LR(1) elemzés

Alulról-felfelé elemezve az elemezendő szimbólumsorozatból indulunk ki, megkeressük a mondatforma nyelét, és ezt a nyelet helyettesítjük a hozzátartozó nemterminális szimbólummal. Ezt ismételve próbáljuk felépíteni a szintaxisfát. A célunk az, hogy elérjük a nyelvtan kezdőszimbólumát, ez lesz majd a szintaxisfa gyökérpontja, a fa levelein pedig az elemezendő programszöveg terminális szimbólumai lesznek.

Ha $A \to \alpha \in P$, akkor a βAx mondatforma ($x \in T^*,\ \alpha, \beta \in (N \cup T)^*$) *legjobboldalibb helyettesítése* $\beta\alpha x$, azaz

$$\beta Ax \underset{legjobb}{\Longrightarrow} \beta\alpha x.$$

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

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

A legjobboldalibb levezetésben a terminális szimbólumok a mondatforma jobb oldalán jelennek meg. A nyél és a legjobboldalibb helyettesítés kapcsolata alapján, ha a legjobboldalibb levezetés lépéseit visszafelé alkalmazzuk, akkor éppen az alulról-felfelé haladó elemzés lépéseit kapjuk meg. Az alulról-felfelé elemzés tehát a legjobboldalibb levezetés inverzének felel meg. Ezért a továbbiakban, amikor alulról-felfelé elemzésről lesz szó, a helyettesítéseket és a levezetéseket jelölő nyilak alá már nem is írjuk oda a *legjobb* szót.

Az általános alulról-felfelé elemzést, a felülről-lefelé elemzésekhez hasonlóan, visszalépéses algoritmussal lehet megvalósítani. A

visszalépések azonban rendkívül lelassíthatják az elemzést, ezért csak olyan nyelvtanokkal fogunk foglalkozni, amelyekre visszalépés nélküli elemzések adhatók meg.

A továbbiakban bemutatunk egy hatékony, a környezetfüggetlen nyelvtanok igen nagy osztályára alkalmazható elemzési módszert.

Az elemzést $LR(k)$ elemzésnek, a nyelvtant $LR(k)$ nyelvtannak nevezzük, ahol az LR a balról jobbra („*L*eft to *R*ight”) történő elemzésre utal, a k pedig azt jelenti, hogy k szimbólumot előreolvasva egyértelműen meghatározható a mondatforma nyele. Az $LR(k)$ elemzés visszalépés nélküli, léptetés-redukálás típusú elemzés.

Elegendő az $LR(1)$-es elemzőkkel foglalkoznunk, mivel minden $LR(k)$ $(k > 1)$ nyelvtanhoz létezik vele ekvivalens $LR(1)$ nyelvtan. Ez rendkívül fontos számunkra, mivel így egy szöveg elemzésekor mindig elég csak egy szimbólumot előreolvasni.

Az $LR(k)$ elemzés hátrányának hozható fel, hogy az elemző táblázatának „kézi” megkonstruálása nem könnyű. Léteznek azonban olyan programok (például a UNIX `yacc` programja), amelyek egy adott nyelvtan szabályaiból létrehozzák a teljes elemző programot, és így az elemző megírása sem jelent problémát.

Az $LR(k)$ nyelvtanok vizsgálata után az $LALR(1)$ elemzést, a programnyelvek fordítóprogramjaiban jelenleg használt elemzési módszert tanulmányozzuk.

Az $LR(k)$ nyelvtanok

Mint már korábban is tettük, jelöljük az elemezendő szöveg, az elemezendő terminális sorozat jobb oldalát a # szimbólummal. Vezessünk be egy új S' nemterminális szimbólumot és egy új $S' \to S$ szabályt.

Legyen a $G = (N, T, P, S)$ nyelvtanhoz tartozó G' kiegészített nyelvtan a következő:

$$G' = (N \cup \{S'\}, T, P \cup \{S' \to S\}, S').$$

Sorszámozzuk meg a helyettesítési szabályokat, az $S' \to S$ szabály legyen a nulladik szabály. Így, ha redukáláskor a nulladik szabályt kell alkalmazni, akkor ez az elemzés végét, és az elemzett szöveg szintaktikus helyességét fogja jelenteni.

Ha az eredeti S kezdőszimbólum nem szerepel egyik helyettesítési szabály jobb oldalán sem, akkor az $S' \to S$ kiegészítésre nincs is szükség. Az általánosság kedvéért azonban az ***LR***(k) tulajdonságot csak kiegészített nyelvtanokra értelmezzük.

Egy G' kiegészített nyelvtan *LR*(k) nyelvtan ($k \geq 0$), ha bármely két

$$S' \overset{*}{\Longrightarrow} \alpha A w \Longrightarrow \alpha\beta w,$$
$$S' \overset{*}{\Longrightarrow} \gamma B x \Longrightarrow \gamma\delta x = \alpha\beta y$$

($A, B \in N,\ x, y, w \in T^*,\ \alpha, \beta, \gamma, \delta \in (N \cup T)^*$) levezetésre

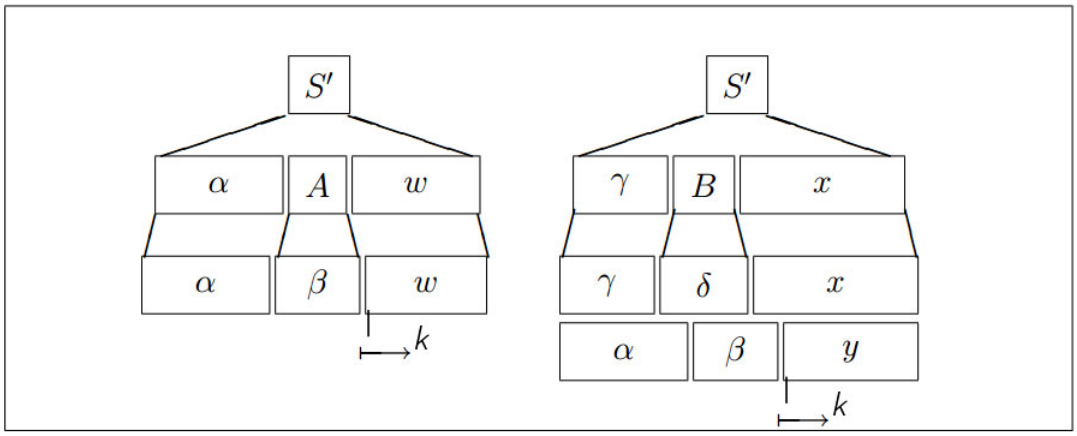
$$\mathit{First}_k(w) = \mathit{First}_k(y)$$

esetén

$$\alpha = \gamma, A = B \text{ } \textit{és} \text{ } x = y.$$

Az ***LR***(k) nyelvtanokra az a jellemző, hogy az $\alpha\beta w$ mondatformában a w első szimbólumától kezdve előreolvasva k darab szimbólumot, egyértelműen meghatározható, hogy valóban β a nyél, és az, hogy

az $A \to \beta$ szabállyal kell redukálni, azaz az $\alpha\beta w$ mondatforma az $\alpha A w$ mondatformára redukálható.



Az $LR(k)$ nyelvtan.

