

A továbbiakban csak a balról-jobbra haladó elemzésekkel foglalkozunk, azaz azokkal az elemzésekkel, amelyek a program jeleit balról jobbra olvasva dolgozzák fel.

A szintaxisfa felépítésére több módszer létezik. Az egyik az, amikor az S szimbólumból kiindulva építjük fel a szintaxisfát, ez felülről-lefelé haladó elemzés. Ha a szintaxisfa építése a levelekből kiindulva halad az S szimbólum felé, akkor alulról-felfelé elemzésről beszélünk.



történő szintaktikai elemzés

LL(k) elemzés

Felülről-lefelé elemezve a nyelvtan kezdőszimbólumától, a szintaxisfa gyökérpontjától indulunk el, és így próbáljuk felépíteni a szintaxisfát. A célunk az, hogy a szintaxisfa levelein az elemezendő programszöveg terminális szimbólumai legyenek.

Egy példa:

A $G = (\{S\}, \{a, b\}, P, S)$ nyelvtan helyettesítési szabályai a következők:

(1) $S \rightarrow ab$

(2) $S \rightarrow aSb$

a nyelv pedig: $L(G) = \{a^n b^n \mid n \geq 1\}$

Vezessük le az $aabb$ szót!
Melyik szabályt válasszuk? Ha megnézzük a szó első két betűjét, akkor látszik, hogy a (2) szabályt kell először alkalmazni:

$$S \Rightarrow aSb$$

Most az első betű (a) illeszkedik a levezetendő szó első betűjére. Ha megnézzük a következő két betűt a szóban (ab), akkor látszik, hogy az első szabályt kell alkalmazni.

$$S \Rightarrow aSb \Rightarrow aabb$$

A következő nyelvtannál:

$G = (\{A, B, S\}, \{a, b, c\}, P, S)$,

$S \rightarrow A \mid B$

$$A \to aAb \mid ab$$
$$B \to aBc \mid ac$$

ez a módszer nem működik.

LL(k) nyelvtanok

Mivel felülről-lefelé építjük a szintaxisfát és a szövegben balról-jobbra haladunk, ezért arra kell törekednünk, hogy az elemzéskor kapott mondatformák bal oldalán a lehető leghamarabb megjelenjenek az elemezendő szöveg terminálisai.

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

$$xA\beta \underset{legbal}{\Longrightarrow} x\alpha\beta.$$

Ha az $S \overset{*}{\Longrightarrow} x$ ($x \in T^*$) levezetésben minden helyettesítés legbaloldalibb helyettesítés, akkor ezt a levezetést legbaloldalibb levezetésnek nevezzük, és így jelöljük:

$$S \underset{legbal}{\overset{*}{\Longrightarrow}} x.$$

A legbaloldalibb levezetésben a terminális szimbólumok a mondatforma bal oldalán jelennek meg, ezért a felülről-lefelé levezetésekben mindig legbaloldalibb helyettesítéseket alkalmazunk,

így a felülről-lefelé elemzés a legbaloldalibb levezetésnek felel meg. Ezért a továbbiakban, amikor felülről-lefelé elemzésről lesz szó, a nyilak alá már nem is írjuk oda a „legbal” szót.

A felülről-lefelé elemzés egyik módszere az lehetne, hogy előállítjuk az összes lehetséges szintaxisfát. Egy szintaxisfa levelein levő szimbólumokat balról-jobbra olvasva a nyelv egy mondatát kapjuk meg. Ha az elemezendő szöveg megegyezik valamelyik mondattal, akkor a mondathoz tartozó szintaxisfáról az elemzés lépései már leolvashatók. Ezt a módszert természetesen nem célszerű és nem is lehet a gyakorlatban megvalósítani.

A gyakorlatban megvalósítható módszer a következő:

Kiindulunk a kezdőszimbólumból, és megpróbálunk legbaloldalibb helyettesítések egymás utáni alkalmazásával eljutni az elemezendő szöveghez. A szintaxisfa építésekor azonban egyáltalán nem biztos, hogy jó helyettesítési szabályokat alkalmazunk, mert például egy lépés után előfordulhat, hogy a következő lépésben nem találunk alkalmazható szabályt, vagy a mondatforma elejére kerülő terminális szimbólumok nem egyeznek meg az elemezendő szöveg terminális szimbólumaival. A terminális szimbólumokra a következőket állíthatjuk:

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

A tétel állítása triviális, egy mondatforma bal oldalán a terminálisokból álló x sorozatot a környezetfüggetlen nyelvtan helyettesítési szabályai nem változtathatják meg.

Ez a „kezdőszelet-egyeztetés” arra használható, hogy megállapíthassuk, ha a szintaxisfa építésekor a bal oldali terminálisok nem egyeznek meg az elemezendő szöveg bal oldalán álló terminálisokkal, akkor a szintaxisfa építése biztosan rossz irányban halad. Ekkor egy lépést vissza kell lépni, és ott egy másik helyettesítési szabályt kell alkalmazni, és még egy lépést vissza kell lépni akkor, ha az adott pontban már nincs több alkalmazható szabály.

Az általános felülről-lefelé elemzést tehát visszalépéses algoritmussal lehet megvalósítani, de ez nem hatékony.

Az LL(k) nyelvtanok alaptulajdonsága az, hogy ha az $S \overset{*}{\Longrightarrow} wx$ $(w, x \in T^*)$ legbaloldalibb levezetés építésekor eljutunk az $S \overset{*}{\Longrightarrow} wA\beta$ mondatformáig $(A \in N,\ \beta \in (N \cup T)^*)$, és az $A\beta \overset{*}{\Longrightarrow} x$-t szeretnénk elérni, akkor az A-ra alkalmazható $A \to \alpha$ helyettesítést egyértelműen meghatározhatjuk, ha ismerjük az x első k darab szimbólumát.

A k szimbólum előreolvasására megadjuk az $First_k$ függvényt. Legyen $First_k(\alpha)$ $(k \geq 0,\ \alpha \in (N \cup T)^*)$ az α-ból levezethető szimbólumsorozatok k hosszúságú kezdő terminális sorozatainak halmaza, azaz

$$\mathit{First}_k(\alpha) = \{x \mid \alpha \overset{*}{\Longrightarrow} x\beta \text{ és } |x| = k\} \cup \{x \mid \alpha \overset{*}{\Longrightarrow} x \text{ és } |x| < k\}, \quad x \in T^*,\ \beta \in (N \cup T)^*.$$

Tehát az $\mathit{First}_k(x)$ halmaz az x első k darab szimbólumát, $|x| < k$ esetén pedig a teljes x-t tartalmazza. Ha $\alpha \overset{*}{\Longrightarrow} \varepsilon$, akkor természetesen $\varepsilon \in \mathit{First}_k(\alpha)$.

