Szintaktikus elemzési módszerek

Fülöp Zoltán
fulopinf.u-szeged.hu

Szegedi Tudományegyetem
Számítástudomány Alapjai Tanszék
6720 Szeged, Árpád tér 2

Ajánlott irodalom

- A. V. Aho, J. D. Ullman, The Theory of Parsing, Translation and Compiling I.,II., Prentice Hall, Englewood Cliffs, New Jersey, 1972.
- Csörnyei Zoltán, Fordítóprogramok elemzési algoritmusai (Informatikai Algoritmusok 2., 20. fej, Szerk. Iványi Antal), ELTE Eötvös Kiadó, 2005.
- Fülöp Zoltán, Formális nyelvek és szintaktikus elemzésük, Polygon Kiadó, SZTE, 1999, bővített kiadás 2004.
- S. Sippu, E. Soisalon-Soininen: Parsing Theory, Springer-Verlag, 1988.

Szintaxis megadása

HOGYAN, MILYEN MÓDSZERREL ADHATÓ MEG PROGRAMOZÁSI NYELVEK SZINTAXISA?

A legelterjedtebb módszer a *generatív nyelvtannal* történő szintaxis megadás.

Adjuk meg az A, B és C változókból, a 0 és 1 konstansokból, a $+$ és a $*$ műveleti jelekből, valamint a (és) zárójelekből felépíthető aritmetikai kifejezések szintaxisát!

Ilyenek például az A, 1, $A+1$, $A+B$, $A*(B+1)$ aritmetikai kifejezések.

1. Példa: *Aritmetikai kifejezéseknek* az A, B és C változó jelekből, a 0 és 1 konstans jelekből, a $+$ és $*$ műveleti jelekből, valamint a (és) csoportosító jelekből, a

$$
\begin{array}{lcl}
\langle kif \rangle & \rightarrow & \langle tag \rangle \,|\, \langle kif \rangle + \langle tag \rangle \\
\langle tag \rangle & \rightarrow & \langle fakt \rangle \,|\, \langle tag \rangle * \langle fakt \rangle \\
\langle fakt \rangle & \rightarrow & (\langle kif \rangle) \,|\, \langle valt \rangle \,|\, \langle konst \rangle \\
\langle valt \rangle & \rightarrow & A \,|\, B \,|\, C \\
\langle konst \rangle & \rightarrow & 0 \,|\, 1
\end{array}
$$

szabályok alkalmazásával felépíthető jelsorozatokat (szavakat) nevezzük.

Ez egy *környezetfüggetlen nyelvtan.*

Levezetés:

$$
\begin{aligned}
& \underline{\langle kif \rangle} \\
\Rightarrow\ & \underline{\langle tag \rangle} \\
\Rightarrow\ & \langle tag \rangle * \underline{\langle fakt \rangle} \\
\Rightarrow\ & \langle tag \rangle * (\underline{\langle kif \rangle}) \\
\Rightarrow\ & \langle tag \rangle * (\underline{\langle kif \rangle} + \langle tag \rangle) \\
\Rightarrow\ & \underline{\langle tag \rangle} * (\underline{\langle tag \rangle} + \langle tag \rangle) \\
\Rightarrow\ & \langle fakt \rangle * (\underline{\langle tag \rangle} + \langle tag \rangle) \\
\Rightarrow\ & \langle fakt \rangle * (\langle fakt \rangle + \underline{\langle tag \rangle}) \\
\Rightarrow\ & \langle fakt \rangle * (\langle fakt \rangle + \langle fakt \rangle)
\end{aligned}
$$

Levezetés folytatása:

$$
\begin{aligned}
\Rightarrow\ & \underline{\langle fakt \rangle} * (\langle fakt \rangle + \langle fakt \rangle) \\
\Rightarrow\ & \langle valt \rangle * (\underline{\langle fakt \rangle} + \langle fakt \rangle) \\
\Rightarrow\ & \langle valt \rangle * (\langle valt \rangle + \underline{\langle fakt \rangle}) \\
\Rightarrow\ & \underline{\langle valt \rangle} * (\langle valt \rangle + \underline{\langle konst \rangle}) \\
\Rightarrow\ & A * (\underline{\langle valt \rangle} + \langle konst \rangle) \\
\Rightarrow\ & A * (B + \underline{\langle konst \rangle}) \\
\Rightarrow\ & A * (B + 1)
\end{aligned}
$$

Jelölés: $\langle kif \rangle \Rightarrow^* A * (B + 1)$

Az $A, B, C, 0, 1, +, *, (\text{ és }) $ jelekből álló aritmetikai kifejezés szintaxisa:

egy jelsorozat (vagy szó) akkor és csak akkor aritmetikai kifejezés, ha a $\langle kif \rangle$-ből a fenti szintaktikai szabályok alkalmazásával történő levezetéssel megkapható.

Röviden:

w szó aritmetikai kifejezés $\Leftrightarrow$ $\langle kif \rangle \Rightarrow^* w$.

2. Példa: Egy egyszerű programozási nyelv, a PÉLDA szintaxisa:

$$
\begin{array}{lcl}
\langle program \rangle & \rightarrow & \langle ut.lista \rangle . \\
\langle ut.lista \rangle & \rightarrow & \langle ut \rangle \mid \langle ut \rangle ; \langle ut.lista \rangle \\
\langle ut \rangle & \rightarrow & \langle ert.ado \rangle \mid \langle ifut \rangle \mid \\
 & & \langle whileut \rangle \mid \langle blokk \rangle \\
\langle ert.ado \rangle & \rightarrow & \langle valt \rangle := \langle kif \rangle \\
\langle ifut \rangle & \rightarrow & \textbf{if } \langle relacio \rangle \textbf{ then } \langle ut \rangle \textbf{ else } \langle ut \rangle \\
\langle whileut \rangle & \rightarrow & \textbf{while } \langle relacio \rangle \textbf{ do } \langle ut \rangle \\
\langle blokk \rangle & \rightarrow & \textbf{begin } \langle ut.lista \rangle \textbf{ end} \\
\langle relacio \rangle & \rightarrow & \langle kif \rangle \langle relaciojel \rangle \langle kif \rangle \\
\langle relaciojel \rangle & \rightarrow & < \mid > \mid = \mid \neq
\end{array}
$$

Továbbá, a már ismert:

$$
\begin{array}{lcl}
\langle kif \rangle & \rightarrow & \langle tag \rangle \,|\, \langle kif \rangle + \langle tag \rangle \\
\langle tag \rangle & \rightarrow & \langle fakt \rangle \,|\, \langle tag \rangle * \langle fakt \rangle \\
\langle fakt \rangle & \rightarrow & (\langle kif \rangle) \,|\, \langle valt \rangle \,|\, \langle konst \rangle \\
\langle valt \rangle & \rightarrow & A \,|\, B \,|\, C \\
\langle konst \rangle & \rightarrow & 0 \,|\, 1
\end{array}
$$

Ez is egy *környezetfüggetlen nyelvtan.*

Egy w jelsorozat akkor és csakis akkor szintaktikusan helyes PÉLDA nyelvű program, ha $\langle program \rangle \Rightarrow^* w$. Ilyen például a következő jelsorozat:

$A := 0;$
while $A < C$ **do**
  begin $A := A + 1;$
    $B := B * C;$
  end;
$C := C * B.$

Kérdés: el tudjuk-e dönteni tetszőleges w esetén, hogy $\langle program \rangle \Rightarrow^* w$?

Általánosabban:

Amennyiben adott egy programozási nyelv szintaxisa és adott egy – ezen a nyelven írott – program, akkor el tudjuk-e eldönteni, hogy az adott program engedelmeskedik-e a szintaxisnak, vagyis szintaktikusan helyes-e?

Még általánosabban:

Adott: A PL programozási nyelv szintaxisa:

$$G_{PL} = (N, \Sigma, P, S)$$

cf nyelvtan és egy (felhasználó által megírt) PL nyelvű w program (szó, sztring).

Kérdés: A w program megfelel-e a szintaktikai előírásoknak?

Tudjuk: A w program szintaktikusan helyes $\Leftrightarrow w \in L(G_{PL})$.

Tehát a kérdés: Teljesül-e, hogy $w \in L(G_{PL})$?

Az elemzés alapfeladata:

Adjunk meg olyan algoritmust, amely tetszőleges $G = (N, \Sigma, P, S)$ cf nyelvtan és $w \in \Sigma^*$ szó esetén eldönti, hogy $w \in L(G)$ teljesül-e!

Továbbá, ha $w \in L(G)$, akkor az algoritmus outputja még a w egy (bal oldali vagy jobb oldali) levezetését biztosító szabálysorozat.

Környezetfüggetlen nyelvtanok
– Ismétlés –

Környezetfüggetlen nyelvtan

Egy $G = (N, \Sigma, P, S)$ nyelvtan környezetfüggetlen, ha minden szabálya $A \to \alpha$ alakú, ahol $\alpha \in (N \cup \Sigma)^*$.

Példa: G_{ar}, melynek szabályai

- $K \to K + T$, $K \to T$,
- $T \to T * F$, $T \to F$,
- $F \to (K)$, $F \to a$.

$L(G_{ar})$ az a-ból valamint a $(,), +$ és $*$ jelekből képezhető aritmetikai kifejezések halmaza.

Jelölések

- Az a, b, c, d szimbólumok Σ elemeit,
- az A, B, C, D, S szimbólumok N elemeit,
- az U, V, W, X, Y, Z szimbólumok $N \cup \Sigma$ elemeit,
- az α, β, γ, δ szimbólumok $(N \cup \Sigma)^*$ elemeit,
- és az u, v, w, x, y, z szimbólumok Σ^* elemeit

fogják jelölni.

Levezetési (derivációs) módok

$G = (N, \Sigma, P, S)$, minden szabály $A \to \alpha$ alakú

Korlátozás nélküli derivváció (levezetés):
$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \Rightarrow \alpha_n$
Bal olodali derivváció
(α_i legbaloldalibb nemterminálisát helyettesítjük):
$\alpha_0 \Rightarrow_l \alpha_1 \Rightarrow_l \ldots \Rightarrow_l \alpha_n$
Jobb olodali derivváció
(α_i legjobboldalibb nemterminálisát helyettesítjük):
$\alpha_0 \Rightarrow_r \alpha_1 \Rightarrow_r \ldots \Rightarrow_r \alpha_n$

Példák

Az aritmetikai kifejezéseket generáló G_{ar} nyelvtanban

$$\begin{array}{llllll} \underline{K} & \Rightarrow & \underline{T} & \Rightarrow & \underline{T} * F & \Rightarrow & F * \underline{F} & \Rightarrow & F * (K) \\ \underline{K} & \Rightarrow_l & \underline{T} & \Rightarrow_l & \underline{T} * F & \Rightarrow_l & \underline{F} * F & \Rightarrow_l & a * F \\ \underline{K} & \Rightarrow_r & \underline{T} & \Rightarrow_r & T * \underline{F} & \Rightarrow_r & T * (\underline{K}) & \Rightarrow_r & T * (K + T) \end{array}$$

Ha α olyan, hogy $S \Rightarrow^* \alpha$ ($S \Rightarrow_l^* \alpha$, $S \Rightarrow_r^* \alpha$), akkor α *mondatforma (bal mondatforma, jobb mondatforma).*

A különböző levezetési módok kapcsolata

Ha $X \Rightarrow_l^* \alpha$ vagy $X \Rightarrow_r^* \alpha$, akkor $X \Rightarrow^* \alpha$ is fennáll.

Fordítva nem igaz, például G_{ar}-ban

$$K \Rightarrow \underline{K} + T \Rightarrow K + \underline{T} + T \Rightarrow K + F + T,$$

de sem $K \Rightarrow_l^* K + F + T$ sem $K \Rightarrow_r^* K + F + T$ nem teljesül.

Ellenben, ha α terminális szó ($\alpha \in \Sigma^*$), akkor akkor az állítás már megfordítható, vagyis igaz lesz, hogy bal oldali (jobb oldali) levezetésekkel ugyanazok a terminális szavak kaphatók meg mint korlátozás nélküli levezetésekkel.

Tétel. Minden $X \in (N \cup \Sigma)$ és $w \in \Sigma^*$ esetén:

$$X \Rightarrow^* w \quad \Leftrightarrow \quad X \Rightarrow_l^* w \quad \Leftrightarrow \quad X \Rightarrow_r^* w.$$

Derivációs fák

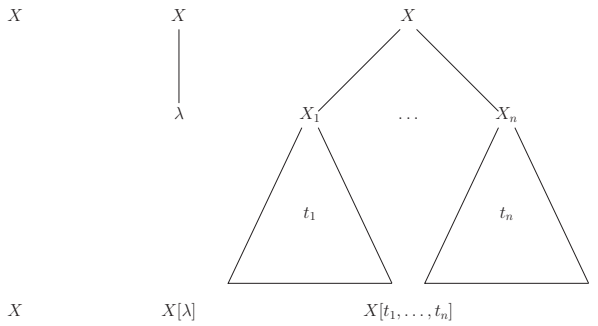
Az $X \in (N \cup \Sigma)$ gyökerű derivációs fák halmaza a legszűkebb olyan D_X halmaz, amelyre

(i) Az a fa, amelynek egyetlen szögpontja (vagyis csak gyökere) az X, eleme D_X-nek. (Ezt a fát X-szel jelöljük.)

(ii) Ha $X \to \lambda \in P$, akkor az a fa, amelynek gyökere X, a gyökerének egyetlen leszármazottja a λ, eleme D_X-nek. (Ezt a fát $X[\lambda]$-val jelöljük.)

(iii) Ha $X \to X_1 \dots X_n \in P$ és $t_1 \in D_{X_1}, \dots, t_n \in D_{X_n}$, akkor az a fa, amelynek gyökere X, a gyökérből n él indul rendre a $t_1, \dots, t_n$ fák gyökeréhez, eleme D_X-nek. (Ezt a fát $X[t_1, \dots, t_n]$-nel jelöljük.

Derivációs fák



Derivációs fák magassága, határa

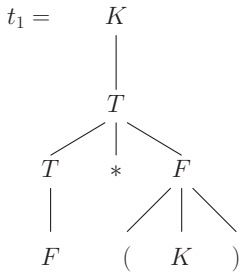
Legyen t egy X gyökerű derivációs fa. Akkor t magasságát $h(t)$-vel, t határát $fr(t)$-vel jelöljük és az alábbi módon definiáljuk:

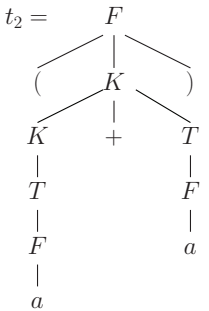
(i) Ha $t = X$, akkor $h(t) = 0$ és $fr(t) = X$.

(ii) Ha $t = X[\lambda]$, akkor $h(t) = 1$ és $fr(t) = \lambda$.

(iii) Ha $t = X[t_1, \dots, t_n]$, akkor $h(t) = 1 + max\{h(t_i) \mid 1 \le i \le n\}$ és $fr(t) = fr(t_1) \dots fr(t_n)$.

Informálisan: $h(t)$ a t-ben lévő olyan utak hosszának maximuma, amelyek t gyökerétől annak valamely leveléhez vezetnek. Továbbá, $fr(t)$ az az $(N \cup \Sigma)^*$-beli szó, amelyet t leveleinek balról jobbra történő leolvasásával kapunk.





$fr(t_1) = F * (K)$

$h(t_1) = 3$

$fr(t_2) = (a + a)$

$h(t_2) = 5$

Kapcsolat a derivációs fák és a derivációk között

Tétel Tetszőleges $X \in (N \cup \Sigma)$ és $\alpha \in (N \cup \Sigma)^*$ esetén

$$X \Rightarrow^* \alpha$$

akkor és csak akkor,

ha van olyan $t \in D_X$ derivációs fa amelyre $fr(t) = \alpha$.

Összefoglalás.

Egy $G = (N, \Sigma, P, S)$ cf nyelvtan által generált nyelv a most bevezetett fogalmak segítségével a következőképpen írható fel.

Levezetésekkel:

$$\begin{aligned} L(G) &= \{w \in \Sigma^* \,|\, S \Rightarrow^* w\} \\ &= \{w \in \Sigma^* \,|\, S \Rightarrow_l^* w\} \\ &= \{w \in \Sigma^* \,|\, S \Rightarrow_r^* w\}. \end{aligned}$$

Derivációs fákkal: $L(G) = \{fr(t) \,|\, t \in D_S,\ fr(t) \in \Sigma^*\}$.

Derivációkra és derivációs fákra vonatkozó megjegyzések

(a) Ha $X \Rightarrow^* \alpha$, akkor általában nem csak egy olyan X gyökerű derivációs fa létezik melynek határa α.

Például legyenek az

$A \ \rightarrow \ aB \,|\, Ab$
$A \ \rightarrow \ a$
$B \ \rightarrow \ b$

szabályok egy környezetfüggetlen nyelvtan szabályai.

Akkor $A \Rightarrow^* ab$. Ugyanakkor a $t_1 = A[a, B[b]]$ és a $t_2 = A[A[a], b]$ fákra $fr(t_1) = ab$ és $fr(t_2) = ab$

Derivációkra és derivációs fákra vonatkozó megjegyzések

(b) A $t \in D_X$ tartalmazásból csak az következik, hogy $X \Rightarrow^* \alpha$, de nem tudunk semmit deriváció lépéseiről.

Például a G_{ar} nyelvtan esetén $t = K[T[T[F], *, F[(, K,)]]]$ egy K gyökerű derivációs fa, melynek határa $F * (K)$. Tudjuk, hogy $K \Rightarrow^* F * (K)$.

