Dr. Hunyadvári László
Manhertz Tamás

Automaták és formális nyelvek

Jegyzet programtervező matematikus és informatika tanári szakos hallgatók számára

Technikai kivitelezés:
Tichler Krisztián

Harmadik kiadás, utolsó javítás dátuma: 2006. január 13.

Készült Donald E. Knuth TEX szövegszedő rendszerével.
Budapest, Eötvös Loránd Tudományegyetem, Informatikai Kar, 2006.

Tartalomjegyzék

I Formális nyelvtanok **3**

1 Alapfogalmak **4**
1.1. Ábécé, szavak, nyelv. . . . 4
1.2. Műveletek szavakkal . . . 4
1.2.1. Konkatenáció. . . . 4
1.2.2. Megfordítás. . . . 5
1.2.3. Szó hossza. . . . 5
1.2.4. Homomorfizmus (átkódolás). . . . 5
1.3. Műveletek nyelvekkel . . . 5
1.3.1. Halmazműveletek. . . . 5
1.3.2. Konkatenáció. . . . 6
1.3.3. Helyettesítés (nemdeterminisztikus átkódolás). . . . 6
1.3.4. Lezárás, iteráció. . . . 6
1.4. Nyelvek definiálásának módszerei . . . 6
1.5. Nyelvek megadása produkciós rendszerrel . . . 8
1.5.1. A Π által leírt nyelv . . . 9
1.5.2. Generatív nyelvtanok . . . 10
1.6. Nyelvtanok osztályozása . . . 11
1.7. Nyelvtanok kiterjesztett típusai . . . 13
1.7.1. Kiterjesztett nyelvtani típusok . . . 13
1.8. Normálformák . . . 19
1.8.1. Kuroda normálforma . . . 19
1.8.2. Chomsky normálforma . . . 21
1.8.3. Greibach normálforma . . . 22
1.8.4. 3-as normálforma . . . 26
1.9. Zártsági tételek . . . 27

2 A Chomsky-osztályok és az algoritmussal definiált nyelvek kapcsolata **30**

3 Veremautomatákkal jellemezhető nyelvek **33**
3.1. 0-vermek . . . 35
3.1.1. Determinisztikus 0-vermek ($\mathcal{L}_{D0v}$) . . . 37
3.1.2. A 3. típusú nyelvek néhány tulajdonsága . . . 38
3.1.3. Minimális automata előállítása . . . 41
3.1.4. Állapotok ekvivalenciájának algoritmikus eldöntése . . . 44

3.1.5. Reguláris nyelvek 45
3.1.6. A 3-as típusú nyelvek néhány további zártsági tulajdonsága 47

4 Algoritmikusan eldönthető problémák 3-as típusú nyelveken 49

5 2-es típusú nyelvek 51
5.1. Algoritmikusan eldönthető problémák 2-es típusú nyelvekre 57
5.2. A veremautomaták és 2-es típusú nyelvek kapcsolata 58
5.3. Determinisztikus 1-vermek 63
5.3.1. Veremautomaták normálformái 64

6 Elemzési módszerek 69
6.1. Az elemzési módszerek célja 69
6.2. A szintaxiselemzési módszerek osztályozásai 69
6.3. Felülről-lefelé elemzések 70
6.4. $LL(k)$ nyelvtanok 73
6.5. Alulról-fölfelé módszerek 74
6.6. $LR(k)$ nyelvtanok 75
6.7. Kapcsolat $\mathcal{L}_{\mathrm{DIt1}\mathcal{V}}$ és az $LR(k)$ nyelvek között 76

I. FORMÁLIS NYELVTANOK

1. Alapfogalmak

1.1. Ábécé, szavak, nyelv.

1.1. definíció. *Ábécének nevezünk egy tetszőleges véges szimbólumhalmazt. Az ábécé elemeit betűknek hívjuk.*

Konvenció: az ábécék jelölésére az X, Y jeleket fogjuk használni, míg T az úgynevezett terminális ábécét fogja jelenteni. Egy tetszőleges ábécé betűit az a, b, c szimbólumok jelölik.

1.2. definíció. *Az X ábécé elemeinek egy tetszőleges véges sorozatát az X ábécé feletti szónak nevezzük. Ha X nem lényeges vagy egyértelmű, akkor szóról beszélünk.*

Világos, hogy egy X ábécé feletti szó tekinthető minden X-nél bővebb ábécé feletti szónak is. Konvenció: szavak jelölésére az u, v, w szimbólumokat, illetve ezek megjelölt változatait fogjuk használni $(u_1, v', \overline{w})$.

Azt a legszűkebb ábécét, mely felett u még szó $X(u)$-val jelöljük.

Az X ábécé feletti összes szavak halmazát jelölje X^*. Az ε jelentse az üres szót és legyen $X^+ := X^* \setminus \{\varepsilon\}$.

1.3. definíció. *Az X^* valamely részhalmazát (azaz 2^{X^*} valamely elemét) az X ábécé feletti nyelvnek nevezzük.*

Konvenció: nyelvek jelölésére az L betűt, illetve ennek megjelölt változatait fogjuk használni. Ha X nem lényeges vagy egyértelmű, akkor nyelvről beszélünk. $X(L)$ jelöli a legszűkebb olyan ábécét, mely felett L nyelv. Megegyezés szerint az u szót azonosítjuk az $\{u\}$ nyelvvel.

1.4. definíció. *Nyelvek valamely összességét nyelvosztálynak, nyelvcsaládnak hívjuk.*

Konvenció: nyelvosztályok jelölésére az $\mathcal{L}$ betűt, illetve ennek megjelölt változatait fogjuk használni.

1.2. Műveletek szavakkal

1.2.1. Konkatenáció.

Legyenek u, v szavak egy adott L nyelvből. Ekkor a két szó konkatenációja uv. (A két szó egymás utáni leírásával kapott szó.) Ez egy asszociatív művelet, melynek ε az egységeleme, így

$$\langle X^*, \text{konkatenáció}, \varepsilon \rangle$$

egységelemes félcsoportot képez.

1.2.2. Megfordítás.

Legyen u egy szó, ekkor a megfordítására az u^R, illetve u^{-1} jelöléseket használjuk.

1.2.3. Szó hossza.

Egy adott u szó hossza a benne lévő, ábécébeli jelek száma. Jelölése: $l(u)$. Egy adott $y \in X$ jel előfordulásainak száma egy adott u szóban: $l_y(u)$. Legyen $H \subseteq X$. Jelölje $l_H(u)$ az u szóban előforduló H-beli szimbólumok számát. Például $l_{\{a,b\}}(abcac) = 3$.

1.2.4. Homomorfizmus (átkódolás).

1.5. definíció. *Legyenek X, Y tetszőleges ábécék. Egy $h : X^* \longmapsto Y^*$ leképezést konkatenációtartónak hívunk, ha tetszőleges $u, v \in X^*$ szavak esetén $h(uv) = h(u)h(v)$.*

Világos, hogy egy h konkatenációtartó leképezés esetén $h(\varepsilon) = \varepsilon$ és elég h-t megadni az X elemein. A $h : X^* \longmapsto Y^*$ konkatenációtartó leképezéseket *homomorfizmusoknak* nevezzük.

Nyelvekre vonatkozó kiterjesztése:

$$h(L) := \bigcup_{u \in L} h(u).$$

1.3. Műveletek nyelvekkel

1.3.1. Halmazműveletek.

Mivel a nyelvek halmazok, ezért a halmazműveletek alkalmazhatóak a nyelvekre is. Véges sok, nyelveken végzett halmazelméleti művelet eredményéhez mindig adható ábécé, amely fölötti nyelv az eredmény. Legyenek $L_1, L_2, \ldots, L_k$ nyelvek. Az ezekkel végzett művelet eredményének megfelelő egy ábécé:

$$\bigcup_{i=1}^{k} X(L_i).$$

Végtelen sok művelet esetén ez nem feltétlenül teljesül, mint ezt az alábbi, gyakorlati szempontból is érdekes példa mutatja:

Legyen

$$\Delta_n := \{(_1, (_2, \ldots, (_n,)_1,)_2, \ldots,)_n\} \text{ egy ábécé, és}$$

$$D_n := HE(\Delta_n),$$

azaz a Δ_n feletti helyes zárójelezések halmaza. Ekkor

$$D = \bigcup_{i=1}^{\infty} D_i,$$

a helyes zárójelezések összessége már nem lesz nyelv, mert nincsen véges ábécéje.

Végtelen sok művelet esetén tehát ügyelni kell arra, hogy az eredményül kapott halmaz nyelv legyen, vagyis létezzen hozzá ábécé.

1.3.2. Konkatenáció.

Legyenek L_1, L_2 nyelvek. Ekkor az L_1 és L_2 nyelvek konkatenációján az $L_1L_2 := \{uv \mid u \in L_1 \wedge v \in L_2\}$ nyelvet értjük.

A konkatenáció rendelkezik az asszociativitás tulajdonságával. Erre vonatkozó egységeleme $\{\varepsilon\}$. Az unió és a konkatenáció között mindemellett kétoldalú disztributivitás áll fenn, azaz

$L(L_1 \cup L_2) = LL_1 \cup LL_2$, valamint

$(L_1 \cup L_2)L = L_1L \cup L_2L$.

Ezekkel a tulajdonságokkal a

$$\left\langle 2^{X^*}, \{\cup, \text{konkatenáció}, \emptyset, \{\varepsilon\}\}\right\rangle$$

struktúra egységelemes félgyűrű.

1.3.3. Helyettesítés (nemdeterminisztikus átkódolás).

Legyenek X, Y ábécék, és $\Phi : 2^{X^*} \longmapsto 2^{Y^*}$. A Φ leképezést helyettesítésnek nevezzük, ha unió- és konkatenációtartó és megtartja a zérus elemet, valamint az egységelemet ($\Phi(\emptyset) = \emptyset, \Phi(\{\varepsilon\}) = \{\varepsilon\}$). ($\Phi$ tehát a 2^{X^*} és a 2^{Y^*} félgyűrűk közötti algebrai homomorfizmus.)

Megjegyzés: A művelettartás miatt Φ-t elegendő X elemein megadni. Tetszőleges, nyelvekre kiterjesztett homomorfizmus tekinthető helyettesítésnek.

1.3.4. Lezárás, iteráció.

Legyen L egy nyelv. Ekkor L iteráltján vagy lezártján a következő nyelvet értjük:

$$L^* := \bigcup_{i=0}^{\infty} L^i,$$

ahol $L^i := \underset{1}{L}\underset{2}{L}\ldots\underset{i}{L}$ (pontosan i darab L nyelv konkatencációja), és $L^0 := \{\varepsilon\}$. Példa:

$$\{ab, ba\}^* = \{\varepsilon, ab, ba, abab, abba, baab, baba, ababab, \ldots\}.$$

Pozitív lezárás (ε nélkül):

$$L^+ := \bigcup_{i=1}^{\infty} L^i.$$

1.4. Nyelvek definiálásának módszerei

Nyelvek definiálására többféle eszköz létezik, de mindegyik esetében követelmény, hogy a leírás véges legyen. A következőkben néhány alapvető módszert vázolunk fel:

(1) Véges nyelvek esetén felsorolhatjuk a nyelv elemeit.

(2) Logikai formulával.

Például legyen $X := \{a, b\}$ és $L := \{u \mid u = a^n b^n \wedge n \in \mathcal{N}\}$.

(3) Struktúrális rekurzióval. A módszer lényege, hogy adott egy — legfeljebb megszámlálhatóan végtelen — leírással már rendelkező nyelvekből álló nyelvkészlet (az elemi nyelvek) és adott egy véges műveletkészlet. Úgy adjuk meg a nyelvet, hogy megmondjuk, mely elemi nyelvekből és milyen megengedett műveletekkel áll elő (véges sok nyelvből véges sok művelettel).

Legyen U egy megszámlálhatóan végtelen szimbólumhalmaz. A továbbiakban feltesszük, hogy minden ábécé ennek részhalmaza (minden nyelv átkódolható ilyenné betűk átkódolásával).

Példa struktúrális rekurzióra:

1.6. definíció. *Reguláris nyelveknek nevezzük azokat a L nyelveket, melyek megfelelnek az alábbi két feltételnek:*

(a) L az $\{a\}, \{\varepsilon\}, \emptyset$ nyelvek valamelyike (ezek az elemi nyelvek), ahol $a \in U$,

(b) L az elemi nyelvekből az unió, konkatenáció és lezárás műveletek véges számú alkalmazásával áll elő.

Az $X := \{a, b\} \subseteq U$ ábécé mellett például $\{\{a\}\{b\} \cup \{b\}\{a\}\}^*$ reguláris nyelv, melyet az előbb látott, úgynevezett reguláris kifejezéssel írtunk le.

(4) Algoritmus segítségével.

1.7. definíció. *Az L nyelv rekurzívan felsorolható $\iff$ ha létezik A algoritmus, mely az elemeit felsorolja.*

Rekurzív felsorolás: Az A algoritmus outputjára szavakat állít elő, s így a nyelv összes szavát (és csak azokat) felsorolja.

1.8. definíció. *Az L nyelv parciálisan rekurzív $\iff$ létezik olyan A parciálisan eldöntő algoritmus, melynek inputjára tetszőleges szót helyezve eldönti, benne van-e a nyelvben ($u \in L$ szó esetén igen válasszal áll le, míg $u \notin L$ esetén nem terminál, vagy ha terminál, akkor nem választ ad).*

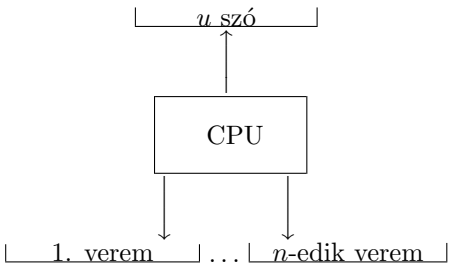
1.9. definíció. *Az L nyelv rekurzív $\iff$ létezik olyan A eldöntő algoritmus, melynek inputjára egy tetszőleges u szót helyezve eldönti, benne van-e az L nyelvben (igen a válasz, ha eleme a nyelvnek, és nem a válasz ellenkező esetben).*

Megjegyzés: Az algoritmus fogalom pontosításával matematikai szigorúsággal bizonyítható, hogy a rekurzívan felsorolható nyelvek összessége megegyezik a parciálisan rekurzív nyelvekkel, míg a rekurzív nyelvek az előbbiek valódi részét képezik.

(5) Matematikai gépekkel:

Az absztrakt matematikai gépek felépítése a következő: Adott egy potenciálisan végtelen input szalag, amelyre fel van írva az elolvasásra megadott szó. A gép ezt az input szalagot egy olvasófejjel olvassa végig. Adott továbbá n darab listaszerkezetként (speciálisan veremként, sorként) használt munkaszalag, melyek mindegyikéhez tartozik egy író/olvasófej. A fejeket az úgynevezett központi egység vezérli, ami véges sok állapottal rendelkezik. Ez az állapot határozza meg a gép további működését.

Példa matematikai gépre az n-verem.



Matemaikai gépek működése:

Adott egy diszkrét időskála, melynek minden ütemére a gép végrehajt egy lépést. Egy ilyen lépés két alapvető részből áll:

(a) Felderítés: Milyen jelet olvastunk, a CPU milyen állapotban van, és mi van a munkaszalagok olvasófeje alatt.

(b) Tényleges működés: A CPU új állapotba kerül(het), a munkaszalagokra új jeleket ír(hat)unk, az olvasófejet, illetve a munkaszalagok iró- és olvasófejeit mozgatjuk.

Ha létezik olyan működéssorozat, amelyre az inputot elolvassa a gép, és a CPU valamely kitüntetettállapotok (úgynevezett végállapotok) egyikébe kerül, akkor a szó a gép által felismert nyelvbe tartozik, különben nem.

(6) Produkciós rendszerrel.

Itt axiómák és következtetési szabályok segítségével adjuk meg a nyelvet. A produkciós rendszer olyan szavakból álló nyelvet ír le, amelyek az axiómákból levezethetők, illetve az axiómákra visszavezethetők. Speciális produkciós rendszer az úgynevezett generatív grammatika (formális nyelvtan). A produkciós rendszerekkel és a generatív grammatikákkal a következőkben részletesebbel foglalkozunk.

1.5. Nyelvek megadása produkciós rendszerrel

1.10. definíció. *A* Π *produkciós rendszer alatt a következő hármast értjük:*

$$\Pi = \langle X, P, A_x \rangle,$$

ahol X az ábécé, $P \subseteq X^* \times X^*$ a produkciós szabályok véges halmaza, és $A_x \subseteq X^*$ véges axiómahalmaz.

Megállapodás szerint $(p, q) \in P$ esetén az írásmód: $p \longrightarrow q$.

1.11. definíció. *Közvetlen levezetés Π-ben: Az $a \in X^*$ szóból közvetlenül levezethető a $b \in X^*$ szó a Π produkciós rendszerben $\iff$ létezik $g_1, g_2 \in X^*$ és $p \longrightarrow q \in P$ szabály, hogy $a = g_1 p g_2$, és $b = g_1 q g_2$.*

Jelölésben: $a \underset{\Pi}{\longrightarrow} b$.

1.12. definíció. *Közvetett levezetés Π-ben: Az $a \in X^*$ szóból közvetett módon levezethető a $b \in X^*$ szó a Π produkciós rendszerben $\iff$ van olyan $k \in \mathcal{N}$ és vannak olyan $\delta_0, \delta_1, \ldots, \delta_k \in X^*$ szavak, hogy $a = \delta_0, b = \delta_k$, és minden $i \in \left[0, k-1\right]$ esetén $\delta_i \underset{\Pi}{\longrightarrow} \delta_{i+1}$.*

Jelölésben: $a \overset{*}{\underset{\Pi}{\longrightarrow}} b$.