A G nyelvtan LL(k) nyelvtan ($k \geq 0$), ha bármely két

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

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

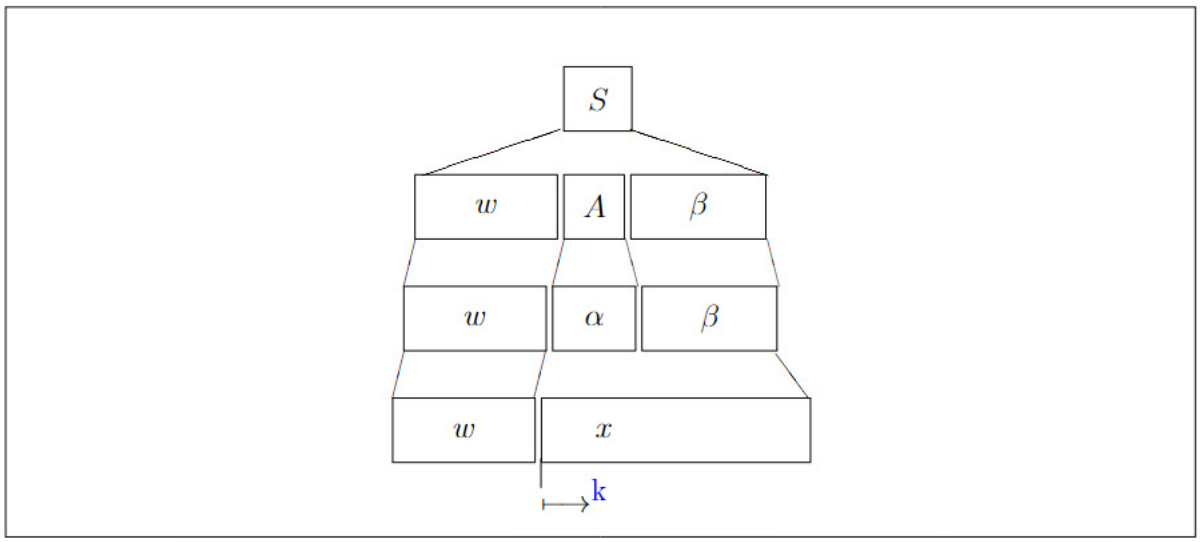
$$\mathit{First}_k(x) = \mathit{First}_k(y)$$

esetén

$$\alpha_1 = \alpha_2.$$

A fenti értelmezés szerint, ha egy nyelvtan LL(k) nyelvtan, akkor a már elemzett w utáni k darab terminális szimbólum az A-ra alkalmazható helyettesítési szabályt egyértelműen meghatározza.

Az értelmezésből az is látható, hogy ha egy nyelvtan LL(k_0) nyelvtan, akkor minden $k > k_0$-ra is LL(k) nyelvtan. Ha LL(k) nyelvtanról beszélünk, akkor k alatt mindig azt a legkisebb k-t



1. ábra. Az LL(k) nyelvtan.

értjük, amelyre az értelmezésben megadott tulajdonság teljesül.

Példa. A következő nyelvtan egy LL(1) nyelvtan. Legyen $G = (\{A, S\}, \{a, b\}, P, S)$, ahol a helyettesítési szabályok a következők:

$S \to AS \mid \varepsilon$
$A \to aA \mid b$

Az S szimbólumra az $S \to AS$ szabályt kell alkalmazni, ha az elemezendő szöveg következő szimbóluma a vagy b, és az $S \to \varepsilon$ szabályt, ha a következő szimbólum a # jel.

Nem minden környezetfüggetlen nyelvtan LL(k) nyelvtan. Például a következő nyelvtan semmilyen k-ra sem LL(k) nyelvtan.

Példa. A $G = (\{A, B, S\}, \{a, b, c\}, P, S)$ nyelvtan helyettesítési szabályai a következők:

$S \to A \mid B$

$A \to aAb \mid ab$

$B \to aBc \mid ac$

Az $L(G)$ az $a^i b^i$ és $a^i c^i$ $(i \geq 1)$ alakú mondatokat tartalmazza. Az $a^{k+1}b^{k+1}$ elemzésének már a kezdetekor sem lehet eldönteni k szimbólum előreolvasásával, hogy az $S \to A$ és az $S \to B$ közül melyik helyettesítést kell először alkalmazni, mivel minden k-ra $First_k(a^k b^k) = First_k(a^k c^k) = a^k$.

Az LL(k) nyelvtan értelmezése szerint, ha legbaloldalibb helyettesítésekkel a $wA\beta$ mondatformát kaptuk, akkor a w-t követő k darab terminális szimbólum egyértelműen meghatározza az A-ra alkalmazható szabályt. Ezt mondja ki a következő tétel.

Tétel. A G kiterjesztett nyelvtan akkor és csak akkor LL(k) nyelvtan, ha minden

$$S \overset{*}{\Longrightarrow} wA\beta, \quad \text{és} \quad A \to \gamma, \ A \to \delta$$

$$(\gamma \neq \delta, \ w \in T^*, \ A \in N, \ \beta, \gamma, \delta \in (N \cup T)^*)$$

esetén

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

Ha a nyelvtanban az A-ra van egy $A \to \varepsilon$ szabály is (kiterjesztett

nyelvtan), akkor a $First_k$ halmazokban a β-ból származó terminális sorozatok k hosszúságú kezdősorozatai is szerepelnek.

Ez pedig azt jelenti, hogy az LL(k) tulajdonság eldöntéséhez nem elegendő csak a szabályokat vizsgálni, hanem a nem feltétlenül véges darabszámú levezetéseket is figyelembe kell venni.

A gyakorlatban jól használható vizsgálati módszert csak az LL(1) nyelvtanokra lehet adni. Ehhez egy új fogalmat értelmezünk, egy szimbólumsorozatot követő k hosszúságú terminális sorozatok halmazát.

$Follow_k(\beta) = \left\{x \mid S \overset{*}{\Longrightarrow} \alpha\beta\gamma \text{ és } x \in First_k(\gamma)\right\}$, és ha $\varepsilon \in Follow_k(\beta)$, akkor legyen $Follow_k(\beta) = Follow_k(\beta) \setminus \{\varepsilon\} \cup \{\#\}$ $(\alpha, \beta, \gamma \in (N \cup T)^*,\ x \in T^*)$.