Ugyanakkor ez a deriváció két különböző lépéssorozattal is elvégezhető:

$K \ \Rightarrow \ T \ \Rightarrow \ T * F \ \Rightarrow \ F * F \quad \Rightarrow \ F * (K)$
$K \ \Rightarrow \ T \ \Rightarrow \ T * F \ \Rightarrow \ T * (K) \ \Rightarrow \ F * (K)$

Deriválciókra és deriválciós fákra vonatkozó megjegyzések

(c) Ha azonban $t \in D_S$ és $fr(t) = x \in \Sigma^*$, vagyis t az x deriválciós fája, akkor t egyértelműen meghatározza x bal oldali (jobb oldali) levezetésének lépéseit.

Az is igaz, hogy ha t' az x egy másik deriválciós fája, akkor a t-ből és t'-ből kapott bal oldali (jobb oldali) levezetések különböznek.

Ezért, mivel x minden levezetése meghatározza a x egy deriválciós fáját, x bal oldali (jobb oldali) levezetései és deriválciós fái között bijekció áll fenn.

Egyértelmű nyelvtanok és nyelvek

Definíció. Egy G nyelvtan *egyértelmű*, ha minden $x \in L(G)$ szóhoz csak egy olyan $t \in D_S$ deriválciós fa létezik, melyre $fr(t) = x$.

Következmény. G akkor és csak akkor egyértelmű, ha minden $x \in L(G)$ szónak pontosan egy bal oldali (jobb oldali) levezetése van.

Bizonyítás. x bal oldali (jobb oldali) levezetései és deriválciós fái között bijekció áll fenn.

Egyértelmű nyelvtanok és nyelvek

Példa.

Tekintsük a következő, ugyancsak aritmetikai kifejezéseket generáló G'_{ar} nyelvtant:

- $K \to K + K$,
- $K \to K * K$,
- $K \to (K)$, $K \to a$.

Ekkor $L(G'_{ar}) = L(G_{ar})$, ugyanakkor G_{ar} egyértelmű, G'_{ar} pedig nem egyértelmű.

Egyértelmű nyelvtanok és nyelvek

Valóban, például az $a + a + a$ szónak G'_{ar}-ban két bal oldali levezetése is van:

$$\begin{aligned} K &\Rightarrow_l K + K \Rightarrow_l a + K && \Rightarrow_l a + K + K \\ & && \Rightarrow_l^2 a + a + a \\ K &\Rightarrow_l K + K \Rightarrow_l K + K + K && \Rightarrow_l a + K + K \\ & && \Rightarrow_l^2 a + a + a \end{aligned}$$

Tétel. Nem létezik olyan algoritmus, amely tetszőleges cf nyelvtanról eldönti, hogy egyértelmű-e.

A Chomsky normálforma

Definíció. Egy $G = (N, \Sigma, P, S)$ környezetfüggetlen nyelvtan *Chomsky-normálformájú* (vagy Chomsky normálalakban van), ha λ-mentes és P-ben csak $S \to \lambda$, $A \to BC$ és $A \to a$ alakú szabályok vannak.

Tétel. Minden $G = (N, \Sigma, P, S)$ környezetfüggetlen nyelvtanhoz van vele ekvivalens Chomsky-normálformájú $G' = (N', \Sigma, P', S)$ környezetfüggetlen nyelvtan.

E l e m z é s

Az elemzés definíciója

Definíció. Legyen adott egy $G = (N, \Sigma, P, S)$ környezetfüggetlen nyelvtan, melynek szabályai meg vannak számozva 1-től p-ig. Egy $\alpha \in (N \cup \Sigma)^*$ szó egy bal oldali (jobb oldali) elemzésének egy olyan $n_1 \dots n_k \in [1, p]^*$ sorozatot nevezünk, amely megadja α egy bal oldali (jobb oldali) levezetését. Például, a

(1) $K \to K + K$,

(2) $K \to K * K$,

(3) $K \to (K)$,

(4) $K \to a$.

nyelvtan estén 11444 és 14144 az $a + a + a$ bal oldali elemzései is és jobb oldali elemzései is.

A CYK elemzési algoritmus

Input: Egy w szó és egy Chomsky normálformában lévő $G = (N, \Sigma, P, S)$ cf nyelvtan.

Output: Ha $w \in L(G)$, akkor w egy bal oldali elemzése, különben "Nem".

Algoritmus: CYK = Cocke-Younger-Kasami algoritmus.

A CYK elemzési algoritmus

Előkészítés: Tfh, $|w| = n \geq 1$ és legyen w_{ij} a w i-edik pozíciójában kezdődő j hosszúságú rész-szava (tehát w_{i1} az i-edik betű).

Minden $j = 1, \ldots, n$-re és $i = 1, \ldots, n-j+1$-re meghatározzuk a $V_{ij} = \{A \in N \mid A \Rightarrow^* w_{ij}\}$ halmazt, a következő elv alapján:

$\underline{j = 1}$: $A \in V_{i1} \iff A \to w_{i1} \in P$.

$\underline{j > 1}$: $A \in V_{ij} \iff \exists 1 \leq k < j$ és $A \to BC \in P$, hogy $B \in V_{ik}$ és $C \in V_{i+k,j-k}$. (Dinamikus programozás.)

Ezután $w \in L(G)$, akkor és csak akkor, ha $S \in V_{1n}$.

A CYK elemzési algoritmus

Algoritmus a V_{ij} halmazok kiszámítására:

begin
for $i := 1$ **to** n **do** $V_{i1} = \{A \mid A \to w_{i1} \in P\}$;
for $j := 2$ **to** n **do**
for $i := 1$ **to** $n - j + 1$ **do**
begin $V_{ij} = \emptyset$;
for $k := 1$ **to** $j - 1$ **do**
$V_{ij} = V_{ij} \cup \{A \mid A \to BC \in P, B \in V_{ik}, C \in V_{i+k,j-k}\}$;
end
end

A CYK elemzési algoritmus

A V_{ij} halmazok kiszímításának

tárbonyolultsága : $O(n^2)$,

időbonyolultsága: $O(n^3)$ vagyis mindkettő polinomiális, ahol n az input szó hossza.

A CYK elemzési algoritmus

Példa.

(1) $S \to AA$
(2) $S \to AS$
(3) $S \to b$
(4) $S \to SA$
(5) $A \to AS$
(6) $A \to a$ $\qquad w = abaab$

Nem egyértelmű nyelvtan

A CYK elemzési algoritmus

Példa folytatása, $w = abaab$.

$V_{11} = \{A\}$ $V_{21} = \{S\}$ $V_{31} = \{A\}$ $V_{41} = \{A\}$ $V_{51} = \{S\}$
$V_{12} = \{A, S\}$ $V_{22} = \{S\}$ $V_{32} = \{S\}$ $V_{42} = \{A, S\}$
$V_{13} = \{A, S\}$ $V_{23} = \{S\}$ $V_{33} = \{A, S\}$
$V_{14} = \{A, S\}$ $V_{24} = \{S\}$
$V_{15} = \{A, S\}$

$S \in V_{15}$, tehát $w \in L(G)$.

A CYK elemzési algoritmus

Az elemzési algoritmusban kapott

$V_{ij} = \{A \in N \mid A \Rightarrow^* w_{ij}\}$, $j = 1, \ldots, n$, $i = 1, \ldots, n - j + 1$

halmazok összességét elemzési táblának nevezzük és T-vel jelöljük.

A CYK algoritmus (az elemzési tábla ismeretében) $w \in L(G)$ esetén megadja w egy bal oldali elemzését, különben hibaüzenet ad.

A CYK elemzési algoritmus

(CYK) elemzési algoritmus:

Input: Egy w szó és egy Chomsky normálformában lévő $G = (N, \Sigma, P, S)$ cf nyelvtan és az előző algoritmus által megkonstruált T elemzőtábla.

Output: Ha $w \in L(G)$, akkor w egy bal oldali elemzése, különben "Nem".

Algoritmus: Ha $S \in V_{1,n}$, akkor hajtsuk végre (az alábbi) $parse(1, n, S)$ eljárást, különben legyen az output "Nem".

A CYK elemzési algoritmus

(CYK) Algoritmus, $parse(i, j, A)$**:**

1) Ha $j = 1$, akkor keressünk egy olyan A-t, melyre $A \to w_{i1}$ egy szabály és adjuk outputra a sorszámát.

2) Ha $j > 1$, akkor keressük meg a legkisebb olyan $1 \leq k < j$ számot, melyre valamely $B \in V_{ik}$ és $C \in V_{i+k,j-k}$ esetén $A \to BC$ egy szabály (mondjuk az m-edik).
Adjuk outputra az m számot és hajtsuk végre a $parse(i, k, B)$ és a $parse(i + k, j - k, C)$ algoritmusokat.

A CYK elemzési algoritmus

Példa, a $w = abaab$ bal oldali elemzése.

Mivel $S \in V_{15}$, végrehajtjuk $parse(1, 5, S)$-et. Kapjuk, hogy $A \in V_{11}$, $S \in V_{24}$ és $S \to AS$ a (2) sorszámú szabály, tehát az output 2.

Végrehajtjuk $parse(1, 1, A)$-t és $parse(2, 4, S)$-et. Az első hívás outputra adja 6-ot (mivel $A \to a$ sorszáma (6)), tehát az output 26. A második hívásra kapjuk, hogy $S \in V_{21}$, $A \in V_{33}$ és $S \to SA$ az (4) sorszámú szabály, tehát az output 264.

A CYK elemzési algoritmus

Példa, a $w = abaab$ bal oldali elemzése, folytatás.

Következnek a $parse(2, 1, S)$-t és $parse(3, 3, A)$ hívások. Az első hívás outputra adja 3-at (mivel $S \to b$ sorszáma (3)), tehát az output 2643. A második hívásra kapjuk ...

Végül az output (vagyis a bal oldali elemzés) 264356263.

Megjegyzés: csak egy bal oldali levezetést kapunk. Általában az összes bal oldali levezetés felsorolása exponenciális ideig tart, mivel a bal oldali levezetések száma az input hosszának exponenciális függvénye lehet.

A CYK elemzési algoritmus

A CYK algoritmus előnye a polinimális időbonyolultság.

Hátránya, hogy a nyelvtant először Chomsky normálformára kell hozni. Emiatt a gyakorlatban (programozási nyelvek esetén) nem alkalmazható, mert a programozási nyelvek szintaxisának áttekinthetőnek kell lennie (lásd a PÉLDA nyelv szintaxisát).

Az Earley elemzési algoritmus

Alapötlet. Legyen $G = (N, \Sigma, P, S)$ egy környezetfüggetlen nyelvtan, $w = a_1 \dots a_n$ egy input szó. Az

$$[A \to X_1 \dots X_k . X_{k+1} \dots X_m, i]$$

alakú kifejezéseket, ahol $A \to X_1 \dots X_m$ egy szabály, *elemnek* hívjuk. (A pont szerepelhet a jobb oldal elején és végén is, λ-szabály esetén alakja $[A \to ., i]$.)

Minden $0 \leq j \leq n$-re megkonstruáljuk elemeknek azon I_j halmazát, melyre teljesül, hogy $[A \to \alpha.\beta, i]$ akkor és csak akkor van I_j-ben, ha van olyan γ és δ, hogy $S \Rightarrow^* \gamma A \delta$, $\gamma \Rightarrow^* a_1 \dots a_i$ és $\alpha \Rightarrow^* a_{i+1} \dots a_j$.

Tehát i és j kijelölik az input szó azon részét, amelyet α-ból le lehet vezetni.

Az Earley elemzési algoritmus

Az $I_0, \ldots, I_n$ sorozatot a w elemzési sorozatának (vagy listájának) nevezzük.

Az I_j halmaz definíciójából következik, hogy $w \in L(G)$, akkor és csak akkor, ha I_n-ben van $[S \to \alpha., 0]$ alakú elem.

Az Earley elemzési algoritmus:

Input: Egy $G = (N, \Sigma, P, S)$ cf nyelvtan és egy $w = a_1 \ldots a_n$ szó.

Output: Az $I_0, \ldots, I_n$ elemzési sorozat.

Az Earley elemzési algoritmus

Először kiszámoljuk I_0-t a következőképpen:

(1) Minden $S \to \alpha$ szabály esetén legyen $[S \to .\alpha, 0] \in I_0$.

Ismételjük az alábbi (2) és (3) pontokat, amíg I_0 már nem bővíthető:

(2) Ha $[B \to \gamma., 0] \in I_0$, akkor minden $[A \to \alpha.B\beta, 0] \in I_0$ esetén adjuk az $[A \to \alpha B.\beta, 0]$ elemet is I_0-hoz.

(3) Ha $[A \to \alpha.B\beta, 0] \in I_0$, akkor minden $B \to \gamma$ szabály esetén adjuk a $[B \to .\gamma, 0]$ elemet is I_0-hoz.

Az Earley elemzési algoritmus

Tegyük fel, hogy az $I_0, \ldots, I_{j-1}$ sorozat már ismert, konstráljuk meg I_j-t:

(4) Minden $[B \to \alpha.a\beta, i] \in I_{j-1}$ esetén, ha $a = a_j$, akkor adjuk a $[B \to \alpha a.\beta, i]$ elemet I_j-hez.

Ismételjük az alábbi (5) és (6) pontokat, amíg I_j már nem bővíthető:

(5) Ha $[B \to \gamma., i] \in I_j$, akkor minden $[A \to \alpha.B\beta, k] \in I_i$ esetén adjuk a $[A \to \alpha B.\beta, k]$ elemet is I_j-hez.

(6) Ha $[A \to \alpha.B\beta, i] \in I_j$, akkor minden $B \to \gamma$ szabály esetén adjuk a $[B \to .\gamma, j]$ elemet is I_j-hez.

Az Earley elemzési algoritmus

Példa. Legyen G a következő nyelvtan

(1) $K \to T + K$
(2) $K \to T$
(3) $T \to F * T$
(4) $T \to F$
(5) $F \to (K)$
(6) $F \to a$

és legyen $w = (a + a) * a$.

Az Earley elemzési algoritmus

I_0 :

$[K \to .T + K, 0]$
$[K \to .T, 0]$
$[T \to .F * T, 0]$
$[T \to .F, 0]$
$[F \to .(K), 0]$
$[F \to .a, 0]$

I_1 :

$[F \to (.K), 0]$
$[K \to .T + K, 1]$
$[K \to .T, 1]$
$[T \to .F * T, 1]$
$[T \to .F, 1]$
$[F \to .(K), 1]$
$[F \to .a, 1]$

I_2 :

$[F \to a., 1]$
$[T \to F. * T, 1]$
$[T \to F., 1]$
$[K \to T. + K, 1]$
$[K \to T., 1]$
$[F \to (K.), 0]$

Az Earley elemzési algoritmus

I_3 :

$[K \to T + .K, 1]$
$[K \to .T + K, 3]$
$[K \to .T, 3]$
$[T \to .F * T, 3]$
$[T \to .F, 3]$
$[F \to .(K), 3]$
$[F \to .a, 3]$

I_4 :

$[F \to a., 3]$
$[T \to F. * T, 3]$
$[T \to F., 3]$
$[K \to T. + K, 3]$
$[K \to T., 3]$
$[K \to T + K., 1]$
$[F \to (K.), 0]$

I_5 :

$[F \to (K)., 0]$
$[T \to F. * T, 0]$
$[T \to F., 0]$
$[K \to T. + K, 0]$
$[K \to T., 0]$

Az Earley elemzési algoritmus

I_6 :

$[T \to F * .T, 0]$
$[T \to .F * T, 6]$
$[T \to .F, 6]$
$[F \to .(K), 6]$
$[F \to .a, 6]$

I_7 :

$[F \to a., 6]$
$[T \to F. * T, 6]$
$[T \to F., 6]$
$[T \to F * T., 0]$
$[K \to T. + K, 0]$
$[K \to T., 0]$

Mivel $[K \to T., 0] \in I_7$, az $(a + a) * a$ szó $L(G)$-ben van.

Az Earley elemzési algoritmus

Az $I_0, \ldots, I_n$ sorozat kiszámításának

tárbonyolultsága : $O(n^2)$,

időbonyolultsága általában: $O(n^3)$.

időbonyolultsága, ha G egyértelmű: $O(n^2)$

Az Earley elemzési algoritmus

Jobb oldali elemzés megadása ciklusmentes nyelvtanokra:

Definíció. Egy $G = (N, \Sigma, P, S)$ cf nyelvtan *ciklusmentes*, ha nincs olyan $A \in N$, melyre $A \Rightarrow^+ A$.

Lemma. (Lásd tankönyv). Tetszőleges G cf nyelvtanhoz konstruálható olyan G' ciklusmentes cf nyelvtan, amelyre $L(G) = L(G')$.

Az Earley elemzési algoritmus

Input: Egy ciklusmentes $G = (N, \Sigma, P, S)$ cf nyelvtan, egy $w = a_1 \dots a_n$ input szó és az $I_0, \dots, I_n$ elemzési sorozat.

Output: Ha $w \in L(G)$, akkor w egy π jobb oldali elemzése, különben "Nem".

Algoritmus: Ha I_n-ben nincs $[S \to \alpha., 0]$ alakú elem, akkor legyen az output "Nem", különben hajtsuk végre az (alábbi) $parse([S \to \alpha., 0], n)$ algoritmust.

Az Earley elemzési algoritmus

A $parse([A \to \beta., i], j)$ **algoritmus :**

(π globális változó, kezdetben $\pi = \lambda$)

(1) Legyen $\pi := \pi h$, ahol h az $A \to \beta$ szabály sorszáma.