1.13. definíció. *Az előbbi definícióban szereplő k számot a levezetés hosszának nevezzük.*

Megjegyzés: Amennyiben $k = 0$, akkor $a = \delta_0 = b$, azaz a definíció közvetlen következménye, hogy minden szó levezethető saját magából, tehát a levezetés reflexív. Jelöljük a pontosan k hosszú levezetéseket a következőképpen: $a \overset{k}{\underset{\Pi}{\longrightarrow}} b$.

Pozitív levezetésről beszélünk, ha $k \geq 1$. Jelölésben: $a \overset{+}{\underset{\Pi}{\longrightarrow}} b$.

1.5.1. A Π által leírt nyelv

1.14. definíció. *A Π produkciós rendszer által (a T ábécére relatív) generált nyelv:*

$$L_T^g(\Pi) = \{u \in T^* \mid \exists a \in A_x : a \overset{*}{\underset{\Pi}{\longrightarrow}} u\}.$$

1.15. definíció. *A Π produkciós rendszer által (a T ábécére relatív) elfogadott nyelv (akceptált nyelv):*

$$L_T^a(\Pi) = \{u \in T^* \mid \exists a \in A_x : u \overset{*}{\underset{\Pi}{\longrightarrow}} a\}.$$

Néhány példa:

Legyen $\Pi_1 := \langle \{(,)\}, \{() \longrightarrow \varepsilon\}, \{\varepsilon\} \rangle$, ekkor a Π_1 által elfogadott nyelv a helyes zárójelezések halmaza, azaz $L^a_{\{(,)\}}(\Pi_1) = HE$.

Legyen $\Pi_2 := \langle \{\mathrm{S}, (,)\}, \mathcal{P}, \{\mathrm{S}\} \rangle$, ahol $\mathcal{P} := \{\mathrm{S} \longrightarrow (\mathrm{S}), \mathrm{S} \longrightarrow \mathrm{SS}, \mathrm{S} \longrightarrow \varepsilon\}$. A $\{(,)\}$-re relatív generált nyelv itt is a helyes zárójelezések halmaza, tehát $L^g_{\{(,)\}}(\Pi_1) = HE$.

Megjegyzés: Az utóbbi példában szereplő $\mathcal{P}$ halmazt — a rövidebb írásmód kedvéért — a következő módon szokták megadni: $\mathcal{P} := \{\mathrm{S} \longrightarrow (\mathrm{S}) \mid \mathrm{SS} \mid \varepsilon\}$.

1.5.2. Generatív nyelvtanok

A generatív nyelvtanok speciális produkciós rendszerek az alábbi négy feltétellel:

(a) $X = T \cup N$ és $T \cap N = \emptyset$, ahol T a terminális jelek, N a nyelvtani jelek halmaza.

(b) $p \longrightarrow q \in P$ esetén p-ben van legalább egy nyelvtani jel.

(c) $A_x = \{S\}$, ahol $S \in N$ a kezdőszimbólum.

(d) Csak a T-re relatív generálás megengedett.

Ezeket a feltételeket figyelembe véve az alábbi közvetlen definiciót adhatjuk a formális nyelvtan fogalmára.

1.16. definíció. *Generatív nyelvtannak (formális nyelvtannak) nevezzük az alábbi négyest:*

$$G = \langle T, N, \mathcal{P}, S \rangle,$$

ahol T a terminális jelek ábécéje, N a nyelvtani jelek ábécéje, $\mathcal{P}$ véges szabályhalmaz, melynek bármely szabálya bal oldalán van legalább egy nyelvtani jel, és $S \in N$ a kezdőszimbólum.

A levezetés fogalom a formális nyelvtanok esetében megegyezik a produkciós rendszereknél definiált levezetés fogalommal. Mivel most kitüntetett szerepük van a terminális jeleknek, ezért külön elnevezést használunk azokra a szavakre, melyek vegyesen tartalmazhatnak terminális jeleket és nyelvtani jeleket, és külön a csak terminális jelekből állókra. Az előbbieket *mondatformának* hívjuk és konvenció szerint a görög ábécé első betűinek kisbetűs formáival (α, β, γ) jelöljük, míg az utóbbiakat *terminális szavaknak* és a latin ábécé végéről vett kis betűkkel (u, v, w, z) jelöljük.

Az S kezdőszimbólumból G-ben levezethető mondatformákat, terminális szavakat *G-ben elérhető* mondatformáknak, terminális szavaknak nevezzük (Ha G rögzített vagy nem lényeges, akkor egyszerűen csak elérhetőségről beszélünk).

1.17. definíció. *A G nyelvtan által generált $L(G)$ nyelv a G-ben elérhető terminális szavak összessége, azaz:*

$$L(G) = \{u \in T^* \mid S \xrightarrow[G]{*} u\}.$$

Példák

1-2. Legyen $G_1 = \langle \{a, b\}, \{\mathrm{S}\}, \{\mathrm{S} \longrightarrow a\mathrm{S}b \mid b\mathrm{S}a \mid ab \mid ba \mid \mathrm{SS}\}, \mathrm{S} \rangle$. Bizonyítsuk be, hogy ekkor a generált nyelv:

$$L(G_1) = \{u \in \{a, b\}^* \mid l_a(u) = l_b(u) > 0\}.$$

Bizonyítás.
„$\supseteq$”:

Először legyen $u \in \{x \in \{a, b\}^* \mid l_a(x) = l_b(x) > 0\}$. Bizonyítani kell, hogy ekkor $u \in L(G_1)$. Meg kell mutatni, hogy létezik olyan S-ből induló levezetés G_1-ben, mely ezt az u szót adja. Ezt az $n := l_a(u) = l_b(u)$-ra vonatkozó teljes indukcióval igazoljuk:

$n = 1$: Ekkor $u = ab$ vagy $u = ba$, mely S kezdőszimbólumból az $S \longrightarrow ab$ vagy az $S \longrightarrow ba$ szabályok alkalmazásával megkapható.

$n + 1$: Most $1, 2, \ldots, n$-re elfogadjuk és bizonyítjuk $n + 1$-re. Könnyen látszik, hogy u felépítése alapján három eset lehetséges:

$u = u_1u_2$ $(u_1, u_2 \in L)$ esetén indukciós feltevésünk alapján $S \xrightarrow[G_1]{*} u_1$ és $S \xrightarrow[G_1]{*} u_2$, ezért

$$S \longrightarrow SS \xrightarrow[G_1]{*} u_1S \xrightarrow[G_1]{*} u_1u_2 = u,$$

$u = awb$ esetén $S \longrightarrow aSb \xrightarrow[G_1]{*} awb = u$, illetve

$u = bwa$ esetén $S \longrightarrow bSa \xrightarrow[G_1]{*} bwa = u$.

„$\subseteq$”:

Most legyen $u \in L(G_1)$ és meg kell mutatni, hogy $u \in \{x \in \{a, b\}^* \mid l_a(x) = l_b(x) > 0\}$. Ehhez nyilván elég belátni azt, hogy a szabályok tetszőleges alkalmazásával csak olyan u szavak vezethetők le, melyekben az a és b szimbólumok száma megegyezik. Ez viszont teljesül, mert $\mathcal{P}$-ben csak olyan szabályok vannak, melyek jobboldala azonos számú a és b jelet tartalmaz, és a legrövidebb levezethető szó is legalább 2 hosszú.

□

1.6. Nyelvtanok osztályozása

A generatív nyelvtanok osztályozását a szabályok alakja alapján tehetjük meg. Egy megkötést már ismerünk: a szabályok bal oldalán mindenképpen szerepelnie kell nyelvtani jelnek.

A szabályok alakja szerint a következő osztályozás lehetséges:

0. típus: Nincs további megkötés, azaz minden nyelvtan egyben 0-ás típusú.

1. típus: Két szabályforma megengedett:

   (a) $a_1Aa_2 \longrightarrow a_1qa_2$, ahol $a_1, a_2 \in (T \cup N)^*$ a környezet, $A \in N$, és $q \in (T \cup N)^+$ (azaz $q \neq \varepsilon$). Ez a környezetfüggő szabályforma.

   (b) $S \longrightarrow \varepsilon$, ahol S a kezdő nyelvtani jel, s ha van ilyen szabály, akkor S nem fordulhat elő szabály jobboldalán.

2. típus: Itt is két szabályforma lehetséges:

   (a) $A \longrightarrow q$, ahol $A \in N$, és $q \in (T \cup N)^+$. Ez a környezetfüggetlen szabályforma.

   (b) hasonlóan, mint az 1. típus b) pontja.

   Megjegyzés: Ez a típus az 1. típus további megszorítása: a_1, a_2 mindig ε-nal egyenlő.

3. típus: Itt három szabályforma megengedett:

   (a) $A \longrightarrow aB$, ahol $A, B \in N$, és $a \in T$.

   (b) $A \longrightarrow a$, ahol $A \in N$ és $a \in T$.

(c) hasonlóan, mint az 1. típus b) pontja.

Megjegyzés: Ez a 2. típus további megszorítása, q csak speciális alakú lehet.

Világos, hogy az $1., 2., 3.$ típusú nyelvtanok esetén az $S \longrightarrow \varepsilon$ szabályalak csak az egyéb szabályokkal nem származtatható ε levezetésére használható.

A különböző típusú nyelvtanok összességét nyelvtani osztályoknak hívjuk és rendre $\mathcal{G}_i$-vel jelölünk. Tehát:

$$\mathcal{G}_i := \{i.\text{ típusú nyelvtanok}\}.$$

A definíciók alapján világos, hogy $\mathcal{G}_3 \subseteq \mathcal{G}_2 \subseteq \mathcal{G}_1 \subseteq \mathcal{G}_0$.

A nyelvtani osztályozást felhasználva a nyelveket is osztályozhatjuk:

$$\mathcal{L}_i := \{L \mid L \text{ nyelv, és van olyan } G \in \mathcal{G}_i, \text{ amelyre } L(G) = L\}.$$

Az előző nyelvtanhierarchia alapján $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$. Ez az ún. **Chomsky-féle hierarchia**.

Igaz a hierarchia erősebb változata, ahol mindenütt a szigorú tartalmazás jele áll. Erre később még kitérünk.

Legyen T véges, nemüres ábécé. Jelölje $\mathcal{L}_0^T$ azokat a T feletti nyelveket, melyek nyelvtannal leírhatók. Kérdés: Vajon minden nyelvhez megadható-e olyan nyelvtan, ami azt a nyelvet generálja? Erre ad választ a következő tétel.

1.18. tétel. *Nem minden nyelv írható le nyelvtannal.*

Bizonyítás.

Elég ehhez belátnunk, hogy tetszőleges nemüres T ábécé esetén $\mathcal{L}_0^T \subset 2^{T^*}$. Mivel T^* megszámlálható számosságú, ezért hatványhalmaza, 2^{T^*} biztosan megszámlálhatónál nagyobb számosságú. Emiatt elég belátni, hogy $\mathcal{L}_0^T$ legfeljebb megszámlálható számosságú.

Legyen $\hat{N}$ a potenciális nyelvtani jelek megszámlálható számosságú halmaza, melyre $\hat{N} \cap T = \emptyset$. Legyen $\mathcal{N}y := \{G \mid G \text{ nyelvtani jelei } \hat{N}\text{-ből, terminális jelei pedig } T\text{-ből valók}\}$.

Állítás: Minden $L \in \mathcal{L}_0^T$ esetén létezik olyan $G \in \mathcal{N}y$, amelyre $L(G) = L$.

Ez az állítás könnyen belátható, hiszen minden ilyen L-hez létezik egy G' nyelvtan, melyre $L(G') = L$. Ha a G' nyelvtan minden nyelvtani jele $\hat{N}$-ből van, akkor készen vagyunk. Ha nem, akkor G' nyelvtani jeleit szisztematikusan átcseréljük $\hat{N}$-beliekre. ($\hat{N}$ megszámlálható volta miatt rendelkezésünkre áll megfelelő mennyiségű nyelvtani jel). Világos, hogy az így kapott nyelvtan $\mathcal{N}y$-ben van és L-et generálja.

Legyen $\Phi : \mathcal{N}y \longmapsto \mathcal{L}_0^T$, melyre $\Phi(G) := L(G)$. Ez a Φ az előbbi állítás miatt ráképezés, tehát a képtere a teljes $\mathcal{L}_0^T$. Emiatt nyilván fennáll, hogy $|\mathcal{N}y| \geq |\mathcal{L}_0^T|$. Ennek alapján elegendő belátni, hogy $\mathcal{N}y$ megszámlálható számosságú.

Egy $G \in \mathcal{N}y$ nyelvtant a következőképp írhatunk fel:

$$G = \big\langle \{t_1, \ldots, t_k\}, \{A_1, \ldots, A_n\}, \{x_1 \ldots x_l \longrightarrow y_1 \ldots y_s, \ldots\}, S \big\rangle,$$

ezért G-t tekinthetjük az $X := \hat{N} \cup T \cup \big\{\langle, \rangle, \longrightarrow, ', \{', \}'\big\} \cup \big\{,\big\}$ megszámlálható halmazból vett jelek véges sorozatának. Tehát $\mathcal{N}y \subseteq X^*$, amiből már következik, hogy $|\mathcal{N}y| \leq |X^*|$. X^*-ról viszont tudjuk, hogy maga is megszámlálható számosságú. □

Bár az előbb bizonyított tétel és annak bizonyítása alapján a nyelvek többsége nem írható le nyelvtan segítségével, a gyakorlatban használt, konstruktív nyelvdefiníciós eszközök mindegyikéről kiderült, hogy az általuk leírt nyelvek nyelvtannal is leírhatók. Ennek alapján általánosan elfogadott, hogy a jövőben sem fogunk olyan, általunk konstruktívnak tekintett nyelvdefiníciókat találni, melyek a nyelvtanoknál többet tudnak. Ezt mondja ki a **Church-tézis**.

Church-tézis: Minden valamilyen konstruktív módon megadható nyelv leírható nyelvtannal.

1.7. Nyelvtanok kiterjesztett típusai

Vizsgáljuk meg, mit mondhatunk akkor, ha a megengedett szabályok alakját kicsit általánosabban adjuk meg.

1.7.1. Kiterjesztett nyelvtani típusok

1.19. definíció. *A G nyelvtan kiterjesztett 1-es típusú nyelvtan, ha szabályai a következő alakúak lehetnek.*

i)$p \longrightarrow q \in \mathcal{P}$ esetén $l(p) \leq l(q)$, azaz a szabály nem csökkenti a hosszt.

ii)$S \longrightarrow \varepsilon$ alakú, ahol S a kezdőszimbólum és ha van ilyen szabály, akkor S nem fordul elő szabály jobboldalán.

Jelölés: $\mathcal{G}_{\text{kit1}}$.

Jelölés: Ha $G = \langle T, N, \mathcal{P}, S \rangle \in \mathcal{G}_1$ és $u \in T^*$, akkor $K(G,u)$ legyen a legfeljebb $l(u)$ hosszúságú G-beli mondatformák száma. $K(G,u) := \left|(T \cup N)^{\leq l(u)}\right|$.

1.20. állítás. *Egy $G \in \mathcal{G}_{\text{kit1}}$ nyelvtan bármely $u \in L(G)$ szavának van legfeljebb $K(G,u)$ hosszú levezetése is G-ben.*

Bizonyítás. Két esetet különböztetünk meg.
$u = \varepsilon$: Ekkor $K(G,\varepsilon) = 1$ (hiszen csak egyetlen nulla hosszú szó létezik), és ez illeszkedik az ε egyetlen $S \longrightarrow \varepsilon$ levezetésének 1 hosszához.

$u \neq \varepsilon$: Ha $u \in L(G)$, akkor létezik $S = \alpha_0 \underset{G}{\longrightarrow} \alpha_1 \underset{G}{\longrightarrow} \ldots \underset{G}{\longrightarrow} \alpha_s = u$ levezetés. Ekkor természetesen van olyan $S = \beta_0 \underset{G}{\longrightarrow} \ldots \underset{G}{\longrightarrow} \beta_{s'} = u$ levezetése is, ahol mindegyik β_i különböző. (Ugyanis, ha $i < j$ esetén $\alpha_i = \alpha_j$, akkor az i és j közötti lépéseket egyszerűen elhagyjuk.)

A hosszúság nem csökkentése miatt $l(\beta_0) \leq l(\beta_1) \leq \ldots \leq l(\beta_{s'}) = l(u)$. Ebből pedig már következik, hogy $\beta_0, \beta_1, \ldots, \beta_{s'} \in \left(T \cup N\right)^{\leq l(u)}$, és mivel ezek mind különbözők, ezért biztosan igaz, hogy $s' \leq \left|\left(T \cup N\right)^{\leq l(u)}\right| = K(G,u)$.

□

1.21. definíció. *A G nyelvtan kiterjesztett 2-es típusú, ha a szabályok alakja $A \longrightarrow q \in \mathcal{P}$, ahol $A \in N, q \in (T \cup N)^*$.*

Jelölés: $\mathcal{G}_{\text{kit2}}$.

A kiterjesztett 2-es típus a környezetfüggetlenség legtágabb megfogalmazása. Egy kiterjesztett 2-es típusú nyelvtanban a mondatforma tetszőleges részét a további részeitől függetlenül alakíthatjuk tovább a végeredmény eléréséig. Ezt fejezi ki az alábbi, kiterjesztett környezetfüggetlen nyelvtanokra érvényes *függetlenségi lemma.*

1.22. lemma. *Legyen $G = \langle T, N, \mathcal{P}, S \rangle$ kiterjesztett 2-es típusú nyelvtan és $\alpha \xrightarrow[G]{*} \beta$ valamely levezetés G-ben. Ekkor α egy tetszőleges $\alpha = \alpha_1 \dots \alpha_k$ felbontása esetén a β szónak létezik olyan $\beta = \beta_1 \dots \beta_k$ felbontása, melyre $\alpha_1 \xrightarrow[G]{*} \beta_1, \dots, \alpha_k \xrightarrow[G]{*} \beta_k$, úgy, hogy ezeknek a levezetéseknek az összhossza megegyezik $\alpha \xrightarrow[G]{*} \beta$ hosszával.*

Bizonyítás.