Az értelmezés második részében levő átalakítás azért szükséges, mert ha az $\alpha\beta\gamma$ levezetésben a β után nem áll semmilyen szimbólum, azaz $\gamma = \varepsilon$, akkor a β utáni jel csak a mondatokat lezáró # jel lehet.

A $Follow_1(A)$ $(A \in N)$ tehát azokat a terminális szimbólumokat tartalmazza, amelyek az

$$S \overset{*}{\Longrightarrow} \alpha A\gamma \overset{*}{\Longrightarrow} \alpha Aw\ (\alpha, \gamma \in (N \cup T)^*,\ w \in T^*)$$

levezetésben közvetlenül az A szimbólum mögött állhatnak.

Tétel. A G kiterjesztett nyelvtan akkor és csak akkor LL(1) nyelvtan, ha minden A nemterminális szimbólum $A \to \gamma, A \to \delta$ helyettesítési szabályaira igaz, hogy

$$First_1\big(\gamma Follow_1(A)\big) \cap First_1\big(\delta Follow_1(A)\big) = \emptyset.$$

A tételben az $First_1\big(\gamma Follow_1(A)\big)$ kifejezés azt jelenti, hogy a γ-t a $Follow_1(A)$ halmaz minden elemével külön-külön konkatenálni kell, és az így kapott halmaz minden elemére alkalmazni kell az $First_1$ függvényt.

Látható, hogy a tétel jól használható annak eldöntésére, hogy egy nyelvtan vajon LL(1)-es-e, hiszen legfeljebb csak annyi halmazt kell a vizsgálathoz előállítani, ahány szabálya van a nyelvtannak.

A továbbiakban az LL(1) nyelvtanok által meghatározott LL(1) nyelvekkel foglalkozunk, az LL(1) nyelvek elemzési módszereit vizsgáljuk. Az egyszerűbb jelölés érdekében az $First_1$ és $Follow_1$ függvények nevéből az indexet elhagyjuk.

Az $First(\alpha)$ halmaz elemei a következő algoritmussal határozhatók meg.

```
FIRST(α)
 1  if α = ε
 2     then F ← {ε}
 3  if α = a, ahol a ∈ T
 4     then F ← {a}
 5  if α = A, ahol A ∈ N
 6     then if A → ε ∈ P
 7             then F ← {ε}
 8             else F ← ∅
 9          for minden A → Y1Y2 ... Ym ∈ P-re  (m ≥ 1)
10              do F ← F ∪ (FIRST(Y1) \ {ε})
11                 for k ← 1 to m − 1
12                     do if Y1Y2 ... Yk ⟹* ε
13                           then F ← F ∪ (FIRST(Yk+1) \ {ε})
14                 if Y1Y2 ... Ym ⟹* ε
15                    then F ← F ∪ {ε}
16  if α = Y1Y2 ... Ym (m ≥ 2)
17     then F ← (FIRST(Y1) \ {ε})
18          for k ← 1 to m − 1
19              do if Y1Y2 ... Yk ⟹* ε
20                    then F ← F ∪ (FIRST(Yk+1) \ {ε})
21          if Y1Y2 ... Ym ⟹* ε
22             then F ← F ∪ {ε}
23  return F
```

Az 1–4. sorokban az ε és egy a terminális szimbólum argumentumra adjuk meg a halmazt, az 5–15. sorokban az A nemterminális szimbólumra határozzuk meg halmaz elemeit. A 6–7. sorokban és a 14–15. sorokban a halmazba betesszük az ε szimbólumot, ha az A-ból az ε levezethető. Az algoritmus 16–22. sorai arra az esetre adják meg a halmaz elemeit, ha az argumentum egy szimbólumsorozat. Megjegyezzük, hogy a 11. és 18. sor for ciklusát már akkor is befejezhetjük, ha $Y_k \in T$, mivel ekkor $Y_1 Y_2 \ldots Y_k$-ból biztosan nem vezethető le az ε.

A tételben, és a továbbiakban is, szükséges a *Follow*(A) halmaz elemeinek ismerete. Ezeknek a meghatározására a következő algoritmust adjuk.

FOLLOW(A)

```
 1  if A = S
 2     then F ← {#}
 3     else F ← ∅
 4  for minden B → αAβ ∈ P szabályra
 5      do if |β| > 0
 6            then F ← F ∪ (FIRST(β) \ {ε})
 7                 if β ⟹* ε
 8                    then F ← F ∪ FOLLOW(B)
 9            else F ← F ∪ FOLLOW(B)
10  return F
```

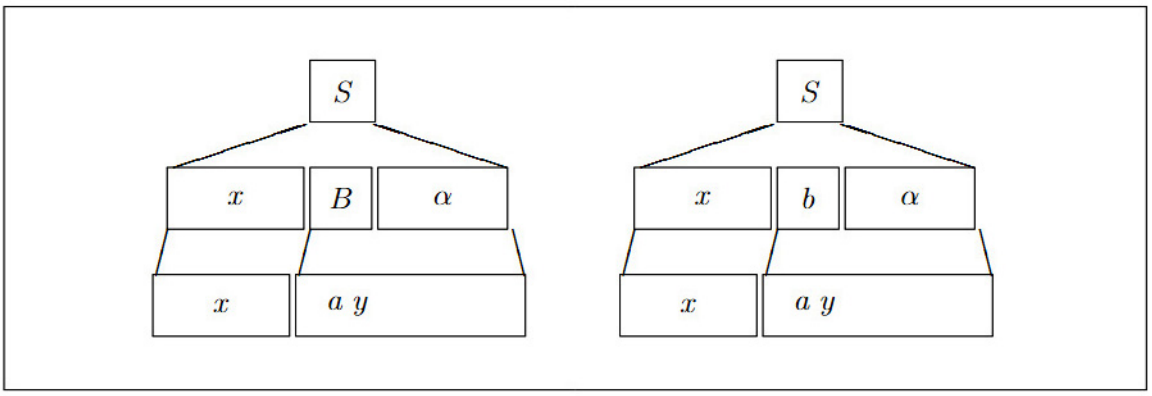
A *Follow*(A) halmaz elemeit az F halmazba helyezzük. A 4–9.

sorokban megvizsgáljuk, hogy ha az argumentum egy szabály jobb oldalán szerepel, milyen terminális szimbólumok állhatnak közvetlenül utána. Látható, hogy az ε nem kerülhet bele a halmazba, és a # is csak akkor, ha az argumentum egy mondatforma legjobboldalibb szimbóluma.