(2) Ha $\beta = X_1 \dots X_m$, legyen $k = m$ és $l = j$.

(3) (a) Ha $X_k \in \Sigma$, legyen $k := k - 1$ és $l := l - 1$.

(3) (b) Ha $X_k \in N$, keressünk egy $[X_k \to \gamma., r]$ alakú elemet I_l-ben, úgy, hogy $[A \to X_1 \dots X_{k-1}.X_k \dots X_m, i] \in I_r$. Hajtsuk végre a $parse([X_k \to \gamma., r], l)$ algoritmust. Legyen $k := k - 1$ és $l := r$.

(4) Repeat (3) until $k = 0$. Halt.

Az Earley elemzési algoritmus

Magyarázat.

A $parse([A \to \beta., i], j)$ hívás hozzáadja az $A \to \beta$ szabály sorszámát az eddigi elemzéshez.

Ha $\beta = v_0 B_1 v_1 B_2 v_2 \dots B_s v_s$, akkor a *parse* rutin minden $1 \leq t \leq s$ esetén meghatározza az első olyan $B_t \to \beta_t$ szabályt, amelyet a B_t-re alkalmaztunk, és a w input szóban azt a pozíciót, amely a B_t-ből levezetett első terminális szimbólum előtt áll.

Az Earley elemzési algoritmus

Ezután a következő rekurzív hívások történnek:
$parse([B_s \to \beta_s., i_s], j_s)$,
$parse([B_{s-1} \to \beta_{s-1}., i_{s-1}], j_{s-1})$,
$\vdots$
$parse([B_1 \to \beta_1., i_1], j_1)$,
ahol
$j_s = j - |v_s|$ és $j_q = i_{q+1} - |v_q|$, minden $1 \leq q < s$-re
és ahol
$\beta = v_0 B_1 v_1 B_2 v_2 \dots B_s v_s$.

Az Earley elemzési algoritmus

Példa: $w = (a + a) * a$, meghívjuk $parse([K \to T., 0], 7)$-et.

Az (1) lépésben $\pi := 2$, majd $k = 1$ és $l = 7$.

A (3)(b) lépés következik, melyben találjuk $[T \to F*T., 0]$ elemet I_7-ben és az $[K \to .T, 0]$ elemet I_0-ban.

Meghívjuk $parse([T \to F * T., 0], 7)$-et, melynek hatására $\pi := 23$, majd $k = 3$ és $l = 7$.

A (3)(b) lépés következik, melyben találjuk $[T \to F., 6]$ elemet I_0-ban és az $[K \to F * .T, 0]$ elemet I_6-ban.

Meghívjuk $parse([T \to F., 6], 7)$-et, melynek hatására $\pi := 234$, majd $k = 2$ és $l = 6$.

A (3)(a) lépés következik, melyben $k = 1$ és $l = 5$ ... végeredmény $\pi = 23465124646$.

Felülről lefelé haladó elemzés

Felülről lefelé (top-down) haladó elemzési algoritmusok: az S kezdőszimbólumból kiindulva próbálunk meg felépíteni egy olyan derivációs fát aminek a határa w.

Ha sikerül, $w \in L(G)$, különben $w \notin L(G)$.

Előkészület: a balrekurzió megszüntetése.

Definíció. Egy $G = (N, \Sigma, P, S)$ egy cf nyelvtan *balrekurzív*, ha van olyan $A \in N$, melyre $A \Rightarrow^+ A\alpha$.

$A \Rightarrow^+ A\alpha \Rightarrow^+ A\alpha\alpha \Rightarrow^+ A\alpha\alpha\alpha \dots$

Lemma. (Lásd tankönyv). Tetszőleges G cf nyelvtanhoz konstruálható olyan G' nem balrekurzív cf nyelvtan, amelyre $L(G) = L(G')$.

Általános felülről lefelé elemzés

Adott $G = (N, \Sigma, P, S)$ cf nyelvtan és $w \in \Sigma^*$ szó, igaz-e, hogy $w \in L(G)$

Csak balrekurziómentes nyelvtanokkal foglalkozunk.

Az S kezdőszimbólumból kiindulva próbálunk meg felépíteni egy olyan derivációs fát aminek a határa w.

Kiterjesztés: A derivációs fában egy nemterminálist helyettesítünk valamely alternatívájával.

Illesztés: Annak az ellenőrzése, hogy a kiterjesztésnél alkalmazott alternatívában szereplő terminálisok illeszkednek-e az elemzendő szó (w) megfelelő részéhez.

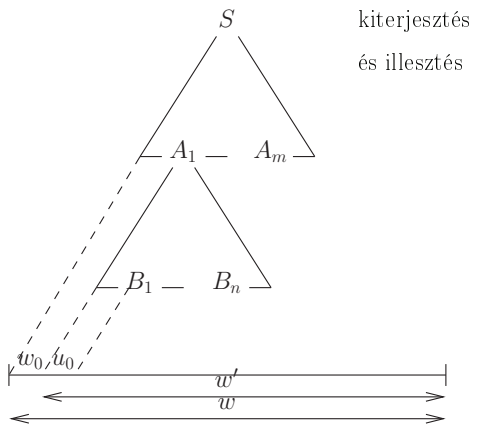
Az általános felülről lefelé haladó elemzés alapötlete.

– Minden $A \in N$-re rögzítsük le az A alternatíváinak egy $A \to \gamma_1 \,|\, \ldots \,|\, \gamma_k$ sorrendjét.

– Legyenek S alternatívái, $S \to \alpha_1 \,|\, \ldots \,|\, \alpha_k$.

– Keressük meg az első olyan $\alpha_i = w_0 A_1 w_1 \ldots A_m w_m$ alternatívát, melyre w_0 illeszkedik-e w elejéhez, azaz fennáll $w = w_0 w'$. ($w_0 = \lambda$ mindig jó!)

– Ha nincs ilyen α_i, akkor azt mondhatjuk, hogy $w \notin L(G)$ és az algoritmus leáll.



kiterjesztés

és illesztés

– Ha van ilyen i, azaz $\alpha_i = w_0 A_1 w_1 \ldots A_m w_m$ és $w = w_0 w'$, akkor A_1-et terjesztjük ki. Az $A_1 \to \beta_1 \,|\, \ldots \,|\, \beta_l$ alternatívák között megkeressük a legelső olyat, ami illeszkedik w' elejéhez: $\beta_j = u_0 B_1 u_1 \ldots B_n u_n$ és $w' = u_0 w''$.

– Ha egy nemterminális kiterjesztése esetén (pl. most A_1) nem találunk olyan alternatívát ami illeszkedik w megmaradt részéhez, akkor nem mondhatjuk azt, hogy $w \notin L(G)$, mivel valamennyi előző kiterjesztés esetében a *legelső* olyan alternatívát választottuk ami illeszkedett.

– Ezért, ha egy szinten nem találunk illeszkedő alternatívát, akkor egy szintet visszamegyünk (ún. *backtrack*-et hajtunk végre) és az ott választott alternatíva után következő első olyan alternatívát kell választanunk ami illeszkedik.

Felülről lefelé haladó általános elemzési algoritmus.

Input Egy nem balrekurzív $G = (N, \Sigma, P, S)$ cf nyelvtan és egy $w = a_1 \dots a_n$, $n \geq 0$ input szó.

Ouput *Igen* és a w egy bal oldali levezetése, ha $w \in L(G)$. Különben *Nem*.

Módszer

• Minden $A \in N$ esetében rögzítsük le A alternatíváinak egy $A \to \gamma_1 \,|\, \gamma_2 \,|\, \dots \,|\, \gamma_k$ sorrendjét. Az A i-edik alternatívájára A_i-vel fogunk hivatkozni.

• Az elemzés (s, i, α, β) alakú konfigurációk sorozata.

– $s \in \{\, q, t, b \,\}$, normál, elfogadó és backtrack *állapot*.

Felülről lefelé haladó általános elemzési algoritmus.

– i egy pointer, $1 \leq i \leq n+1$, $a_{n+1} := \$$.

– α egy verem, melynek teteje a jobb végén van. Tartalmazza az elemzés mindenkori „történetét”. Kezdő értéke: λ.

– β egy verem, melynek teteje a bal végén van. Tartalmazza a levezetett bal oldali mondatformának azt a részét amelyet még ki kell terjeszteni. Kezdőértéke: S.

• A konfigurációk halmazán megadunk egy $\vdash$ *átmeneti relációt*. Minden (s, i, α, β) konfigurációhoz legfeljebb egy $(s', i', \alpha', \beta')$ konfiguráció van, melyre $(s, i, \alpha, \beta) \vdash (s', i', \alpha', \beta')$.

• $w \in L(G) \Leftrightarrow$, ha $(q, 1, \lambda, S) \vdash^* (t, n+1, \alpha, \lambda)$.

Felülről lefelé haladó általános elemzési algoritmus.

Az elemzési algoritmus

1. $C := (q, 1, \lambda, S)$;
2. Amíg van olyan C', melyre $C \vdash C'$, legyen $C := C'$;
3. Ha $C := (t, n+1, \alpha, \lambda)$ valamely α-ra, akkor output: *Igen*. Ekkor az α veremben lévő alternatívák adják a bal oldali levezetést. Különben output: *Nem*.

Az algoritmusban szereplő $\vdash$ átmeneti relációt a következőképpen definiáljuk: az alábbi (1) – (6) pontok közül sorrendben a legelső alkalmazhatót alkalmazzuk a C konfigurációra.

Felülről lefelé haladó általános elemzési algoritmus.

Átmeneti reláció

(1) *Kiterjesztés:* Ha $C = (q, i, \alpha, A\beta)$, vagyis az aktív szimbólum nemterminális, akkor $C \vdash C'$, ahol $C' = (q, i, \alpha A_1, \gamma_1 \beta)$, és γ_1 az A első alternatívája.

(2) *Sikeres input illesztés:* Ha $C = (q, i, \alpha, a\beta)$ és $a = a_i$, vagyis az aktív szimbólum terminális és illeszkedik az input pointer által mutatott input szimbólumhoz, akkor $C \vdash C'$, ahol $C' = (q, i+1, \alpha a, \beta)$. (A lépés $i = n+1$ esetén soha nem alkalmazható, mert $a_{n+1} = \$ \neq a$, semmilyen $a \in \Sigma$-ra.)

(3) *Sikeres elemzés:* Ha $C = (q, n+1, \alpha, \lambda)$, akkor $C \vdash C'$, ahol $C' = (t, n+1, \alpha, \lambda)$, vagyis elérjük a befejező konfigurációt.

Felülről lefelé haladó általános elemzési algoritmus.

(4) *Sikertelen input illesztés:* Ha $C = (q, i, \alpha, a\beta)$ de $a \neq a_i$, akkor $C \vdash C'$, ahol $C' = (b, i, \alpha, a\beta)$. (Átmegy backtrack állapotba.)

(5) *Backtrack az inputban:* Ha $C = (b, i, \alpha a, \beta)$, akkor $C \vdash C'$, ahol $C' = (b, i-1, \alpha, a\beta)$. (A passzív veremből visszatesszük az inputot.)

Felülről lefelé haladó általános elemzési algoritmus.

(6) *Backtrack a kiterjesztésben:* Ha $C = (b, i, \alpha A_j, \gamma_j\beta)$ akkor az alábbi esetek lehetségesek:

(i) Ha A-nak van $j+1$-edik alternatívája is, akkor $C \vdash C'$, ahol $C' = (q, i, \alpha A_{j+1}, \gamma_{j+1}\beta)$. ($A$-t a következő, vagyis a $j+1$-edik alternatívájával terjesztjük ki. Utána ezt illesztjük ezért átmegyünk normál állapotba.)

(ii) Ha $i = 1, A = S$ és S-nek csak j alternatívája van, akkor nincs átmenet semelyik konfigurációba.

(iii) Ha az előző feltételek egyike sem teljesül, akkor $C \vdash C'$, ahol $C' = (b, i, \alpha, A\beta)$. ($A$-nak már minden alternatíváját kipróbáltuk, ezért vissza kell térnünk az előző szintre.)

Felülről lefelé haladó általános elemzési algoritmus.

Példa. G_e: $K \to T + K \mid T$
$T \to a \mid b$

Az alternatívák sorrendje a fenti, tehát K_1 a $K \to T + K$, K_2 a $K \to T$, T_1 a $T \to a$ és végül T_2 a $T \to b$ alternatívát jelentik. Elemezzük a $w = b{+}a$ szót az előbb ismertetett általános elemzési algoritmussal. (Tehát most $n = 3$, $a_1 = b$, $a_2 = +$, $a_3 = a$ és $a_4 = \$$.)

Felülről lefelé haladó általános elemzési algoritmus.

Példa. $K \to T + K \mid T$
$T \to a \mid b$

$(q, 1, \lambda, K) \vdash (q, 1, K_1, T + K) \vdash (q, 1, K_1T_1, a + K) \vdash$
$(b, 1, K_1T_1, a + K) \vdash (q, 1, K_1T_2, b + K) \vdash (q, 2, K_1T_2b, {+}K) \vdash$
$(q, 3, K_1T_2b{+}, K) \vdash (q, 3, K_1T_2b + K_1, T + K) \vdash$
$(q, 3, K_1T_2b + K_1T_1, a + K) \vdash (q, 4, K_1T_2b + K_1T_1a, {+}K) \vdash$
$(b, 4, K_1T_2b + K_1T_1a, {+}K) \vdash (b, 3, K_1T_2b + K_1T_1, a + K) \vdash$
$(q, 3, K_1T_2b + K_1T_2, b + K) \vdash (b, 3, K_1T_2b + K_1, T + K) \vdash$
$(q, 3, K_1T_2b + K_2, T) \vdash (q, 3, K_1T_2b + K_2T_1, a) \vdash$
$(q, 4, K_1T_2b + K_2T_1a, \lambda) \vdash (t, 4, K_1T_2b + K_2T_1a, \lambda)$

Következésképpen $b + a \in L(G_e)$.

A $b + a$ szó bal oldali levezetése: $K_1T_2K_2T_1$.

Felülről lefelé haladó általános elemzési algoritmus.

Az általános felülről lefelé haladó elemzés

tárbonyolultsága : $O(n)$ (lineáris),

időbonyolultsága: $O(c^n)$ (= exponenciális), ez nagyon nagy.

Felülről lefelé haladó általános elemzési algoritmus.

Felgyorsítási technikák.

(1) A szabályokat az alkalmazásuk valószínűségének a sorrendjébe rakjuk.

Kérdés, hogy meg tudjuk-e állapítani. Továbbá, nem segít abban az esetben, ha $w \notin L(G)$.

Felülről lefelé haladó általános elemzési algoritmus.

Tetszőleges $k \geq 0$ és $\alpha \in (N \cup \Sigma)^*$-ra, legyen
$FIRST_k(\alpha) = \{w \,|\, \alpha \Rightarrow^* wx \in \Sigma^* \text{ és } |w| = k \text{ vagy } (|w| < k \text{ és } x = \lambda)\}$

Rögzítsünk egy k-t (pl. $k = 1$) és minden $A \to \alpha$ szabály estén számoljuk ki $FIRST_k(\alpha)$ halmazt.

(2) (Look-ahead technika.) Az inputnak "előre nézzük" a k hosszúságú prefixét, legyen ez x. Ha az A nemterminálist terjesztjük ki, akkor csak olyan $A \to \alpha$ alternatíva jöhet szóba, melyre $x \in FIRST_k(\alpha)$. Bizonyos nyelvtanokra nagyon hatékony, lásd később.

Felülről lefelé haladó általános elemzési algoritmus.

(3) "Nyilvántartást" vezetünk a backtrackről. Például, ha tudjuk, hogy az utolsó m darab alkalmazott szabály esetén mindegyiknek az utolsó alternatíváját alkalmaztuk, akkor illesztési konfliktus estén közvetlenül visszaléphetünk az első olyan szabályra, aminek még van nem kipróbált alternatívája.

Felülről lefelé haladó általános elemzési algoritmus.

Az exponenciális időbonyolultság mellett másik hátránya, hogy a hiba behatároló képessége szegényes.

Ugyanis, ha $w \notin L(G)$, akkor elvárható, hogy az elemző azonosítsa az első olyan helyet, ahol az inputban hiba van. Sőt, elvárható, hogy a hibán továbblépve folytassa az elemzést és derítse fel a további hibákat. Az általános felülről lefelé haladó elemzési algoritmus a jelenelgi formájában $w \notin L(G)$ esetén csak jelzi a hibát és az input pointert a legelső pozícióra állítja.

$LL(k)$ nyelvtanok és elemzésük

Az $LL(k)$ elemzés (mint speciális felülről lefelé elemzés) azon alapul, hogy amikor egy A nemterminális kiterjesztését keressük akkor pótlólagos információként megnézzük az input még feldolgozatlan részének k hosszúságú prefixét és ennek ismeretében egyértelműen ki tudjuk választani A-nak azt az alternatíváját amely szerint ki kell őt terjeszteni, ha van ilyen. Amennyiben nincs ilyen alternatíva, akkor azt tudjuk mondani, hogy az elemzett szó nem eleme $L(G)$-nek.

Csak azokat nyelvtanokat lehet $LL(k)$ módon elemezni, amelyek kielégítik az ún. $LL(k)$ feltételt.

A $FIRST_k$ halmaz definíciója

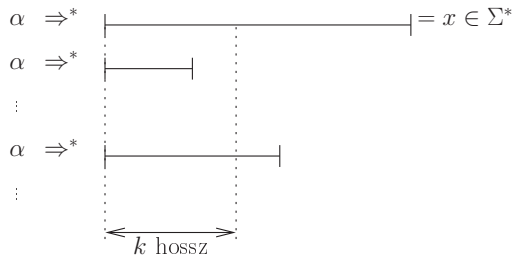
Definíció. Legyen $k \geq 0$ egy egész szám.