Az állítást a levezetés hossza szerinti indukcióval végezzük. $h = 0$ hosszúság esetén a reflexivitás alapján $\alpha = \beta$ és β számára a $\beta = \alpha_1 \dots \alpha_k$ felbontás nyilván megfelelő.

Tegyük fel, hogy a legfeljebb h hoszúságú levezetésekre ($h \geq 0$) az állítás már igaz, s bizonyítsuk $h+1$ hosszra. Legyen tehát $\alpha \xrightarrow[G]{*} \beta$ egy $h+1$ hosszúságú levezetés és $\alpha = \alpha_1 \dots \alpha_k$ tetszőleges felbontás. Írjuk ki ennek utolsó lépését részletesebben: $\alpha \xrightarrow[G]{h} \beta' \xrightarrow[G]{1} \beta$. Indukciós feltevésünk alapján létezik a $\beta' = \beta'_1 \dots \beta'_k$ felbontás, melyre $\alpha_1 \xrightarrow[G]{*} \beta'_1, \dots, \alpha_k \xrightarrow[G]{*} \beta'_k$, és ezen levezetések összhossza h. A $\beta' = \beta'_1 \dots \beta'_k \xrightarrow[G]{1} \beta$ lépésben alkalmazott szabály legyen $A \longrightarrow \gamma$, ahol A egyetlen nyelvtani jel (nyelvtanunk kiterjesztett 2-es típusú). Kell legyen tehát olyan j index, hogy az alkalmazott szabály baloldala (mely maga ez az A nyelvtani jel) β'_j-nek eleme. Legyen $\beta'_j = \gamma_1 A \gamma_2$. A $\beta' = \beta'_1 \dots \beta'_k \xrightarrow[G]{1} \beta$ helyettesítési lépés ezt a részt $\gamma_1\gamma\gamma_2$-vé alakítja át. A $\beta_i := \beta'_i$ $i \neq j$ és $\beta_j := \gamma_1\gamma\gamma_2$ választás mellett nyilván $\beta = \beta_1 \dots \beta_k$. Az $\alpha_i \xrightarrow[G]{*} \beta_i$ levezetések $i \neq j$ esetén az indukciós feltevés közvetlen következményei. A j indexre a $\beta'_j = \gamma_1 A \gamma_2 \xrightarrow[G]{1} \gamma_1\gamma\gamma_2 = \beta_j$, ahonnan felhasználva az indukciós feltevést is megkapjuk az $\alpha_j \xrightarrow[G]{*} \beta'_j \xrightarrow[G]{1} \beta_j$ levezetést, s az is világos, hogy a kapott levezetések összhossza $h + 1$. □

1.23. definíció. *A G nyelvtan kiterjesztett 3-as típusú, ha a szabályok alakja $A \longrightarrow uB$ vagy $A \longrightarrow u$, ahol $A, B \in N$, és $u \in T^*$.*

Jelölés: $\mathcal{G}_{\text{kit3}}$.

Ezekkel a jelölésekkel nyilván $\mathcal{G}_i \subseteq \mathcal{G}_{\text{kit}i}$, hiszen látszik, hogy csak gyengítettünk a szabályok alakjának megkötésein. Ha hasonló jelöléseket a nyelveken is bevezetünk, akkor az előbbiek alapján $\mathcal{L}_i \subseteq \mathcal{L}_{\text{kit}i}$ is igaz lesz.

1.24. tétel. Kiterjesztési tétel

$$\mathcal{L}_i = \mathcal{L}_{\text{kit}i}, \quad \textit{ahol } i = 1, 2, 3$$

Bizonyítás. Csak $\mathcal{L}_{\text{kit}i} \subseteq \mathcal{L}_i$-t kell belátni, mert a másik irány nyilvánvaló. Ehhez viszont elég a következő lemmát belátni:

1.25. lemma. *Tetszőleges $G \in \mathcal{G}_{\text{kit}i}$ nyelvtanhoz található olyan $G' \in \mathcal{G}_i$, melyre $L(G') = L(G)$.*

Ezt mindhárom kiterjesztésre be kell látni.
$i = 1$:

Feltehető, hogy G-ben terminális jel csak $A \longrightarrow a \quad (a \in T, A \in N)$ alakú szabályokban fordul elő. (Ha ez nem így lenne, akkor G-t a következőképpen alakítjuk át: minden $t \in T$-hez bevezetjük az x_t-vel jelölt álterminálisokat, melyek új nyelvtani jelek lesznek. Ezek különbözzenek mind egymástól, mind pedig N jeleitől. $\mathcal{P}$-ben minden szabályban a terminális jeleket kicseréljük a megfelelő álterminális jelekre. Végül felvesszük az $x_t \longrightarrow t$ új szabályokat. Az így átalakított nyelvtan nyilván ugyanazt tudja, mint az eredeti, és terminális jel csak az $x_t \longrightarrow t$ alakú szabályokban van, ami pontosan a kívánt alakú.)

Cél olyan G' nyelvtan konstrukciója, mely 1-es típusú, és $L(G') = L(G)$. Ha történetesen G minden szabálya 1-es típusú, akkor készen vagyunk ($G' = G$). Ha nem, akkor vegyük ki G olyan szabályait, melyek nem 1-es típusúak („rossz" szabályok). Milyen ezeknek az alakja?

Rossz szabály a következő formájú lehet: $X_1X_2 \dots X_n \longrightarrow Y_1Y_2 \dots Y_m \quad (m \geq n)$. Ennek hatását csupa 1-es típusú szabállyal szimuláljuk. Ehhez bevezetünk új nyelvtani jeleket, $Z_1, Z_2, \dots, Z_n$-et. A szabályok:

$$
\begin{aligned}
&X_1X_2 \dots X_n \longrightarrow Z_1X_2 \dots X_n,\\
&Z_1X_2 \dots X_n \longrightarrow Z_1Z_2X_3 \dots X_n,\\
&\qquad\vdots\\
&Z_1Z_2 \dots Z_{n-1}X_n \longrightarrow Z_1Z_2 \dots Z_nY_{n+1} \dots Y_m \quad (n \leq m),\\
&Z_1Z_2 \dots Z_nY_{n+1} \dots Y_m \longrightarrow Y_1Z_2 \dots Z_nY_{n+1} \dots Y_m,\\
&\qquad\vdots\\
&Y_1 \dots Y_{n-1}Z_nY_{n+1} \dots Y_m \longrightarrow Y_1Y_2 \dots Y_m.
\end{aligned}
$$

$Z_1, \dots, Z_n$ új volta miatt a szabályokat csak ebben a sorrendben lehet és kell végrehajtani, ezért az új nyelvtan is ugyanazt a nyelvet generálja. Csináljuk meg ezt a szabálytranszformációt az összes „rossz" szabályra. Az így kapott G' nyelvtan már 1-es típusú és $L(G)$-t generálja.

$i = 2$:

Itt a problémát az jelenti, hogy a kiterjesztett 2-es típusú nyelvtanok esetén megengedett, hogy üres szó álljon a szabályok jobb oldalán. Legyen G ilyen kiterjesztett 2-es típusú nyelvtan. A gond tehát az $A \longrightarrow \varepsilon$ alakú, úgynevezett **ε-szabályokkal** van. Hogyan szimulálhatjuk hatásukat 2-es típusú szabályokkal?

Példa: Legyen $G = \langle T, N, \mathcal{P}, \text{S} \rangle$, ahol $T = \{a, b, c\}$, $N = \{\text{A}, \text{B}, \text{C}, \text{S}\}$, $\mathcal{P}$ pedig a következő szabályokból áll:

$$
\begin{aligned}
&\text{S} \longrightarrow \text{AB}c|\text{AA},\\
&\text{B} \longrightarrow \text{CC},\\
&\text{A} \longrightarrow \varepsilon|a,
\end{aligned}
$$

$\mathrm{C} \longrightarrow \varepsilon | b.$

Vegyük itt a következő, ε-szabályokat tartamazó levezetést:

$$\mathrm{S} \underset{G}{\longrightarrow} \mathrm{AB}c \underset{G}{\longrightarrow} \mathrm{B}c \underset{G}{\longrightarrow} \mathrm{CC}c \underset{G}{\longrightarrow} b\mathrm{C}c \underset{G}{\longrightarrow} bc.$$

Az ε-szabállyal elhagyott nyelvtani jeleket a mondatformába már ne is hozzuk be, tehát ilyen levezetést várunk:

$$\mathrm{S} \underset{G'}{\longrightarrow} \mathrm{B}c \underset{G'}{\longrightarrow} \mathrm{C}c \underset{G'}{\longrightarrow} bc.$$

Azt az algoritmust, mely megoldja a szabályok előbb kitűzött irányú módosítását, ε-mentesítési eljárásnak hívjuk.

Legyen $H := \{A \in N \mid A \underset{G}{\overset{*}{\longrightarrow}} \varepsilon\}$. Mivel G' konstrukciójában használni fogjuk a H halmazt, fontos kérdés, hogyan lehet H-t megtalálni? Fokozatos közelítéssel állítjuk elő:

$$H_1 := \{A \in N \mid A \longrightarrow \varepsilon \in \mathcal{P}\},$$

$$H_{i+1} := H_i \cup \{A \in N \mid \exists A \longrightarrow Q \in \mathcal{P} : Q \in H_i^*\}.$$

Ekkor nyilván teljesül a következő összefüggés: $H_1 \subseteq H_2 \subseteq \ldots \subseteq H_i \subseteq \ldots$. Mivel minden i-re $H_i \subseteq N$, és az N halmaz véges, ezért egy i_0 indextől kezdődően biztosan azonosak lesznek ezek a halmazok. Legyen i_0 a legkisebb ilyen index.

Ekkor $H_1 \subset H_2 \subset \ldots \subset H_{i_0} = H_{i_0+1}$. Belátjuk, hogy $H = H_{i_0}$. Ehhez elég bizonyítani, hogy $H_j = H_{j+1}$ esetén $H_{j+1} = H_{j+2}$, hiszen ebből következik, hogy H_{i_0}-tól kezdve a halmazok ugyanazok lesznek. Tegyük fel, hogy $H_j = H_{j+1}$. Ekkor nyilván $H_j^* = H_{j+1}^*$, és a definíció alapján:

$$H_{j+2} = H_{j+1} \cup \{A \in N \mid \exists A \longrightarrow Q \in \mathcal{P} : Q \in H_{j+1}^*\} =$$

$$= H_j \cup \{A \in N \mid \exists A \longrightarrow Q \in \mathcal{P} : Q \in H_j^*\} = H_{j+1}.$$

Tehát a H halmaz megkonstruálható.

Következmény: $S \in H \Longleftrightarrow \varepsilon \in L(G)$.

A G' nyelvtan konstrukciója legyen a következő: $G' := \langle T, N, \bar{\mathcal{P}}, S \rangle$, ahol $A \longrightarrow \bar{q} \in \bar{\mathcal{P}}$ akkor és csak akkor, ha $\bar{q} \neq \varepsilon \ \wedge \ \exists \mathrm{A} \longrightarrow q \in \mathcal{P}$, hogy $\bar{q}$-t q-ból néhány (esetleg nulla) H-beli jel elhagyásával kapjuk.

Az előbbi példát tekintve az ε-mentesített nyelvtan konstrukciója:

$H_1 = \{\mathrm{A},\mathrm{C}\}$,

$H_2 = \{\mathrm{A},\mathrm{C},\mathrm{B},\mathrm{S}\}$,

$H_3 = H_2 = H = \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{S}\}$.

$\bar{\mathcal{P}}$ elemei:

$\mathrm{S} \longrightarrow \mathrm{AB}c \,|\mathrm{B}c\, |\mathrm{A}c \mid c \,|\mathrm{AA}|\mathrm{A}$,

$\mathrm{B} \longrightarrow \mathrm{CC}|\mathrm{C}$,

$\mathrm{A} \longrightarrow a$,

$\mathrm{C} \longrightarrow b$.

1.26. állítás. $L(G') = L(G) \setminus \{\varepsilon\}$

Bizonyítás.

$L(G') \subseteq L(G) \setminus \{\varepsilon\}$:

Mivel G'-ben nincs ε jobboldalú szabály, ezért nyilván $u \neq \varepsilon$. Legyen $u \in L(G')$ egy tetszőleges szó. Vizsgáljuk valamely G'-beli levezetését:

$$S \xrightarrow[G']{*} \alpha_1 A \alpha_2 \xrightarrow[G']{} \alpha_1 \bar{q} \alpha_2 \xrightarrow[G']{*} u.$$

A kiemelt szabály helyettesíthető a G nyelvtan következő levezetésével:

$$A \xrightarrow[G]{} q \xrightarrow[G]{*} \bar{q},$$

ahol a $\bar{q}$-ból elhagyott H-beli elemekből levezetjük ε-t. Ha ezt minden G'-beli szabályra elvégezzük, akkor u egy G-beli levezetését kapjuk. Tehát $S \xrightarrow[G]{*} u$, így $u \in L(G)$. Nyilván $u \neq \varepsilon$ miatt teljesül $u \in L(G) \setminus \{\varepsilon\}$ is.

$L(G') \supseteq L(G) \setminus \{\varepsilon\}$:

Ehhez elég bebizonyítani, hogy ha $S \xrightarrow[G]{*} u \neq \varepsilon$, akkor $S \xrightarrow[G']{*} u$. Ehhez belátjuk a következő lemmát:

1.27. lemma. *Ha* $A \xrightarrow[G]{*} \alpha \neq \varepsilon$, *akkor* $A \xrightarrow[G']{*} \alpha$, *bármely* $A \in N$ *esetén.*

Bizonyítás.

A lemma állítását a levezetés hossza szerinti indukcióval látjuk be. Legyen i a levezetés hossza.

$i = 0$-ra az állítás nyilván teljesül, mert $A \xrightarrow[G]{0} \alpha \iff A = \alpha \iff A \xrightarrow[G']{0} \alpha$.

Most tegyük fel, hogy minden i-nél rövidebb levezetésre az állítás igaz. Legyen $A \xrightarrow[G]{i} \alpha$, $(i \geq 1)$, és tekintsük a levezetés legelső lépését:

$$A \xrightarrow[G]{1} Z_1 \dots Z_k \xrightarrow[G]{i-1} \alpha \neq \varepsilon.$$

A függetlenségi lemma miatt α felbontható az $\alpha = \alpha_1 \alpha_2 \dots \alpha_k$ alakra úgy, hogy

$$Z_1 \xrightarrow[G]{<i} \alpha_1, \ \dots \ , Z_k \xrightarrow[G]{<i} \alpha_k.$$

Az α_i-k között még lehetnek üres szavak, ezeket vegyük ki. Így kapunk $\alpha_{i_1}, \dots, \alpha_{i_p} \neq \varepsilon$ részszavakat, melyekre igaz, hogy $\alpha = \alpha_{i_1} \dots \alpha_{i_p}$. Ezekre az indexekre vagy $Z_{i_j} \in T$, vagy alkalmazható az indukciós feltétel, s mindkét esetben

$$Z_{i_1} \xrightarrow[G']{*} \alpha_{i_1}, \ \dots \ , Z_{i_k} \xrightarrow[G']{*} \alpha_{i_k}.$$

Ebből következik, hogy $Z_{i_1} Z_{i_2} \dots Z_{i_p} \xrightarrow[G']{*} \alpha_{i_1} \alpha_{i_2} \dots \alpha_{i_p}$. Viszont $A \longrightarrow Z_{i_1} Z_{i_2} \dots Z_{i_p} \in \bar{\mathcal{P}}$, ugyanis láttuk, hogy $A \longrightarrow Z_1 Z_2 \dots Z_k \in \mathcal{P}$, míg $Z_{i_1} Z_{i_2} \dots Z_{i_p}$-t $Z_1 Z_2 \dots Z_k$-ból néhány H-beli elem elhagyásával kaptuk.

□

Ha ε nem eleme $L(G)$-nek, akkor készen vagyunk, $L(G) = L(G')$. Ha viszont $\varepsilon \in L(G)$, akkor természetesen gondoskodni kell ε levezethetőségéről. Ennek megoldására vegyünk fel egy S' új kezdőszimbólumot, és vegyük fel a következő két szabályt:

$$S' \longrightarrow S \mid \varepsilon.$$

Így már tényleg $L(G) = L(G')$, és G' típusa is megfelelő.

□

$i = 3$:

Itt azt kell belátnunk, hogy tetszőleges $G \in \mathcal{G}_{\text{kit3}}$ esetén van olyan $G' \in \mathcal{G}_3$, melyre $L(G) = L(G')$. Ehhez a következő három transzformációt kell elvégezni:

a) ε-mentesítés:

— Ez a kiterjesztett 2-es nyelvtanok ε-mentesítő eljárása. Ha ezt kiterjesztett 3-as nyelvtanra alkalmazzuk, az eredmény is kiterjesztett 3-as típusú lesz.

b) Láncmentesítés:

— Ezt is mentesítő algoritmussal végezzük. (1-es típusú nyelvtanra alkalmazható algoritmust adunk.)

c) Hosszredukció.

Láncmentesítés:

Legyen G egy 1-es típusú nyelvtan. Láncszabályoknak nevezzük az $A \longrightarrow B$ $(A, B \in N)$ alakú szabályokat. Ezeket fogjuk szimulálni nem láncszabályokkal, hogy ne legyenek „fölösleges" lépések a levezetés során.