Táblázatos elemzés

Tegyük fel, hogy az elemezendő terminális sorozat xay, amelyből az x szöveget már szintaktikai hiba detektálása nélkül elemeztük.

Mivel felülről lefelé elemzünk, tehát legbaloldalibb helyettesítéseket alkalmazunk, az elemezendő mondatformánk $xY\alpha$, azaz $xB\alpha$ vagy $xb\alpha$ alakú ($Y \in (N \cup T)$, $B \in N$, $a, b \in T$, $x, y \in T^*$, $\alpha \in (N \cup T)^*$).



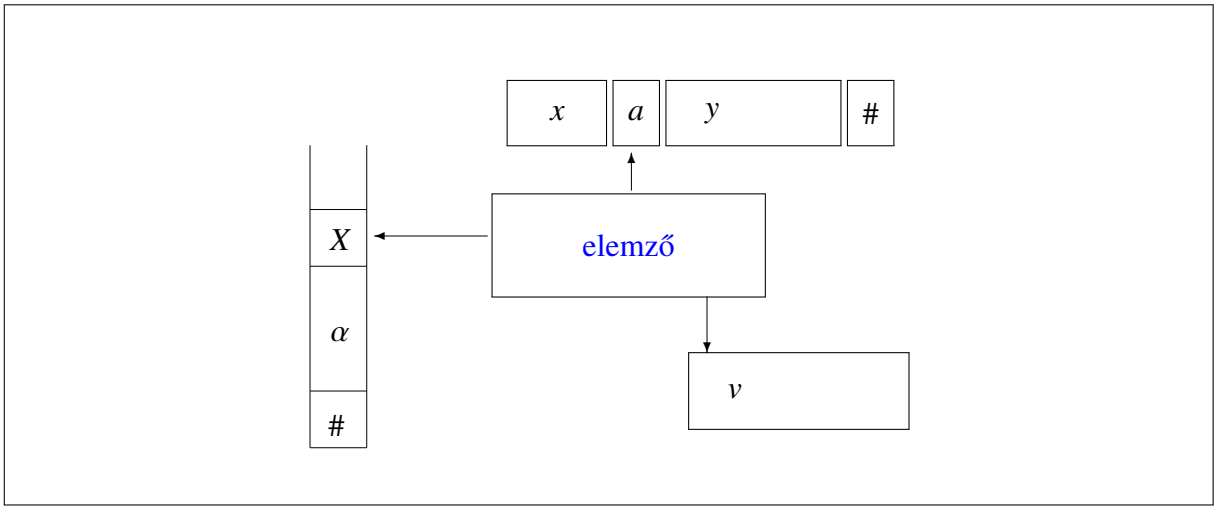
2. ábra. A mondatforma és az elemezendő szöveg

Az első esetben a szintaxisfa építésében most a B egy helyettesítése a következő lépés. Ismerjük a bemenő szimbólumsorozat következő a elemét, ezért egyértelműen meghatározhatjuk azt a $B \to \beta$ szabályt, amelyikre $a \in \mathit{First}(\beta \mathit{Follow}(B))$. Ha van ilyen szabály, akkor az LL(1) nyelvtan értelmezése alapján pontosan egy van, ha nincs, akkor ez az eset egy szintaktikai hiba megtalálását jelenti.

A második esetben a mondatforma következő szimbóluma a b terminális szimbólum, tehát azt várjuk, hogy az elemezendő szöveg következő szimbóluma is b legyen. Ha ez teljesül, azaz $a = b$, akkor az a szimbólum egy helyes szimbólum, továbbléphetünk, mind a mondatformában, mind az elemezendő szövegben az a szimbólum átkerülhet a már elemzett jelsorozatba. Ha $a \neq b$, akkor ez egy szintaktikai hibát jelent.

Az elemző működését a következőképpen írjuk le.
• Jelöljük a # jellel az elemezendő szöveg utolsó szimbólumát.
• Az elemzéshez egy vermet is használunk, a verem alja legyen szintén #.
• A helyettesítési szabályokat valamilyen sorrendben sorszámozzuk meg.
• Az elemzés során alkalmazott szabályok sorszámát felírjuk egy listába, az elemzés végén ez a lista arra fog szolgálni, hogy az elemzett szöveg szintaxisfáját felépíthessük.

Az elemzés állapotait az $(ay\#, X\alpha\#, v)$ hármassal jelöljük. Az



3. ábra. Az LL(1) elemző szerkezete.

$ay\#$ a még nem elemzett szöveg, $X\alpha\#$ az elemzés mondatformájának még nem elemzett része, ez van a veremben, úgyhogy X van a verem tetején, v pedig a szabályok sorszámait tartalmazó lista. Az elemzés úgy fog működni, hogy mindig a verem tetején levő X szimbólumot és a még nem elemzett szöveg első szimbólumát, az a-t fogjuk vizsgálni. Az a-t aktuális szimbólumnak nevezzük. A verem tetejére és az aktuális szimbólumra az elemzőből egy-egy pointer mutat.

Mivel felülről lefelé elemzünk, a verem kezdeti tartalma legyen $S\#$. Ha a teljes elemezendő szöveget xay-nal jelöljük, akkor az elemzés kezdetén az elemzés állapotát, azaz a kezdőállapotot az $(xay\#, S\#, \varepsilon)$ hármassal írhatjuk le, ahol most ε az üres listát jelöli.

Az elemzést egy T elemző táblázat segítségével végezzük. A

táblázat sorai a verem tetején álló szimbólumot, az oszlopai pedig a bemenet következő elemezendő szimbólumát jelölik, a # jelet a táblázat utolsó sorához és utolsó oszlopához írjuk.
A táblázat $T[X, a]$ eleme legyen a következő:

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

vagy

$$T[X,a] = \begin{cases} (\beta, i), & \text{ha } X \to \beta \text{ az } i\text{-edik szabály,} \\ & a \in \mathit{First}(\beta\mathit{Follow}(X)) \\ \mathit{pop}, & \text{ha } X = a \\ \mathit{elfogad}, & \text{ha } X = \# \text{ és } a = \# \\ \mathit{hiba} & \text{egyébként} \end{cases}$$

1. példa.