Tetszőleges $\alpha \in (N \cup \Sigma)^*$-ra

$FIRST_k(\alpha) = \{w \,|\, \alpha \Rightarrow^* wx \in \Sigma^* \text{ és } |w| = k \text{ vagy } (|w| < k \text{ és } x = \lambda)\}$

Tetszőleges $L \subseteq (N \cup \Sigma)^*$-ra

$$FIRST_k(L) = \bigcup_{\alpha \in L} FIRST_k(\alpha)$$



Megjegyzés.

1. $FIRST_k(\alpha), FIRST_k(L) \subseteq \Sigma^{*,k}$ (és így véges halmazok).

2. $u \in \Sigma^*$-ra

$$FIRST_k(u) = \begin{cases} \{\, u \,\} & \text{ha } |u| < k \\ \{\, w \,\} & \text{ha } u = wx, \text{ ahol } |w| = k. \end{cases}$$

Ilyenkor $FIRST_k(u) = \{\, u \,\}$ és $FIRST_k(u) = \{\, w \,\}$ helyett rendre $FIRST_k(u) = u$ és $FIRST_k(u) = w$-t írunk.

3. A továbbiakban a $FIRST_k$ jelölés helyett a rövidebb FI_k-t használjuk.

4. $FI_1(aba) = a$, $FI_2(aba) = ab$ és $FI_k(aba) = aba$ minden $k \geq 3$-ra.

Az $LL(k)$ nyelvtan definíciója

Definíció.

Legyen $k \geq 1$ egy egész szám. Azt mondjuk, hogy a G nyelvtan $LL(k)$, ha valahányszor teljesülnek az

- $S \Rightarrow_l^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx$
- $S \Rightarrow_l^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy$
- $FI_k(x) = FI_k(y)$

feltételek, mindannyiszor $\beta = \gamma$.

Észrevétel. Ha egy nyelvtan $LL(k)$, akkor az $LL(k+1)$ is.

Az $LL(k)$ feltétel átfogalmazása

Tétel. G akkor és csakis akkor $LL(k)$, ha valahányszor $S \Rightarrow_l^* wA\alpha$ levezetés, $A \to \beta$ és $A \to \gamma$ pedig különböző P-beli szabályok, mindannyiszor $FI_k(\beta\alpha) \cap FI_k(\gamma\alpha) = \emptyset$.

Bizonyítás. Mindkét irány: indirekt bizonyítás.

a) Tfh, hogy a lemma feltételei teljesülnek és a nyelvtan mégsem $LL(k)$:

- $S \Rightarrow_l^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx$ és
- $S \Rightarrow_l^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy$

melyekre $FI_k(x) = FI_k(y)$, és mégis $\beta \neq \gamma$.

Ekkor az $u = FI_k(x) = FI_k(y)$ szóra teljesül, hogy $u \in FI_k(\beta\alpha) \cap FI_k(\gamma\alpha)$. Tehát a $wA\alpha$ bal modatformára, az $A \to \beta$ és az $A \to \gamma$ (különböző) szabályokra, $FI_k(\beta\alpha) \cap FI_k(\gamma\alpha) \neq \emptyset$, ami ellentmondás, mivel feltettük, hogy a lemma feltételei teljesülnek.

b) Tfh, hogy G $LL(k)$, de a lemma feltételei nem teljesülnek, vagyis van egy $wA\alpha$ bal mondatforma, van két különböző szabály, az $A \to \beta$ és az $A \to \gamma$, melyekre $FI_k(\beta\alpha) \cap FI_k(\gamma\alpha) \neq \emptyset$.

Ez utóbbi azt jelenti, hogy valamely $x, y \in \Sigma^*$ szavakra $\beta\alpha \Rightarrow^* x$, $\gamma\alpha \Rightarrow^* y$ és $FI_k(x) = FI_k(y)$. Ez viszont ellentmond annak, hogy G $LL(k)$, mivel a levezetéseket összerakva kapjuk, hogy

- $S \Rightarrow_l^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx$,
- $S \Rightarrow_l^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy$,
- $FI_k(x) = FI_k(y)$

és mégis $\beta \neq \gamma$. Tehát G nyelvtan $LL(k)$.

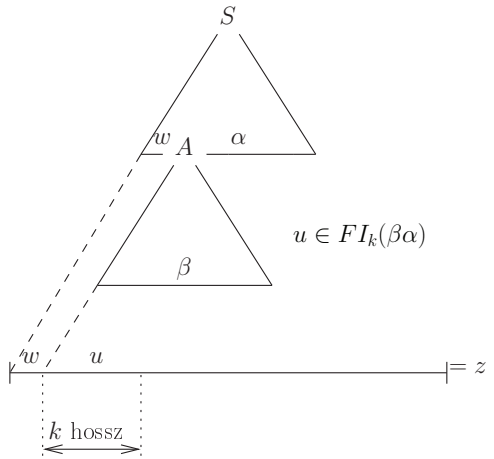
$LL(k)$ elemzés alapötlete.

Kérdés: $z \in L(G)$?

Tudjuk: $S \Rightarrow_l^* wA\alpha$ és $z = wz'$. Előrenézés: $u = FI_k(z')$.

Akkor A-t azzal az egyértelmű $A \to \beta$ alternatívával terjesztjük ki, melyre $u \in FI_k(\beta\alpha)$.

Ha nincs ilyen β, akkor $z \notin L(G)$.



Példa. G_{ar}: $K \to K + T$, $K \to T$,
$T \to T * F$, $T \to F$,
$F \to (K)$, $F \to a$

Nem $LL(1)$, mert

- $K \Rightarrow_l^* K \Rightarrow K + T \Rightarrow^* a + a$
- $K \Rightarrow_l^* K \Rightarrow T \Rightarrow^* a$
- $FI_1(a + a) = FI_1(a) = a$

mégis $K + T \neq T$.

A következőkben két kérdést szeretnénk tisztázni:

(1) Ha adott egy G nyelvtan és egy k szám akkor eldönthető-e, hogy G teljesíti-e az $LL(k)$ feltételt?

(2) Hogyan működik az $LL(k)$ elemzés?

Általánosan csak (1)-et válaszoljuk meg.
(2)–t lásd: Fülöp Zoltán, Formális nyelvek és szintaktikus elemzésük, 7.2 fejezet.

Az egyszerűbb, ún. erősen $LL(k)$ esetben mindkét kérdést megválaszoljuk.

Mindkét kérdés megválaszolásához szükség van a $FI_k(\alpha)$-t kiszámító algoritmusra. Ehhez szükség lesz a következőkre.

Definíció. Legyenek $L_1, L_2 \subseteq \Sigma^*$ nyelvek és $k \geq 0$. Akkor

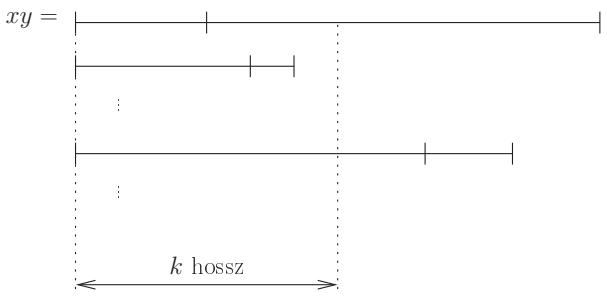
$$L_1 \oplus_k L_2 = \{FI_k(xy) \,|\, x \in L_1, y \in L_2\}.$$

Példa. Ha $L_1 = \{\lambda, abb\}$ és $L_2 = \{b, bab\}$, akkor $L_1 \oplus_2 L_2 = \{b, ba, ab\}$.

Megjegyzés. $L_1 \oplus_k L_2$ mindig véges, mivel $L_1 \oplus_k L_2 \subseteq \Sigma^{*,k}$.

Továbbá a $\oplus_k$ művelet asszociatív és könnyen igazolható, hogy minden α, β-ra

$$FI_k(\alpha\beta) = FI_k(\alpha) \oplus_k FI_k(\beta).$$



$FI_k(\alpha)$-t kiszámítása

Legyen $\alpha = X_1 \ldots X_n$. Ekkor az előzőek miatt

$$FI_k(\alpha) = FI_k(X_1) \oplus_k \ldots \oplus_k FI_k(X_n).$$

Mivel a $\oplus_k$ műveletet könnyű végrehajtani, elegendő csak $FI_k(X_i)$-t kiszámítani, ahol $X_i \in (N \cup \Sigma)$.

Sőt, a $FI_k(\alpha)$ definíciójából következik, hogy ha $X_i \in \Sigma$, akkor $FI_k(X_i) = X_i$.

Mindezt egybevetve, azt kapjuk, hogy amennyiben minden $A \in N$-re $FI_k(A)$-t ki tudjuk számolni, akkor minden α szóra $FI_k(\alpha)$-t is ki tudjuk számolni. Tehát elegendő $FI_k(A)$-val foglalkozni.

$FI_k(A)$ halmazok kiszámítása.

Input Egy G környezetfüggetlen nyelvtan.

Output Minden $A \in N$-re a $FI_k(A)$ halmaz.

Módszer $FI_k(A)$-t iteráljuk a $H_0(A), H_1(A), \ldots$ sorozattal.

Algoritmus

(1) Legyen minden $a \in \Sigma$ és $i \geq 0$ esetén $H_i(a) = \{a\}$.

(2) Legyen minden $A \in N$-re $H_0(A) =$
$\{x \in \Sigma^* \mid A \to x\alpha \in P$
ahol $(|x| = k)$ vagy $(|x| < k$ és $\alpha = \lambda)\}$ és legyen $i = 0$.

(3) Minden $A \in N$-re $H_0(A), \ldots, H_i(A)$ már ismertek.
Legyen $H_{i+1}(A) =$
$H_i(A) \cup \{x \in \Sigma^* \mid x \in H_i(X_1) \oplus_k \ldots \oplus_k H_i(X_n)$
valamely $A \to X_1 \ldots X_n \in P$ esetén $\}$.

(4) Ha minden $A \in N$-re $H_i(A) = H_{i+1}(A)$, akkor álljunk meg, különben legyen $i = i + 1$ és menjünk vissza (3)-ra.

Példa. G_{ar}: $K \to K + T$, $K \to T$,
$T \to T * F$, $T \to F$,
$F \to (K)$, $F \to a$

Számítsuk ki a $FI_1(K), FI_1(T)$ és $FI_1(F)$ halmazokat. A $H_0, H_1, \ldots$ közelítések egy táblázat soraiban ábrázolhatók a következő módon:

	K	T	F
H_0	$\emptyset$	$\emptyset$	$\{(, a\}$
H_1	$\emptyset$	$\{(, a\}$	$\{(, a\}$
H_2	$\{(, a\}$	$\{(, a\}$	$\{(, a\}$
H_3	$\{(, a\}$	$\{(, a\}$	$\{(, a\}$

Példa. Jelöljük G'_{ar}-ral azt a nyelvtant melynek szabályai:

1: $K \to TT'$
2: $T' \to +TT'$
3: $T' \to \lambda$
4: $T \to FF'$
5: $F' \to *FF'$
6: $F' \to \lambda$
7: $F \to (K)$
8: $F \to a$

G'_{ar} ugyancsak az aritmetikai kifejezéseket generálja, tehát $L(G_{ar}) = L(G'_{ar})$.

Számoljuk ki $FI_1(X)$-et minden $X \in \{K, T, T', F, F'\}$-re.

	K	T	T'	F	F'
H_0	$\emptyset$	$\emptyset$	$\{+, \lambda\}$	$\{(, a\}$	$\{*, \lambda\}$
H_1	$\emptyset$	$\{(, a\}$	$\{+, \lambda\}$	$\{(, a\}$	$\{*, \lambda\}$
H_2	$\{(, a\}$	$\{(, a\}$	$\{+, \lambda\}$	$\{(, a\}$	$\{*, \lambda\}$
H_3	$\{(, a\}$	$\{(, a\}$	$\{+, \lambda\}$	$\{(, a\}$	$\{*, \lambda\}$

Tehát $FI_1(K) = FI_1(T) = FI_1(F) = \{(, a\}$, $FI_1(T') = \{+, \lambda\}$ és $FI_1(F') = \{*, \lambda\}$.

Az $LL(k)$ feltétel eldöntése

Definíció. Tetszőleges $A \in N$-re legyen

$\sigma(A) = \{L \subseteq \Sigma^{*,k} \mid$ van olyan w és α, hogy $S \Rightarrow^* wA\alpha$ és $L = FI_k(\alpha)\}$.

Rögtön észrevehető, hogy $\sigma(A)$ halmaz véges, mert $\sigma(A) \subseteq \mathcal{P}(\Sigma^{*,k})$.

A $\sigma(A)$ halmazok kiszámíthatók. A kiszámításukra alkalmas algoritmus lényege, hogy iterációval kiszámoljuk minden $A, B \in N$-re a

$\bar{\sigma}(A, B) = \{L \subseteq \Sigma^{*,k} \mid$ van olyan w és α, hogy $A \Rightarrow^* wB\alpha$ és $L = FI_k(\alpha)\}$

halmazt. Ezután $\sigma(A) = \bar{\sigma}(S, A)$ lesz.

Az $LL(k)$ feltétel eldöntése.

A $\bar{\sigma}(A, B)$ halmazok kiszámítása.

Input Egy G környezetfüggetlen nyelvtan.

Output Minden $A, B \in N$-re a $\bar{\sigma}(A, B)$ halmaz.

Módszer $\bar{\sigma}(A, B)$-t iteráljuk a $H_0(A, B), H_1(A, B), \ldots$ sorozattal.

Algoritmus

(1) Legyen minden $A, B \in N$-re $H_0(A, B) = \{L \subseteq \Sigma^* \mid A \to \gamma B\alpha \in P$ ahol $L = FI_k(\alpha)\}$ és legyen $i = 0$.

(2) Minden $A, B \in N$-re $H_0(A, B), \ldots, H_i(A, B)$ már ismertek. Legyen $H_{i+1}(A, B) = H_i(A, B) \cup \{L_1 \oplus_k L_2 \mid \exists C \in N : L_2 \in H_i(A, C)$ és $L_1 \in H_i(C, B)\}$.

(3) Ha minden $A, B \in N$-re $H_i(A, B) = H_{i+1}(A, B)$, akkor álljunk meg, különben legyen $i = i + 1$ és menjünk vissza (2)-re.

Ekkor $\sigma(A) = \bar{\sigma}(S, A)$.

Az $LL(k)$ feltétel eldöntése.

A $\sigma(A)$ kiszámíthatóságából valamint az $LL(k)$ feltétel átfogalmazásából következik, hogy tetszőleges G-re az $LL(k)$ feltétel algoritmikusan eldönthető.

Input Egy G környezetfüggetlen nyelvtan és egy $k \geq 1$ egész szám.

Output *Igen* ha G $LL(k)$ és *Nem* ha G nem $LL(k)$.

Algoritmus

(1) Válasszunk egy olyan A nemterminálist, melynek legalább két alternatívája van és számoljuk ki $\sigma(A)$-t.

(2) Válasszunk két különböző $A \to \beta$ és $A \to \gamma$ szabályt és minden $L \in \sigma(A)$-ra számoljuk ki a

$$(FI_k(\beta) \oplus_k L) \cap (FI_k(\gamma) \oplus_k L)$$

halmazt. Ha van olyan L melyre ez a metszet nem üres, akkor G nem $LL(k)$ tehát álljunk meg és adjunk outputra *Nem* jelzést.

(3) Ha választható A-nak újabb két alternatíva-párja, akkor ismételjük (2)-t.

(4) Ha választható újabb nemterminális, akkor ismételjük (1)-et.

(5) Adjunk outputra *Igen* jelzést (mert az átfogalmazott $LL(k)$ feltétel teljesül).

Példa $LL(k)$ feltétel eldöntésére.

A G_{ar} nyelvtan nem $LL(1)$. Vegyük például a $K \to K + T \,|\, T$ szabálypárt. Itt, minden $L \in \sigma(K)$-ra, függetlenül attól, hogy mi az L, $FI_1(K+T) \oplus_1 L = FI_1(K)$ és $FI_1(T) \oplus_1 L = FI_1(T)$, mert az előző Példa a) része szerint $\lambda \notin FI_1(K)$ és $\lambda \notin FI_1(T)$. Továbbá, ugyancsak az előző Példa szerint $FI_1(K) = \{(, a\}$ és $FI_1(T) = \{(, a\}$, tehát

$$(FI_1(K+T) \oplus_1 L) \cap (FI_1(T) \oplus_1 L) = \{(, a\} \neq \emptyset.$$

Ugyanakkor G'_{ar} viszont $LL(1)$, mivel erősen $LL(1)$, de ezt később mutatjuk be.

Az erősen $LL(k)$ nyelvtan definíciójának előkészítése

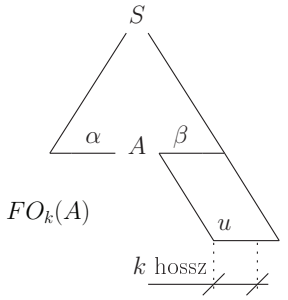
A $FOLLOW_k$ halmaz definíciója.

Definíció. Legyen $A \in N$ és $k \geq 1$. Akkor

$FOLLOW_k(A) =$
$\bigcup\{FI_k(\beta) \mid S \Rightarrow^* \alpha A\beta,$ valamilyen $\alpha, \beta \in (N \cup \Sigma)^*$-ra$\}$.