Tegyük fel, hogy G-ben a következő láncszabályokat tartalmazó levezetést tudjuk elvégezni:

$$\beta_1\alpha_1 A\alpha_2\beta_2 \underset{G}{\longrightarrow} \beta_1\alpha_1 C_1\alpha_2\beta_2 \underset{G}{\longrightarrow} \ldots \underset{G}{\longrightarrow} \beta_1\alpha_1 C_k\alpha_2\beta_2 \underset{G}{\longrightarrow} \beta_1\alpha_1 B\alpha_2\beta_2 \underset{G}{\longrightarrow} \beta_1\alpha_1 q\alpha_2\beta_2,$$

$C_1, C_2, \ldots, C_k, A, B \in N, A \underset{G}{\longrightarrow} C_1, C_1 \underset{G}{\longrightarrow} C_2, \ldots, C_k \underset{G}{\longrightarrow} B$ láncszabályok és $\alpha_1 B\alpha_2 \underset{G}{\longrightarrow} \alpha_1 q\alpha_2$ nem láncszabály. Ezt a levezetést tudjuk szimulálni az $\alpha_1 A\alpha_2 \underset{G'}{\longrightarrow} \alpha_1 q\alpha_2$ új szabállyal a következő módon:

$$\beta_1\alpha_1 A\alpha_2\beta_2 \underset{G'}{\longrightarrow} \beta_1\alpha_1 q\alpha_2\beta_2.$$

A megvalósításhoz meg kell határozni az új szabályokat. Legyen $A \in N$ esetén $H(A) := \{B \in N \mid A \overset{*}{\underset{G}{\longrightarrow}} B\}$. Ezt a halmazt – mivel szabály jobboldalán szereplő nyelvtani jelekre nincsenek ε-szabályok – a következő módon határozhatjuk meg:

$$H_0(A) := \{A\}$$

$$H_{i+1}(A) := H_i(A) \cup \{B \mid \exists C \in H_i(A) \wedge C \underset{G}{\longrightarrow} B\}$$

Az ε-mentesítésnél látottakhoz hasonlóan ezek a halmazok is egy i_0 indextől kezdve azonosak lesznek és megegyeznek $H(A)$-val.

Ha $G = \langle T, N, \mathcal{P}, S \rangle$, akkor $G' = \langle T, N, \mathcal{P}', S \rangle$ lesz az új nyelvtan, ahol

$$\mathcal{P}' = \{\alpha_1 A \alpha_2 \longrightarrow \alpha_1 q \alpha_2 \mid q \notin N \wedge \exists B \in H(A) : \alpha_1 B \alpha_2 \longrightarrow \alpha_1 q \alpha_2 \in \mathcal{P}\}.$$

Ekkor $L(G) = L(G')$, melynek bizonyítását az olvasóra bízzuk.

Ha G kiterjesztett, ε-szabályoktól mentes 3-as típusú nyelvtan volt, akkor az eredmény is kiterjesztett, ε-szabályoktól mentes 3-as típusú lesz, melyben természetesen láncszabályok sincsenek.

Hosszredukció:

Az előző lépés miatt a nem megfelelő alakú szabályok már csak a következő formájúak lehetnek:

$$A \longrightarrow uB,$$

$$A \longrightarrow u,$$

ahol $l(u) \geq 2$. u-t betűnként kiírva az $A \longrightarrow t_1 t_2 \dots t_k B$-t könnyen szimulálhatjuk a következő 3-as típusú szabályrendszerrel, ahol $Z_1, Z_2, \dots, Z_{k-1}$ új nyelvtani jelek:

$$A \longrightarrow t_1 Z_1,$$

$$Z_1 \longrightarrow t_2 Z_2,$$

$$\vdots$$

$$Z_{k-1} \longrightarrow t_k B.$$

Az $A \longrightarrow u$ szabály esetén hasonlóan járunk el, csak az utolsó szabály $Z_{k-1} \longrightarrow t_k$ lesz.

Ezzel bebizonyítottuk a kiterjesztési tételt.

□

1.8. Normálformák

A normálformák további megszorításokat jelentenek az egyes nyelvtani osztályokra. A továbbiakban az 1, 2, 3-as osztályokra adunk újabb megszorításokat.

1.8.1. Kuroda normálforma

$i = 1$:

1.28. definíció. *Legyen $G = \langle T, N, \mathcal{P}, S \rangle$ egy nyelvtan. A G nyelvtan Kuroda-féle normálformájú, ha szabályai a következő alakúak:*

(i) $S \longrightarrow \varepsilon$, *ilyenkor S a kezdőszimbólum, és S szabály jobboldalán nem szerepelhet,*

(ii) $A \longrightarrow t$,

(iii) $A \longrightarrow BC$,

(iv) $AB \longrightarrow AC$,

(vi) $BA \longrightarrow CA$,

ahol $A, B, C \in N$, *és* $t \in T$.

1.29. tétel. (Kuroda-normálforma tétel.) *Tetszőleges* $L \in \mathcal{L}_1$ *nyelv esetén létezik olyan* G *nyelvtan, amely Kuroda-normálformájú, és* $L(G) = L$.

Bizonyítás.

Elegendő belátni, hogy tetszőleges G kiterjesztett 1-es nyelvtanhoz létezik olyan G' Kuroda-normálformájú nyelvtan, melyre $L(G) = L(G')$. Legyen $G = \langle T, N, \mathcal{P}, S \rangle$. Feltehető, hogy G-ben terminális jel csak $A \longrightarrow a$ alakú szabályban fordul elő, ahol $A \in N$, és $a \in T$.

A normálformának nem megfelelő szabályokat szimuláljuk jó szabályokkal. Hogyan néznek ki ezek a „rossz" szabályok? Egyrészt csak nyelvtani jelek vannak bennük, másrészt ha a bal oldal hossza 1, akkor alakjuk:

$$A \longrightarrow X_1 X_2 \dots X_k, \text{ ahol } k = 1 \text{ vagy } k \geq 3.$$

Az $A \longrightarrow X$ alakú láncszabályok esetén a kiterjesztett 1-es típusú nyelvtanokra bevezetett láncmentesítés algoritmusát alkalmazhatjuk. Ha $k \geq 3$, akkor vezessük be a $Z_1, \dots, Z_{k-2}$ új nyelvtani jeleket, és következő szabályokat:

$$\begin{aligned} A &\longrightarrow X_1 Z_1, \\ Z_1 &\longrightarrow X_2 Z_2, \\ &\vdots \\ Z_{k-2} &\longrightarrow X_{k-1} X_k. \end{aligned}$$

Ha a bal oldal hossza legalább 2, akkor a „rossz" szabály a következőképpen írható le:

$$X_1 X_2 \dots X_m \longrightarrow Y_1 Y_2 \dots Y_n \ (m \geq 2, n \geq m)$$

azaz hosszúságot nem csökkentő szabály. Szimulációja a $Z_1, Z_2, \dots, Z_{n-2}$ új nyelvtani jelek bevezetésével:

$$\begin{aligned} X_1 X_2 &\longrightarrow Y_1 Z_1, \\ Z_1 X_3 &\longrightarrow Y_2 Z_2, \\ &\vdots \\ Z_{m-3} X_{m-1} &\longrightarrow Y_{m-2} Z_{m-2}, \\ Z_{m-2} &\longrightarrow Y_{m-1} Z_{m-1}, \\ &\vdots \\ Z_{n-2} &\longrightarrow Y_{n-1} Y_n. \end{aligned}$$

Ezek között még vannak nem Kuroda-normálformájúak, sémájuk $AB \longrightarrow CD$. Átalakításokkal egy W új nyelvtani jelet vezetünk be, melyek segítségével a „rossz” szabály hatását „jó” szabályokkal szimuláljuk:

$AB \longrightarrow AW$,

$AW \longrightarrow CW$,

$CW \longrightarrow CD$.

□

1.8.2. Chomsky normálforma

1.30. definíció. *Legyen $G = \langle T, N, \mathcal{P}, S \rangle$ egy nyelvtan. A G nyelvtan Chomsky-féle normálformájú, ha szabályai a következő alakúak:*

(i) $S \longrightarrow \varepsilon$, *ahol S a kezdőszimbólum, és ha van ilyen szabály, akkor S szabály jobboldalán nem szerepelhet,*

(ii) $A \longrightarrow t, \quad t \in T, A \in N$,

(iii) $A \longrightarrow BC$, *ahol* $A, B, C \in N$.

1.31. tétel. (Chomsky-normálforma tétel.) *Tetszőleges $L \in \mathcal{L}_2$ esetén van olyan G Chomsky-normálformájú nyelvtan, amelyre $L(G) = L$.*

Bizonyítás.

Azt bizonyítjuk be, hogy tetszőleges $G \in \mathcal{G}_{\text{kit2}}$ esetén van olyan G' Chomsky-normálformájú nyelvtan, melyre $L(G) = L(G')$. A konstrukció lépései:

a) ε-mentesítés,

b) álterminálisok bevezetése,

c) lánc-mentesítés,

d) hosszredukció.

Az ε-, és a lánc-mentesítés algoritmusát már láttuk. Az álterminálisok bevezetésének célja, hogy az $A \longrightarrow q \in \mathcal{P}$ $\big(l(q) \geq 2\big)$ szabályokban a terminális jeleket álterminális (nyelvtani) jelekre cseréljük. Ezért a $\mathcal{P}$' szabályrendszerhez hozzá kell venni az $X_t \longrightarrow t$ szabályokat. Így ezen lépések után a következő szabályformák lesznek:

$S \longrightarrow \varepsilon$, ahol S a kezdőszimbólum, és ha van ilyen szabály, akkor S szabály jobboldalán nem fordulhat elő,

$A \longrightarrow t$, ahol $A \in N$, és $t \in T$,

$A \longrightarrow$Q, ahol Q$\in N^*$ és $l(Q) \geq 2$.

A hosszredukció az $A \longrightarrow Q$ alakú szabályokra $l(Q) \geq 3$ esetén hasonlóan végezhető el, mint a Kuroda-normálforma esetében.

□

1.8.3. Greibach normálforma

1.32. definíció. *Legyen $G = \langle T, N, \mathcal{P}, S \rangle$ egy nyelvtan. A G nyelvtan Greibach-féle normálformájú, ha szabályai a következő alakúak:*

$S \longrightarrow \varepsilon$, ahol S kezdőszimbólum és ha van ilyen szabály, akkor S nem fordul elő szabály jobboldalán,

$A \longrightarrow aQ$, ahol $A \in N, \quad a \in T, \quad$ és $Q \in N^*$.

1.33. tétel. Greibach-féle normálforma tétel *Tetszőleges $L \in \mathcal{L}_2$ nyelvhez létezik olyan G Greibach normálformájú nyelvtan, melyre $L(G) = L$.*

A tétel bizonyítása előtt szükségünk lesz még néhány olyan előkészítő fogalomra, mint a rekurzív nyelvtani jel, a rekurzív nyelvtan, a rendezett nyelvtan és a kvázi-Greibach nyelvtan fogalma.

1.34. definíció. *Egy A nyelvtani jel rekurzív (önbeágyazott), ha indul belőle $A \xrightarrow[G]{(i)} \alpha_1 A \alpha_2$ alakú levezetés, ahol $\alpha_1, \alpha_2 \in (T \cup N)^*$ és $i > 0$.*

1.35. definíció. *Egy $A \in N$ nyelvtani jel balrekurzív, illetve jobbrekurzív, ha rekurzív és $\alpha_1 = \varepsilon$, illetve $\alpha_2 = \varepsilon$.*

Vegyük észre, hogy ha egy nyelvtani jel nem balrekurzív és nem jobbrekurzív, akkor attól még lehet rekurzív.

A Greibach normálforma lényege, hogy kizárja a nyelvtanból a balrekurzió lehetőségét. Kérdés: Csökken-e a második típusú nyelvtanok hatóereje, ha nem csak a bal, hanem mindenféle rekurziót kizárjunk?

1.36. tétel. *Ha a G 2-es típusú nyelvtanban nincs rekurzív nyelvtani jel, akkor $L(G)$ véges (és így 3-as típusú).*

Bizonyítás. A nyelvtani jelek számára vonatkozó indukcióval. A részleteket az olvasóra bízzuk.

□

2-es típusú nyelvtanok redukálása

Legyen kiinduló nyelvtanunk egy kiterjesztett 2-es típusú nyelvtan. A nyelvtan redukálása felesleges nyelvtani jelei, illetve szabályai elhagyását jelenti. Milyenek ezek a felesleges nyelvtani jelek? Az első csoportot az olyan nyelvtani jelek képezik, melyek a levezetés során zsákutcába visznek, azaz belőlük nem vezethető le T^*-beli szó. A második csoportba pedig a kezdőszimbólumból nem elérhető nyelvtani jelek tartoznak.

Zsákutcák megkeresése, zsákutca-mentesítés:

Legyen $H := \{A \in N \mid \exists u \in T^* : A \xrightarrow[G]{*} u\}$. Ezen H halmaz konstrukciós lépései:

$H_1 := \{A \in N \mid \exists q \in T^* : A \longrightarrow q \in \mathcal{P}\}$.

$H_{i+1} := H_i \cup \{A \mid \exists q \in (H_i \cup T)^* : A \longrightarrow q \in \mathcal{P}\}$.

A H_i halmazok monoton növekvő, felülről korlátos sorozatot alkotnak, ezért — mint ahogy már több hasonló konstrukciónál láttuk — egy bizonyos i_0 indextől kezdvődően mind azonosak lesznek, és erre az indexre $H_{i_0} = H$. Az $N \setminus H$-beli jelek lesznek a zsákutcák, a zsákutcamentes nyelvtan nyelvtani jelei a H elemei, míg szabályai a csak $H \cup T$-beli jeleket tartalmazó eredeti szabályok.

Az elérhető nyelvtani jelek megkeresése, összefüggővé alakítás:

Legyen a K halmaz az S kezdőszimbólumból levezetéssel elérhető nyelvtani jelek halmaza. Konstrukciós lépései:

$$K_0 = \{S\},$$

$$K_{i+1} = K_i \cup \{A \in N \mid \exists B \in K_i : B \longrightarrow \alpha_1 \mathrm{A} \alpha_2 \in \mathcal{P}\}.$$

A K_i monoton növő, felülről korlátos sorozat itt is K-ban stabilizálódik, ez lesz az összefüggő nyelvtan nyelvtani jeleinek halmaza, míg szabályai a csak $K \cup T$-beli elemeket tartalmazó eredeti szabályok.

Felmerülhet a kérdés, milyen sorrendben kell ezt a két lépést végrehajtani, ha a kapott nyelvtantól azt várjuk el, hogy zsákutcamentes és összefüggő legyen.

1.37. állítás. *Csak az előbbi sorrend a megfelelő, azaz először a zsákutcákat kell kiszűrni, utána pedig a nem elérhető jeleket.*

Bizonyítás. Miért nem jó a másik sorrend? Ellenpélda:

$$\mathrm{S} \longrightarrow a \mid \mathrm{AB},$$

$$\mathrm{A} \longrightarrow a.$$

Ez összefüggő. Ha most kiszűrjük a zsákutcákat:

$$\mathrm{S} \longrightarrow a,$$

$$\mathrm{A} \longrightarrow a.$$

Ez pedig nem összefüggő.

Az adott sorrend azért megfelelő, mert ha egy G nyelvtan zsákutcamentes, akkor összefüggővé alakítás után is zsákutcamentes lesz. Ennek bizonyítását az olvasóra bízzuk. □

1.38. definíció. *A zsákutcamentes és összefüggő 2-es típusú nyelvtanokat redukáltnak nevezzük.*

1.39. definíció. *A G nyelvtant rendezett nyelvtannak nevezzük, ha nyelvtani jelei megszámozhatóak úgy, hogy ha $N = \{A_1, A_2, \ldots, A_n\}$ és $A_i \longrightarrow A_j q \in \mathcal{P}$, akkor $i < j$, $q \in (T \cup N)^*$.*

Megjegyzés: Ha egy ε-szabályokat nem tartalmazó 2-es típusú nyelvtan rendezett, akkor biztosan nem balrekurzív. (Ugyanis mindig nőnie kell a levezetés elején lévő nyelvtani jel indexének.)

1.40. definíció. *A G nyelvtan kvázi Greibach-normálformájú, ha szabályai $A \longrightarrow aq$ alakúak, ahol $q \in (T \cup N)^*, a \in T$, és $A \in N$.*

Térjünk rá most a Greibach-féle normálforma tétel bizonyítására: Elegendő bizonyítani, hogy tetszőleges $G \in \mathcal{G}_2$ nyelvtan esetén létezik olyan G' Greibach-normálformájú nyelvtan, melyre $L(G') = L(G)$. Feltehető, hogy $\varepsilon \notin L(G)$, hisz a konstrukció a másra úgysem használható $S \longrightarrow \varepsilon$ elhagyásával, illetve a végeredményhez való hozzávételével kiterjeszthető az általános esetre. Ennek megfelelően tehát G legyen ε-szabály mentes.

A Greibach-normálformára alakítás egymás után alkalmazandó lépései a következők:

a) redukció,

b) rendezetté alakítás,

c) kvázi Greibach-normálformájúvá alakítás, és

d) Greibach-normálformájúvá alakítás.

A redukció a már megismert algoritmus, a b) és c) lépéseket a következőkben megadjuk. A d) lépés álterminálisok bevezetésével egyszerűen megvalósítható.

Elemi transzformációk

A) típusú transzformáció:

Legyen G 2-es típusú nyelvtan. Tegyük fel, hogy

$s : A \longrightarrow \alpha_1 B \alpha_2 \in \mathcal{P}$, és

$B \longrightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_k$ a B-re vonatkozó összes szabály.

Ez a transzformáció az s szabályt a következőkkel helyettesíti:

$$A \longrightarrow \alpha_1\beta_1\alpha_2 \mid \alpha_1\beta_2\alpha_2 \mid \ldots \mid \alpha_1\beta_k\alpha_2.$$

Ha B ezzel az átalakítással elérhetetlenné válik, akkor őt és a rá vonatkozó szabályokat elhagyjuk. A többi szabály változatlan marad. A kapott G' nyelvtanra nyilván fennáll, hogy $L(G') = L(G)$.