Tegyük fel ugyanis, hogy az $\alpha\beta w$ és az $\alpha\beta y$ mondatformákban, amelyeknek tehát az $\alpha\beta$ prefixük azonos, $\mathit{First}_k(w) = \mathit{First}_k(y)$, és mégis az $\alpha\beta w$ az $\alpha A w$-re, az $\alpha\beta y$ pedig a $\gamma B x$-re redukálható. Az $LR(k)$ tulajdonság miatt ekkor csak $\alpha = \gamma$ és $A = B$ lehet. Ez azt jelenti, hogy a nyél vagy soha nem a β, vagy pedig mindig az (?? ábra).

Példa. A $G' = (\{S', S\}, \{a\}, P', S')$ nyelvtan, ahol a helyettesítési szabályok

$S' \to S$
$S \to Sa \mid a$

nem $LR(0)$ nyelvtan, mivel, feltüntetve az értelmezés jelöléseit,

$$\begin{array}{lcccccccccc} S' & \overset{*}{\Longrightarrow} & \varepsilon & S' & \varepsilon & \Longrightarrow & \varepsilon & S & \varepsilon, & & \\ & & \alpha & A & w & & \alpha & \beta & w & & \\ S' & \overset{*}{\Longrightarrow} & \varepsilon & S & \varepsilon & \Longrightarrow & \varepsilon & Sa & \varepsilon & = & \varepsilon\ S\ a, \\ & & \gamma & B & x & & \gamma & \delta & x & & \alpha\ \beta\ y \end{array}$$

esetén $\mathit{First}_0(\varepsilon) = \mathit{First}_0(a) = \varepsilon$, de $\gamma Bx \neq \alpha Ay$.

Példa. A következő nyelvtan $LR(1)$ nyelvtan.
$G = (\{S', S\}, \{a, b\}, P', S')$, ahol a helyettesítési szabályok:
$S' \to S$
$S \to SaSb \mid \varepsilon$

Példa. Ebben példában megmutatjuk, hogy van olyan környezetfüggetlen nyelvtan, amelyik nem $LR(k)$ nyelvtan egyetlen k-ra sem $(k \geq 0)$.
Legyenek a $G' = (\{S', S\}, \{a\}, P', S')$ nyelvtan helyettesítési szabályai a következők:
$S' \to S$
$S \to aSa \mid a$
Ekkor minden k-ra $(k \geq 0)$
$S' \overset{*}{\Longrightarrow} a^k S a^k \Longrightarrow a^k a a^k$
$S' \overset{*}{\Longrightarrow} a^{k+1} S a^{k+1} \Longrightarrow a^{k+1} a a^{k+1} = a^k a a a^{k+1}$
innen $w = a^k$, $y = aa^{k+1}$ és
$\mathit{First}_k(a^k) = \mathit{First}_k(aa^{k+1}) = a^k$,
de
$\alpha = a^k \neq a^{k+1} = \gamma$ és
$x = a^{k+1} \neq aa^{k+1} = y$.

Míg egy tetszőleges $LL(k)$ $(k > 1)$ nyelvtanra nem biztos, hogy lehet vele ekvivalens $LL(1)$ nyelvtant megadni, addig az $LR(k)$ nyelvtanokra jobb eredményt lehet elérni:

Tétel. *Minden $LR(k)$ $(k > 1)$ nyelvtanhoz létezik vele ekvivalens $LR(1)$ nyelvtan.*

A fenti tétel rendkívül nagy jelentősége az, hogy $LR(k)$ $(k > 1)$ nyelvtanok és nyelvek helyett elegendő csak az $LR(1)$ nyelvtanokkal és nyelvekkel foglalkozni.

LR(1) kanonikus halmazok

Legyen az $\alpha\beta x$ ($\alpha, \beta \in (N \cup T)^*, x \in T^*$) mondatforma nyele β. Ekkor az $\alpha\beta$ jelsorozat prefixeit az $\alpha\beta x$ *járható prefixeinek* nevezzük.

Példa. Tekintsük a következő nyelvtant: $G' = (\{E, T, S'\}, \{i, +, (,)\}, P', S')$, ahol a helyettesítési szabályok a következők (a helyettesítési szabályokat sorszámmal láttuk el):

(0) $S' \to E$
(1) $E \to T$
(2) $E \to E + T$
(3) $T \to i$
(4) $T \to (E)$

A nyelvtan egy mondatformája $E + (i + i)$, ahol az első i a mondatforma nyele. Ennek a mondatformának a járható prefixei a következők: E, $E+$, $E + ($, $E + (i$.

Az értelmezés szerint a járható prefixek a mondatforma nyele utáni szimbólumokat nem tartalmazhatják. Így, mivel az alulról-felfelé elemzésben a feladat a mondatforma nyelének a meghatározása, ez a feladat visszavezethető a mondatforma leghosszabb járható prefixének meghatározására.

Ha adott egy nyelvtan, akkor a nyelvtan helyettesítési szabályaiból a járható prefixek halmaza meghatározható.

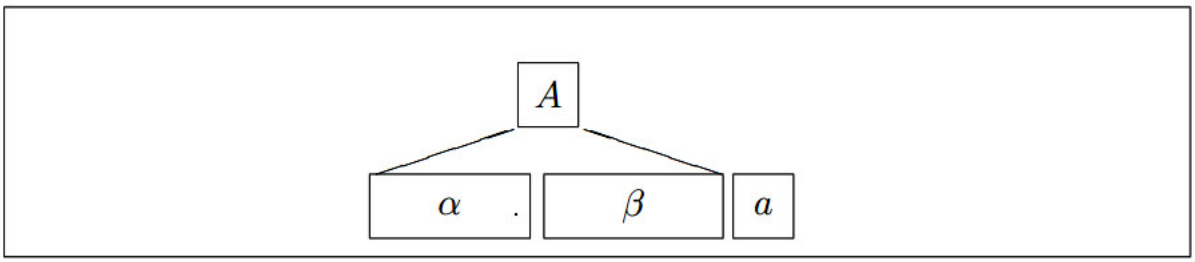
A járható prefixek jelentősége az elemzésben a következő: a nyelvtan járható prefixeiből képezett halmazokhoz hozzárendelhetők egy determinisztikus véges automata állapotai, az állapotátmenetekhez pedig a nyelvtan szimbólumai úgy, hogy kiindulva az automata kezdőállapotából, egy állapothoz mindig egy járható prefix szimbólumain keresztül jutunk el. Ezt a tulajdonságot ismerve fogunk egy

olyan módszert adni, amellyel az elemzést végző automata meghatározható.

Ha a G' nyelvtan egy helyettesítési szabálya $A \to \alpha\beta$, akkor a nyelvtan *LR*(1)-*eleme*

$$[A \to \alpha.\beta, a]\,, \quad (a \in T \cup \{\#\}),$$

ahol az $A \to \alpha.\beta$ az *LR*(1)-elem *magva*, és a az *LR*(1)-elem *előreolvasási szimbóluma*.



Az $[A \to \alpha.\beta, a]$ *LR*(1)-elem.

Az előreolvasási szimbólumnak csak akkor van szerepe, ha az *LR*(1)-elem redukciót ír elő, azaz $[A \to \alpha., a]$ alakú. Ez azt jelenti, hogy redukciót majd csak abban az esetben szabad végrehajtani, ha az α-t, azaz a mondat nyelét az a szimbólum követi.