(forrás: internet)

$$S \to aS \mid A$$
$$A \to bAc \mid d \mid \epsilon$$

$$FOLLOW_1(S) = \{\#\}$$
$$FOLLOW_1(A) = \{c, \#\}$$

$$FIRST_1(aS\ FOLLOW_1(S)) = \{a\}$$
$$FIRST_1(A\ FOLLOW_1(S)) = \{b, d, \#\}$$

$$FIRST_1(bAc\ FOLLOW_1(A)) = \{b\}$$
$$FIRST_1(d\ FOLLOW_1(A)) = \{d\}$$
$$FIRST_1(\epsilon\ FOLLOW_1(A)) = \{c, \#\}$$

	a	b	c	d	#
S	$(aS, 1)$	$(A, 2)$		$(A, 2)$	$(A, 2)$
A		$(bAc, 3)$	$(\varepsilon, 5)$	$(d, 4)$	$(\varepsilon, 5)$
a	pop				
b		pop			
c			pop		
d				pop	
#					$elfogad$

Az elemzés az abc szóra a következő:

$$
\begin{array}{lll}
(abc\#,\, S\#,\, \varepsilon) & \xrightarrow{(aS,1)} & (\quad abc\#,\quad aS\#,\quad 1\quad) \\
& \xrightarrow{pop} & (\quad bc\#,\quad S\#,\quad 1\quad) \\
& \xrightarrow{(A,2)} & (\quad bc\#,\quad A\#,\quad 12\quad) \\
& \xrightarrow{(bAc,3)} & (\quad bc\#,\quad bAc\#,\quad 123\quad) \\
& \xrightarrow{pop} & (\quad c\#,\quad Ac\#,\quad 123\quad) \\
& \xrightarrow{(\varepsilon,5)} & (\quad c\#,\quad c\#,\quad 1235\quad) \\
& \xrightarrow{pop} & (\quad \#,\quad \#,\quad 1235\quad) \\
& \xrightarrow{elfogad} & \text{OK}
\end{array}
$$

Az abc szó levezetése az 1, 2, 3, 5 helyettesítési szabályok alkalmazásával:

$$S \Rightarrow aS \Rightarrow aA \Rightarrow abAc \Rightarrow abc$$

Elemzés az aba szóra:

$$
\begin{array}{lll}
(aba\#,\, S\#,\, \varepsilon) & \xrightarrow{(aS,1)} & (\quad aba\#,\quad aS\#,\quad 1\quad) \\
& \xrightarrow{pop} & (\quad ba\#,\quad S\#,\quad 1\quad) \\
& \xrightarrow{(A,2)} & (\quad ba\#,\quad A\#,\quad 12\quad) \\
& \xrightarrow{(bAc,3)} & (\quad ba\#,\quad bAc\#,\quad 123\quad) \\
& \xrightarrow{pop} & (\quad a\#,\quad Ac\#,\quad 123\quad) \\
& \rightarrow & \text{HIBA}
\end{array}
$$

2. példa. Legyen a G nyelvtan a következő:

$G = (\{E, E', T, T', F\}, \{+, *, (,), i\}, P, E)$,

ahol a P helyettesítési szabályok a következők:

$E \to TE'$

$E' \to +TE' \mid \varepsilon$

$T \to FT'$

$T' \to *FT' \mid \varepsilon$

$F \to (E) \mid i$

	+	*	(	)	i	#
E			$(TE', 1)$		$(TE', 1)$	
E'	$(+TE', 2)$			$(\varepsilon, 3)$		$(\varepsilon, 3)$
T			$(FT', 4)$		$(FT', 4)$	
T'	$(\varepsilon, 6)$	$(*FT', 5)$		$(\varepsilon, 6)$		$(\varepsilon, 6)$
F			$((E), 7)$		$(i, 8)$	
+	pop					
*		pop				
(			pop			
)				pop		
i					pop	
#						elfogad

A táblázat kitöltését a következő algoritmussal végezhetjük:

LL(1)-TÁBLÁZATOT-KITÖLT(G)

```
 1  for minden A ∈ N-re
 2      do if A → α ∈ P az i-edik szabály
 3            then for minden a ∈ First(α)-ra
 4                    do T[A, a] ← (α, i)
 5                    if ε ∈ First(α)
 6                       then for minden a ∈ Follow(A)-ra
 7                               do T[A, a] ← (α, i)
 8  for minden a ∈ T-re
 9      do T[a, a] ← pop
10  T[#, #] ← elfogad
11  for minden X ∈ (N ∪ T ∪ {#}) és minden a ∈ (T ∪ {#})-ra
12      do if T[X, a] = „üres”
13            then T[X, a] ← hiba
14  return T
```

A 10. sorban a táblázat jobb alsó sorába az elfogad szöveget írjuk, a 8–9. sorokban a terminálisokkal címkézett sorok és oszlopok rész-táblázatának főátlójába a pop szöveget írjuk, az 1–7. sorokban levő algoritmussal a megadott pozícióra a szabály jobb oldalának szimbólumai és a szabály sorszáma kerül.

Az algoritmus 12–13. sorában a táblázat üresen maradt helyeire a szintaktikai hibát jelző hiba szöveget írjuk.

Az elemzés működése állapotátmenetekkel adható meg. A kezdőállapot tehát $(x\#, S\#, \varepsilon)$, ha az elemezendő szöveg x, és az

elemzés sikeresen befejeződik akkor, ha az elemző a $(\#, \#, w)$ állapotba, a végállapotba kerül. Ha a még nem elemzett szöveg az $ay\#$, és a verem tetején az X szimbólum áll, az állapotátmenetek a következők:

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

Az OK azt jelenti, hogy az elemzett szimbólumsorozat szintaktikailag helyes, a HIBA pedig egy szintaktikai hiba detektálását jelzi.

Az elemző működésére a következő algoritmust adhatjuk meg.

```
LL(1)-ELEMEZ(xay#, T)
 1  s ← (xay#, S#, ε), s′ ← elemez
 2  repeat
 3        if s = (ay#, Aα#, v) és T[A, a] = (β, i)
 4          then s ← (ay#, βα#, vi)
 5          else if s = (ay#, aα#, v)
 6                  then s ← (y#, α#, v)     ▷Ekkor T[a, a] = pop.
 7                  else if s = (#, #, v)
 8                          then s′ ← OK ▷Ekkor T[#, #] = elfogad.
 9                          else s′ ← HIBA ▷Ekkor T[A, a] = hiba.
10  until s′ = OK vagy s′ = HIBA
11  return s′, s
```

Az algoritmus bemenő paramétere az xay elemezendő szöveg és a T elemző táblázat. Az s' változó az elemző működését jelzi, működés közben az s' értéke elemez, az elemzés befejezésekor OK vagy HIBA. Az elemző az elemzett szöveg a aktuális szimbóluma és a verem tetején levő szimbólum alapján a T táblázatból meghatározza az elvégzendő műveletet. A 3–4. sorban a szintaxisfát építi az $A \to \beta$ szabály alapján. Az 5–6. sorban léptetés történik, mivel a verem tetején is az a szimbólum található. Az algoritmus a 8–9. sorban az elemzés befejezését jelzi, ha a verem kiürült és az elemezendő szöveg végére ért, akkor az elemzett szöveg helyes, egyebként az elemző egy szintaktikai hibát fedezett fel. Az algoritmus végeredménye

ennek megfelelően az OK vagy HIBA jelzés, és kimenetként mindkét esetben megjelenik az elemző állapotának s hármasa is. Helyes szöveg esetén a hármas harmadik eleméből, v-ből a szabályok sorszámai alapján felépíthető a szintaxisfa, szintaktikai hiba esetén a hármas első elemének első szimbóluma a hiba helyét adja meg.