Jelölés: Atrans(s, l) az az A típusú transzformáció, mely az s szabály l-edik nyelvtani jelére végzi el az előbbi átalakítást. Ha nincs l-edik nyelvtani jel, akkor nem csinál semmit.

Megjegyzés: Ha G ε-szabálymentes és redukált volt, akkor G' is az.

B) típusú transzformáció:

Legyen G 2-es típusú ε-szabálymentes, redukált nyelvtan. Tegyük fel, hogy az $A \in N$ nyelvtani jele közvetlenül balrekurzív, azaz legyen az összes szabály A-ra:

$$A \longrightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_k \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_l,$$

ahol a zsákutcamentesség miatt $l \geq 1$, továbbá minden $i = 1, \ldots, l$-re β_i első jele nem A, és $\alpha_j \neq \varepsilon$ $(j = 1, \ldots, k)$.

Hogyan néz ki egy ilyen szabály hatása? Az A nyelvtani jelből a következő alakú szavak vezethetők le (mint közbülső eredmény):

$$\{\beta_1, \ldots, \beta_l\}\{\alpha_1, \ldots, \alpha_k\}^*.$$

Az új szabályok legyenek olyanok, hogy ugyancsak a $\{\beta_1, \ldots, \beta_l\}\{\alpha_1, \ldots, \alpha_k\}^*$ elemeit vezethetjük le velük A-ból, de most már jobbrekurzív módon. Ehhez vezessük be a Z új nyelvtani jelet és a következő szabályokat:

$$A \longrightarrow \beta_1 \mid \ldots \mid \beta_l \mid \beta_1 Z \mid \ldots \mid \beta_l Z,$$

$$Z \longrightarrow \alpha_1 Z \mid \ldots \mid \alpha_k Z \mid \alpha_1 \mid \ldots \mid \alpha_k.$$

Az így kapott G' nyelvtanra itt is teljesül, hogy $L(G) = L(G')$.

Jelölés: Az előbb leírt transzformáció eredménye legyen Btrans(A, Z), ahol ha az $A \in N$ nyelvtani jel nem közvetlenül balrekurzív, akkor az eljárás nem csinál semmit és Z-t sem vezetjük be. Megjegyzés: A G' eredménynyelvtan ugyancsak ε-szabálymentes és redukált lesz.

Rendezetté alakítás

Legyen a G nyelvtanunk ε-szabálymentes, redukált és rendezetlen. Számozzuk meg G nyelvtani jeleit: $N = \{A_1, \ldots, A_n\}$ és legyen A_1 a kezdőszimbólum. Legyen a rendezetté alakító algoritmus elő-, illetve utófeltétele:

$$Q : A\ G \text{ nyelvtan } \varepsilon\text{-szabály mentes, redukált}$$

$$R : Q \wedge (\forall i \in [1, n] : (A_i \longrightarrow A_j\alpha \in \mathcal{P} \Longrightarrow i < j)).$$

Ezt nyilván egy ciklussal oldjuk meg, melynek i-edik fázisában a következő invariáns teljesül:

$$P : Q \wedge (\forall k \in [1, i] : (s : A_k \longrightarrow A_j\alpha \in \mathcal{P} \Longrightarrow k < j)).$$

Ez $i = 0$-ra nyilván teljesül. Az algoritmus a ciklusokra vonatkozó általános megoldó sémát használva:

$i := 0$
$i < n$
$i := i + 1$
$(\exists s : A_i \longrightarrow A_j\alpha \in \mathcal{P}) \ \wedge \ j < i$
Atrans$(s, 1)$
Btrans(A_i, Z_i)

A belső ciklus azt éri el, hogy az A_i nyelvtani jelre vonatkozó szabályok jobboldalának első jele vagy terminális vagy i-nél nagyobb egyenlő indexű nyelvtani jel legyen. A Btrans(A_i, Z_i) ezek közül eltűnteti az A_i-vel kezdődőket. Az ε-szabálymentességet és redukáltságot minden lépés megtartja.

Már csak egy dologra kell vigyáznunk. Az újonnan bevezetett Z_i nyelvtani jelekre még ellenőrizni kell, hogy teljesül-e a rendezettség. Ezt egyszerű módon biztosíthatjuk, átszámozzuk a nyelvtani jeleket a következő sorrend szerint:

$$Z_n, Z_{n-1}, \ldots, Z_1, A_1, A_2, \ldots, A_n.$$

A Z_j-k keletkezésének sorrendje garantálja, hogy az új G' nyelvtan rendezett lesz.

Kvázi Greibach-normálformájúvá alakítás

Legyen a G nyelvtan ε-szabálymentes, redukált és rendezett. Vezessük be az $\mathrm{lh}(\alpha)$ jelölést az α szó legelső jelére. (Az lh az angol lefthead szó rövidítése.)

Legyen az algoritmus elő-, illetve utófeltétele:

$Q : G$ ε-szabálymentes, redukált és rendezett,

$R : Q \wedge \forall i \in [1, n] : (s : A_i \longrightarrow \alpha \in \mathcal{P} \Longrightarrow \mathrm{lh}(\alpha) \notin N)$.

Ezt ismét ciklussal oldjuk meg, melynek invariánsa:

$P : Q \wedge \forall k \in [i, n] : (s : A_k \longrightarrow \alpha \in \mathcal{P} \Longrightarrow \mathrm{lh}(\alpha) \notin N)$.

Az algoritmus struktogramja (ismét a ciklusokra vonatkozó általános megoldó sémát használva):

$i := n$
$i > 1$
$i := i - 1$
$\exists s : A_i \longrightarrow \alpha \in \mathcal{P} \wedge \mathrm{lh}(\alpha) \in N$
$\mathrm{Atrans}(s, 1)$

A ciklus minden egyes lépése egy újabb nyelvtani jelre éri el azt, hogy jobboldalai terminálissal kezdődjenek, míg az ε-szabálymentességet és a redukáltságot minden transzformáció megtartja.

1.8.4. 3-as normálforma

1.41. definíció. *Legyen $G = \langle T, N, \mathcal{P}, S \rangle$ egy nyelvtan. A G nyelvtan 3-as normálformájú, ha szabályai a következő alakúak:*

$A \longrightarrow \varepsilon$,

$A \longrightarrow aB$.

Kiindulás egy tetszőleges 3-as típusú G nyelvtan. Ebben a rossz alakú szabályok $A \longrightarrow a$ alakúak. Könnyen látható, hogy ezt egy új nyelvtani jel és a következő szabályok segítségével szimulálhatjuk:

$A \longrightarrow aF$,

$F \longrightarrow \varepsilon$,

ahol F egy új nyelvtani jel.

1.9. Zártsági tételek

Legyen Φ n-változós nyelvi operátor, azaz ha $L_1, \ldots, L_n$ nyelvek, akkor $\Phi(L_1, \ldots, L_n)$ is legyen nyelv.

1.42. definíció. *Az $\mathcal{L}$ nyelvcsalád zárt a Φ nyelvi operátorra, ha $L_1, \ldots, L_n \in \mathcal{L}$ esetén $\Phi(L_1, \ldots, L_n) \in \mathcal{L}$.*

1.43. tétel. Zártsági tétel *Az $\mathcal{L}_i$ $(i = 0, 1, 2, 3)$ nyelvosztályok zártak az unió, konkatenáció és a lezárás műveletekre.*

Bizonyítás. Elég bizonyítani, hogy ha $G_1, G_2 \in \mathcal{G}_{\text{kit}i}$, akkor léteznek olyan $G_\cup$, G_{konk}, $G^* \in \mathcal{G}_{\text{kit}i}$ nyelvtanok, melyekre

$$L(G_\cup) = L(G_1) \cup L(G_2),$$
$$L(G_{\text{konk}}) = L(G_1)L(G_2),$$
$$L(G^*) = (L(G_1))^*.$$

Legyenek $G_1 = \langle T_1, N_1, \mathcal{P}_1, S_1 \rangle$ és $G_2 = \langle T_2, N_2, \mathcal{P}_2, S_2 \rangle$. Feltehető, hogy $N_1, N_2, T_1 \cup T_2$ páronként diszjunktak. (Különben N_1, N_2 elemeit átnevezzük.) Továbbá feltehető, hogy $i = 0, 1, 2$-re terminális jel csak $A \longrightarrow t (t \in T)$ alakú szabályokban fordul elő. Legyen $S \notin N_1 \cup N_2$ új nyelvtani jel. Az alapkonstrukció:

$$G_\cup := \langle T_1 \cup T_2, N_1 \cup N_2 \cup \{S\}, \mathcal{P}_1 \cup \mathcal{P}_2 \cup \{S \longrightarrow S_1 \mid S_2\}, S \rangle,$$
$$G_{\text{konk}} := \langle T_1 \cup T_2, N_1 \cup N_2 \cup \{S\}, \mathcal{P}_1 \cup \mathcal{P}_2 \cup \{S \longrightarrow S_1 S_2\}, S \rangle,$$
$$G^* := \langle T_1, N_1 \cup \{S\}, \mathcal{P}_1 \cup \{S \longrightarrow \varepsilon \mid S_1 S\}, S \rangle.$$

A konstrukció alapján a „$\supseteq$” tartalmazás nyilván fennáll. Bizonyítani kell, hogy a „$\subseteq$” irányú tartalmazás is teljesül, és hogy a fent definiált nyelvtanok kiterjesztett i-típusú nyelvtanok.

1) Unió:

Bizonyítani kell, hogy $S \xrightarrow[G_\cup]{*} u \implies S_1 \xrightarrow[G_1]{*} u \vee S_2 \xrightarrow[G_2]{*} u$. Ez nyilván teljesül, hiszen kezdésként csak két szabály közül alkalmazhatunk egyet, azaz $S \longrightarrow S_1$ vagy $S \longrightarrow S_2$ valamelyikét.

Tegyük föl, hogy az $S \longrightarrow S_1$ szabályt alkalmaztuk. Ekkor nyilván $S_1 \xrightarrow[G_1]{*} u$, mert a levezetés során csak N_1-beli nyelvtani jelek kerülnek be, és $N_1 \cap N_2 = \emptyset$ miatt csak $\mathcal{P}_1$-beli szabályok alkalmazhatók. Ugyanez az $S \longrightarrow S_2$ szabályra is igaz. Tehát unióra a „$\subseteq$” irány is teljesül.

Kérdés, hogy megmarad-e a típus? Világos, hogy $i = 0, 2, 3$-ra igen, de az $i = 1$ esetnél az alapkonstrukción módosítani kell az esetleges $S_1 \longrightarrow \varepsilon, S_2 \longrightarrow \varepsilon$ szabályok miatt:

$$\varepsilon \notin L(G_1) \cup L(G_2) \implies \text{nincs módosítás},$$
$$\varepsilon \in L(G_1) \cup L(G_2) \implies \mathcal{P}_\cup := \mathcal{P}_1 \setminus \{S_1 \longrightarrow \varepsilon\} \cup \mathcal{P}_2 \setminus \{S_2 \longrightarrow \varepsilon\} \cup \{S \longrightarrow S_1 \mid S_2 \mid \varepsilon\}.$$

2) Konkatenáció:

A „$\subseteq$” irányú tartalmazás bizonyításához be kell látni, hogy $S \xrightarrow[G_{\text{konk}}]{*} u$ esetén létezik az u szónak olyan $u = vw$ felbontása, amelyre $S_1 \xrightarrow[G_1]{*} v$, és $S_2 \xrightarrow[G_2]{*} w$ teljesül.

Tekintsünk egy tetszőleges $u \in L(G_{\text{konk}})$ szót. Erre létezik levezetés G_{konk}-ban, azaz $S \xrightarrow[G_{\text{konk}}]{*} u$. Mivel $N_1 \cap N_2 = \emptyset$, és szabályok baloldalán csak nyelvtani jelek vannak, ez a levezetés elvégezhető a következő módon is:

$$S \underset{G_{\text{konk}}}{\longrightarrow} S_1S_2 \xrightarrow[G_{\text{konk}}]{*} vS_2 \xrightarrow[G_{\text{konk}}]{*} vw = u,$$

ahol S_1-ből $\mathcal{P}_1$-beli szabályokkal vezettük le a v szót, S_2-ből pedig a $\mathcal{P}_2$-beli szabályokat alkalmaztuk w levezetésére. Tehát a v és w szavaknak rendre létezik levezetése az eredeti G_1, G_2 nyelvtanokban, $S_1 \xrightarrow[G_1]{*} v$ és $S_2 \xrightarrow[G_2]{*} w$. Ez viszont pontosan azt jelenti, hogy $L(G_{\text{konk}}) \subseteq L(G_1)L(G_2)$.

További kérdés, hogy milyen az így kapott nyelvtan típusa? Világos, hogy az $i = 0, 2$ esetekben G_{konk} típusa megfelelő. Viszont $i = 3$ esetén csak az $A \longrightarrow u$ és az $A \longrightarrow uB (u \in T^*)$ alakú szabályok vannak megengedve, az $S \longrightarrow S_1S_2$ szabály már nem. Ezt a konstrukció módosításával szimulálni fogjuk. Tegyük fel, hogy az $A \longrightarrow a \in \mathcal{P}_1$ szabállyal fejeződik be a G_1-beli levezetés. Ekkor ettől az a jeltől kell folytatnunk a második (G_2-beli) levezetést a következő szabállyal:

$$A \longrightarrow aS_2.$$

Innen már változatlan a levezetés, mint G_2-ben. Jelöljük ezt a $\mathcal{P}_1$ szabályokra vonatkozó transzformációt $\Psi(\mathcal{P}_1, S_2)$-vel. Így a módosított nyelvtan:

$$G_{\text{konk}} := \big\langle T_1 \cup T_2, N_1 \cup N_2, \Psi(\mathcal{P}_1, S_2) \cup \mathcal{P}_2, S_1 \big\rangle.$$

Ekkor már $i = 3$-ra is megfelelő a típus.

Az $i = 1$ esetben ismét az ε-levezetését szolgáló szabályokkal van probléma. Itt négy eset lehetséges. Ha sem a G_1 nyelvtanban, sem a G_2 nyelvtanban nem lehet ε-t levezetni, akkor nincs probléma, az eredeti konstrukció is 1-es típusú lesz. Ha viszont vagy a G_1, vagy a G_2 nyelvtanban levezethető ε, akkor nyilván változtatnunk kell G_{konk} szabályrendszerének definícióján az alábbi módon:

$$\mathcal{P}_{\text{konk}} := \begin{cases} (\mathcal{P}_1 \cup \mathcal{P}_2 \cup \{S \longrightarrow S_1S_2 \mid S_2\}) \setminus \{S_1 \longrightarrow \varepsilon\}, & \text{ha } \varepsilon \in L(G_1), \varepsilon \notin L(G_2), \\ (\mathcal{P}_1 \cup \mathcal{P}_2 \cup \{S \longrightarrow S_1S_2 \mid S_1\}) \setminus \{S_2 \longrightarrow \varepsilon\}, & \text{ha } \varepsilon \notin L(G_1), \varepsilon \in L(G_2), \\ (\mathcal{P}_1 \cup \mathcal{P}_2 \cup \{S \longrightarrow \varepsilon \mid S_1S_2 \mid S_1 \mid S_2\}) \setminus \\ \qquad\qquad \{S_1 \longrightarrow \varepsilon, S_2 \longrightarrow \varepsilon\}, & \text{ha } \varepsilon \in L(G_1), \varepsilon \in L(G_2). \end{cases}$$

Így G_{konk} már nyilván 1-es típusú lesz.

3) Lezárás:

Bizonyítani kell, hogy $S \xrightarrow[G^*]{*} u$ esetén létezik olyan k konstans, hogy $u = u_1u_2 \dots u_k$, ahol minden i-re $S_1 \xrightarrow[G_1]{*} u_i$.

Ezt az alábbi állítás teljes indukciós bizonyításával próbálhatjuk meg belátni:

1.44. állítás. *Ha* $S \xrightarrow[G^*]{*} \alpha$, *akkor* $\exists k \geq 1$ *és* $\alpha = \alpha_1 \dots \alpha_k$ *felbontás, hogy minden* $h = 1, \dots, k-1 : S_1 \xrightarrow[G_1]{*} \alpha_h$ *és vagy* $\alpha_k = S$ *vagy* $S_1 \xrightarrow[G_1]{*} \alpha_k$.

Az indukciós lépés akkor működik, ha az $\alpha = \alpha_1 \dots \alpha_k$-ra valamely G^*-beli szabályt csak úgy lehet alkalmazni, ha annak baloldala valamely α_j részszava. Ez $i = 0$ és 1 esetben általában nem igaz, adható ellenpélda. $i = 2$-re viszont nyilvánvaló, a részleteket az olvasóra bízzuk.

Tehát $i = 2$-re és $i = 3$-ra $L(G^*) = (L(G_1))^*$. Ha G_1 típusa 2-es, akkor G^* is az.

Ha $i = 3$, akkor az előzőek miatt ugyan $L(G^*) = L(G)^*$, de az $S \longrightarrow S_1 S$ szabály nem 3-as típusú. Ezért ennek hatását a konkatenációnál látottakhoz hasonlóan szimulálni kell. A szabályrendszer ezért $\mathcal{P}_1 \cup \Psi(\mathcal{P}_1, S_1) \cup \{S \longrightarrow \varepsilon \mid S_1\}$-re módosul.

Ugyanakkor az $i = 0, 1$ esetekben $L(G^*) = L(G_1)^*$ általában nem is igaz, az eredeti konstrukción változtatni kell úgy, hogy ne fordulhasson elő a fenti állítás bizonyításában α_j-ken átívelő szabályalkalmazás. Ezt úgy tesszük meg, hogy a következő α_j példányt csak akkor kezdjük el generálni, amikor az előző végén megjelent már egy terminális jel.