Az olyan terminális szavak FI_k-i, amelyek az A-t tartalmazó mondatformák (mint $\alpha A\beta$) $\underline{A\text{-t követő}}$ részéből (vagyis β-ból) vezethetők le.

A továbbiakban a $FOLLOW_k$ jelölés helyett a rövidebb FO_k-t használjuk.



A $FO_k(A)$ halmazok kiszámítása.

Input Egy G cf nyelvtan és egy $k \geq 1$ egész szám.
Output Minden $A \in N$-re $FO_k(A)$.
Módszer $FO_k(A)$-t iteráljuk a $H_0(A), H_1(A), \ldots$ sorozattal.

Algoritmus

(1) Legyen $i = 0$, $H_0(S) = \{\lambda\}$ és minden $A \neq S$-re legyen $H_0(A) = \emptyset$.

(2) Minden $A \in N$-re $H_0(A), \ldots, H_i(A)$ már ismertek. Legyen $H_{i+1}(A) = H_i(A) \cup \{x \in \Sigma^* \mid$
$x \in FI_k(\beta H_i(B))$ valamely $B \to \alpha A\beta$ szabály esetén $\}$.

(3) Ha minden $A \in N$-re $H_i(A) = H_{i+1}(A)$, akkor álljunk meg, különben legyen $i = i + 1$ és menjünk (2)-re.

Minden A-ra $FO_k(A) = H_i(A)$.

Példa. Jelöljük G'_{ar}-ral azt a nyelvtant melynek szabályai:

1: $K \to TT'$
2: $T' \to +TT'$
3: $T' \to \lambda$
4: $T \to FF'$
5: $F' \to *FF'$
6: $F' \to \lambda$
7: $F \to (K)$
8: $F \to a$

Számoljuk ki $FO_1(X)$-et minden $X \in \{K, T, T', F, F'\}$-re.

Számoljuk ki $FO_1(X)$-et minden $X \in \{K, T, T', F, F'\}$-re.

	K	T	T'	F	F'
H_0	$\{\lambda\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
H_1	$\{\lambda\}$	$\{+, \lambda\}$	$\{\lambda\}$	$\emptyset$	$\emptyset$
H_2	$\{\lambda\}$	$\{+, \lambda\}$	$\{\lambda\}$	$\{+, *, \lambda\}$	$\{+, \lambda\}$
H_3	$\{), \lambda\}$	$\{+, \lambda\}$	$\{\lambda\}$	$\{+, *, \lambda\}$	$\{+, \lambda\}$
H_4	$\{), \lambda\}$	$\{+,), \lambda\}$	$\{), \lambda\}$	$\{+, *, \lambda\}$	$\{+, \lambda\}$
H_5	$\{), \lambda\}$	$\{+,), \lambda\}$	$\{), \lambda\}$	$\{+, *,), \lambda\}$	$\{+,), \lambda\}$
H_6	$\{), \lambda\}$	$\{+,), \lambda\}$	$\{), \lambda\}$	$\{+, *,), \lambda\}$	$\{+,), \lambda\}$

Tehát $FO_1(K) = FO_1(T') = \{), \lambda\}$, $FO_1(T) = FO_1(F') = \{+,), \lambda\}$ és $FO_1(F) = \{+, *,), \lambda\}$.

Az erősen $LL(k)$ nyelvtan definíciója

Definíció. Legyen $k \geq 1$. Azt mondjuk, hogy G erősen $LL(k)$, ha tetszőleges $A \in N$ és $A \to \beta$, $A \to \gamma$ különböző szabályok esetén teljesül, hogy

$$FI_k(\beta FO_k(A)) \cap FI_k(\gamma FO_k(A)) = \emptyset,$$

(ahol $\beta FO_k(A) = \{\beta x \,|\, x \in FO_k(A)\}$.)

Tétel. Ha G erősen $LL(k)$, akkor $LL(k)$.

Bizonyítás. Indirekt: tfh G erősen $LL(k)$, de nem $LL(k)$.

Ha G nem $LL(k)$ akkor teljesülnek az

- $S \Rightarrow_l^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx$
- $S \Rightarrow_l^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy$
- $FI_k(x) = FI_k(y)$

feltételek, és mégis $\beta \neq \gamma$. Legyen $u = FI_k(x)$.

Mellesleg: $FI_k(\alpha) \subseteq FO_k(A)$.

A $FI_k(\alpha) \subseteq FO_k(A)$ feltétel miatt

$u = FI_k(x) \in FI_k(\beta\alpha) = FI_k(\beta) \oplus_k FI_k(\alpha) \subseteq$
$FI_k(\beta) \oplus_k FO_k(A) = FI_k(\beta FO_k(A))$.

Hasonlóan megmutatható, hogy $u \in FI_k(\gamma FO_k(A))$, tehát

$$u \in FI_k(\beta FO_k(A)) \cap FI_k(\gamma FO_k(A)) \neq \emptyset,$$

ami ellentmondás, mert G erősen $LL(k)$.

A következőkben két kérdést fogjuk tisztázni:

(1) Ha adott egy G nyelvtan és egy k szám akkor eldönthető-e, hogy G teljesíti-e az erősen $LL(k)$ feltételt?

(2) Hogyan működik az erősen $LL(k)$ elemzés?

Az erősen $LL(k)$ feltétel eldöntése.

Input Egy G környezetfüggetlen nyelvtan és egy $k \geq 1$ egész szám.
Output *Igen* ha G erősen $LL(k)$, különben *Nem*.

Algoritmus

(1) Válasszunk egy olyan A nemterminálist melynek legalább két alternatívája van.

(2) Válasszunk két különböző $A \to \beta$ és $A \to \gamma$ szabályt és számoljuk ki a $(FI_k(\beta FO_k(A))) \cap (FI_k(\gamma FO_k(A)))$ halmazt. Ha ez nemüres, akkor G nem erősen $LL(k)$, tehát álljunk meg és output: *Nem*.

(3) Ha választható A-nak újabb két alternatíva-párja, akkor ismételjük (2)-t.

(4) Ha választható újabb nemterminális, akkor ismételjük (1)-et.

(5) Adjunk outputra *Igen* jelzést.

Példa.

G'_{ar}-nak két olyan szabálypárja van, melyeknek bal oldala ugyanaz.

1) $T' \to +TT' \mid \lambda$: $FI_1(+TT'\,FO_1(T')) = \{+\}$ és $\overline{FI_1(\lambda\,FO_1(T'))} = FI_1(\{\,),\lambda\}) = \{\,),\lambda\}$. Tehát

$$FI_1(+TT'\,FO_1(T')) \cap FI_1(\lambda\,FO_1(T')) = \{+\} \cap \{\,),\lambda\} = \emptyset.$$

2) $F' \to *FF' \mid \lambda$: $FI_1(*FF'\,FO_1(F')) = \{*\}$ és $\overline{FI_1(\lambda\,FO_1(F'))} = FI_1(\{+,),\lambda\}) = \{+,),\lambda\}$. Tehát

$$FI_1(*FF'FO_1(F')) \cap FI_1(\lambda FO_1(F')) = \{*\} \cap \{+,),\lambda\} = \emptyset.$$

Következésképpen G'_{ar} erősen $LL(1)$.

Az $LL(1)$ eset.

Tétel. G akkor és csakis akkor erősen $LL(1)$, ha $LL(1)$.

Bizonyítás. $\Leftarrow$: Már beláttuk, hogy ha G erősen $LL(k)$, akkor $LL(k)$ is, minden k-ra.

$\Rightarrow$: Indirekt: tfh G nyelvtan $LL(1)$, de nem erősen $LL(1)$.

Van két olyan különböző $A \to \beta$ és $A \to \gamma$ szabály, (tehát $\beta \neq \gamma$) melyekre

$$FI_1(\beta FO_1(A)) \cap FI_1(\gamma FO_1(A)) \neq \emptyset,$$

vagyis van olyan $a \in (\Sigma \cup \{\lambda\})$, hogy $a \in FI_1(\beta FO_1(A)) \cap FI_1(\gamma FO_1(A))$.

(a) eset: $a \in \Sigma$. Ekkor az $a \in FI_1(\beta FO_1(A)) \cap FI_1(\gamma FO_1(A))$ tartalmazás négyféleképpen valósulhat meg.

(a1) aleset: $a \in FI_1(\beta)$ és $a \in FI_1(\gamma)$. Ekkor léteznek az

- $S \Rightarrow_l^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wax'$
- $S \Rightarrow_l^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* way'$

levezetések, továbbá teljesül $FI_1(ax') = FI_1(ay') = a$. Ezért, mivel G nyelvtan $LL(1)$, $\beta = \gamma$-nak is teljesülnie kell, ami ellentmondás.

(a) eset: $a \in \Sigma$. Ekkor az $a \in FI_1(\beta FO_1(A)) \cap FI_1(\gamma FO_1(A))$ tartalmazás négyféleképpen valósulhat meg.

(a2) aleset: $\beta \Rightarrow^* \lambda, a \in FO_1(A)$ és $a \in FI_1(\gamma)$. Ekkor léteznek az

- $S \Rightarrow_l^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* w\alpha \Rightarrow^* wax'$
- $S \Rightarrow_l^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* way'$

levezetések, továbbá teljesül $FI_1(ax') = FI_1(ay')$. Ezért, G nyelvtan $LL(1)$ volta miatt $\beta = \gamma$, ami ellentmondás.

(a) eset: $a \in \Sigma$. Ekkor az $a \in FI_1(\beta FO_1(A)) \cap FI_1(\gamma FO_1(A))$ tartalmazás négyféleképpen valósulhat meg.

(a3) aleset: $a \in FI_1(\beta)$, $\gamma \Rightarrow^* \lambda$ és $a \in FO_1(A)$. Ennek az esetnek a bizonyítása az (a2)-höz hasonlóan történik.

(a4) aleset: $\beta \Rightarrow^* \lambda, a \in FO_1(A)$, továbbá $\gamma \Rightarrow^* \lambda$.

Hasonlóan.

(b) eset: $a = \lambda$. Ekkor a $\lambda \in FI_1(\beta FO_1(A)) \cap FI_1(\gamma FO_1(A))$ tartalmazás úgy valósulhat meg, hogy $\beta \Rightarrow^* \lambda, \gamma \Rightarrow^* \lambda$, továbbá $\lambda \in FO_1(A)$. Ezért léteznek az

- $S \Rightarrow_l^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* w\alpha \Rightarrow^* w$
- $S \Rightarrow_l^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* w\alpha \Rightarrow^* w$

levezetések, továbbá w után λ áll és teljesül, hogy $FI_1(\lambda) = FI_1(\lambda)$. Ezért, mivel G nyelvtan $LL(1)$ $\beta = \gamma$-nak is teljesülne kell, ami megint csak ellentmondás.

A tétel $k = 2$-re már nem igaz!

Az erősen $LL(k)$ elemzés alapötlete

Kérdés: $z \in L(G)$?

Tudjuk: $S \Rightarrow_l^* wA\alpha$ és $z = wz'$. Előrenézés: $u = FI_k(z')$.

Akkor A-t azzal az egyértelmű $A \to \beta$ alternatívával terjesztjük ki, melyre $u \in FI_k(\beta\alpha)$.

Mivel $FI_k(\alpha) \subseteq FO_k(A)$, annak is teljesülnie kell, hogy $u \in FI_k(\beta FO_k(A))$.

Az erősen $LL(K)$ definíciója szerint legfeljebb egy olyan szabály van, amire ez a tartalmazás teljesül, ha pedig nincs ilyen szabály, akkor $z \notin L(G)$.

A $FI_k(\beta FO_k(A))$ halmazok kiszámítása alapvető fontosságú!

Erősen $LL(k)$ nyelvtanok esetében egy M elemző tábla konstruálható a következőképpen.

M sorai az $N \cup \Sigma \cup \{\$\}$ halmaz elemeivel, oszlopai pedig a $\Sigma^{*,k}$ halmaz elemeivel vannak címkézve.

Ha $A \in N$, akkor M-nek az A sorhoz és $u \in \Sigma^{*,k}$ oszlophoz tartozó $M(A, u)$ eleme azt a tevékenységet írja le amit akkor kell végezni, ha az elemzés során A-t kell kiterjeszteni és u az előre nézett szó.

Ha $a \in \Sigma$, akkor M-nek az a sorhoz és az $u \in \Sigma^{*,k}$ oszlophoz tartozó $M(a, u)$ eleme azt adja meg, hogy mit kell csinálni, ha az előre nézett szó u és a-t kell illeszteni az elemzendő szóhoz.

M elemzőtábla definíciója:

(1) Minden $A \in N$ és $u \in \Sigma^{*,k}$ esetén, ha $u \in FI_k(\beta(FO_k(A))$ és $A \to \beta$ a j-edik szabály, akkor

$$M(A,u) = (\beta, j),$$

(2) Minden $a \in \Sigma$ és $u \in \Sigma^{*,k}$ esetén, ha $u = av$, akkor legyen $M(a,u)$=**pop**.

(3) Legyen $M(\$, \lambda)$=**elfogadás**,

(4) Minden más $X \in (N \cup \Sigma \cup \{\$\})$ és $u \in \Sigma^{*,k}$ esetén legyen $M(X,u)$=**hiba**.

Példa. G'_{ar} elemzőtáblája:

$M:$	a	$($	$)$	$+$	$*$	λ
K	$TT',1$	$TT',1$	**h**	**h**	**h**	**h**
T	$FF',4$	$FF',4$	**h**	**h**	**h**	**h**
T'	**h**	**h**	$\lambda,3$	$TT',2$	**h**	$\lambda,3$
F	$a,8$	$(K),7$	**h**	**h**	**h**	**h**
F'	**h**	**h**	$\lambda,6$	$\lambda,6$	$*FF',5$	$\lambda,6$
a	**p**	**h**	**h**	**h**	**h**	**h**
$($	**h**	**p**	**h**	**h**	**h**	**h**
$)$	**h**	**h**	**p**	**h**	**h**	**h**
$+$	**h**	**h**	**h**	**p**	**h**	**h**
$*$	**h**	**h**	**h**	**h**	**p**	**h**
$\$$	**h**	**h**	**h**	**h**	**h**	**e**

Erősen $LL(k)$ elemzési algoritmus.

Input G nyelvtan (ami erősen $LL(k)$) M elemző táblája és $z \in \Sigma^*$ szó.

Output *Igen* és z egy bal oldali levezetése, ha $z \in L(G)$, *Nem* különben.

Módszer

- Sorszámozzuk meg G szabályait.
- Az elemzés $(x, \alpha\$, \pi)$ alakú konfigurációk sorozata, ahol
  - $x \in \Sigma^*$ egy szó, az elemzendő szó hátralévő része,
  - $\alpha \in (N \cup \Sigma)^*$ egy verem, a levezetett bal mondatforma azon része, amely még további kiterjesztéseket és illesztéseket tartalmaz.
  - $\$$ az elemzendő szó végét jelző szimbólum, $\$ \notin (N \cup \Sigma)$.
  - π egy sor, szabályok sorszámaiból alkotott sorozat.
- Kezdő konfiguráció: $(z, S\$, \lambda)$. $z \in L(G)$ akkor és csakis akkor teljesül, ha $(z, S\$, \lambda) \vdash^* (\lambda, \$, \pi)$.

Az elemzési algoritmus

1. $C := (z, S\$, \lambda)$;
2. Amíg van olyan C', melyre $C \vdash C'$, legyen $C := C'$;
3. Ha $C = (\lambda, \$, \pi)$ alakú, valamely π-re, akkor output: *Igen* különben output: *Nem*.

Amennyiben az output *Igen*, π adja z egy bal oldali levezetését.

Példa. Input G'_{ar} és a $z = a * (a + a)$ szó.

$(\underline{a} * (a + a), K\$, \lambda) \vdash (\underline{a} * (a + a), TT'\$, 1) \vdash$
$(\underline{a} * (a + a), FF'T'\$, 14) \vdash (\underline{a} * (a + a), aF'T'\$, 148) \vdash$
$(\underline{*}(a + a), F'T'\$, 148) \vdash (\underline{*}(a + a), *FF'T'\$, 1485) \vdash$
$((\underline{a} + a), FF'T'\$, 1485) \vdash ((\underline{a} + a), (K)F'T'\$, 14857) \vdash$
...
$(\underline{\lambda}, F'T'\$, 14857148624863) \vdash (\underline{\lambda}, T'\$, 148571486248636) \vdash$
$(\underline{\lambda}, \$, 1485714862486363)$

Tehát $a * (a + a) \in L(G'_{ar})$ és 1485714862486363 a szó bal oldali levezetése.

Átmeneti reláció Tegyük fel, hogy az $(x, X\alpha, \pi)$ konfigurációban vagyunk (ahol $X\alpha \in (N \cup \Sigma)^*\$$). Képezzük $u = FI_k(x)$-et.

(1) Ha $X = A \in N$ és $M(A, u) = (\beta, j)$, akkor $(x, X\alpha, \pi) \vdash (x, \beta\alpha, \pi j)$. (A verem tetején egy nemterminális állt amelyet kiterjesztettünk.)

(2) Ha $X = a \in \Sigma$ és $M(a, u) =$**pop**, akkor $(x, X\alpha, \pi) \vdash (x', \alpha, \pi)$, ahol $x = ax'$. (Az illesztendő terminális illeszkedik az elemzendő szóhoz.)

(3) Ha $M(X, u) =$**elfogadás** (ami csak úgy lehet, ha $X = \$, \alpha = \lambda$ és $u = \lambda$), akkor nincs átmenet.

(4) Ha $M(X, u)=$**hiba**, akkor nincs átmenet.

Egy egyszerű példa $LL(1)$ nyelvtanra.