Példa. Legyen $G = (\{E, E', T, T', F\}, \{+, *, (,), i\}, P, E)$, ahol P:

$E \to TE'$

$E' \to +TE' \mid \varepsilon$

$T \to FT'$

$T' \to *FT' \mid \varepsilon$

$F \to (E) \mid i$

Az elemző táblázat kitöltéséhez a következő halmazok szükségesek:

$\mathit{First}(TE') = \{(, i\}$,

$\mathit{First}(+TE') = \{+\}$,

$\mathit{First}(FT') = \{(, i\}$,

$\mathit{First}(*FT') = \{*\}$,

$\mathit{First}((E)) = \{(\}$,

$\mathit{First}(i) = \{i\}$,

$\mathit{Follow}(E') = \{), \#\}$,

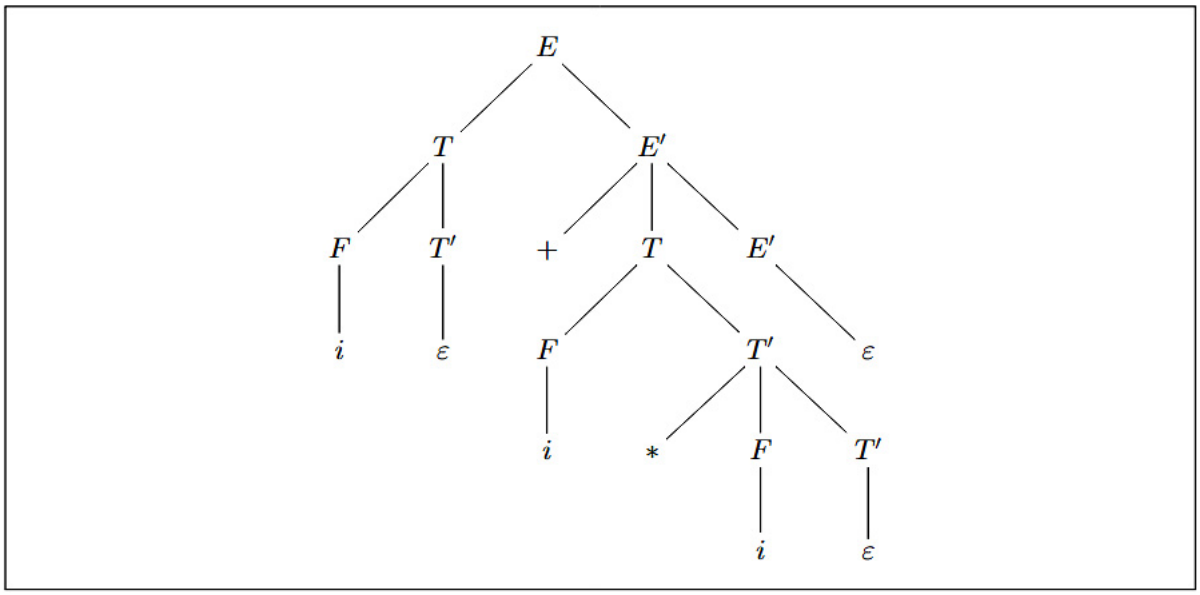
$\mathit{Follow}(T') = \{+,), \#\}$.

Az elemző táblázat a következő, a táblázatban az üres helyek a hibát jelentik.

	+	*	(	)	i	#
E			$(TE', 1)$		$(TE', 1)$	
E'	$(+TE', 2)$			$(\varepsilon, 3)$		$(\varepsilon, 3)$
T			$(FT', 4)$		$(FT', 4)$	
T'	$(\varepsilon, 6)$	$(*FT', 5)$		$(\varepsilon, 6)$		$(\varepsilon, 6)$
F			$((E), 7)$		$(i, 8)$	
+	pop					
*		pop				
(			pop			
)				pop		
i					pop	
#						elfogad

Elemezzük az $i + i * i$ szöveget. Az elemzés a következő:

$$
\begin{array}{lllll}
(i+i*i\#, S\#, \varepsilon) & \xrightarrow{(TE',1)} & (\ i+i*i\#, & TE'\#, & 1\ ) \\
& \xrightarrow{(FT',4)} & (\ i+i*i\#, & FT'E'\#, & 14\ ) \\
& \xrightarrow{(i,8)} & (\ i+i*i\#, & iT'E'\#, & 148\ ) \\
& \xrightarrow{pop} & (\ +i*i\#, & T'E'\#, & 148\ ) \\
& \xrightarrow{(\varepsilon,6)} & (\ +i*i\#, & E'\#, & 1486\ ) \\
& \xrightarrow{(+TE',2)} & (\ +i*i\#, & +TE'\#, & 14862\ ) \\
& \xrightarrow{pop} & (\ i*i\#, & TE'\#, & 14862\ ) \\
& \xrightarrow{(FT',4)} & (\ i*i\#, & FT'E'\#, & 148624\ ) \\
& \xrightarrow{(i,8)} & (\ i*i\#, & iT'E'\#, & 1486248\ ) \\
& \xrightarrow{pop} & (\ *i\#, & T'E'\#, & 1486248\ ) \\
& \xrightarrow{(*FT',5)} & (\ *i\#, & *FT'E'\#, & 14862485\ ) \\
& \xrightarrow{pop} & (\ i\#, & FT'E'\#, & 14862485\ ) \\
& \xrightarrow{(i,8)} & (\ i\#, & iT'E'\#, & 148624858\ ) \\
& \xrightarrow{pop} & (\ \#, & T'E'\#, & 148624858\ ) \\
& \xrightarrow{(\varepsilon,6)} & (\ \#, & E'\#, & 1486248586\ ) \\
& \xrightarrow{(\varepsilon,3)} & (\ \#, & \#, & 14862485863\ ) \\
& \xrightarrow{elfogad} & \text{OK} &&
\end{array}
$$



4. ábra. Az $i + i * i$ mondat szintaxisfája.