Egy G' nyelvtan $[A \to \alpha.\beta, a]$ *LR*(1)-eleme *érvényes* a $\gamma\alpha$ járható prefixre nézve, ha

$$S' \overset{*}{\Longrightarrow} \gamma A x \Longrightarrow \gamma\alpha\beta x \ (\gamma \in (N \cup T)^*,\ x \in T^*),$$

és az a az x első szimbóluma, vagy ha $x = \varepsilon$, akkor $a = \#$.

Példa.

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

(0) $S' \to S$
(1) $S \to AA$
(2) $A \to aA$
(3) $A \to b$

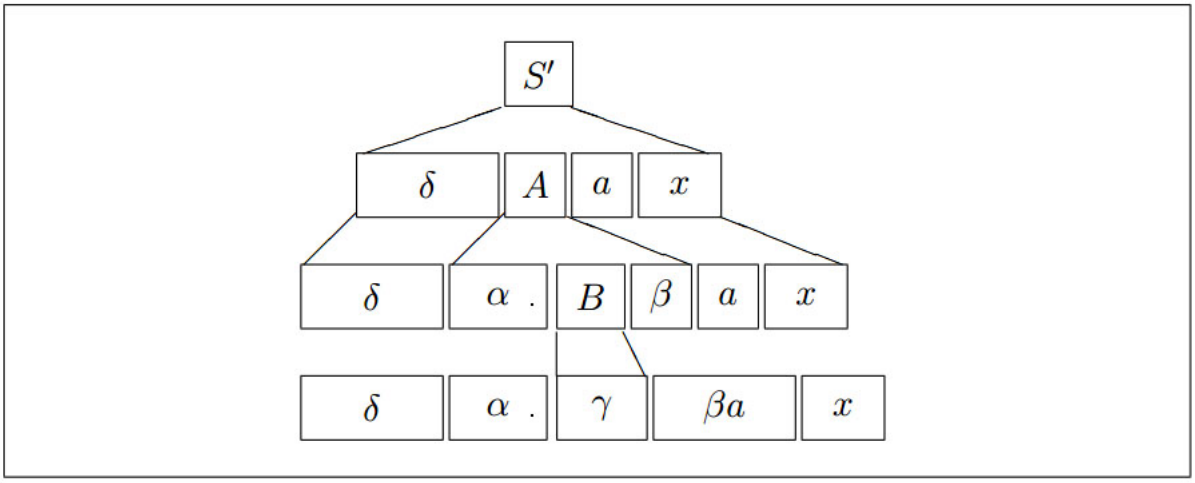
Ekkor $S' \overset{*}{\Longrightarrow} aaAab \Longrightarrow aaaAab$. Az aaa egy járható prefix, és az $[A \to a.A, a]$ érvényes elem erre a járható prefixre nézve. Hasonlóan, $S' \overset{*}{\Longrightarrow} AaA \Longrightarrow AaaA$. Az Aaa járható prefixre nézve az $[A \to a.A, \#]$ ***LR***(1)-elem érvényes.

Az ***LR***(1) elemző felépítéséhez meg kell konstruálni az ***LR***(1)-elemek kanonikus halmazait, és ehhez értelmezni kell az ***LR***(1)-elemhalmazokra a *closure* „lezárás" és a *read* „olvasás" függvényeket.

Legyen a $\mathcal{H}$ halmaz egy nyelvtan egy ***LR***(1)-elemhalmaza. Ekkor a ***closure***$(\mathcal{H})$ halmaz a következő ***LR***(1)-elemeket tartalmazza:

1. a $\mathcal{H}$ halmaz minden eleme legyen eleme a ***closure***$(\mathcal{H})$ halmaznak is,
2. ha $[A \to \alpha.B\beta, a] \in$ ***closure***$(\mathcal{H})$ és $B \to \gamma$ a nyelvtan egy helyettesítési szabálya, akkor legyen $[B \to .\gamma, b] \in$ ***closure***$(\mathcal{H})$ minden $b \in$ ***First***(βa)-ra,
3. a ***closure***$(\mathcal{H})$ halmazt a 2. pontban leírt művelettel addig kell bővíteni, ameddig az lehetséges.

Az értelmezés szerint, ha a $\delta\alpha$ járható prefixre nézve az $[A \to \alpha.B\beta, a]$ egy érvényes ***LR***(1)-elem, akkor ugyanerre a prefixre a $[B \to .\gamma, b]$ is egy érvényes ***LR***(1)-elem lesz, ahol $b \in$ ***First***(βa). (l. ábra). Látható az is, hogy a *closure* művelet a $\delta\alpha$ prefixre az összes érvényes LR(1)-elemet meghatározza.



A *closure*$([A \to \alpha.B\beta, a])$ függvény.

Egy $\mathcal{H}$ *LR*(1)-elemhalmaz lezárását, azaz a *closure*$(\mathcal{H})$ halmazt a következő algoritmussal határozhatjuk meg. A lezárás eredménye a $\mathcal{K}$-ba kerül.

Elemhalmaz-lezár$(\mathcal{H})$

1 $\mathcal{K} \leftarrow \emptyset$
2 **for minden** $E \in \mathcal{H}$ **LR(1)-elemre**
3 **do** $\mathcal{K} \leftarrow \mathcal{K} \cup$ ELEM-LEZÁR(E)
4 **return** $\mathcal{K}$

ELEM-LEZÁR(E)

```
 1  K_E ← {E}
 2  if E LR(1)-elem [A → α.Bβ, a] alakú
 3     then I ← ∅
 4          J ← K_E
 5          repeat
 6                for minden [C → γ.Dδ, b] ∈ J alakú LR(1)-elemre
 7                   do for minden D → η ∈ P szabályra
 8                         do for minden c ∈ FIRST(δb) szimbólumra
 9                               do I ← I ∪ [D → .η, c]
10                J ← I
11                if I ≠ ∅
12                   then K_E ← K_E ∪ I
13                        I ← ∅
14          until J ≠ ∅
15  return K_E
```

Az ELEM-LEZÁR algoritmus egy E elem $\mathcal{K}_E$ lezárását adja meg. Ha az argumentumban a „pont" után egy terminális szimbólum van, akkor az eredmény halmazában csak ez az egy elem lesz (1. sor). Ha ez a szimbólum egy B nemterminális szimbólum, akkor a B bal oldalú szabályok mindegyikéből tudunk egy új LR(1)-elemet készíteni, ez található a 9. sorban. Mivel az elemek vizsgálatát minden új elemre is el kell végezni, az 5–14. sorok egy repeat ciklust tartalmaznak. Ezeket a lépéseket addig kell végezni, amíg új elemeket kapunk (14. sor). A J halmaz tartalmazza a megvizsgálandó elemeket, és I az új elemeket, a $J \leftarrow I$ művelet a 10. sorban látható.

Legyen a $\mathcal{H}$ halmaz egy nyelvtan egy ***LR***(1)-elemhalmaza. Ekkor a ***read***$(\mathcal{H}, X)$ ($X \in (N \cup T)$) halmaz a következő ***LR***(1)-elemeket

tartalmazza:

1. ha $[A \to \alpha.X\beta, a] \in \mathcal{H}$, akkor a ***closure***$([A \to \alpha X.\beta, a])$ minden eleme legyen a ***read***$(\mathcal{H}, X)$ halmaz eleme,
2. a ***read***$(\mathcal{H}, X)$ halmazt az 1. művelettel addig kell bővíteni, ameddig az lehetséges.