$\mathcal{P}^* := \mathcal{P}_1 \cup \{S \longrightarrow \varepsilon \mid S_1 \mid S_1 S'\} \cup \{tS' \longrightarrow tS_1 \mid \forall t \in T_1\} \cup \{tS' \longrightarrow tS_1 S' \mid \forall t \in T_1\}$.

A kötelezően behozandó terminális jelek megakadályozzák az α_j szavak határain átívelő szabályalkalmazást, az indukciós bizonyítás $i = 2$-höz hasonlóan befejezhető.

Tehát $i = 0, 1$ esetén is elértük, hogy a módosított G^*-ra $L(G^*) = L(G)^*$. A típusok megfelelésében még annyit kell tennünk, hogy $i = 1$ esetén az esetleges $S_1 \longrightarrow \varepsilon$ szabályt kihagyjuk.

□

2. A Chomsky-osztályok és az algoritmussal definiált nyelvek kapcsolata

Legyen $\mathcal{L}_{\text{RekFel}}$ a rekurzívan felsorolható nyelvek osztálya, $\mathcal{L}_{\text{ParcRek}}$ a parciálisan rekurzív nyelvek osztálya, $\mathcal{L}_{\text{Rek}}$ pedig a rekurzív nyelvek osztálya. A következő tétel azt bizonyítja, hogy a Chomsky-osztályokba tartozó nyelvek algoritmusok segítségével is megadhatók.

2.1. tétel. $\mathcal{L}_0 \subseteq \mathcal{L}_{\text{RekFel}}$, $\mathcal{L}_0 \subseteq \mathcal{L}_{\text{ParcRek}}$, $\mathcal{L}_1 \subseteq \mathcal{L}_{\text{Rek}}$.

Bizonyítás. Megfelelő algoritmusok konstrukciójával bizonyítjuk. Legyen $G = \langle T, N, \mathcal{P}, S \rangle$ tetszőleges 0-ás típusú nyelvtan. Konstruáljuk meg a G-beli összes levezetések t_G gráfját.

A t_G gráf definíciója: pontjai $(T \cup N)^*$ elemeivel, élei $\mathbb{N} \times \mathbb{N}$ elemeivel vannak címkézve.

Egy α pontból kiinduló, β végpontú és (i, j)-vel címkézett él akkor és csak akkor lesz éle a t_G gráfnak, ha $\alpha \underset{G}{\longrightarrow} \beta$ és a levezetés során G-nek a j-edik szabályát használjuk, míg a helyettesítés első jele α-nak az i-edik pozícióján van. (Előzőleg számozzuk meg a szabályokat.)

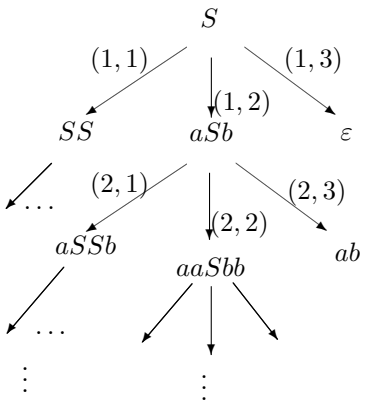
Nyilván ez a gráf G összes levezetését tartalmazni fogja. Például legyen a G nyelvtan szabályrendszere a következő:

1) S$\longrightarrow$SS,

2) S$\longrightarrow a$Sb,

3) S$\longrightarrow \varepsilon$.

Ekkor ennek az S csúcsból elérhető részgráfja a következőképpen néz ki:



Az S-ből kiinduló levezetések gráfjának egy része

Kérdés, hogy mikor vezethető le S-ből egy adott u szó. Nyilván akkor, ha a t_G gráf S-ből elérhető részének van olyan pontja, ami u-val van címkézve. $L(G)$ felsorolásához tehát elég az előbbi t_G gráfot S-ből kiinduló szélességi bejárással bejárni.

Ha a bejárási algoritmus csúcsfeldolgozó eljárása:

címke(c) $\in T^*$	különben
kiír(címke(c))	SKIP

akkor pontosan $L(G)$-t felsoroló algoritmust kapunk. Ha a bejárási algoritmus csúcsfeldolgozó eljárása:

címke(c) $= u$	különben
Exit(+, bejárás)	SKIP

ahol u az input szó, akkor pontosan egy $L(G)$-t parciálisan eldöntő algoritmust kapunk. Ha a parciálisan eldöntő algoritmus csúcsfeldolgozó eljárása:

szintszám(c) $> K(G, u)$	szintszám(c) $\leq K(G, u)$ $\wedge$ címke(c) $= u$	különben
Exit($-$, bejárás)	Exit(+, bejárás)	SKIP

akkor $G \in \mathcal{G}_1$ esetben $L(G)$-t rekurzívan eldöntő algoritmust kapunk, hiszen $u \in L(G)$ esetén ilyenkor van legfeljebb $K(G, u)$ hosszú levezetés.

Az „Exit(+, bejárás)” és az „Exit(−, bejárás)” az jelentik, hogy a szélességi bejárást befejezzük, és pozitív, illetve negatív választ adunk vissza.

□

Ha a Church-tézist elfogadjuk, akkor $\mathcal{L}_0 = \mathcal{L}_{\text{ParcRek}} = \mathcal{L}_{\text{RekFel}}$.

3. Veremautomatákkal jellemezhető nyelvek

Ebben a fejezetben a matematikai gépek egy speciális változatával, a veremautomatákkal foglalkozunk. Azt vizsáljuk, hogy az általuk felismert nyelveknek mi a viszony a Chomsky nyelvosztályokhoz.

3.1. definíció. *n-verem alatt a következő $(2n+5)$-öst értjük:*

$$\mathcal{V} = \langle A, T, \Sigma_1\ , \ldots, \Sigma_n\ , \delta, a_0, \sigma_0^1\ , \ldots, \sigma_0^n\ , F \rangle,$$

ahol

A *az állapotok halmaza (ez legyen véges halmaz),*

T *egy ábécé, a bemenő ábécé,*

Σ_i *az i-edik verem ábécéje,*

δ *az állapotátmeneti függvény,*

$a_0 \in A$ *kezdőállapot,*

σ_0^i *az i. verem kezdőszimbóluma, ahol* $\sigma_0^i\ \in \Sigma_i$,

$F \subseteq A$ *a végállapotok halmaza.*

Az állapotátmeneti függvény $\delta : A \times (T \cup \{\varepsilon\}) \times \Sigma_1\ \times \ldots \times \Sigma_n\ \longrightarrow\ 2^{A \times \Sigma_1^* \times \ldots \times \Sigma_n^*}$ *alakú függvény, melyre megköveteljük, hogy értékkészlete véges halmazokból álljon.*

A $\mathcal{V}$, mint matematikai gép működése ütemekben történik. Egy ütemben kiolvassa a központi egység állapotát, az input szimbólumot és a vermek tetőelemeit. Ezek függvényében új állapotba kerül, s a vermek eddigi tetőelemeit felülírja 0 vagy több jellel. Az olvasó fej egyet jobbra lép (kivéve az ún. ε-mozgás esetén), a tetőmutatók az új tetőelemekre állnak rá. A δ állapotátmeneti függvény egy ilyen, egy ütemben történő működés leírására szolgál. Formálisan a konfiguráció fogalmával lehet definiálni a veremautomaták működését.

3.2. definíció. *Konfigurációnak nevezzük azoknak az adatoknak az összességét, melyektől a gép elkövetkezendő működése függ.*

A konfigurációk a következő alakú $(n+2)$-esek:

$$[a, v, \alpha_1, \ldots, \alpha_n],$$

ahol:

a az aktuális állapot,

v az input szó még elolvasatlan része,

α_i az i-edik verem tartalma.

3.3. definíció. *Közvetlen konfigurációátmenetről beszélünk, ha $\mathcal{V}$ egy lépésben vált át egyik konfigurációból a másikba, azaz $[a, u, \alpha_1, \ldots, \alpha_n] \underset{\mathcal{V}}{\vdash} [b, v, \beta_1, \ldots, \beta_n]$ akkor és csak akkor, ha van olyan $t \in T \cup \{\varepsilon\}$, hogy $u = tv$, továbbá minden $i \in [1, n]$ esetén van olyan $\sigma_i \in \Sigma_i$ és $\gamma_i, \tau_i \in \Sigma_i^*$, amelyekre $\alpha_i = \sigma_i\gamma_i, \beta_i = \tau_i\gamma_i$, valamint $(b, \tau_1, \ldots, \tau_n) \in \delta(a, t, \sigma_1, \ldots, \sigma_n)$. Ha itt $t = \varepsilon$, akkor ε-mozgásról beszélünk.*

3.4. definíció. *A közvetett konfigurációátmenet a közvetlen átmenet reflexív, tranzitív lezártja. Jelölése: $\overset{*}{\underset{\mathcal{V}}{\vdash}}$*

3.5. definíció. *Az u szóhoz tartozó kezdőkonfiguráció: $[a_0, u, \sigma_0^1, \ldots, \sigma_0^n]$.*

3.6. definíció. *Egy K konfiguráció terminális konfiguráció, ha nincs rákövetkezője, azaz $\nexists K'$ konfiguráció, hogy $K \underset{\mathcal{V}}{\vdash} K'$.*

3.7. definíció. *Végállapottal illetve üres veremmel elfogadó egy $[f, \varepsilon, \beta_1, \ldots, \beta_n]$ konfiguráció, ha $f \in F$ illetve $\beta_1 = \varepsilon$.*

A $\mathcal{V}$ n-verem végállapottal illetve üres veremmel elfogadja az u szót, ha létezik átmenet az u szóhoz tartozó kezdőkonfigurációból végállapottal illetve üres veremmel elfogadó konfigurációba. A $\mathcal{V}$ által végállapottal illetve üres veremmel elfogadott nyelv:

$$L^F(\mathcal{V}) = \{u \in T^* \mid [a_0, u, \sigma_0^1, \ldots, \sigma_0^n] \overset{*}{\underset{\mathcal{V}}{\vdash}} [f, \varepsilon, \beta_1, \beta_2, \ldots, \beta_n] \text{ valamely } f \in F\text{-re}\},$$

$$L^\varepsilon(\mathcal{V}) = \{u \in T^* \mid [a_0, u, \sigma_0^1, \ldots, \sigma_0^n] \overset{*}{\underset{\mathcal{V}}{\vdash}} [f, \varepsilon, \varepsilon, \beta_2, \ldots, \beta_n]\}.$$

3.8. definíció. *Egy adott $\mathcal{V}$ n-verem determinisztikus, ha minden konfigurációnak legfeljebb egy rákövetkezője van. Ez a következő két feltétel teljesülése esetén áll fenn:*

1) Minden $(a, t, \sigma_1, \ldots, \sigma_n) \in D_\delta$ esetén $\mid \delta(a, t, \sigma_1, \ldots, \sigma_n) \mid \leq 1$ és

2) $\delta(a, \varepsilon, \sigma_1, \ldots, \sigma_n) \neq \emptyset$ esetén minden $t \in T$-re $\mid \delta(a, t, \sigma_1, \ldots, \sigma_n) \mid = 0$.

A második feltétel szerint ha valamely konfigurációban lehetséges ε-mozgás, ott „valódi" olvasással történő működés nem lehetséges.

Jelölések:

$\mathcal{L}_{n\mathcal{V}}^{\text{Fin}}$ illetve $\mathcal{L}_{n\mathcal{V}}^{\varepsilon}$ jelentse az n-vermek által végállapottal illetve üres veremmel elfogadott nyelvek osztályát,

$\mathcal{L}_{Dn\mathcal{V}}^{\text{Fin}}$ illetve $\mathcal{L}_{Dn\mathcal{V}}^{\varepsilon}$ jelentse a determinisztikus n-vermek által végállapottal illetve üres veremmel elfogadott nyelvek osztályát.

A veremautomaták működésének néhány érdekes tulajdonságát fogalmazzák meg az alábbi állítások:

3.9. állítás. *A* $\overset{*}{\underset{\nu}{\dashv}}$ *reláció független a szó elolvasatlan részétől, azaz*

$$[a, u, \alpha_1, \ldots, \alpha_n] \overset{*}{\underset{\nu}{\dashv}} [b, \varepsilon, \beta_1, \ldots, \beta_n] \Longleftrightarrow [a, uw, \alpha_1, \ldots, \alpha_n] \overset{*}{\underset{\nu}{\dashv}} [b, w, \beta_1, \ldots, \beta_n].$$

3.10. állítás. *A konfigurációátmenetek folytathatók, azaz* $[a, u, \alpha_1, \ldots, \alpha_n] \overset{*}{\underset{\nu}{\dashv}} [b, \varepsilon, \beta_1, \ldots, \beta_n]$ *és* $[b, v, \beta_1, \ldots, \beta_n] \overset{*}{\underset{\nu}{\dashv}} [c, \varepsilon, \gamma_1, \ldots, \gamma_n]$ *esetén* $[a, uv, \alpha_1, \ldots, \alpha_n] \overset{*}{\underset{\nu}{\dashv}} [c, \varepsilon, \gamma_1, \ldots, \gamma_n]$.

Bizonyítás. Tegyük fel, hogy

$$[a, u, \alpha_1, \ldots, \alpha_n] \overset{*}{\underset{\nu}{\dashv}} [b, \varepsilon, \beta_1, \ldots, \beta_n] \text{ és}$$

$$[b, v, \beta_1, \ldots, \beta_n] \overset{*}{\underset{\nu}{\dashv}} [c, \varepsilon, \gamma_1, \ldots, \gamma_n].$$

Ekkor az előző állítás szerint $[a, uv, \alpha_1, \ldots, \alpha_n] \overset{*}{\underset{\nu}{\dashv}} [b, v, \beta_1, \ldots, \beta_n]$, és a $\overset{*}{\underset{\nu}{\dashv}}$ reláció tranzitivitásából adódik az állítás.

□

3.1. 0-vermek

Milyen egy 0-verem? Mivel itt nincsenek kiegészítő tárak, ezért a működés csak az elolvasott input jeltől és az aktuális állapottól függ. A szavak elfogadása is nyilván csak végállapottal történhet, ezért elegendő csak elfogadásrol beszélni. 0-vermeket általában állapotátmeneti diagrammal ábrázoljuk, melynek pontjai állapotokkal vannak címkézve, élei pedig $T \cup \{\varepsilon\}$-beli elemekkel. Ez így egy irányított gráf, ahol egy adott $a \in A$-val címkézett pontból a $b \in B$ címkéjű pontba pontosan akkor vezet egy $t \in T \bigcup \{\varepsilon\}$ elemmel címkézett él, ha $b \in \delta(a, t)$.
0-vermek osztályozása:

— Nincs megkötés (szokásos elnevezése véges, ε-átmenetes, nemdeterminisztikus automata), az elfogadott nyelvek osztályát jelölje $\mathcal{L}_{\varepsilon\mathrm{NDA}}$.

— ε-átmenet megtiltása (véges, nemdeterminisztikus automata), az elfogadott nyelvek osztálya: $\mathcal{L}_{\mathrm{NDA}}$.

— A nemdeterminisztikusság és az ε-átmenetek megtiltása (véges, parciálisan determinisztikus automata), az elfogadott nyelvek osztálya: $\mathcal{L}_{\mathrm{PDA}}$.

— A nemdeterminisztikusság és az ε-átmenetek megtiltása, mindenütt definiáltság ($\forall a \in A$ és $t \in T$ esetén $\mid \delta(a, t) \mid = 1$, és $\mid \delta(a, \varepsilon) \mid = 0$) (véges, determinisztikus automata), az elfogadott nyelvek osztálya: $\mathcal{L}_{\mathrm{DA}}$.

Ezen osztályozás alapján nyilvánvaló, hogy $\mathcal{L}_{0\mathcal{V}} = \mathcal{L}_{\varepsilon\mathrm{NDA}} \supseteq \mathcal{L}_{\mathrm{NDA}} \supseteq \mathcal{L}_{\mathrm{PDA}} \supseteq \mathcal{L}_{\mathrm{DA}}$. A 0-vermek jelölésére tradícionálisan az $\mathcal{A}$ szimbólumot fogjuk használni.

3.11. tétel. $\mathcal{L}_{0\mathcal{V}} = \mathcal{L}_3$. *A kétirányú tartalmazást látjuk be.*

Bizonyítás.

„$\supseteq$”:

Minden 3-as típusú nyelvhez létezik 3-as normálformájú nyelvtan, így tehát elég belátni, hogy tetszőleges, 3-as normálformában adott nyelvtanhoz létezik $\mathcal{A}$ 0-verem, hogy $L(\mathcal{A}) = L(G)$.

Legyen $G = \langle T, N, \mathcal{P}, S \rangle$. Ennek a nyelvtannak a szabályai a következő alakúak:

$A \longrightarrow tB$

$A \longrightarrow \varepsilon$, ahol $A, B \in N$ és $t \in T$.

Legyen az s levezetés a következő:

$$B_0 \underset{G}{\longrightarrow} t_1 B_1 \underset{G}{\longrightarrow} t_1 t_2 B_2 \underset{G}{\longrightarrow} \ldots \underset{G}{\longrightarrow} t_1 t_2 \ldots t_k B_k \underset{G}{\longrightarrow} t_1 t_2 \ldots t_n.$$

A kérdés az, hogyan tudjuk ezt automatával szimulálni? A t_i jel generálásának feleljen meg a t_i jel olvasása és B_i-k legyenek az állapotok. Ha $A \longrightarrow \varepsilon$ szabályt alkalmazunk, akkor A legyen végállapot. Ekkor a következő konfigurációátmeneteket várjuk:

$$[B_0, t_1 \ldots t_k] \underset{\mathcal{V}}{\vdash} [B_1, t_2 \ldots t_k] \underset{\mathcal{V}}{\vdash} \ \ldots \ \underset{\mathcal{V}}{\vdash} [B_k, \varepsilon].$$

Konstrukció: Legyen $\mathcal{A} = \langle N, T, \delta, S, F \rangle$, ahol

$C \in \delta(B, t) \iff B \longrightarrow tC \in \mathcal{P}$, ahol $t \in T$, és

$B \in F \iff B \longrightarrow \varepsilon \in \mathcal{P}$.