Elemezzük az $i+)$ hibás szöveget:

$$
\begin{array}{lcllll}
(i+)\#, S\#, \varepsilon) & \xrightarrow{(TE',1)} & (& i+)\#, & TE'\#, & 1 \quad) \\
& \xrightarrow{(FT',4)} & (& i+)\#, & FT'E'\#, & 14 \quad) \\
& \xrightarrow{(i,8)} & (& i+)\#, & iT'E'\#, & 148 \quad) \\
& \xrightarrow{pop} & (& +)\#, & T'E'\#, & 148 \quad) \\
& \xrightarrow{(\varepsilon,6)} & (& +i)\#, & E'\#, & 1486 \quad) \\
& \xrightarrow{(+TE',2)} & (& +)\#, & +TE'\#, & 14862 \quad) \\
& \xrightarrow{pop} & (&)\#, & TE'\#, & 14862 \quad) \\
& \rightarrow & \text{HIBA} & & &
\end{array}
$$

A rekurzív leszállás módszere

A visszalépés nélküli felülről-lefelé elemzésekre a táblázatos módszeren kívül gyakran alkalmazunk egy olyan módszert, amelynek lényege az, hogy a nyelvtanhoz egy programot rendelünk.

A nyelvtan szimbólumaihoz eljárásokat adunk meg, és elemzés közben a rekurzív eljáráshívásokon keresztül a programnyelv implementációja valósítja meg az elemző vermét és a veremkezelést. A felülről-lefelé elemzés és a rekurzív eljáráshívások miatt ezt a módszert a rekurzív leszállás módszerének nevezzük. A terminális szimbólumok vizsgálatára vezessük be a Vizsgál eljárást. Legyen az eljárás paramétere a „várt szimbólum”, azaz a mondatforma legbaloldalibb, még nem vizsgált terminális szimbóluma, és tartalmazza az aktuális_szimbólum globális változó a vizsgált terminális sorozat soron köv. szimbólumát.

```
procedure Vizsgál(a);
begin
   if aktuális_szimbólum = a
      then Következő_szimbólum
      else Hibajelzés
end;
```

A Következő_szimbólum az előreolvasásra szolgál, ez a lexikális elemzőt meghívó eljárás neve. Ez az eljárás a bemenő szimbólumsorozatból meghatározza a következő szimbólumot és ezt az aktuális_szimbólum változóba tölti. A Hibajelzés el-

járás szintaktikai hibajelzést ad, és ezután befejezi az elemző program futását.

A nyelvtan minden nemterminális szimbólumához rendeljünk hozzá egy eljárást. Az A szimbólumhoz tartozó eljárás legyen a következő:

```
procedure  A;
begin
   T(A)
end;
```

ahol `T(A)`-t az A-ra vonatkozó helyettesítési szabályok jobb oldalán álló szimbólumok határozzák meg.

Az elemzéshez használt nyelvtanok redukáltak, ami többek között azt jelenti, hogy nincs bennük „felesleges” nemterminális szimbólum, minden nemterminális szimbólum szerepel legalább egy helyettesítési szabály bal oldalán.

Tehát ha az A szimbólumot vizsgáljuk, biztosan van legalább egy $A \to \alpha$ helyettesítési szabály.

1. Ha az A szimbólumra csak egy helyettesítési szabály van:
   (a) az $A \to a$ szabályhoz rendelt program legyen a `Vizsgál(a)`,
   (b) az $A \to B$ szabályhoz rendeljük hozzá a `B` eljáráshívást,

(c) az $A \to X_1 X_2 \ldots X_n$ $(n \geq 2)$ szabályhoz tartozzon a következő blokk:

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

2. Ha az A szimbólumra több szabály van:

(a) Ha az $A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$ szabályok ε-mentesek, azaz α_i-ből $(1 \leq i \leq n)$ nem vezethető le az ε, akkor `T(A)` legyen

```
case aktuális_szimbólum of
   First(alpha_1) : T(alpha_1);
   First(alpha_2) : T(alpha_2);
   ...
   First(alpha_n) : T(alpha_n)
end;
```

ahol `First(alpha_i)` az *First*(α_i) programbeli jelölése. Felhívjuk a figyelmet arra, hogy a rekurzív leszállás módszerében most először használjuk ki azt, hogy a nyelvtan LL(1)-es.

(b) Az LL(1) nyelvtan programnyelvet ír le, ezért a nyelvtan ε-mentességét nem célszerű megkövetelni. Az $A \to$

$\alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_{n-1} \mid \varepsilon$ szabályokhoz a következő `T(A)` programot rendeljük:

```
case aktuális_szimbólum of
   First(alpha_1)      : T(alpha_1);
   First(alpha_2)      : T(alpha_2);
   ...
   First(alpha_(n-1)) : T(alpha_(n-1));
   Follow(A)           : skip
end;
```

ahol `Follow(A)` a *Follow*(A)-nak felel meg.

Speciálisan, ha az $A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$ szabály esetén egy i-re $(1 \leq i \leq n)$ $\alpha_i \overset{*}{\Longrightarrow} \varepsilon$, azaz $\varepsilon \in$ *First*(α_i), akkor a `case` utasítás i-edik sora lesz a `Follow(A) : skip` sor.

A `T(A)`-ban a `case` helyett, ha lehetséges, használhatunk `if-then-else` vagy `while` utasítást is.

A rekurzív leszállás módszerével készített elemző program kezdő eljárása, azaz főprogramja a nyelvtan kezdőszimbólumához írt eljárás lesz. A rekurzív leszállás módszerével működő elemző programot a köv. Rek-leszáll-készít algoritmussal hozhatjuk létre. Az algoritmus bemenete a G nyelvtan, és az algoritmus eredményül az elemző P programját adja. Az algoritmusban használunk egy Programot-ír eljárást, ami az argumentumában megadott programsorokat a már meglévő P programhoz fűzi. Ezt az algoritmust nem részletezzük.

```
REK-LESZÁLL-KÉSZÍT(G)
 1  P ← ∅
 2  PROGRAMOT-ÍR(
 3     procedure Vizsgál(a);
 4     begin
 5     if aktuális_szimbólum = a
 6        then Következő_szimbólum
 7        else Hibajelzés
 8     end;
 9     )
10  for a G nyelvtan minden A ∈ N szimbólumára
11     do if A = S
12           then PROGRAMOT-ÍR(
13                  program S;
14                  begin
15                    REK-LESZÁLL-UT(S, P)
16                  end.
17                  )
18           else PROGRAMOT-ÍR(
19                  procedure A;
20                  begin
21                    REK-LESZÁLL-UT(A, P)
22                  end;
23                  )
24  return P
```

Az algoritmus a 2–9. sorokban elkészíti a Vizsgál eljárást, majd a bemeneteként megadott G nyelvtan minden nemterminális szimbólumára a REK-LESZÁLL-UT algoritmus felhasználásával készíti a szimbólumhoz tartozó eljárást. A 11–17. sorokban látható, hogy a nyelvtan kezdőszimbólumához az elemző főprogramja fog tartozni. Az algoritmus kimenete az elemző program lesz.