Definíció. G *egyszerű* $LL(1)$, ha λ-mentes és valahányszor $A \to \beta$ és $A \to \gamma$ két különböző szabály, mindannyiszor β és γ különböző terminális szimbólumokkal kezdődnek.

Egyszerű $LL(1) \Rightarrow$ erősen $LL(1) \Leftrightarrow LL(1)$

Ha $A \to a\beta'$ és $A \to b\gamma'$ különböző szabályok, akkor $FI_1(a\beta' FO_1(A)) = FI_1(a\beta') = \{a\}$ és $FI_1(b\gamma'(FO_1(A)) = \{b\}$, ezért

$$FI_1(a\beta' FO_1(A)) \cap FI_1(a\gamma' FO_1(A)) = \emptyset.$$

Faktorizálás

Egy módszer az $LL(1)$ konfliktus megszüntetésére.

Tekintsük az if-then-else nyelvtant:
$S \to iEtSeS \mid iEtS \mid x$
$E \to y$,
ami nyilvánvalóan nem $LL(1)$ az $S \to iEtSeS \mid iEtS$ szabályok miatt:

$$FI_1(iEtSeSFO_1(S)) \cap FI_1(iEtSFO_1(S)) = \{i\} \neq \emptyset.$$

Faktorizálás

Kiemeljük a közös prefixet és bevezetünk egy új változót:

$S \to iEtSeS \mid iEtS \mid x$
$E \to y$
helyett:
$S \to iEtSA \mid x$
$A \to eS \mid \lambda$
$E \to y$.

Ez utóbbi nyelvtan már $LL(1)$.
Persze a módszer nem működik minden nyelvtan esetén.

Faktorizálás

FACTOR(G)
Input: $G = (N, \Sigma, P, S)$ cf nyelvtan.

1) Minden $A \in N$-re:
2) Keressük meg A-szabályok jobb oldalainak leghosszabb $\alpha \neq \lambda$ közös prefixét.
- helyettesítsük az $A \to \alpha\beta_1 \mid \ldots \mid \alpha\beta_m \mid \gamma_1 \mid \ldots \mid \gamma_m$ szabályokat
- az $A \to \alpha A' \mid \gamma_1 \mid \ldots \mid \gamma_m$ és $A' \to \beta_1 \mid \ldots \mid \beta_m$ szabályokkal
3) Ismételjük 2)-t, amíg már nincsenek A-szabályok, melyek jobb oldalainak közös a prefixe.

Faktorizálás

Példa. A

- $K \to T + K \mid T$
- $T \to F * T \mid F$
- $F \to (K) \mid a$

nyelvtanból kapjuk a következő $LL(1)$ nyelvtanokat:

$K \to TT'$	$K \to TT'$
$T' \to +K \mid \lambda$	$T' \to +TT' \mid \lambda$
$T \to FF'$	$T \to FF'$
$F' \to *T \mid \lambda$	$F' \to *FF' \mid \lambda$
$F \to (K) \mid a$	$F \to (K) \mid a$

Alulról felfelé haladó elemzés

Alulról felfelé (bottom-up) haladó elemzési algoritmusok: a w szóból kiindulva próbálunk meg felépíteni egy olyan derivációs fát aminek a gyökere S, a határa pedig w.

Ha sikerül, $w \in L(G)$, különben $w \notin L(G)$.

Előkészület: ciklusmentesítés.

Definíció. Egy $G = (N, \Sigma, P, S)$ egy cf nyelvtan *ciklusmentes*, ha nincs olyan $A \in N$, melyre $A \Rightarrow^+ A$.

$A \Rightarrow^+ A \Rightarrow^+ A \Rightarrow^+ A \ldots$

Lemma. (Lásd tankönyv). Tetszőleges G cf nyelvtanhoz konstruálható olyan G' ciklusmentes cf nyelvtan, amelyre $L(G) = L(G')$.

Általános alulról felfelé elemzés

Adott $G = (N, \Sigma, P, S)$ cf nyelvtan és $w \in \Sigma^*$ szó, igaz-e, hogy $w \in L(G)$

Csak λ-mentes és ciklusmentes nyelvtanokkal foglalkozunk.

A w szóból kiindulva próbálunk meg felépíteni egy olyan derivációs fát aminek gyökere az S kezdőszimbólum.

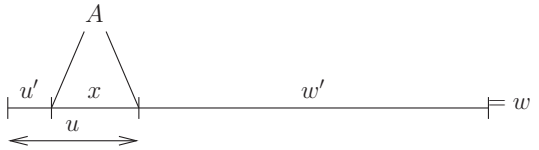
Két művelet:

Shiftelés: egy szimbólummal tovább olvassuk az input szót (ami kezdetben w).

Redukálás: egy szabály jobb oldalát a bal oldalával helyettesítjük.

Redukálás az $A \to x$ szabály szerint.

redukálás



Az általános alulról felfelé haladó elemzés alapötlete.

Olvassuk w-t az elejéről (vagyis shifteljünk) addig, amíg az elolvasott rész jobb oldali végén ki nem alakul egy szabály jobb oldala. Tehát legyen u olyan, amire teljesül, hogy $w = uw'$ és van olyan u' és x, hogy $u = u'x$ és $A \to x \in P$. Helyettesítsük x-et a szabály A bal oldalával, vagyis redukáljunk. Eredményül kapjuk az $u'Aw'$ szót, amelyre $u'Aw' \Rightarrow w$

az $u'Aw'$ szóra végezzük ezt tovább úgy, hogy w' elejétől kezdve

annak betűit hozzá shifteljük $u'A$-hoz addig, amíg az így kapott szó jobb oldali végén megint ki nem alakul egy szabály jobb oldala. Ha kialakult akkor redukálunk. Így próbáljuk meg elérni S-et.

Két probléma.

Shiftelés-redukálás konfliktus: Ha addig shifteltünk, hogy elértünk egy olyan pontot ahol redukálni is lehet, akkor nem tudjuk, hogy elvégezzük-e a redukálást vagy tovább shifteljünk egy későbbi olyan redukálás reményében, amitől azt várjuk, hogy sikeresen elvezet S-hez.

Redukálás-redukálás konfliktus: Ha elértünk egy olyan pontot, ahol redukálni lehet, akkor több olyan szabály is lehet, ami alkalmas a redukálásra. Ekkor nem tudjuk eldönteni, hogy melyik szabály szerint redukáljunk.

Megoldás.

Rögzítsük le a P-beli szabályok egy sorrendjét valamilyen módon. Ha shiftelés-redukálás konfliktussal találkozunk akkor végezzünk mindig redukálást. Továbbá, ha redukálás-redukálás konfliktusba ütközünk, akkor mindig a kisebb sorszámú szabály szerint redukáljunk.

Mivel a szabályok általunk felállított sorrendje önkényes, nem biztos, hogy a fenti elv mindig a helyes irányba visz. Mégsem mondhatjuk, hogy ekkor $w \notin L(G)$, mivel nem vettünk figyelembe minden lehetőséget. Ezért (csakúgy mint a top-down elemzésnél) fenn kell tartanunk a visszalépés, vagyis a backtrack lehetőségét.

Alulról felfelé haladó általános elemzési algoritmus.

Input Egy $G = (N, \Sigma, P, S)$ ciklusmentes és λ-mentes környezetfüggetlen nyelvtan és egy $w = a_1 \dots a_n \in \Sigma^*, n \geq 1$ szó.

Output *Igen* jelzés és a w szónak egy jobb oldali levezetése ha $w \in L(G)$. Különben *Nem* jelzés.

Módszer

- Rögzítsük le a szabályok egy sorrendjét.
- Az elemzés (p, i, α, β) alakú konfigurációk sorozata, ahol

- $p \in \{ q, b, t \}$, normál, elfogadó és b=backtrack állapot. Kezdő értéke: q.

- i az input szóba mutató pointer, $1 \leq i \leq n+1$.

- α egy verem, teteje a jobb végén van. Tartalma az input szó elolvasott részéből redukálásokkal keletkezett $(N \cup \Sigma)^*$-beli szó. Kezdőértéke: λ.

- β egy verem, teteje a bal végén van. Tartalma az elemzés "története" vagyis a végrehajtott shiftelések és redukálások sorozata. Kezdőértéke: λ.

• A konfigurációk halmazán megadunk egy $\vdash$ átmeneti relációt: $(p, i, \alpha, \beta) \vdash (p', i', \alpha', \beta')$ determinisztikus reláció. Az átmeneti relációt definiáló rész (1) lépésétől elindulva az első olyan lépést alkalmazzuk, ami alkalmazható a konfigurációra.

• Kezdő konfiguráció: $(q, 1, \lambda, \lambda)$. A befejező konfigurációk alakja $(t, n+1, S, \gamma)$ ahol γ egy s (shift) szimbólumból és szabályok sorszámaiból képzett sorozat. Az teljesül, hogy $w \in L(G)$ akkor és csakis akkor, ha $(q, 1, \lambda, \lambda) \vdash^* (t, n+1, S, \gamma)$.

Az elemzési algoritmus

(0) *Kezdő konfiguráció beállítása:* Legyen $C := (q, 1, \lambda, \lambda)$.

(1) *Redukálás:* Amíg $C = (q, i, \alpha\gamma, \beta)$ alakú és van $A \to \gamma$ alakú szabály, addig $\overline{C \vdash C'}$, ahol $C' = (q, i, \alpha A, j\beta)$ és j az $A \to \gamma$ szabály sorszáma, és a j-edik az első olyan szabály, amely szerint redukálás hajtható végre.

(Ha már nem lehet redukálni, akkor továbblépünk.)

(2) *Shiftelés:* Ha $C = (q, i, \alpha, \beta)$ és $i \neq n+1$, akkor $C \vdash C'$, ahol $C' = (q, i+1, \alpha a_i, s\beta)$ (a_i pedig az elemzendő szó i-edik betűje). Menjünk (1)-re (mert ha lehet, akkor redukálunk).

(3) *Elfogadás:* Ha $C = (q, n+1, S, \beta)$, akkor $C \vdash C' = (t, n+1, S, \beta)$. Az algoritmus álljon le és adjon ki *Igen* jelzést.

(4) *Átmenet backtrack állapotba:* Ha $C = (q, n+1, \alpha, \beta)$ de $\alpha \neq S$, akkor (nem értük el S-et tehát) $C \vdash C'$, ahol $C' = (b, n+1, \alpha, \beta)$. Menjünk (5)-re.

(5) Backtrack végrehajtása: (Egymást kizáró esetek.)

(i) Ha $C = (b, i, \alpha A, j\beta)$, a j-edik szabály $A \to \gamma$ és az $\alpha\gamma$ (vagyis a visszaállított szó) redukálható egy további $B \to \gamma'$ szabály szerint is (vagyis $\alpha\gamma = \alpha'\gamma'$), akkor $C \vdash C'$, ahol $C' = (q, i, \alpha' B, k\beta)$, és $k > j$ a $B \to \gamma'$ szabály sorszáma (az általunk rögzített sorrendben) és a j-edik után ez a legelső olyan szabály, amely szerint $\alpha\gamma$ szón redukálás hajtható végre.

Menjünk (1)-re (mert ha lehet, akkor redukálunk).

(5) Backtrack végrehajtása: (Egymást kizáró esetek.)

(ii) Ha $C = (b, i, \alpha A, j\beta)$, a j-edik szabály $A \to \gamma$ és $i \neq n+1$ de $\alpha\gamma$ nem redukálható más szabállyal, akkor $C \vdash C' = (q, i+1, \alpha\gamma a_i, s\beta)$.

Menjünk (1)-re (mert ha lehet, akkor redukálunk).

(iii) Ha $C = (b, n+1, \alpha A, j\beta)$ a j-edik szabály $A \to \gamma$ de $\alpha\gamma$ nem redukálható más szabállyal és már shiftelni sem tudunk akkor $C \vdash C' = (b, n+1, \alpha\gamma, \beta)$. Menjünk (5)-re.

(iv) Ha $C = (b, i, \alpha a, s\beta)$ és $i > 1$, akkor $C \vdash C' = (b, i-1, \alpha, \beta)$. Menjünk (5)-re.

(v) Ha egyik feltétel sem teljesül akkor $w \notin L(G)$, az algoritmus álljon le és adjon *Nem* jelzést outputra.

Alulról felfelé haladó általános elemzési algoritmus.

Példa 1. G_e: $1 : K \to K + T$ $2 : K \to T$
$3 : T \to a$ $4 : T \to b$.

Elemezzük a $w = b + a$ szót.

$(q, 1, \lambda, \lambda) \vdash (q, 2, b, s) \vdash (q, 2, T, 4s) \vdash (q, 2, K, 24s) \vdash$
$(q, 3, K+, s24s) \vdash (q, 4, K + a, ss24s) \vdash (q, 4, K + T, 3ss24s) \vdash$
$(q, 4, K, 13ss24s) \vdash (t, 4, K, 13ss24s)$,

Tehát $b + a \in L(G_e)$. Továbbá $13ss24s$-ből törölve az s-eket, $b + a$ jobb oldali levezetését kapjuk 1324.

Példa 2. Legyen G a következő nyelvtan

(1) $E \to E + T$
(2) $E \to T$
(3) $T \to T * F$
(4) $T \to F$
(5) $F \to a$

Elemezzük a $w = a * a$ szót.

Elemezzük a $w = a * a$ szót.

$(q, 1, \lambda, \lambda) \vdash (q, 2, a, s) \vdash (q, 2, F, 5s) \vdash (q, 2, T, 45s) \vdash$
$(q, 2, E, 245s) \vdash (q, 3, E*, s245s) \vdash (q, 4, E * a, ss245s) \vdash$
$(q, 4, E * F, 5ss245s) \vdash (q, 4, E * T, 45ss245s) \vdash (q, 4, E * E, 245ss245s) \vdash$
$(b, 4, E * E, 245ss245s) \vdash (b, 4, E * T, 45ss245s) \vdash (b, 4, E * F, 5ss245s) \vdash$
$(b, 4, E * a, ss245s) \vdash (b, 3, E*, s245s) \vdash (b, 2, E, 245s) \vdash (q, 3, T*, s45s) \vdash$
$(q, 4, T * a, ss45s) \vdash (q, 4, T * F, 5ss45s) \vdash (q, 4, T, 35ss45s) \vdash$
$(q, 4, E, 235ss45s) \vdash (t, 4, E, 235ss45s)$

Tehát $a * a \in L(G)$, egy jobb oldali levezetése pedig 23545.

Alulról felfelé haladó általános elemzési algoritmus.

Az alulról felfelé haladó általános elemzési algoritmus

tárbonyolultsága : $O(n)$ (lineáris),

időbonyolultsága: $O(c^n)$ (= exponenciális), ez nagyon nagy.

Alulról felfelé haladó általános elemzési algoritmus.

Felgyorsítási technikák.

(1) A szabályokat az alkalmazásuk valószínűségének a sorrendjébe rakjuk.

(2) Előrenézünk az inputban egy k hosszúságú u szót és abból következtetünk. Például, ha tudjuk, hogy u nem következhet egy A nemterminális után, akkor redukálunk A-ra.

(3) Backtrack felgyorsítása különböző technikákkal.

$LR(k)$ nyelvtanok és elemzésük

$G = (N, \Sigma, P, S)$ egy tetszőleges cf nyelvtan.

Az $LR(k)$ nyelvtanok, ahol $k \geq 0$ most is egy egész szám, olyan speciális környezetfüggetlen nyelvtanok, melyek esetében az alulról felfelé elemzéskor megnézzük a jobb oldali mondatforma még feldolgozatlan részének k hosszúságú prefixét és ennek segítségével fel tudjuk oldani mind a shiftelés-redukálás mind a redukálás-redukálás konfliktust.

Technikai okok miatt feltesszük, hogy S-nek csak egyetlen alternatívája van és S nem szerepel egyetlen szabály jobb oldalán sem.

Definíció. Legyen γ jobb mondatforma, $\gamma \neq S$. Azt mondjuk, hogy β a γ nyele, ha van olyan $A \to \beta$ szabály, melyre $S \Rightarrow_r^* \alpha A w \Rightarrow_r \alpha\beta w = \gamma$.

Az elemzés szempontjából fontos a nyél meghatározása: ha minden γ jobb mondatformának meg tudnánk határozni egy nyelét, akkor minden jobb mondatformát tudnánk redukálni is. Mivel ekkor újra jobb mondatformát kapunk, el tudnánk dönteni, hogy egy $z \in \Sigma^*$ szó redukálható-e (több lépésben) S-re, vagyis teljesül-e $z \in L(G)$.

Az $LR(k)$ nyelvtan definíciója

Definíció. Legyen $k \geq 0$ egy egész szám. Azt mondjuk, hogy a G nyelvtan $LR(k)$, ha valahányszor teljesülnek az

- $S \Rightarrow_r^* \alpha Aw \Rightarrow_r \alpha\beta w$
- $S \Rightarrow_r^* \gamma Bx \Rightarrow_r \gamma\delta x = \alpha\beta y$
- $FI_k(w) = FI_k(y)$

feltételek, mindannyiszor $\alpha = \gamma, A = B$ és $\beta = \delta$.

Észrevétel. Ha egy nyelvtan $LR(k)$, akkor az $LR(k+1)$ is.