Könnyen belátható az u szó hosszára vonatkozó indukció alapján, hogy

$$[A, u] \overset{*}{\underset{\mathcal{A}}{\vdash}} [B, \varepsilon] \iff A \overset{*}{\underset{G}{\longrightarrow}} uB.$$

Ekkor már valóban $L(\mathcal{A}) = L(G)$, ugyanis $u \in L(\mathcal{A}) \iff \exists B \in F : ([S, u] \overset{*}{\underset{\mathcal{A}}{\vdash}} [B, \varepsilon]) \iff \exists B : (B \underset{G}{\longrightarrow} \varepsilon, S \overset{*}{\underset{G}{\longrightarrow}} uB) \iff S \overset{*}{\underset{G}{\longrightarrow}} u \iff u \in L(G)$.

Milyen az így kapott automata? ε-átmenet nélküli, ezért egy kicsit többet láttunk be, nevezetesen $\mathcal{L}_3 \subseteq \mathcal{L}_{\text{NDA}}$-t.

„$\subseteq$”:

Itt elég belátni, hogy tetszőleges $\mathcal{A}$ 0-veremhez létezik olyan $G \in \mathcal{G}_{\text{kit3}}$, hogy $L(G) = L(\mathcal{A})$. Az előző konstrukció megfordítását alkalmazzuk.

Legyen $\mathcal{A} = \langle A, T, \delta, a_0, F \rangle$, és legyen $G = \langle T, A, \mathcal{P}, a_0 \rangle$ a hozzá konstruálandó nyelvtan. Milyenek legyenek $\mathcal{P}$ szabályai?

$a \longrightarrow tb \in \mathcal{P} \iff b \in \delta(a, t)$, ahol $t \in T \cup \{\varepsilon\}$,

$f \longrightarrow \varepsilon \in \mathcal{P} \iff f \in F$.

Ez kiterjesztett 3-as nyelvtan lesz, mert t lehet ε is, ugyanis $\mathcal{A}$-ban nem volt megtiltva az ε-átmenet. Ekkor a levezetés hossza szerinti indukcióval belátható, hogy

$$a \overset{*}{\underset{G}{\longrightarrow}} ub \iff [a, u] \overset{*}{\underset{\mathcal{A}}{\vdash}} [b, \varepsilon].$$

Ebből már következik, hogy $L(G) = L(\mathcal{A})$.

Következmény: az előző tétel bizonyításából az is adódik, hogy $\mathcal{L}_3 = \mathcal{L}_{\mathrm{NDA}}$.

□

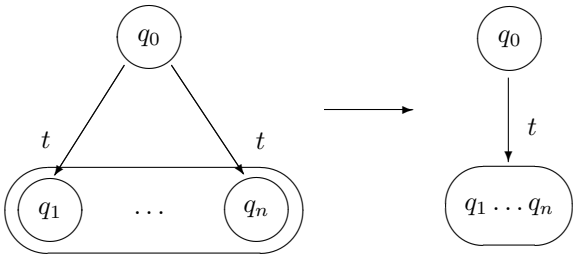
3.1.1. Determinisztikus 0-vermek ($\mathcal{L}_{D0\nu}$)

3.12. tétel. $\mathcal{L}_{\mathrm{DA}} = \mathcal{L}_{\mathrm{NDA}}$.

Bizonyítás. Mivel a $\subseteq$ irány nyilvánvaló, elég bebizonyítani, hogy $\mathcal{L}_{\mathrm{NDA}} \subseteq \mathcal{L}_{\mathrm{DA}}$.

Belátjuk, hogy tetszőleges $\mathcal{A} = \langle A, T, \delta, a_0, F \rangle$ véges, nemdeterminisztikus automatához konstruálható olyan $\mathcal{A}'$ véges, determinisztikus automata, amelyre az elfogadott nyelvek azonosak, azaz $L(\mathcal{A}) = L(\mathcal{A}')$ (determinisztikussá tétel).

Tekintsünk egy $\mathcal{A}$-beli átmenetet. Legyen q_0 az állapotátmeneti gráf egy csúcsa, és egy adott $t \in T$ jel hatására az összes átmenet vezessen $q_1, \ldots, q_n$-be. Ha őket párhuzamos működésként fogjuk fel, akkor készíthetünk olyan gépet, melynek ezen párhuzamos működések együttesei az alapműködései.



Nemdeterminisztikus átmenetek átalakítása

Legyen tehát $\mathcal{A}' = \langle 2^A, T, \delta', \{a_0\}, \mathcal{F} \rangle$. Definiáljuk δ'-t a következő módon:

$$\delta'(\{b_1, \ldots, b_s\}, t) := \bigcup_{i=1}^{s} \delta(b_i, t),$$

míg az $\mathcal{A}'$ végállapotaira

$$B \in \mathcal{F} \iff B \cap F \neq \emptyset.$$

3.13. állítás. $[B, u] \overset{*}{\underset{\mathcal{A}'}{\vdash}} [C, \varepsilon]$ *akkor és csak akkor teljesül, ha egyrészt minden* $b \in B$ *és* $a \in A$ *esetén* $[b, u] \overset{*}{\underset{\mathcal{A}}{\vdash}} [a, \varepsilon]$*-ből következik* $a \in C$*, másrészt minden* $c \in C$*-re van olyan* $b \in B$*, amelyre* $[b, u] \overset{*}{\underset{\mathcal{A}}{\vdash}} [c, \varepsilon]$.

Ezt az állítást a konfigurációátmenetek hosszára vonatkozó teljes indukcióval láthatjuk be, melynek részletezését az olvasóra bízzuk.

Az $\mathcal{A}'$ automata ugyanazt tudja, mint az eredeti, hiszen: $u \in L(\mathcal{A}') \iff \exists B \in \mathcal{F} : ([\{a_0\}, u] \overset{*}{\underset{\mathcal{A}'}{\vdash}} [B, \varepsilon]) \iff \exists B : (B \cap F \neq \emptyset \wedge \forall b \in B : [a_0, u] \overset{*}{\underset{\mathcal{A}}{\vdash}} [b, \varepsilon]) \iff \exists b \in F : [a_0, u] \overset{*}{\underset{\mathcal{A}}{\vdash}} [b, \varepsilon] \iff u \in L(\mathcal{A})$.

Ezért tényleg $L(\mathcal{A}') = L(\mathcal{A})$, és $\mathcal{A}'$ egy véges, determinisztikus automata.

(Gyakorlatban általában $\mathcal{A}'$-nek csak az összefüggő részét konstruálják meg.)

□

Következmény:

$$\mathcal{L}_{0\mathcal{V}} = \mathcal{L}_{\varepsilon\text{NDA}} = \mathcal{L}_{\text{NDA}} = \mathcal{L}_{\text{PDA}} = \mathcal{L}_{\text{DA}} = \mathcal{L}_{D0\mathcal{V}}.$$

A definíciók alapján világos ugyanis, hogy ezen nyelvosztályok közül $\mathcal{L}_{0\mathcal{V}}$ a legtágabb, $\mathcal{L}_{\text{DA}}$ a legszűkebb. A már bizonyított $\mathcal{L}_{0\mathcal{V}} = \mathcal{L}_3 \subseteq \mathcal{L}_{\text{NDA}} = \mathcal{L}_{\text{DA}}$ miatt viszont ők — s így az összes közöttük elhelyezkedő nyelvosztályok is — egyenlőek.

3.1.2. A 3. típusú nyelvek néhány tulajdonsága

3.14. tétel. Kis Bar-Hillel lemma: *Minden $L \in \mathcal{L}_3$ nyelvhez van olyan $n = n(L) \in \mathbb{N}$ nyelvfüggő konstans, hogy minden $u \in L$ szó esetén ha tekintjünk egy tetszőleges $u = \alpha_1 u' \alpha_2$ olyan felbontását, ahol $l(u') \geq n$, akkor van u'-nek olyan v részszava ($u' = \beta_1 v \beta_2$), hogy $0 < l(v) \leq n$, és minden $i \geq 0$ esetén $\alpha_1 \beta_1 v^i \beta_2 \alpha_2 \in L$.*

Kevésbé formálisan a lényeget a következőképpen fejezhetjük ki: L minden szavának elég hosszú részszavában létezik elég rövid, nemüres, beiterálható részszó.

Bizonyítás. Tudjuk, hogy minden $L \in \mathcal{L}_3$ nyelvhez van olyan $\mathcal{A}$ véges, determinisztikus automata, melyre $L(\mathcal{A}) = L$. Legyen $\mathcal{A} = \langle A, T, \delta, a_0, F \rangle$, és $n := \big|A\big| \big(= n(L)\big)$. Vizsgáljunk meg egy megfelelő tulajdonságú szót, azaz legyen $u \in L(\mathcal{A}), u = \alpha_1 u' \alpha_2$, és $l(u') \geq n = \big|A\big|$.

Tegyük fel, hogy $u' = t_1 t_2 \dots t_m$, és $m \geq n$. Az u-t elfogadó konfigurációátmenetet részletezve:

$$[a_0, \alpha_1 t_1 t_2 \dots t_m \alpha_2] \overset{*}{\underset{\mathcal{A}}{\vdash}} [c_0, t_1 t_2 \dots t_m \alpha_2] \underset{\mathcal{A}}{\vdash} [c_1, t_2 \dots t_m \alpha_2] \underset{\mathcal{A}}{\vdash} \dots$$

$$\dots \underset{\mathcal{A}}{\vdash} [c_m, \alpha_2] \overset{*}{\underset{\mathcal{A}}{\vdash}} [f, \varepsilon] \quad (f \in F).$$

Tekintsük a $(c_0, c_1, \dots, c_m)$ sorozatot. Ez $m + 1 \geq n + 1$ elemű. A skatulyaelv alapján biztosan léteznek k, j számok, melyekre $0 \leq j < k \leq n$ és $c_j = c_k$, ugyanis $n = \big|A\big|$.

Az u' szó felbontása legyen $\beta_1 := t_1 \dots t_j, v := t_{j+1} \dots t_k$, és $\beta_2 := t_{k+1} \dots t_m$. Világos, hogy j, k választása miatt: $0 < l(v) \leq n$.

Továbbá az is igaz, hogy

$$[c_0, \beta_1] \overset{*}{\underset{\mathcal{A}}{\vdash}} [c_j, \varepsilon],$$

$$[c_j, v] \overset{*}{\underset{\mathcal{A}}{\vdash}} [c_k, \varepsilon] = [c_j, \varepsilon],$$

$$[c_k, \beta_2] \overset{*}{\underset{\mathcal{A}}{\vdash}} [c_m, \varepsilon].$$

De ezekből már nyilván következik, hogy tetszőleges $i \geq 0$ esetén

$$[c_j, v^i] \overset{*}{\underset{\mathcal{A}}{\vdash}} [c_j, \varepsilon].$$

Összegezve az eddigieket:

$$[a_0, \alpha_1\beta_1 v^i \beta_2 \alpha_2] \overset{*}{\underset{\mathcal{A}}{\vdash}} [c_0, \beta_1 v^i \beta_2 \alpha_2] \overset{*}{\underset{\mathcal{A}}{\vdash}} [c_j, v^i \beta_2 \alpha_2] \overset{*}{\underset{\mathcal{A}}{\vdash}} [c_j, \beta_2 \alpha_2] \overset{*}{\underset{\mathcal{A}}{\vdash}} [c_m, \alpha_2] \overset{*}{\underset{\mathcal{A}}{\vdash}} [f, \varepsilon],$$

és ebből már következik, hogy $\alpha_1\beta_1 v^i \beta_2 \alpha_2 \in L(\mathcal{A}) = L$.

□

Most nézzünk egy példát a Kis Bar-Hillel lemma alkalmazására. Jelölje HE a $\{(,)\}$ ábécé feletti helyes zárójelezések halmazát. Erre fogalmazzuk meg a következő állítást:

3.15. állítás. *$HE \notin \mathcal{L}_3$.*

Bizonyítás. Indirekt módon. Tegyük fel, hogy $HE \in \mathcal{L}_3$. A Kis Bar-Hillel lemma alapján ekkor létezik $n = n(L)$. Vegyük a következő szót: $u := (^n)^n$, és legyen $u' = (^n$. Mivel $l(u') \geq n$, így alkalmazhatjuk a Kis Bar-Hillel lemmát.

Ezek szerint u'-ben létezik v nemüres, beiterálható részszó. Legyen $v := (^d$, ahol $d > 0$. A lemma szerint $(^{n-d-k}\{(^d\}^i(^k)^n = (^{n+(i-1)d})^n$ eleme a HE halmaznak. (Az előző képletben „{}" metazárójelek!) Ez viszont $i \neq 1$ esetén HE definíciója miatt nem teljesül, tehát ellentmondásra jutottunk. □

Következmény:

A programozási nyelvek szintaxisa nem írható le tisztán 3-as típusú nyelvtannal. Viszont a programozási nyelvek alapszimbólumainak (azonosítók, konstansok, alapszavak, elhatároló jelek) feldolgozására már használható automata, a fordítóprogramokban ez a lexikális elemző (scanner).

Véges, determinisztikus automata esetén kiterjeszthetjük a δ függvényt nemcsak egy jelből álló u szóra, hiszen minden $a \in A$ és $u \in T^*$ esetén létezik pontosan egy $b \in A$, amelyre $[a, u] \overset{*}{\underset{\mathcal{A}}{\vdash}} [b, \varepsilon]$.

3.16. definíció. *A kiterjesztett $\hat{\delta}$ függvényt a következő módon definiáljuk:*

$$\hat{\delta}(a, u) := b \Longleftrightarrow [a, u] \overset{*}{\underset{\mathcal{A}}{\vdash}} [b, \varepsilon].$$

Definíció alapján nyilvánvaló összefüggések az alábbiak:

$$\hat{\delta}(a, \varepsilon) = a,$$

$$\hat{\delta}(a, uv) = \hat{\delta}(\hat{\delta}(a, u), v),$$

$$[a, t] \underset{\mathcal{A}}{\vdash} [\hat{\delta}(a, t), \varepsilon], \ a \in A \text{ és } t \in T.$$

A konfigurációátmenet fogalmát felhasználva az utóbbiból $\delta(a,t) = \hat{\delta}(a,t)$. Tehát a $\hat{\delta}$ tényleg valódi kiterjesztése δ függvénynek, így tehát a $\hat{}$ jel elhagyható.

3.17. definíció. *Adott L nyelv $p \in T^*$-ra vonatkozó maradéknyelve L_p, ahol*

$$L_p := \{v \mid pv \in L\}.$$

Tulajdonságai:

$$\big(L_p\big)_q = L_{pq},$$
$$L_\varepsilon = L,$$
$$\varepsilon \in L_p \iff p \in L.$$

3.18. definíció. *Az $\mathcal{A}$ determinisztikus automata $a \in A$ állapotra vonatkozó maradékát jelölje $L_a^{\mathcal{A}}$. Ennek jelentése:*

$$L_a^{\mathcal{A}} := \{v \mid \delta(a,v) \in F\}.$$

Az $L_a^{\mathcal{A}}$ tehát azokat a v szavakat tartalmazza, melyek hatására az automata a-ból végállapotba kerül. Az automata maradéknyelvének tulajdonságai:

$$(L_a^{\mathcal{A}})_q = L_{\delta(a,q)}^{\mathcal{A}} \quad (q \in T^*),$$
$$L_{a_0}^{\mathcal{A}} = L(\mathcal{A}),$$
$$\varepsilon \in L_a^{\mathcal{A}} \iff a \in F.$$

3.19. tétel. Myhill—Nerode tétel: *$L \in \mathcal{L}_3$ akkor és csak akkor, ha $\big|\{L_p\}_{p\in T^*}\big| < \infty$, ahol $T = T(L)$ az L nyelv ábécéje.*

Bizonyítás.

„$\Longleftarrow$”

Konstruálunk egy véges, determinisztikus automatát, melynek az állapothalmaza $\{L_p\}_{p\in T^*}$. Ez a feltétel szerint véges halmaz. Az L_p állapottól azt várjuk, hogy a kezdőállapotból p input elolvasása után ő legyen az aktuális állapot.

Legyen $\mathcal{A}_*^L = \big\langle \{L_p\}_{p\in T^*}, T, \delta, L_\varepsilon, F\big\rangle$, ahol $F := \{L_p \mid \varepsilon \in L_p\}$, és $\delta(L_p,t) := L_{pt}$. Ekkor $\mathcal{A}_*^L$ tulajdonságai:

1) δ jól definiált, azaz $L_p = L_q$ esetén $\delta(L_p,t) = \delta(L_q,t)$.

   — Ugyanis $L_p = L_q \implies (L_p)_t = (L_q)_t \implies L_{pt} = L_{qt} \implies \delta(L_p,t) = \delta(L_q,t)$.

2) Minden $u \in T^*$ esetén $\delta(L_p,u) = L_{pu}$.

   — A δ definíciója segítségével az u szó hosszára vonatkozó indukcióval belátható.

3) $L_p \in F \iff p \in L$.

Már csak azt kell belátni, hogy $L(\mathcal{A}_*^L) = L$. Ez pedig könnyen igazolható, hiszen $u \in L(\mathcal{A}_*^L) \iff \delta(L_\varepsilon,u) \in F \iff L_{\varepsilon u} = L_u \in F \iff u \in L$.