Szemléletesen, a ***read***$(\mathcal{H}, X)$ függvény a $\mathcal{H}$ halmaz elemeiben az X szimbólumot olvassa, a „pont" jel az eredmény halmaz elemeiben már az X jobb oldalán van. Ha $\mathcal{H}$ a γ járható prefixekre nézve érvényes ***LR***(1)-elemeket tartalmazza, akkor a ***read***$(\mathcal{H}, X)$ a γX-re nézve érvényes ***LR***(1)-elemek halmaza lesz.

A *read* műveletet az Elemhalmaz-olvas algoritmus valósítja meg, az eredményt a $\mathcal{K}$-ban kapjuk meg.

Elemhalmaz-olvas$(\mathcal{H}, Y)$

```
1  K ← ∅
2  for minden E ∈ H
3       do K ← K ∪ Elem-olvas(E, Y)
4  return K
```

Elem-olvas(E, Y)

```
1  if E = [A → α.Xβ, a] és X = Y
2     then K_{E,Y} ← Elem-lezár([A → αX.β, a])
3     else K_{E,Y} ← ∅
4  return K_{E,Y}
```

A második algoritmus 2. sorában látható, hogy az összes olyan LR(1)-elemet meghatározzuk, ami az olvasás utáni állapotot írja le.

Az $LR(1)$-elemek felsorolásának rövidebb leírása érdekében vezessük be a következő jelölést:

$$[A \to \alpha.X\beta, a/b]$$

jelentse az

$$[A \to \alpha.X\beta, a] \text{ és } [A \to \alpha.X\beta, b]$$

$LR(1)$-elemeket.

Példa. A példában szereplő nyelvtan egy $LR(1)$-eleme $[S' \to .S, \#]$. Erre

$$\textit{closure}([S' \to .S, \#]) = \{[S' \to .S, \#], [S \to .AA, \#], [A \to .aA, a/b], [A \to .b, a/b]\}.$$

Az $LR(1)$-*elemek kanonikus halmazait*, vagy röviden az $LR(1)$-*kanonikus halmazokat* a következő módszerrel határozzuk meg:

A $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_m$ $LR(1)$-elemek *kanonikus halmazai* a következők:

- Legyen $\mathcal{H}_0 = \textit{closure}([S' \to .S, \#])$,
- Ezután képezzük egy X szimbólumra a $\textit{read}(\mathcal{H}_0, X)$ halmazt. Ha az így kapott halmaz nem üres, és nem egyezik meg a $\mathcal{H}_0$ kanonikus halmazzal, akkor legyen ez a következő kanonikus halmaz, azaz $\mathcal{H}_1$.
  Ismételjük meg ezt a műveletet az összes lehetséges X terminális és nemterminális szimbólumra úgy, hogy ha olyan nem üres halmazt kapunk, amelyik nem egyezik meg egyik korábbi kanonikus halmazzal sem, akkor ez a halmaz legyen egy új kanonikus halmaz, és indexe legyen 1-gyel nagyobb, mint az eddigi maximális index.
- Ezután ismételjük meg ezt a műveletet a már korábban előállított összes kanonikus halmazra és a nyelvtan minden szimbólumára, egészen addig, amíg csak új kanonikus halmazt kapunk.

Az így létrehozott

$$\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_m$$

halmazokat nevezzük a G nyelvtan *LR*(1)-kanonikus halmazainak.

Mivel egy nyelvtanra az *LR*(1)-elemek darabszáma véges, az *LR*(1)-kanonikus halmazok létrehozása biztosan véges lépésben befejeződik.

A G nyelvtan kanonikus halmazait a következő algoritmus állítja elő:

KANONIKUS-HALMAZOKAT-KÉSZÍT(G)

```
 1 i ← 0
 2 H_i ← ELEM-LEZÁR([S' → .S, #])
 3 I ← {H_i}, K ← {H_i}
 4 repeat
 5         L ← K
 6         for minden M ∈ I-re
 7             do I ← I \ M
 8                 for minden X ∈ T ∪ N-re
 9                     do J ← ELEMHALMAZ-LEZÁR
                                (ELEMHALMAZ-OLVAS(M, X))
10                        if J ≠ ∅ és J ∉ K
11                           then i ← i + 1
12                                H_i ← J
13                                K ← K ∪ {H_i}
14                                I ← I ∪ {H_i}
15 until K = L
16 return K
```

Az algoritmus a K-ban adja a kanonikus halmazokat, a 2. sorban

látható, hogy az első kanonikus halmaz a $\mathcal{H}_0$ lesz. További halmazokat a már meglévő kanonikus halmazokból az ELEMHALMAZ-LEZÁR(ELEMHALMAZ-OLVAS) függvénnyel képezünk a 9. sorban. A 10. sor programja azt vizsgálja, hogy ez az új halmaz vajon különbözik-e az eddigiektől, és ha igen, akkor a 11–12. sorban ez a halmaz egy új kanonikus halmaz lesz. A 6–14. sorok for ciklusa azt biztosítja, hogy ezeket a műveleteket a már meglévő minden kanonikus halmazra elvégezzük. A 3–14. sorokban lévő repeat ciklus szerint a kanonikus halmazok generálását addig végezzük, amíg új kanonikus halmazokat kapunk.

Példa.

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

(0) $S' \to S$
(1) $S \to AA$
(2) $A \to aA$
(3) $A \to b$

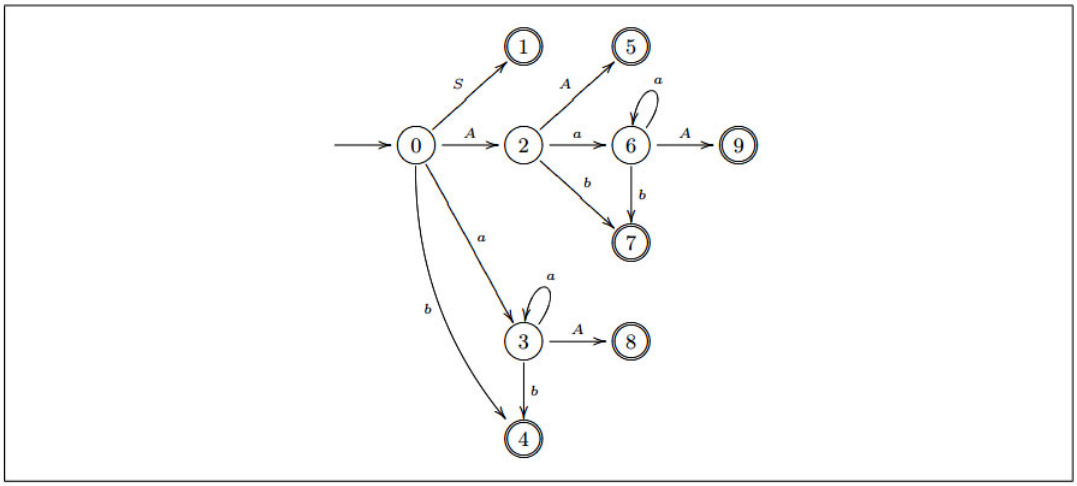
LR(1)**-elemeinek kanonikus halmazai a következők:**