Tfh, hogy G $LR(k)$ és el akarjuk dönteni, hogy $z \in L(G)$ teljesül-e. Tfh, azt már tudjuk, hogy $\omega \Rightarrow_r^* z$. Ekkor az ω szónak nem lehet két különböző nyele. Ha ugyanis egyrészt $\omega = \alpha\beta w$, másrészt $\omega = \gamma\delta x$, továbbá vannak olyan $A \to \beta$ és $B \to \delta$ szabályok melyekre

- $S \Rightarrow_r^* \alpha Aw \Rightarrow_r \alpha\beta w$ és
- $S \Rightarrow_r^* \gamma Bx \Rightarrow_r \gamma\delta x = \alpha\beta w$,

akkor $FI_k(w) = FI_k(w)$, és az $LR(k)$ feltétel miatt azt kapjuk, hogy $\alpha = \gamma$, $A = B$ és $\beta = \delta$, tehát mind a nyél, mind a redukció egyértelműen meghatározott.

A továbbiakban azt is látni fogjuk, hogy az $LR(k)$ feltétel szintén feloldja a shiftelés-redukálás konfliktust is.

Az $LR(k)$ nyelvtanok esetében is a következő két kérdés megválaszolására koncentrálunk:

(1) Ha adott egy G nyelvtan és egy k szám, akkor eldönthető-e, hogy G nyelvtan $LR(k)$-e?

(2) Hogyan működik az $LR(k)$ elemzés?

Mindkét kérdés megválaszolásához szükségünk van az $LR(k)$ feltétel egy olyan átfogalmazására, amely gyakorlati szempontból jobban használható. Ehhez azonban szükségünk van a következő fogalmakra.

Definíció. Egy $\gamma \in (N \cup \Sigma)^*$ *járható prefix*, ha van olyan $\alpha\beta w$ jobb mondatforma melynek nyele β és teljesül rá, hogy γ prefixe $\alpha\beta$-nak.

A járható prefix egy olyan szó, amely prefixe egy jobb mondatformának, de nem nyúlik túl a nyél jobb oldali végén.

Az elemzés során lényegében járható prefixeket állítunk elő. Célunk az, hogy a járható prefixekhez addig shifteljünk újabb szimbólumokat, amíg el nem érjük a nyél jobb oldali végét, vagyis ki nem alakul a nyél. A definíció szerint az így kapott szó is járható prefix, ami azonban már redukálható.

Definíció. Az $[A \to \beta_1.\beta_2, u]$ alakú párokat $LR(k)$ *elemeknek* nevezzük, ahol $A \to \beta_1\beta_2 \in P$ és $u \in \Sigma^{*,k}$.) (A $\beta_1 = \lambda$ és/vagy a $\beta_2 = \lambda$ is lehetséges.)

Definíció. Az $[A \to \beta_1.\beta_2, u]$ $LR(k)$ elem *érvényes az* $\alpha\beta_1$ *járható prefixre*, ha létezik $S \Rightarrow_r^* \alpha Aw \Rightarrow \alpha\beta_1\beta_2 w$ deriváció úgy, hogy $u = FI_k(w)$.

Definíció. Az $[A \to \beta_1.\beta_2, u]$ $LR(k)$ elem *érvényes az* $\alpha\beta_1$ *járható prefixre*, ha létezik $S \Rightarrow_r^* \alpha Aw \Rightarrow \alpha\beta_1\beta_2 w$ deriváció úgy, hogy $u = FI_k(w)$.

Annak, hogy egy $LR(k)$ elem érvényes egy jártható prefixre, a szemléletes jelentése a következő:

- Ha az $LR(k)$ elem $[A \to \beta_1.\beta_2, u]$ alakú, ahol $\beta_2 \neq \lambda$, akkor az $A \to \beta_1\beta_2$ szabály szóba jöhet egy későbbi redukálásnál, mert ezen szabály jobb oldalának „." előtti része (vagyis β_1) már megegyezik a jártható prefix „végével".

- Ha az $LR(k)$ elem $[A \to \beta., u]$ alakú, akkor az $A \to \beta$ szabály szóba jöhet a játható prefix redukálásánál.

Adott γ járható prefix esetén a γ-ra érvényes $LR(k)$ elemek halmazát $V_k(\gamma)$-val jelöljük. Minden γ-ra $V_k(\gamma)$ véges halmaz. A $V_k(\gamma)$-kat $LR(k)$ *tábláknak* hívjuk (mely elnevezés onnan adódik, hogy egy $V_k(\gamma)$ halmazban lévő járható prefixeket egymás alá írva táblázatszerű elrendezést kapunk).

Bevezetjük a

$$\mathcal{T} = \{V_k(\gamma) \,|\, \gamma \text{ járható prefix}\}$$

jelölést.

Ismét megjegyezzük, hogy – bár általában végtelen sok γ járható prefix van – $\mathcal{T}$ is véges halmaz, ugyanis részhalmaza az $LR(k)$ elemek halmaza hatványhalmazának, ami megint csak véges.

$V_k(\gamma)$ meghatározása.

Input G nyelvtan és $\gamma = X_1 \ldots X_n$ szó, ahol $X_1, \ldots, X_n \in (N \cup \Sigma)$.

Output $V_k(\gamma)$ halmaz.

Módszer Kiszámoljuk a $V_k(\lambda), V_k(X_1), \ldots, V_k(X_1 \ldots X_n)$ halmazokat.

Algoritmus

(a) $V_k(\lambda)$ kiszámítása

(1) (Inicializálás.) Minden $S \to \alpha \in P$ esetén legyen $[S \to .\alpha, \lambda] \in V_k(\lambda)$.

(2) (Lezárás.) Mindaddig, amíg $V_k(\lambda)$ bővíthető, a $V_k(\lambda)$ minden $[A \to .B\beta, u]$ alakú elemére és $B \to \delta$ P-beli szabályra, legyen $[B \to .\delta, v] \in V_k(\lambda)$ minden $v \in FI_k(\beta u)$-ra.

Algoritmus

(b) Tegyük fel, hogy $V_k(X_1 \dots X_{i-1})$-et már kiszámítottuk. Ekkor kiszámítjuk $V_k(X_1 \dots X_i)$-t.

(1) (Léptetés.) Minden $[A \to \alpha.X_i\beta, u]$ alakú $V_k(X_1 \dots X_{i-1})$-beli elem esetén legyen $[A \to \alpha X_i.\beta, u] \in V_k(X_1 \dots X_i)$.

(2) (Lezárás.) Mindaddig, amíg $V_k(X_1 \dots X_i)$ bővíthető, a $V_k(X_1 \dots X_i)$ minden $[A \to \alpha.B\beta, u]$ alakú elemére és $B \to \delta$ P-beli szabályra, legyen $[B \to .\delta, v] \in V_k(X_1 \dots X_i)$ minden $v \in FI_k(\beta u)$-ra.

Példa. ($LR(1)$ táblák) G_{list}

$$\begin{aligned} S &\to L \\ L &\to L * E \mid E \\ E &\to a \mid b \end{aligned}$$

$$\begin{aligned} T_0 = V_1(\lambda) = {} & [S \to .L, \lambda] \\ & [L \to .L * E, \lambda \,/\, *] \\ & [L \to .E, \lambda \,/\, *] \\ & [E \to .a, \lambda \,/\, *] \\ & [E \to .b, \lambda \,/\, *] \end{aligned}$$

$$\begin{aligned} T_1 = V_1(L) = {} & [S \to L., \lambda] \\ & [L \to L. * E, \lambda \,/\, *] \end{aligned}$$

$\mathcal{T} = \{V_k(\gamma) \mid \gamma$ járható prefix $\}$ **kiszámítása.**

Input G nyelvtan.

Output $\mathcal{T} = \{V_k(\gamma) \mid \gamma$ járható prefix $\}$.

Algoritmus

(1) Legyen $\mathcal{T} = \{V_k(\lambda)\}$ ($V_k(\gamma)$ Algoritmus (a) pont).

(2) Mindaddig amíg $\mathcal{T}$-ben van megjelöletlen táblázat tegyük a következőt:
- Vegyünk egy megjelöletlen $T \in \mathcal{T}$ elemet. Legyen ez $T = V_k(\gamma)$.
- Jelöljük meg T-t.
- Minden $X \in (N \cup \Sigma)$-ra számoljuk ki $T' = V_k(\gamma X)$-et ($V_k(\gamma)$ Algoritmus (b) pont). Ha $T' \notin \mathcal{T}$ akkor T'-t vegyük fel $\mathcal{T}$-be, különben lépjünk tovább.

Mivel az $LR(k)$ táblák halmaza (vagyis $\mathcal{T}$) véges, a fenti algoritmus véges számú lépés után terminál.

Példa. ($LR(1)$ táblák) G_{list}: $S \to L$
$L \to L * E \mid E$
$E \to a \mid b$

$T_0 = V_1(\lambda) = [S \to .L, \lambda]$
$[L \to .L * E, \lambda \,/\, *]$
$[L \to .E, \lambda \,/\, *]$
$[E \to .a, \lambda \,/\, *]$
$[E \to .b, \lambda \,/\, *]$

$T_1 = V_1(L) = [S \to L., \lambda]$
$[L \to L. * E, \lambda \,/\, *]$

$T_2 = V_1(E) = [L \to E., \lambda \,/\, *]$

$T_3 = V_1(a) = [E \to a., \lambda \,/\, *]$

$T_4 = V_1(b) = [E \to b., \lambda \,/\, *]$

$T_5 = V_1(L*) = [L \to L * .E, \lambda \,/\, *]$
$[E \to .a, \lambda \,/\, *]$
$[E \to .b, \lambda \,/\, *]$

$T_6 = V_1(L * E) = [L \to L * E., \lambda \,/\, *]$

Az $LR(k)$ tétel átfogalmazása.

Definíció. Tetszőleges α szóra
$EFF_k(\alpha) = \{FI_k(x) \mid \alpha \Rightarrow_r^* \beta \Rightarrow_r x$ úgy, hogy $\beta \neq Ax$ semmilyen $A \in N$-re$\}$.

Tétel. ($LR(k)$ tétel.) G akkor és csakis akkor $LR(k)$, ha minden $T \in \mathcal{T}$ $LR(k)$ táblára igaz, hogy ha T-ben van $[A \to \beta., u]$ alakú elem, akkor nincs benne egy olyan másik $[B \to \beta_1.\beta_2, v]$ elem, melyre $u \in EFF_k(\beta_2 v)$. (Ezt röviden úgy fejezzük ki, hogy minden $LR(k)$ tábla konzisztens.)

Tétel. G-ről eldönthető, hogy $LR(k)$-e.

Bizonyítás. Számoljuk ki $\mathcal{T}$-t, majd vegyük sorra $\mathcal{T}$-nek minden T elemét. Amennyiben T-ben nincs $[A \to \beta., u]$ alakú elem, válasszuk a következő táblázatot.

Ha T-ben találunk $[A \to \beta., u]$ alakú elemet, akkor T minden $[B \to \beta_1.\beta_2, v]$ alakú elemére vizsgáljuk meg, hogy $u \in EFF_k(\beta_2 v)$ teljesül-e. (Itt kihasználtuk, hogy $EFF_k(\beta_2 v)$ kiszámítható.) Ha találunk ilyen elemet T-ben, akkor G nem $LR(k)$. Ha egyetlen T táblázat esetén sem találunk két olyan elemet, amelyek kielégítik ezt a feltételt, akkor G $LR(k)$.

Példa. G_{list} $LR(1)$:

$$\begin{array}{rcl} T_1 = V_1(L) & = & [S \to L., \lambda] \\ & & [L \to L. * E, \lambda \,/\, *] \end{array}$$

Az $[S \to L., \lambda]$-t és az $[L \to L. * E, \lambda \,/\, *]$ elemek konfliktusban lehetnek. De $EFF_1(*E\lambda) = EFF_1(*E*) = \{*\}$ és $\lambda \notin \{*\}$, így az $LR(1)$ feltétel megint csak nem sérül.

Hasonlóan látható a $T_2, \ldots, T_6$ táblák esetén is.

Példa. G_{list} $LR(1)$:

$$\begin{array}{rcl} T_0 = V_1(\lambda) & = & [S \to .L, \lambda] \\ & & [L \to .L * E, \lambda \,/\, *] \\ & & [L \to .E, \lambda \,/\, *] \\ & & [E \to .a, \lambda \,/\, *] \\ & & [E \to .b, \lambda \,/\, *] \end{array}$$

Ebben nincs $[A \to \beta., u]$ alakú elem, tehát nem lehet konfliktus.

Az $LR(k)$ elemzés működése (Tfh G $LR(k)$)

Megkonstruáljuk az $LR(k)$ *elemző táblát*, amely két részre osztható: az f *tevékenység táblára* és a g *goto táblára*. Mindkét tábla sorai $\mathcal{T}$ elemeivel vannak címkézve.

Az f tevékenység tábla oszlopai $\Sigma^{*,k}$ elemeivel vannak címkézve. A $T \in \mathcal{T}$ sorhoz és az $u \in \Sigma^{*,k}$ oszlophoz tartozó $f(T, u)$ sor azt írja le, hogy mit kell tenni, ha a vizsgált járható prefixre érvényes $LR(k)$ elemek halmaza T, az előrenézett szó pedig u.

A g goto tábla oszlopai $N \cup \Sigma$ elemeivel vannak címkézve. A $T = V_k(\gamma)$ sor és az X oszlop által meghatározott elem $V_k(\gamma X)$, tehát $g(T, X) = V_k(\gamma X)$.

$f(T,u)$ pontos definíciója.

(1) Ha T-ben van $[A \to \beta., u]$ alakú elem, akkor legyen $f(T,u) =$(**redukálás**, j), ahol j az $A \to \beta$ szabály sorszáma.
(2) ha T-ben van $[A \to \beta_1.\beta_2, v]$ alakú elem, ahol $\beta_2 \neq \lambda$ és $u \in EFF_k(\beta_2 v)$, akkor legyen $f(T,u) =$**shift**.
(3) Ha $u = \lambda$ és T-ben van $[S \to \alpha., \lambda]$ alakú elem, akkor $f(T,u) =$**elfogadás**.
(4) Minden más esetben legyen $f(T,u) =$**hiba**.

A g goto tábla oszlopai $N \cup \Sigma$ elemeivel vannak címkézve. A $T = V_k(\gamma)$ sor és az X oszlop által meghatározott elem $V_k(\gamma X)$, tehát $g(T,X) = V_k(\gamma X)$.

Példa. G_{list}: 1: $S \to L$
2: $L \to L * E$ 3: $L \to E$
4: $E \to a$ 5: $E \to b$ tevékenység táblája:

$f:$	a	b	$*$	λ
T_0	**s**	**s**	**h**	**h**
T_1	**h**	**h**	**s**	**e**
T_2	**h**	**h**	**r** 3	**r** 3
T_3	**h**	**h**	**r** 4	**r** 4
T_4	**h**	**h**	**r** 5	**r** 5
T_5	**s**	**s**	**h**	**h**
T_6	**h**	**h**	**r** 2	**r** 2

$g:$	S	L	E	a	b	$*$
T_0		T_1	T_2	T_3	T_4	
T_1						T_5
T_2						
T_3						
T_4						
T_5			T_6	T_3	T_4	
T_6						

$LR(k)$ elemzési algoritmus

Input G ($LR(k)$) nyelvtan $LR(k)$ elemző (tevékenység és goto) táblája és $w \in \Sigma^*$ szó.

Output *Igen* és w jobb oldali elemzése, ha $w \in L(G)$, *Nem* különben.

Módszer

• Az elemzés $(x, \alpha T, \pi)$ alakú konfigurációk sorozata, ahol

- $x \in \Sigma^*$ egy szó, az elemzendő szó hátralévő (vagyis a járható prefixen túli) része,

- $\alpha T \in (\mathcal{T} \cup N \cup \Sigma)^*$, ahol α tartalmazza a járható prefixet úgy, hogy ennek minden prefixe után α-ban benne van a rá érvényes $LR(k)$ tábla is. (Így T érvényes α'-re, ahol α'-t úgy kapjuk α-ból, hogy töröljük belőle a $\mathcal{T}$-beli szimbólumokat.) Egy veremnek tekintjük, melynek teteje a jobb végén van.

- π egy, a szabályok sorszámaiból alkotott sorozat. Egy sornak tekintjük, melynek teteje a bal végén van.

Módszer

• Kezdő konfiguráció: (w, T_0, λ), ahol $T_0 = V_k(\lambda)$ (vagyis a λ járható prefixre érvényes $LR(k)$ tábla). A befejező konfigurációk $(\lambda, T_0\alpha T, \pi)$ alakúak.

A konfigurációk közötti átmenetet $\vdash$-val jelöljük.

Az átmeneti reláció úgy van definiálva, hogy $w \in L(G)$ akkor és csakis akkor teljesül, ha a kezdő konfigurációból elértünk egy olyan $(\lambda, T_0\alpha T, \pi)$ konfigurációt, melyre $f(T, \lambda) =$**elfogadás**. (Ekkor α-ból elhagyva az $LR(k)$ tábla szimbólumokat, az S egyetlen alternatíváját kapjuk.)

Az elemzési algoritmus

1. $C := (w, T_0, \lambda)$;
2. Amíg van olyan C', melyre $C \vdash C'$, legyen $C := C'$;
3. Ha $C = (\lambda, T_0\alpha T, \pi)$ alakú, valamely π-re és $f(T, \lambda) =$**elfogadás**, akkor output: *Igen* különben output: *Nem*.

Amennyiben az output *Igen*, akkor 1π adja w egy jobb oldali levezetését, ahol 1 az egyetlen S bal oldalú szabály sorszáma.

Átmeneti reláció Tfh az $(x, \alpha T, \pi)$ konfigurációban vagyunk. Képezzük $u = FI_k(x)$-et.

(1) Ha $f(T, u) =$ (**redukálás**, j), akkor αT veremből törlünk $2 \cdot |\beta|$ darab szimbólumot, ahol $A \to \beta$ a j-edik szabály. Tegyük fel, hogy a veremben marad $\gamma T'$. Akkor $(x, \alpha T, \pi) \vdash (x, \gamma T' A T'', j\pi)$ ahol $T'' = g(T', A)$.