REK-LESZÁLL-UT(A, P)
1 if csak egy $A \to \alpha$ szabály van
2 then REK-LESZÁLL-UT1(α, P) $\triangleright$ $A \to \alpha$.
3 else REK-LESZÁLL-UT2($A, (\alpha_1, \ldots, \alpha_n), P$) $\triangleright$ $A \to \alpha_1 \mid \cdots \mid \alpha_n$.
4 return P

Mivel a létrehozandó elemző program utasításai lényegesen függnek attól, hogy az A nemterminális szimbólumra a nyelvtanban hány helyettesítési szabály van, a REK-LESZÁLL-UT algoritmus a további műveleteket két részre osztja. A REK-LESZÁLL-UT1 algoritmus foglalkozik azzal az esettel, amikor csak egy helyettesítési szabály van, és a REK-LESZÁLL-UT2 készíti az alternatívákra vonatkozó elemző programot.

```
REK-LESZÁLL-UT1(α, P)
 1 if α = a
 2    then PROGRAMOT-ÍR(
 3            Vizsgál(a)
 4            )
 5 if α = B
 6    then PROGRAMOT-ÍR(
 7            B
 8            )
 9 if α = X_1 X_2 ... X_n (n ≥ 2)
10    then PROGRAMOT-ÍR(
11            begin
12              REK-LESZÁLL-UT1(X_1, P) ;
13              REK-LESZÁLL-UT1(X_2, P) ;
14              ...
15              REK-LESZÁLL-UT1(X_n, P)
16            end;
17 return P
```

```
REK-LESZÁLL-UT2(A, (α_1, ..., α_n), P)
 1 if α_1, ..., α_n szabályok ε-mentesek
 2   then PROGRAMOT-ÍR(
 3             case aktuális_szimbólum of
 4              First(alpha_1) : REK-LESZÁLL-UT1(α_1, P);
 5              ...
 6              First(alpha_n) : REK-LESZÁLL-UT1(α_n, P)
 7             end;
 8             )
 9 if van ε-szabály, α_i = ε (1 ≤ i ≤ n)
10   then PROGRAMOT-ÍR(
11             case aktuális_szimbólum of
12              First(alpha_1)     : REK-LESZÁLL-UT1(α_1, P);
13              ...
14              First(alpha_(i-1)) : REK-LESZÁLL-UT1(α_{i-1}, P);
15              Follow(A)          : skip;
16              First(alpha_(i+1)) : REK-LESZÁLL-UT1(α_{i+1}, P);
17              ...
18              First(alpha_n)     : REK-LESZÁLL-UT1(α_1, P)
19             end;
20             )
21 return P
```

A fenti két algoritmus a korábban már részletesen leírt programot hozza létre.

Az elemezendő szöveg végének az ellenőrzése a rekurzív leszállás módszerével úgy valósítható meg, hogy a szöveg végét jelző # szimbólumot beépítjük egy új helyettesítési szabályba. Ha a nyelvtan kezdőszimbóluma S, akkor létrehozunk egy $S' \to S\#$ szabályt, és az új S' lesz az új nyelvtan kezdőszimbóluma. A # szimbólumot terminális szimbólumnak tekintjük. Az így kibővített nyelvtanra készítjük el a rekurzív leszállás elemző programját.

Példa. A példában szereplő nyelvtant egészítsük ki a fenti módon. A szabályok tehát a következők.

$S' \to E\#$
$E \to TE'$
$E' \to +TE' \mid \varepsilon$
$T \to FT'$
$T' \to *FT' \mid \varepsilon$
$F \to (E) \mid i$

A példában megadtuk a táblázatos elemző létrehozásához szükséges *First* és *Follow* halmazokat. Ezek közül most a következőkre van szükség:

First$(+TE') = \{+\}$,
First$(*FT') = \{*\}$,
First$((E)) = \{(\}$,
First$(i) = \{i\}$,
Follow$(E') = \{), \#\}$,
Follow$(T') = \{+,), \#\}$.

A halmazok felhasználásának helyét a program sorainak kommentjeiben adjuk meg, a kommenteket a -- karakterpárral kezdjük.

A rekurzív leszállás módszerének elemző programja a következő lesz.

```
program S';
begin
   E;
   Vizsgál(#)
end.
procedure E;
begin
   T;
   E'
end;
procedure E';
begin
   case aktuális_szimbólum of
   +     : begin                  -- First(+TE')
              Vizsgál(+);
              T;
              E'
           end;
   ),#   : skip                   -- Follow(E')
   end
```

```
end;
procedure T;
begin
   F;
   T'
end;
procedure T';
begin
   case aktuális_szimbólum of
   *     : begin                 -- First(*FT')
              Vizsgál(*);
              F;
              T'
           end;
   +,),# : skip                  -- Follow(T')
   end
end;
procedure F;
begin
   case aktuális_szimbólum of
   (     : begin                 -- First((E))
              Vizsgál((;
              E;
              Vizsgál())
           end;
```

```
   i      : Vizsgál(i)              -- First(i)
   end
end;
```

Látható, hogy az elemző főprogramja a nyelvtan S' kezdőszimbólumához tartozó eljárás lett.

Példa C++-ban

Helyettesítési szabályok:

$S' \to S$

$S \to aS \mid A$

$A \to bA \mid \varepsilon$

```
# include < iostream >
using namespace std;

char aktszimb;
char w[1024];
int i;

void S();
void A();

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

```
void S(){
    switch (aktszimb){
         case 'a' : Vizsgal('a') ; S(); break;
         case 'b' : A(); break;
         case '#' : ;
    }
}

void A(){
    switch (aktszimb) {
        case 'b' : Vizsgal('b'); A(); break;
        case '#' : ;
     }
}

int main(){
  cout << "w = ";
  cin >> w;
  i = 0 ;
  aktszimb = w[i];
  S() ;
  Vizsgal ('#');
  return 0;
}
```