$$
\begin{aligned}
\mathcal{H}_0 &= \textbf{\textit{closure}}([S' \to .S, \#]) &&= \{[S' \to .S, \#], [S \to .AA, \#], \\
&&&\quad [A \to .aA, a/b], [A \to .b, a/b]\} \\
\mathcal{H}_1 = \textbf{\textit{read}}(\mathcal{H}_0, S) &= \textbf{\textit{closure}}([S' \to S., \#]) &&= \{[S' \to S., \#]\} \\
\mathcal{H}_2 = \textbf{\textit{read}}(\mathcal{H}_0, A) &= \textbf{\textit{closure}}([S' \to A.A, \#]) &&= \{[S \to A.A, \#], [A \to .aA, \#], \\
&&&\quad [A \to .b, \#]\} \\
\mathcal{H}_3 = \textbf{\textit{read}}(\mathcal{H}_0, a) &= \textbf{\textit{closure}}([A \to a.A, a/b]) &&= \{[A \to a.A, a/b], [A \to .aA, a/b], \\
&&&\quad [A \to .b, a/b]\} \\
\mathcal{H}_4 = \textbf{\textit{read}}(\mathcal{H}_0, b) &= \textbf{\textit{closure}}([A \to b., a/b]) &&= \{[A \to b., a/b]\} \\
\mathcal{H}_5 = \textbf{\textit{read}}(\mathcal{H}_2, A) &= \textbf{\textit{closure}}([S \to AA., \#]) &&= \{[S \to AA., \#]\} \\
\mathcal{H}_6 = \textbf{\textit{read}}(\mathcal{H}_2, a) &= \textbf{\textit{closure}}([A \to a.A, \#]) &&= \{[A \to a.A, \#], [A \to .aA, \#], \\
&&&\quad [A \to .b, \#]\} \\
\mathcal{H}_7 = \textbf{\textit{read}}(\mathcal{H}_2, b) &= \textbf{\textit{closure}}([A \to b., \#]) &&= \{[A \to b., \#]\} \\
\mathcal{H}_8 = \textbf{\textit{read}}(\mathcal{H}_3, A) &= \textbf{\textit{closure}}([A \to aA., a/b]) &&= \{[A \to aA., a/b]\} \\
\textbf{\textit{read}}(\mathcal{H}_3, a) &= \mathcal{H}_3 \\
\textbf{\textit{read}}(\mathcal{H}_3, b) &= \mathcal{H}_4 \\
\mathcal{H}_9 = \textbf{\textit{read}}(\mathcal{H}_6, A) &= \textbf{\textit{closure}}([A \to aA., \#]) &&= \{[A \to aA., \#]\} \\
\textbf{\textit{read}}(\mathcal{H}_6, a) &= \mathcal{H}_6 \\
\textbf{\textit{read}}(\mathcal{H}_6, b) &= \mathcal{H}_7
\end{aligned}
$$

Megjegyzés:
$\mathcal{H}_0$ esetében a számítások a következők:

$\mathcal{H}_0 =$ ***closure***$([S' \to .S, \#])$

Mivel van $S \to AA$ szabály, ezért $[S \to .AA, \#]$ is bekerül a halmazba. Ez utóbbi miatt bekerül $[A \to .aA, a]$ és $[A \to .aA, b]$. Ez utóbbi kettő röviden így írható: $[A \to .aA, a/b]$
Hasonlóképpen kerül be: $[A \to .b, a/b]$



A példa járható prefixeit felismerő automata.

Az *LR*(1) elemző

Ha egy G' kiegészített nyelvtanhoz meghatároztuk az *LR*(1)-elemek

$$\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_m$$

kanonikus halmazait, akkor egy automata k állapotához rendeljük hozzá a $\mathcal{H}_k$ halmazt. Az automata állapotai és az *LR*(1)-elemek kanonikus halmazai közötti kapcsolatot a következő, az *LR*(1)-*elemzés nagy tételének* is nevezett állítás mondja ki:

Tétel. *Egy γ járható prefixre érvényes LR(1)-elemek halmaza az a $\mathcal{H}_k$ kanonikus elemhalmaz, amelyik az elemző véges determinisztikus automatájának ahhoz a k állapotához tartozik, amelyikbe az automata a kezdőállapotból a γ hatására kerül.*

A tétel azt mondja ki, hogy a járható perfixeket felismerő automata felépíthető a kanonikus halmazok ismeretében. Állítsuk elő tehát az *LR*(1)-elemek kanonikus halmazaiból az *LR*(1) elemzőt.

A járható prefixeket felismerő determinisztikus véges automata leírható egy táblázattal, ezt *LR*(1) *elemző táblázatnak* nevezzük. A táblázat sorait az automata állapotaihoz rendeljük hozzá.

Az elemző táblázat két részből áll. Az első neve az action táblázat. Mivel az elemezendő szöveg szimbóluma határozza meg az elvégzendő műveletet, az action táblázatot oszlopokra bontjuk, és az oszlopokhoz a terminális szimbólumokat rendeljük. Az action táblázat azt tartalmazza, hogy az adott állapotban, ha az oszlophoz tartozó terminális szimbólum a bemenő jel, léptetést vagy redukciót kell-e végrehajtani. A léptetés műveletét jelöljük sj-vel, ahol s a léptetést, j a léptetés utáni állapotot jelenti. A redukció jele legyen ri, ahol i az alkalmazott helyettesítési szabály sorszáma. Mivel a nulladik szabály szerinti redukció azt jelenti, hogy elemzés befejeződött és az elemzett szöveg szintaktikusan helyes, jelöljük ezt a táblázatban az elfogad szóval.

A második rész a goto táblázat. Ebbe az az információ kerül, hogy a nemterminális szimbólumok hatására az automata egy adott állapotból melyik állapotba megy át. (A terminális szimbólumok állapot-átmeneteit az action táblázat sj bejegyzései tartalmazzák.)

Az automata állapotainak halmaza legyen a $\{0, 1, \ldots, m\}$ halmaz, az elemző táblázatok i-edik sorát a $\mathcal{H}_i$ *LR*(1)-elemeiből töltjük ki.

Az *action* táblázat i-edik sora:

- ha $[A \to \alpha.a\beta, b] \in \mathcal{H}_i$ és ***read***$(\mathcal{H}_i, a) = \mathcal{H}_j$, akkor legyen ***action***$[i, a] = sj$,
- ha $[A \to \alpha., a] \in \mathcal{H}_i$ és $A \neq S'$, akkor legyen ***action***$[i, a] = rl$, ahol az $A \to \alpha$ a nyelvtan l-edik szabálya,
- ha $[S' \to S., \#] \in \mathcal{H}_i$, akkor legyen ***action***$[i, \#] =$ ***elfogad***.

A *goto* táblázat kitöltésének módszere:

- ha ***read***$(\mathcal{H}_i, A) = \mathcal{H}_j$, akkor legyen ***goto***$[i, A] = j$.
- Mindkét táblázatban az üresen maradt helyeket a *hiba* szöveggel töltsük ki.

Az $LR(1)$-elemek kanonikus halmazaiból létrehozott *action* és *goto* táblázatokat $LR(1)$ vagy *kanonikus elemző táblázatoknak* nevezzük.