(2) Ha $f(T, u) =$**shift**, akkor $(x, \alpha T, \pi) \vdash (x', \alpha T a T', \pi)$, ahol $x = ax'$ és $T' = g(T, a)$.

(3) Ha $f(T, u) =$**elfogadás** (ami csak úgy lehet, ha T-ben van $[S \to \alpha., \lambda]$ alakú elem és $u = \lambda$ egyidejűleg teljesülnek) akkor nincs átmenet.

(4) Ha $f(T, u)=$**hiba**, akkor nincs átmenet.

Tekintsük a G_{list} nyelvtant és a $w = a * b * a$ szót. Adjuk meg szó elemzését.

$(a * b * a, T_0, \lambda) \vdash (*b * a, T_0 a T_3, \lambda) \vdash (*b * a, T_0 E T_2, 4) \vdash$
$(*b * a, T_0 L T_1, 34) \vdash (b * a, T_0 L T_1 * T_5, 34) \vdash$
$(*a, T_0 L T_1 * T_5 b T_4, 34) \vdash (*a, T_0 L T_1 * T_5 E T_6, 534) \vdash$
$(*a, T_0 L T_1, 2534) \vdash (a, T_0 L T_1 * T_5, 2534) \vdash$
$(\lambda, T_0 L T_1 * T_5 a T_3, 2534) \vdash$
$(\lambda, T_0 L T_1 * T_5 E T_6, 42534) \vdash (\lambda, T_0 L T_1, 242534)$

Mivel $f(T_1, \lambda) =$**elfogadás**, az $a * b * a$ szó eleme $L(G_{list})$ nyelvnek. Továbbá, az 1242534 szabály sorszám sorozat ezen szó jobb oldali levezetését adja.

Összefüggések $LL(k)$ és $LR(k)$ nyelvek között

Tétel. Ha egy nyelvtan $LL(k)$, akkor az $LR(k)$ is.

Egy L nyelvet $LL(k)$-nak (hasonlóan $LR(k)$-nak) mondunk, ha van olyan G $LL(k)$ (hasonlóan $LR(k)$) nyelvtan, melyre $L = L(G)$.

Tétel. Minden $LR(k)$ nyelv generálható $LR(1)$ nyelvtannal is. Továbbá, az $LR(1)$ nyelvek osztálya megegyezik a determinisztikus nyelvek osztályával.

$LALR(k)$ nyelvtanok és elemzésük

A $LALR(k)$ nyelvtanok lényegében speciális $LR(k)$ nyelvtanok: ebben az esetben a $LALR(k)$ táblákat egy, az $LR(k)$ táblák kiszámításához képest „egyszerűsített" algoritmussal számoljuk ki. Az egyszerűsítés által a $LALR(k)$ táblák száma általában kevesebb mint az $LR(k)$ táblák száma és így az elemzési algoritmus gyorsabb lehet. Ugyanakkor az egyszerűsítés eredményeképpen előfordulhat, hogy valamely $LALR(k)$ tábla nem lesz konzisztens, habár az $LR(k)$ táblák halmaza konzisztens.

$LALR(k)$ nyelvtanok és elemzésük

A $LALR(k)$ táblákat kiszámító algoritmus megadása előtt bevezetünk egy jelölést.

Tetszőleges T $LR(k)$ tábla esetén legyen

$$szab(T) = \{A \to \beta_1.\beta_2 \in P \mid [A \to \beta_1.\beta_2, u] \in T, \text{ valamely } u \in \Sigma^{*,k}\text{-ra}\},$$

vagyis legyen $szab(T)$ a T-ben előforduló „pontozott" szabályok halmaza.

Algoritmus $LALR(k)$ táblák kiszámítása.

Input G nyelvtan.

Output G $LALR(k)$ tábláinak $\mathcal{T}$ halmaza.

Algoritmus

(1) Legyen $\mathcal{T} = \{V_k(\lambda)\}$ (járható prefix algoritmus (a) pont).

(2) Mindaddig amíg $\mathcal{T}$-ben van megjelöletlen táblázat, do:

- Vegyünk egy megjelöletlen $T \in \mathcal{T}$ elemet.
- Jelöljük meg T-t.
- Minden $X \in (N \cup \Sigma)$-ra számoljuk ki $T' = g(T, X)$-et (járható prefix algoritmus (b) pont) és do:
  * Ha T' már $\mathcal{T}$-ben van, akkor lépjünk tovább.
  * Ha minden $T'' \in \mathcal{T}$-re teljesül $szab(T') \neq szab(T'')$, akkor T'-t vegyük fel $\mathcal{T}$-be és lépjünk tovább.
  * Ha valamely $T'' \in \mathcal{T}$-re teljesül $szab(T') = szab(T'')$, akkor képezzük $\bar{T} = T' \cup T''$ táblát és vegyük fel $\mathcal{T}$-be.

$LALR(k)$ nyelvtan definíciója

Definíció. Egy G nyelvtanról akkor mondjuk, hogy $LALR(k)$, ha a fenti módon kiszámított $LALR(k)$ tábláinak a halmaza konzisztens, vagyis ha minden $T \in \mathcal{T}$ $LALR(k)$ táblára igaz, hogy ha T-ben van $[A \to \beta., u]$ alakú elem, akkor nincs benne egy olyan másik $[B \to \beta_1.\beta_2, v]$ elem, melyre $u \in EFF_k(\beta_2 v)$.

Az $LR(k)$ Tétel, valamint a $LL(k)$ és a $LALR(k)$ táblákat kiszámoló algoritmusok egybevetéséből azonnal adódik, hogy minden $LALR(k)$ nyelvtan egyben $LR(k)$ is.

$LALR(k)$ elemzés

Amennyiben G $LALR(k)$ tulajdonságú, úgy az $LR(k)$ nyelvtanokról szóló részben leírtaknak megfelelően megkonstruálható G $LALR(k)$ *elemző táblája* azzal a különbséggel, hogy most a $LALR(k)$ táblák halmazával dolgozunk. Továbbá, a $LALR(k)$ elemző tábla ismeretében az $LR(k)$ elemzési algoritmussal (ld. $LR(k)$ Algoritmus) analóg módon adható meg a $LALR(k)$ elemzési algoritmus.

Precedencia nyelvtanok és elemzésük

Definíció. Egy $G = (N, \Sigma, P, S)$ nyelvtan *egyértelműen redukálható*, ha valahányszor $A \to \beta$ és $B \to \beta$ P-beli szabályok, mindannyiszor $A = B$ teljesül.

Egyszerű precedencia nyelvtanok

Egyértelműen redukálható, továbbá a nyelvtani szimbólumok (terminálisok, nemterminálisok) között fennállnak a $\prec, \doteq$ és $\succ$ *precedencia relációk* melyek intuitív jelentése a következő. Legyen $\alpha\beta w$ egy jobb oldali mondatforma, melynek a nyele β. Akkor

- α bármely két betűje között $\prec$ vagy $\doteq$ reláció áll fenn,
- α utolsó és β első betűje között a $\prec$ reláció áll fenn,
- a nyél, vagyis β bármely két betűje között az $\doteq$ reláció áll fenn,
- β utolsó és w első betűje között a $\succ$ reláció áll fenn.

Egy jobb mondatforma nyelét úgy találhatjuk meg, hogy addig shifteljük az inputot (még akkor is, ha az már redukálható), amíg azt nem találjuk, hogy a legutoljára beolvasott betű és a hátralévő input első betűje között a $\succ$ reláció áll fenn.

Ha ez teljesül, akkor megtaláltuk a nyél jobb oldali végét.

Ezután a beolvasott szimbólumok között addig olvasunk visszafelé, amíg két betű között a $\prec$ relációt nem találjuk. Ahol ez áll fenn, ott van a nyél bal oldali vége (vagyis az eleje).

Ha a nyelet megtaláltuk, akkor a jobb mondatforma redukálása egyértelműen elvégezhető (tehát nem lesz redukálás-redukálás konfliktus sem).

Az egyszerű precedencia nyelvtanok esetében is a következő két kérdés megválaszolására koncentrálunk:

(1) Ha adott egy G nyelvtan, akkor eldönthető-e, hogy G egyszerű precedencia nyelvtan-e?

(2) Hogyan működik az egyszerű precedencia elemzés?

Legyen $G = (N, \Sigma, P, S)$ egy környezetfüggetlen nyelvtan. A $\prec$, $\doteq$ és $\succ$ relációk az $N \cup \Sigma \cup \{\$\}$ halmazon értelmezett legszűkebb olyan relációk, melyekre az alábbi feltételek teljesülnek:

- minden $X \in (N \cup \Sigma)$-ra, ha $S \Rightarrow^+ X\alpha$, valamely α-ra, akkor $\$ \prec X$;
- minden $X, Y \in (N \cup \Sigma)$-ra, ha van olyan $A \to \alpha X B \beta$ szabály, melyre $B \Rightarrow^+ Y\gamma$, valamely γ-ra, akkor $X \prec Y$;
- minden $X, Y \in (N \cup \Sigma)$-ra, ha van $A \to \alpha X Y \beta$ alakú szabály, akkor $X \doteq Y$;

- minden $X \in (N \cup \Sigma)$-ra és $a \in \Sigma$-ra, ha van olyan $A \to \alpha B Y \beta$ szabály, melyre $B \Rightarrow^+ \gamma X$ és $Y \Rightarrow^* a\delta$, valamely γ-ra és δ-ra, akkor $X \succ a$ (ahol az $Y \Rightarrow^* a\delta$ levezetés nulla lépésben is teljesülhet, tehát úgy, hogy $Y = a$);
- minden $X \in (N \cup \Sigma)$-ra, ha $S \Rightarrow^+ \alpha X$, valamely α-ra, akkor $X \succ \$$.

Az előbbi precedencia relációk könnyem meghatározhatók. Pl. $\prec$ esetén: ha van olyan $A \to \alpha X B \beta$ szabály, melyre $B \Rightarrow^+ Y\gamma$, valamely γ-ra, akkor $X \prec Y$.

$H_B = \{Y \in (N \cup \Sigma) \,|\, B \Rightarrow^+ Y\gamma\}$ halmaz kiszámítása:

1. Legyen $i = 0$ és $H_i = \{Y \in (N \cup \Sigma) \,|\, B \to Y\alpha \in P\}$;
2. Legyen $H_{i+1} = H_i \cup \{Y \in (N \cup \Sigma) \,|\, \exists (A \in H_i) : A \to Y\alpha \in P\}$;
3. Ha $H_{i+1} = H_i$ akkor álljunk meg, különben legyen $i = i + 1$, és menjünk 2-re;

Nyilvánvaló annak igazolása, hogy az algoritmus véges lépés után $H_i = H_B$ mellett terminál.

Definíció. Egy $G = (N, \Sigma, P, S)$ cf nyelvtan egyszerű precedencia nyelvtan, ha rá a következők teljesülnek:

- G ciklusmentes és λ-mentes,
- G egyértelműen redukálható,
- minden $X, Y \in (N \cup \Sigma)$-ra a $\prec, \doteq$ és $\succ$ relációk közül *legfeljebb egy* áll fenn X és Y között. □

Tétel. Tetszőleges G környezetfüggetlen nyelvtanról eldönthető, hogy egyszerű precedencia nyelvtan-e.

Az egyszerű precedencia elemzés

Tétel. Legyen $G = (N, \Sigma, P, S)$ egy λ-mentes cf nyelvtan. Továbbá tegyük fel, hogy

$$\begin{aligned} \$S\$ &\Rightarrow_r^n X_1 \dots X_k A a_1 \dots a_l \\ &\Rightarrow_r X_1 \dots X_k Y_1 \dots Y_m a_1 \dots a_l. \end{aligned}$$

Akkor fennállnak a következő relációk:

(1) minden $1 \le i < k$ esetén $X_i \prec X_{i+1}$ vagy $X_i \doteq X_{i+1}$,

(2) $X_k \prec Y_1$,

(3) minden $1 \le i < m$ esetén $Y_i \doteq Y_{i+1}$,

(4) $Y_m \succ a_1$.

Megjegyzés. Azt nem kell feltenni, hogy G egyszerű precedencia nyelvtan, mivel az állítás csak azt mondja ki, hogy az (1)-(4) pontokban szereplő relációk fennállnak de azt nem, hogy csak ezek állnak fenn.

Következmény. Az előző tétel feltételei mellett tegyük fel még azt is, hogy G egyszerű precedencia nyelvtan. Ekkor az (1) állításban a vagy feltétel kizáró vagyot jelent, továbbá az (1)-(4) állítások mindegyike kiegészíthető azzal, hogy a megfelelő helyeken más reláció pedig nem áll fenn.

Egyszerű precedencia elemzési algoritmus.

Input G (egyszerű precedencia) nyelvtan, precedencia mátrix és $w \in \Sigma^*$ szó. G szabályai meg vannak számozva.

Output *Igen* és w jobb oldali elemzése, ha $w \in L(G)$, *Nem* különben.

Módszer

- Az elemzés $(\$\alpha, x\$, \pi)$ alakú konfigurációk sorozata, ahol
  - $\alpha \in (N \cup \Sigma)^*$, az elemzendő szó már beolvasott részéből redukált járható prefix. Egy veremnek tekintjük, melynek teteje a jobb végén van.

  - $x \in \Sigma^*$ egy szó, az elemzendő szó hátralévő része,
  - teljesül továbbá, hogy $\alpha x \Rightarrow_r^* w$
  - π szabályok sorszámaiból alkotott sorozat. Egy sornak tekintjük, melynek teteje a bal végén van.
- Kezdő konfiguráció: $(\$, w\$, \lambda)$. A befejező konfigurációk $(\$S, \$, \pi)$ alakúak. A konfigurációk közötti átmenetet $\vdash$-val jelöljük. Az átmeneti reláció úgy van definiálva, hogy $w \in L(G)$ akkor és csakis akkor teljesül, ha $(\$, w\$, \lambda) \vdash^* (\$S, \$, \pi)$ valamely π-re és ekkor π adja a w egy jobb oldali levezetését.

Az elemzési algoritmus

1. $C := (\$, w\$, \lambda)$;
2. Amíg van olyan C', melyre $C \vdash C'$, legyen $C := C'$;
3. Ha $C = (\$S, \$, \pi)$ alakú, valamely π-re, akkor output: *Igen*, különben output: *Nem*.

Amennyiben az output *Igen*, akkor π adja w egy jobb oldali levezetését.

Átmeneti reláció Tegyük fel, hogy a $C = (\$\alpha, x\$, \pi)$ konfigurációban vagyunk, ahol $\alpha \in (N \cup \Sigma)^*$ és $x \in \Sigma^*$. Legyen X a $\$\alpha$ verem tetején lévő szimbólum (vagyis $\$\alpha = \alpha' X$) és legyen a az $x\$$ első betűje (vagyis $ax' = x\$$). (Megjegyezzük, hogy $\alpha = \lambda$ esetén $X = \$$, $x = \lambda$ esetén $a = \$$.)

(1) Ha (a precedencia mátrix szerint) $X \prec a$ vagy $X \doteq a$, akkor $C \vdash C'$, ahol $C' = (\$\alpha a, x', \pi)$.

(2) Ha (a precedencia mátrix szerint) $X \succ a$, akkor keressük meg a $\$\alpha$ veremben fölülről lefelé az első olyan helyet, ahol két veremszimbólum között $\prec$ áll fenn, vagyis legyen β a legrövidebb szó, melyre $\$\alpha = \alpha'\beta$ és α' utolsó és β első betűje között $\prec$ áll fenn. (Előfordulhat, hogy $\alpha' = \$$.) Ekkor

- ha β egy $A \to \beta$ szabály (mondjuk a j-edik) jobb oldala, akkor redukáljunk, vagyis $C \vdash C'$, ahol $C' = (\alpha' A, x\$, j\pi)$,
- különben, vagyis ha β egyetlen szabálynak sem jobb oldala, akkor nincs átmenet.

(3) Különben nincs átmenet.

V É G E

Írásbeli vizsga kérdések a
„Szintaktikus elemzési módszerek" c. tárgyhoz

1. A CYK és az Earley elemzési algoritmusok.
2. Az általános felülről lefelé haladó elemzési algoritmus.
3. A FI_k halmazok és az LL(k) nyelvtan definíciója, az LL(k) feltétel átfogalmazása.
4. FI_k kiszámítása és az LL(k) feltétel eldöntése.
5. FO_k definíciója, kiszámítása és az erősen LL(k) nyelvtan definíciója.
6. LL(k) és erősen LL(k) nyelvtanok közötti kapcsolat ($k = 1$ eset is). Faktorizálás.

7. Az erősen LL(k) feltétel eldönthetősége és az erősen LL(k) nyelvek elemzése: elemző tábla kosntukciója és az elemzési algoritmus.
8. Az általános alulról felfelé haladó elemzési algoritmus.
9. Az LR(k) nyelvtan definíciója és az LR(k) elemzéshez szükséges fogalmak bevezetése (nyél, járható prefix, LR(k) elem, érvényesség).
10. LR(k) táblák és azok kiszámítása.
11. Az LR(k) feltétel átfogalmazása (LR(k) tétel), a tevékenység és a goto táblák definíciója. Az LR(k) elemzési algoritmus.
12. LALR(k) nyelvtanok.

13. Egyszerű precedencia nyelvtanok definíciója, eldönthetősége. Egyszerű precedencia elemzési algoritmus.

A példák nem feltétlenül kellenek (viszont jó ha vannak).