Tétel. *A G' kiegészített nyelvtan akkor és csak akkor $LR(1)$ nyelvtan, ha a nyelvtanhoz készített kanonikus elemző táblázatok kitöltése konfliktusmentes.*

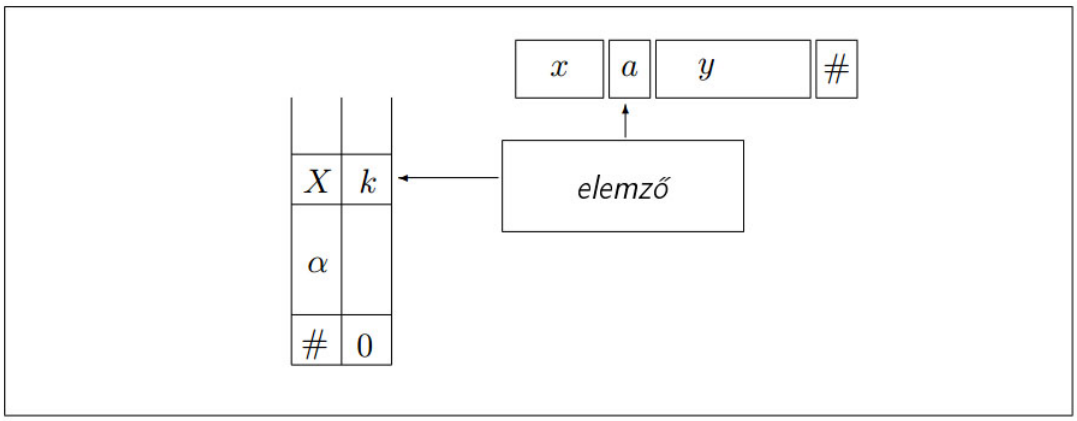
A táblázat kitöltését a következő algoritmussal végezhetjük:

LR(1)-TÁBLÁZATOT-KITÖLT(G)

```
 1  for minden H_i LR(1) kanonikus halmazra
 2      do for minden LR(1)-elemre
 3             if [A → α.aβ, b] ∈ H_i és read(H_i, a) = H_j
 4                then action[i, a] = sj
 5             if [A → α., a] ∈ H_i és A ≠ S' és A → α az l-edik szabály
 6                then action[i, a] = rl
 7             if [S' → S., #] ∈ H_i
 8                then action[i, #] = elfogad
 9             if read(H_i, A) = H_j
10                then goto[i, A] = j
11      for minden a ∈ (T ∪ {#})
12          do if action[i, a] = „üres”
13                then action[i, a] ← hiba
14      for minden X ∈ N
15          do if goto[i, X] = „üres”
16             then goto[i, X] ← hiba
17  return action, goto
```

A táblázatokat soronként töltjük ki, a 2–6. sorokban az *action* táblázatot, a 9–10. sorokban a *goto* táblázatot. Az algoritmus 11–13. soraiban a táblázatok sorainak üresen maradt helyeire a szintaktikai hibát jelző *hiba* szöveget írjuk.

Az *LR*(1) elemző működése a következőképpen adható meg:



Az $LR(1)$ elemző szerkezete.

Az elemző verme egy „dupla verem”, azaz egy push vagy pop művelettel két információt írunk vagy olvasunk. A verem szimbólumpárokat tartalmaz, a párok első elemében egy terminális vagy nemterminális szimbólumot tárolunk, a második elemben pedig az automata állapotának sorszámát. A verem kezdeti tartalma legyen $\#0$.

Az elemző állapotát egy kettőssel írjuk le, a kettős első eleme legyen a verem tartalma, a második elem pedig a bemenő szimbólumsorozat még nem elemzett része. Az elemző kezdőállapota tehát $(\#0, z\#)$, ahol z az elemezendő szimbólumsorozat. Az elemzés sikeresen befejeződik, azaz az elemző a végállapotba kerül, ha a verem tartalma ismét $\#0$, és az elemzéssel az elemezendő szimbólumsorozat végére értünk.

Tegyük fel, hogy az elemző pillanatnyi állapota a $(\#0 \dots Y_k i_k, ay\#)$ kettőssel írható le. Ekkor az elemző következő lépését az ***action***$[i_k, a]$ adat határozza meg.

Az állapotátmenetek a következők:

- Ha ***action***$[i_k, a] = sl$, azaz az automata egy léptetést hajt végre, akkor a bemenet soron következő a szimbóluma és az új állapot i_l sorszáma kerüljön a verembe, azaz

$$(\#0 \dots Y_k i_k, ay\#) \to (\#0 \dots Y_k i_k a i_l, y\#).$$

- Ha ***action***$[i_k, a] = rl$, akkor az l-edik szabály, az $A \to \alpha$ szabály szerint kell redukálni. Először töröljük a verem $|\alpha|$ darab sorát, azaz $2|\alpha|$ elemét. Ezután határozzuk meg a *goto* táblázatból, hogy az automata a törlés után a verem tetejére kerülő állapotból az A hatására melyik állapotba kerül, majd az A szimbólumot és a meghatározott állapotsorszámot írjuk be a verembe.

$$(\#0 \dots Y_{k-r} i_{k-r} Y_{k-r+1} i_{k-r+1} \dots Y_k i_k, y\#) \to$$
$$(\#0 \dots Y_{k-r} i_{k-r} A i_l, y\#),$$

  ahol $|\alpha| = r$, és ***goto***$[i_{k-r}, A] = i_l$.

- Ha ***action***$[i_k, a] =$ ***elfogad***, akkor az elemzés a veremből való törlés után befejeződik, az elemző az elemzett szöveget elfogadja.
- Ha ***action***$[i_k, a] =$ ***hiba***, akkor az elemzés befejeződik, az elemző az elemzett szövegben az a szimbólumnál egy *szintaktikai hibát* detektált.

Az *LR*(1) elemzőt gyakran *kanonikus LR*(1) *elemzőnek* is nevezik.

Ha T-vel jelöljük az *action* és *goto* táblákat, az elemző működésére a következő algoritmust adhatjuk meg.

LR(1)-ELEMEZ$(xay\#, T)$

```
 1  s ← (#0, xay#), s' ← elemez
 2  repeat
 3          s = (#0 ... Y_{k-r} i_{k-r} Y_{k-r+1} i_{k-r+1} ... Y_k i_k, ay#)
 4          if action[i_k, a] = sl
 5             then s ← (#0 ... Y_k i_k a i_l, y#)
 6             else if action[i_k, a] = rl és A → α az l-edik szabály és
 7                     |α| = r és goto[i_{k-r}, A] = i_l
 8                     then s ← (#0 ... Y_{k-r} i_{k-r} A i_l, ay#)
 9                     else if action[i_k, a] = elfogad
10                             then s' ← O.K.
11                             else  s' ← HIBA
12  until s' = O.K. vagy s' = HIBA
13  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 *O.K.* vagy *HIBA*. A 3. sorban az elemző állapotát írjuk fel részletesen, erre majd a 6–8. sorokban levő művelet esetén lesz szükség. Az elemző az automatának a verem tetején levő x_k állapota és az a aktuális szimbólum alapján a *action* táblázatból meghatározza az elvégzendő műveletet. A 4–5. sorban a léptetés műveletét hajtjuk végre, a 6–8. sorokban a redukálás művelete található. Az algoritmus a 9–11. sorban az elemzés befejezését jelzi, ha az elemezendő szöveg végére ért és a verem tetején a 0 állapot van, akkor az elemzett szöveg helyes, egyebként az elemző egy szintaktikai hibát fedezett fel. Az algoritmus végeredménye ennek megfelelően az *O.K.* vagy

HIBA **jelzés, és kimenetként mindkét esetben megjelenik az elemző állapota is. szintaktikai hiba esetén az elemző állapot második elemének első szimbóluma a hiba helyét adja meg.**

Példa.

A következő nyelvtan

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

(0) $S' \to S$
(1) $S \to AA$
(2) $A \to aA$
(3) $A \to b$

A nyelvtan ***LR***(1)**-elemzőjének** ***action*** **és** ***goto*** **táblázatai a következők, az üres helyek most is a** ***hibát*** **jelentik.**

állapot	*action*			*goto*	
	a	b	#	S	A
0	$s3$	$s4$		1	2
1			*elfogad*		
2	$s6$	$s7$			5
3	$s3$	$s4$			8
4	$r3$	$r3$			
5			$r1$		
6	$s6$	$s7$			9
7			$r3$		
8	$r2$	$r2$			
9			$r2$		

(0) $S' \to S$
(1) $S \to AA$
(2) $A \to aA$
(3) $A \to b$

állapot	*action*			*goto*	
	a	b	$\#$	S	A
0	$s3$	$s4$		1	2
1			*elfogad*		
2	$s6$	$s7$			5
3	$s3$	$s4$			8
4	$r3$	$r3$			
5			$r1$		
6	$s6$	$s7$			9
7			$r3$		
8	$r2$	$r2$			
9			$r2$		

(0) $S' \to S$
(1) $S \to AA$
(2) $A \to aA$
(3) $A \to b$

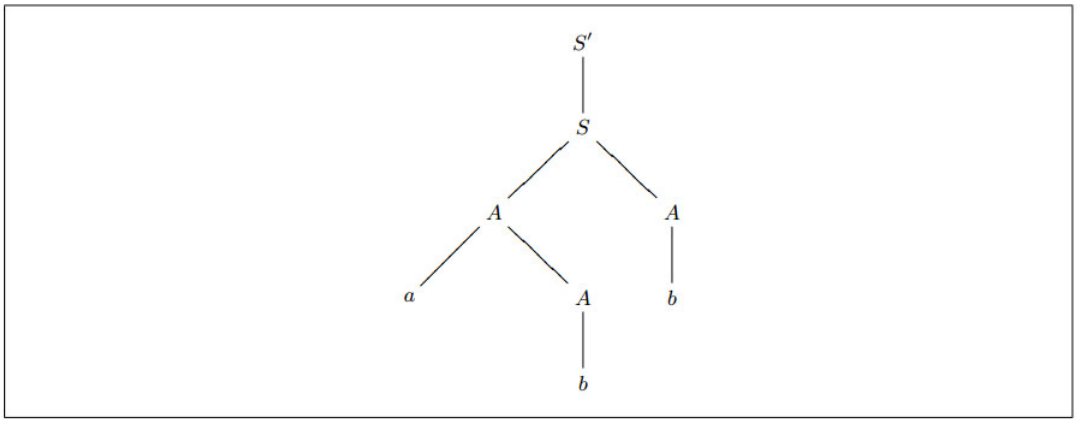
Példa.

Elemezzük az $abb\#$ szöveget.

				szabály
$(\#0,$	$abb\#)$	$\xrightarrow{s3}$	$(\#0a3,\quad bb\#)$	
		$\xrightarrow{s4}$	$(\#0a3b4,\quad b\#)$	
		$\xrightarrow{r3}$	$(\#0a3A8,\quad b\#)$	$A \to b$
		$\xrightarrow{r2}$	$(\#0A2,\quad b\#)$	$A \to aA$
		$\xrightarrow{s7}$	$(\#0A2b7,\quad \#)$	
		$\xrightarrow{r3}$	$(\#0A2A5,\quad \#)$	$A \to b$
		$\xrightarrow{r1}$	$(\#0S1,\quad \#)$	$S \to AA$
		$\xrightarrow{\textit{elfogad}}$	*O.K.*	

Az abb levezetése: $S' \Rightarrow S \Rightarrow AA \Rightarrow Ab \Rightarrow aAb \Rightarrow abb$

Az elemzett mondat szintaxisfája:



Az abb mondat szintaxisfája.

2. példa. bba **elemzése.**

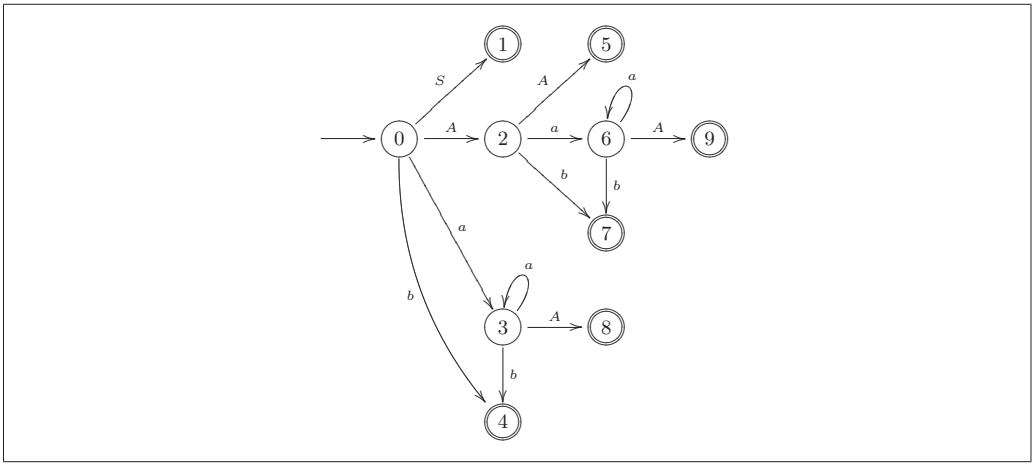
				szabály
$(\#0,$	$bba\#)$	$\xrightarrow{s4}$	$(\#0b4,\quad ba\#)$	
		$\xrightarrow{r3}$	$(\#0A2,\quad ba\#)$	$A \to b$
		$\xrightarrow{s7}$	$(\#0A2b7,\quad a\#)$	
		$\xrightarrow{\textit{hiba}}$	***HIBA***	

3. példa. $abab$ **elemzése.**

					szabály
$(\#0,$	$abab\#)$	$\xrightarrow{s3}$	$(\#0a3,$	$bab\#)$	
		$\xrightarrow{s4}$	$(\#0a3b4,$	$ab\#)$	
		$\xrightarrow{r3}$	$(\#0a3A8,$	$ab\#)$	$A \to b$
		$\xrightarrow{r2}$	$(\#0A2,$	$ab\#)$	$A \to aA$
		$\xrightarrow{s6}$	$(\#0A2a6,$	$b\#)$	
		$\xrightarrow{s7}$	$(\#0A2a6b7,$	$\#)$	
		$\xrightarrow{r3}$	$(\#0A2a6A9,$	$\#)$	$A \to b$
		$\xrightarrow{r2}$	$(\#0A2A5,$	$\#)$	$A \to aA$
		$\xrightarrow{r1}$	$(\#0S1,$	$\#)$	$S \to AA$
		$\xrightarrow{\textit{elfogad}}$	***O.K.***		

(0) $S' \to S$
(1) $S \to AA$
(2) $A \to aA$
(3) $A \to b$

	action			*goto*	
	a	b	$\#$	S	A
0	$s3$	$s4$		1	2
1					
2	$s6$	$s7$			5
3	$s3$	$s4$			8
4	$r3$	$r3$			
5			$r1$		
6	$s6$	$s7$			9
7			$r3$		
8	$r2$	$r2$			
9			$r2$		



Rövid összefoglaló példával

Legyen a $\mathcal{H}$ halmaz egy nyelvtan egy ***LR***(1)-elemhalmaza. Ekkor a ***closure***$(\mathcal{H})$ halmaz a következő ***LR***(1)-elemeket tartalmazza:

1. a $\mathcal{H}$ halmaz minden eleme legyen eleme a ***closure***$(\mathcal{H})$ halmaznak is,
2. ha $[A \to \alpha.B\beta, a] \in$ ***closure***$(\mathcal{H})$ és $B \to \gamma$ a nyelvtan egy helyettesítési szabálya, akkor legyen $[B \to .\gamma, b] \in$ ***closure***$(\mathcal{H})$ minden $b \in$ ***First***(βa)-ra,
3. a ***closure***$(\mathcal{H})$ halmazt a 2. pontban leírt művelettel addig kell bővíteni, ameddig az lehetséges.

Legyen a $\mathcal{H}$ halmaz egy nyelvtan egy ***LR***(1)-elemhalmaza. Ekkor a ***read***$(\mathcal{H}, X)$ ($X \in (N \cup T)$) halmaz a következő ***LR***(1)-elemeket tartalmazza:

1. ha $[A \to \alpha.X\beta, a] \in \mathcal{H}$, akkor a ***closure***$([A \to \alpha X.\beta, a])$ minden eleme legyen a ***read***$(\mathcal{H}, X)$ halmaz eleme,
2. a ***read***$(\mathcal{H}, X)$ halmazt az 1. művelettel addig kell bővíteni, ameddig az lehetséges.

Példa.

(0) $S' \to S$
(1) $S \to A$
(2) $A \to aA$
(3) $A \to b$

$$\mathcal{H}_0 = \textsf{closure}([S' \to .S, \#]) = \left\{[S' \to .S, \#],\ [S \to .A, \#],\ [A \to .aA, \#],\ [A \to .b, \#]\right\}$$

$$\mathcal{H}_1 = \textsf{read}(\ \mathcal{H}_0, S\ ) = \textsf{closure}([S' \to S., \#]) = \left\{[S' \to S., \#]\right\}$$

$$\mathcal{H}_2 = \textsf{read}(\mathcal{H}_0, A) = \textsf{closure}([S \to A., \#]) = \left\{[S \to A., \#]\right\}$$

$$\mathcal{H}_3 = \textsf{read}(\mathcal{H}_0, a) = \textsf{closure}([A \to a.A, \#]) = \left\{[A \to a.A, \#],\ [A \to .aA, \#],\ [A \to .b, \#]\right\}$$

$$\mathcal{H}_4 = \textsf{read}(\mathcal{H}_0, b) = \textsf{closure}([A \to b., \#]) = \left\{[A \to b., \#]\right\}$$

$$\mathcal{H}_5 = \textsf{read}(\mathcal{H}_3, A) = \textsf{closure}([A \to aA., \#]) = \left\{[A \to aA., \#]\right\}$$

$$\textsf{read}(\mathcal{H}_3, a) = \textsf{closure}([A \to a.A, \#]) = \left\{[A \to a.A, \#],\ [A \to .aA, \#],\ [A \to .b, \#]\right\} = \mathcal{H}_3$$

$$\textsf{read}(\mathcal{H}_3, b) = \textsf{closure}([A \to b., \#]) = \left\{[A \to b., \#]\right\} = \mathcal{H}_4$$

Az action táblázat i-edik sora:

- **ha** $[A \to \alpha.a\beta, b] \in \mathcal{H}_i$ **és** ***read***$(\mathcal{H}_i, a) = \mathcal{H}_j$, **akkor legyen** ***action***$[i, a] = sj$,
- **ha** $[A \to \alpha., a] \in \mathcal{H}_i$ **és** $A \neq S'$, **akkor legyen** ***action***$[i, a] = rl$, **ahol az** $A \to \alpha$ **a nyelvtan** l**-edik szabálya,**
- **ha** $[S' \to S., \#] \in \mathcal{H}_i$, **akkor legyen** ***action***$[i, \#] =$ ***elfogad***.

A goto táblázat kitöltésének módszere:

- **ha** ***read***$(\mathcal{H}_i, A) = \mathcal{H}_j$, **akkor legyen** ***goto***$[i, A] = j$.

Példánkban

Mivel $[A \to .aA, \#] \in \mathcal{H}_0$ **és read**$(\mathcal{H}_0, a) = \mathcal{H}_3$, **action**$[0, a] = s3$.
Mivel $[A \to .b, \#] \in \mathcal{H}_0$ **és read**$(\mathcal{H}_0, b) = \mathcal{H}_4$, **action**$[0, b] = s4$.
Mivel $[A \to .aA, \#] \in \mathcal{H}_3$ **és read**$(\mathcal{H}_3, a) = \mathcal{H}_3$, **action**$[3, a] = s3$.
Mivel $[A \to .b, \#] \in \mathcal{H}_3$ **és read**$(\mathcal{H}_3, b) = \mathcal{H}_4$, **action**$[3, b] = s4$.
Mivel $[S' \to S., \#] \in \mathcal{H}_1$, **action**$[1, \#] = elfogad$.

Mivel $[S \to A., \#] \in \mathcal{H}_2$, **action**$[2, \#] = r1$ **(mivel a szabály a 1.).**
Mivel $[A \to b., \#] \in \mathcal{H}_4$, **action**$[4, \#] = r3$ **(mivel a szabály a 3.).**
Mivel $[A \to aA., \#] \in \mathcal{H}_5$, **action**$[5, \#] = r2$ **(mivel a szabály a 2.).**

read$(\mathcal{H}_0, S) = \mathcal{H}_1$, **ezért goto[0,S] = 1.**
read$(\mathcal{H}_0, A) = \mathcal{H}_2$, **ezért goto[0,A] = 2.**
read$(\mathcal{H}_3, A) = \mathcal{H}_5$, **ezért goto[3,A] = 5.**

	action			*goto*	
	a	b	$\#$	S	A
0	$s3$	$s4$		1	2
1			$elfogad$		
2			$r1$		
3	$s3$	$s4$			5
4			$r3$		
5			$r2$		

				szabály
$(\#0,$	$aab\#)$	$\xrightarrow{s3}$	$(\#0a3,\quad ab\#)$	
		$\xrightarrow{s3}$	$(\#0a3a3,\quad b\#)$	
		$\xrightarrow{s4}$	$(\#0a3a3b4,\quad \#)$	
		$\xrightarrow{r3}$	$(\#0a3a3A5,\quad \#)$	$A \to b$
		$\xrightarrow{r2}$	$(\#0a3A5,\quad \#)$	$A \to aA$
		$\xrightarrow{r2}$	$(\#0A2,\quad \#)$	$A \to aA$
		$\xrightarrow{r1}$	$(\#0S1,\quad \#)$	$S \to A$
		$\xrightarrow{\textbf{elfogad}}$	***O.K.***	

Levezetés: $S \Rightarrow A \Rightarrow aA \Rightarrow aaA \Rightarrow aab$