„$\Longrightarrow$”

Legyen $L \in \mathcal{L}_3$. Tudjuk, hogy létezik olyan $\mathcal{A} = \langle A, T, \delta, a_0, F \rangle$ véges, determinisztikus automata, amely az L nyelvet fogadja el, azaz $L = L(\mathcal{A})$. Ekkor biztos, hogy $|\{L_a^{\mathcal{A}}\}_{a \in A}| \leq |A| < \infty$, mert $\mathcal{A}$ véges. Vegyük tetszőleges $u \in T^*$-ra az L_u maradéknyelvet. Ekkor $L_u = \left(L_{a_0}^{\mathcal{A}}\right)_u = L_{\delta(a_0,u)}^{\mathcal{A}}$. Így nyilván fennáll, hogy

$$\{L_u\}_{u \in T^*} \subseteq \{L_a^{\mathcal{A}}\}_{a \in A},$$

amivel a tételt beláttuk.

□

Mivel $\mathcal{A}_*^L$ állapothalmaza $\{L_u\}_{u \in T^*}$, ezért az alábbi következményt kapjuk:
Következmény:

Az $\mathcal{A}_*^L$ automata állapotszáma kisebb vagy egyenlő, mint tetszőleges, az L nyelvhez adott véges, determinisztikus automata állapotszáma.

3.20. definíció. *Valamely $L \in \mathcal{L}_3$-hoz adott minimális állapotszámú véges, determinisztikus automatát L minimális automatájának nevezzük.*

Világos, hogy $\mathcal{A}_*^L$ minimális automatája L-nek. A továbbiakban algoritmust adunk egy $L \in \mathcal{L}_3$ nyelvet elfogadó véges, determinisztikus automata állapotszámának csökkentésére és kimutatjuk, hogy így az izomorfia erejéig egyetlen L-et felismerő minimális automatához jutunk.

Megjegyzés: A scanner programokhoz célszerű minimális automatát használni, mert ezáltal jelentősen csökkenthető a lexikális elemzés tárigénye.

3.1.3. Minimális automata előállítása

Ebben a pontban automata alatt mindig véges, determinisztikus automatát fogunk érteni.

Adott egy $L \in \mathcal{L}_3$ nyelv, mely automatával van megadva. Próbáljunk ebből megkonstruálni egy minimális, L-et felismerő automatát. Egyelőre állapotszám-csökkentő transzformációkat adunk.

1) Összefüggővé alakítás:

Nyilván el lehet hagyni az automata által a felismerés során nem használt állapotokat.

3.21. definíció. *Egy $\mathcal{A}$ automata összefüggő, ha minden $a \in A$ esetén létezik $u \in T^*$ szó, hogy $\delta(a_0, u) = a$.*

Ha egy automata nem összefüggő, elhagyhatjuk a kezdőállapotból nem elérhető állapotokat, hiszen ezeknek amúgy sincs szerepük a nyelv felismerésében. Legyen az így kapott automata $\mathcal{A}_{\text{össz}}$. Ennek állapothalmaza:

$$A_{\text{össz}} = \{a \in A \mid \exists u \in T^* : \delta(a_0, u) = a\}.$$

$A_{\text{össz}}$ most is előállítható az alábbi iterációval:

$$H_0 := \{a_0\},$$

$$H_{i+1} := H_i \cup \{a \mid \exists b \in H_i \wedge \exists t \in T : \delta(b,t) = a\},$$

A H_i halmazok itt is csak bővülhetnek, ezért az A állapothalmaz végessége miatt egy i_0 indextől kezdődően végig megegyeznek, és ekkor $A_{\text{össz}} = H_{i_0}$. Legyen $\mathcal{A}_{\text{össz}} = \langle A_{\text{össz}}, T, \delta\big|_{A_{\text{össz}} \times T}, a_0, F \cap A_{\text{össz}} \rangle$, erre nyilván teljesül, hogy $L(\mathcal{A}_{\text{össz}}) = L(\mathcal{A})$.

2) Redukció:

Legyen $\mathcal{A} = \langle A, T, \delta, a_0, F \rangle$. Két állapot nem megkülönböztethető a nyelvelfogadás szempontjából, ha ugyanazon szavak hatására kerülnek végállapotba. Célunk az, hogy ezeket eggyé összevonjuk úgy, hogy az elfogadott nyelv változatlan maradjon.

3.22. definíció. *Legyenek $a, b \in A$ állapotok. a és b ekvivalensek, ha $L_a^{\mathcal{A}} = L_b^{\mathcal{A}}$. Jelölése: $a \sim b$.*

Tulajdonságai:

— ekvivalenciareláció,

— jobbkongruenciareláció.

3.23. definíció. *Egy $\rho : A \times A$ reláció jobbkongruencia, ha minden $u \in T^*$ esetén $a \rho b \Longrightarrow \delta(a,u) \rho \delta(b,u)$.*

3.24. állítás. *$A \sim$ reláció jobbkongruencia.*

Bizonyítás. Ugyanis $a \sim b \Longrightarrow L_a^{\mathcal{A}} = L_b^{\mathcal{A}} \Longrightarrow \forall u \in T^*\text{-ra } \left(L_a^{\mathcal{A}}\right)_u = \left(L_b^{\mathcal{A}}\right)_u \Longleftrightarrow \forall u \in T^*\text{-ra } L_{\delta(a,u)}^{\mathcal{A}} = L_{\delta(b,u)}^{\mathcal{A}} \Longleftrightarrow \forall u \in T^*\text{-ra } \delta(a,u) \sim \delta(b,u)$.

□

3.25. definíció. *Az „$\sim$” reláció két automata állapotaira vonatkozó kiterjesztése:*

$$a_1 \sim a_2, \text{ ha } L_{a_1}^{\mathcal{A}_1} = L_{a_2}^{\mathcal{A}_2}, \text{ ahol}$$

$\mathcal{A}_1, \mathcal{A}_2$ a két automata, és $a_1 \in A_1, a_2 \in A_2$.

3.26. definíció. *Legyenek $\mathcal{A}_1, \mathcal{A}_2$ két automata $a_0^{(1)}$ és $a_0^{(2)}$ kezdőállapotokkal. $\mathcal{A}_1$ ekvivalens $\mathcal{A}_2$-vel, jelölésben:*

$$\mathcal{A}_1 \sim \mathcal{A}_2, \Longleftrightarrow a_0^{(1)} \sim a_0^{(2)} \ \big(\text{vagyis ha } L(\mathcal{A}_1) = L(\mathcal{A}_2)\big).$$

Annak érdekében, hogy egyszerűbb automatához jussunk, vegyük az $\mathcal{A}$ automata $\sim$-re vonatkozó a faktorautomatáját, jelölésben: $\mathcal{A}/_{\sim}$.

$$\mathcal{A}/_{\sim} := \langle \{C_a\}_{a \in A}, T, \delta', C_{a_0}, \mathcal{F} \rangle,$$

ahol

C_a az a-val ekvivalens állapotok osztálya, melynek reprezentánsa a,

$\mathcal{F} = \{C_a \mid a \in F\}$,

$\delta'(C_a, t) = C_{\delta(a,t)}$, azaz δ'-t egy tetszőleges reprezentánssal definiáljuk.

Az $\mathcal{A}/_{\sim}$ automata tulajdonságai:

1) δ' jól definiált. Ez nyilvánvaló, mert $\sim$ jobbkongruencia.

2) $\delta'(C_a, u) = C_{\delta(a,u)}$, ahol $u \in T^*$.

3) $C_b \in \mathcal{F} \iff b \in F$.

3.27. definíció. *Egy adott $\mathcal{A}$ (véges, determinisztikus) automata redukált automata, ha minden $a, b \in A$ esetén $a \sim b \iff a = b$, azaz nincsenek különböző ekvivalens állapotai.*

3.28. állítás. *Az $\mathcal{A}/_{\sim}$ automata redukált, és $L(\mathcal{A}/_{\sim}) = L(\mathcal{A})$.*

Bizonyítás. $\mathcal{A}/_{\sim}$ automata tulajdonságai alapján $u \in L_a^{\mathcal{A}} \iff \delta(a,u) \in F \iff C_{\delta(a,u)} \in \mathcal{F} \iff \delta'(C_a, u) \in \mathcal{F} \iff u \in L_{C_a}^{\mathcal{A}/\sim}$, tehát beláttuk, hogy tetszőleges $a \in A$ állapotra $L_a^{\mathcal{A}} = L_{C_a}^{\mathcal{A}/\sim}$. Ebből egyrészt $a = a_0$ választásával adódik $L(\mathcal{A}/_{\sim}) = L(\mathcal{A})$, másrészt $\sim$ definíciója alapján fennáll $a \sim C_a$. Innen következik, hogy $\mathcal{A}/_{\sim}$ redukált, ugyanis ha $C_a \sim C_b$, akkor $L_a^{\mathcal{A}} = L_{C_a}^{\mathcal{A}/\sim} = L_{C_b}^{\mathcal{A}/\sim} = L_b^{\mathcal{A}}$ miatt $a \sim b$, és emiatt ugyanazt az osztályt reprezentálják, azaz $C_a = C_b$.

□

Megjegyzés:

Ha az automata a redukció előtt már összefüggő volt, akkor a redukció után is az lesz. Tehát ha adott egy $\mathcal{A}$ automata, akkor abból már tudunk készíteni egy összefüggő, redukált automatát, kisebb állapotszámmal.

Kérdés, hogy ezzel mennyire közelítettünk a minimalitás felé.

3.29. definíció. *Legyenek $\mathcal{A}_i = \langle A_i, T, \delta_i, a_0^{(i)}, F_i \rangle (i = 1, 2)$ véges, determinisztikus automaták. Ekkor $\mathcal{A}_1$ és $\mathcal{A}_2$ izomorfak, ha létezik $\phi : A_1 \longrightarrow A_2$ kölcsönösen egyértelmű ráképezés, melyre a következők teljesülnek:*

1) $\phi\big(a_0^{(1)}\big) = a_0^{(2)}$,

2) $\phi\big(F_1\big) = F_2$,

3) δ-t megőrzi, azaz minden $a_1 \in A_1$ és minden $t \in T$ esetén $\phi\big(\delta_1(a_1, t)\big) = \delta_2\big(\phi(a_1), t\big)$. (Könnyen belátható, hogy ekkor ez minden $u \in T^*$-ra is igaz.)

Jelölés: $\mathcal{A}_1 \cong \mathcal{A}_2$.

3.30. tétel. Izomorfia tétel: *Legyenek $\mathcal{A}_1$ és $\mathcal{A}_2$ összefüggő, redukált és egymással ekvivalens automaták. Ekkor $\mathcal{A}_1 \cong \mathcal{A}_2$.*

Bizonyítás. Az összefüggőségből adódik a következő két állítás:

$$A_i = \{\delta_i(a_0^{(i)}, p) \mid p \in T^*\},$$

$$F_i = \{\delta_i(a_0^{(i)}, p) \mid p \in L(\mathcal{A}_i)\}.$$

Így már könnyű a megfeleltetést megadni:

$$\phi : \delta_1(a_0^{(1)}, p) \longmapsto \delta_2(a_0^{(2)}, p) \text{ minden } p \in T^*\text{-ra.}$$

A ϕ értelmezési tartománya nyilván A_1, értékkészlete A_2. Már csak annak ellenőrzése van hátra, hogy bijektív függvény.

$$\delta_1(a_0^{(1)}, p) = \delta_1(a_0^{(1)}, q) \overset{\text{red}}{\iff} \delta_1(a_0^{(1)}, p) \sim \delta_1(a_0^{(1)}, q) \iff L^{\mathcal{A}_1}_{\delta_1(a_0^{(1)}, p)} = L^{\mathcal{A}_1}_{\delta_1(a_0^{(1)}, q)} \iff \big(L^{\mathcal{A}_1}_{a_0^{(1)}}\big)_p = \big(L^{\mathcal{A}_1}_{a_0^{(1)}}\big)_q \overset{\text{ekv}}{\iff} \big(L^{\mathcal{A}_2}_{a_0^{(2)}}\big)_p = \big(L^{\mathcal{A}_2}_{a_0^{(2)}}\big)_q \iff L^{\mathcal{A}_2}_{\delta_2(a_0^{(2)}, p)} = L^{\mathcal{A}_2}_{\delta_2(a_0^{(2)}, q)} \iff \delta_2(a_0^{(2)}, p) \sim \delta_2(a_0^{(2)}, q) \overset{\text{red}}{\iff} \delta_2(a_0^{(2)}, p) = \delta_2(a_0^{(2)}, q).$$

Tehát ϕ jól definiált és bijektív leképezés. Meg kell még vizsgálni, teljesülnek-e az 1) — 3) feltételek.

$$\phi\big(a_0^{(1)}\big) = \phi\big(\delta_1(a_0^{(1)}, \varepsilon)\big) = \delta_2(a_0^{(2)}, \varepsilon) = a_0^{(2)},$$

$\phi\big(F_1\big) = F_2$, ez F_1, F_2 előállításából következik.

Legyen $\delta_1(a_0^{(1)}, p) \in A_1$ tetszőleges.

$$\phi\big(\delta_1(\delta_1(a_0^{(1)}, p), t)\big) = \phi\big(\delta_1(a_0^{(1)}, pt)\big) = \delta_2(a_0^{(2)}, pt), \text{ és}$$

$$\delta_2\big(\phi(\delta_1(a_0^{(1)}, p)), t\big) = \delta_2\big(\delta_2(a_0^{(2)}, p), t\big) = \delta_2(a_0^{(2)}, pt).$$

□

Következmény:

Mivel egy adott L nyelvhez készített minimális automaták összefüggők és redukáltak (különben csökkenthető lenne az állapotszámuk), ezért a tételből következik, hogy egymással izomorfak. Az is teljesül tehát, hogy bármely L-hez készített automatát összefüggővé téve és redukálva — az izomorfia erejéig — a minimális automatát kapjuk meg.

3.1.4. Állapotok ekvivalenciájának algoritmikus eldöntése

Két állapot ekvivalenciáját egyelőre nem tudjuk a definíció alapján eldönteni, ugyanis

$$a \sim b \iff L^{\mathcal{A}}_a = L^{\mathcal{A}}_b \iff \forall u \in T^* : \big(\delta(a, u) \in F \iff \delta(b, u) \in F\big).$$

Itt T^* végtelen, a probléma a definíció alapján nem megoldható. Megpróbáljuk $\sim$-t approximálni végesen eldönthető relációkkal.

3.31. definíció. *Legyen $\overset{i}{\sim}$, az i-edik ekvivalencia ($i \geq 0$) a következőképpen definiálva:*

$a \overset{i}{\sim} b$ *pontosan akkor, ha minden* $u \in T^{\leq i}$ *esetén* $\big(\delta(a, u) \in F \iff \delta(b, u) \in F\big)$.

Szükségünk lesz még a következő fogalomra is:

3.32. definíció. *Legyenek ρ_1, ρ_2 bináris relációk. Ekkor ρ_1 finomabb, mint ρ_2, ha bármely $a, b \in A$ esetén $a\rho_1 b \implies a\rho_2 b$.*
Jelölése: $\rho_1 \succ \rho_2$.

Világos, hogy ha ρ_1, ρ_2 ekvivalenviarelációkra $\rho_1 \succ \rho_2$, akkor $|\rho_1| \geq |\rho_2|$ (a $|\rho_1|$ és $|\rho_2|$ az ekvivalenciaosztályok száma). Az $\overset{i}{\sim}$ ekvivalencia tulajdonságai:

1) ekvivalenciareláció,

2) minden $a, b \in A$ esetén $a \overset{i+1}{\sim} b \iff a \overset{i}{\sim} b \wedge (\forall t \in T : \delta(a,t) \overset{i}{\sim} \delta(b,t))$,

3) $\overset{0}{\sim} \prec \overset{1}{\sim} \prec \overset{2}{\sim} \prec \ldots \prec \sim$, továbbá $a \sim b \iff \forall i \geq 0 : a \overset{i}{\sim} b$.

A folyamatos finomodás miatt az osztályszámokra fennáll, hogy

$$|\overset{0}{\sim}| \leq |\overset{1}{\sim}| \leq |\overset{2}{\sim}| \leq \ldots \leq |\sim| \leq |A| < \infty.$$

Ekkor viszont létezik olyan $l \in \mathbb{N}$, melyre $|\overset{l}{\sim}| = |\overset{l+1}{\sim}|$. Legyen a legkisebb ilyen szám i_0. Tehát

$$i_0 := \min\{l \mid |\overset{l}{\sim}| = |\overset{l+1}{\sim}|\}.$$

A rekurzív összefüggés alapján nyilván minden $j \geq i_0$ esetén $\overset{j}{\sim} = \overset{i_0}{\sim}$, ugyanis ha $\overset{j}{\sim} = \overset{j+1}{\sim}$, akkor $\overset{j+1}{\sim} = \overset{j+2}{\sim}$, mivel $\overset{j}{\sim} = \overset{j+1}{\sim}$ esetén minden a, b-re $a \overset{j+2}{\sim} b \iff a \overset{j+1}{\sim} b \wedge (\forall t \in T : \delta(a,t) \overset{j+1}{\sim} \delta(b,t)) \iff a \overset{j}{\sim} b \wedge (\forall t \in T : \delta(a,t) \overset{j}{\sim} \delta(b,t)) \iff a \overset{j+1}{\sim} b$.

Tehát az is fennáll, hogy minden $j \geq 0$-ra $\overset{i_0}{\sim} \succ \overset{j}{\sim}$, amiből adódik, hogy $\overset{i_0}{\sim} \succ \sim$. Azaz $\overset{i_0}{\sim}$ és $\sim$ kölcsönösen finomabbak egymásnál, ami csak úgy lehetséges, hogy megegyeznek, tehát $\overset{i_0}{\sim} = \sim$. Az ekvivalencia eldöntését ezzel algoritmikusan végrehajthatóvá tettük.

Milyen becslést tudunk adni i_0-ra? Az nyilvánvaló, hogy $1 \leq |\overset{0}{\sim}|$, i_0 pedig akkor lesz a legnagyobb, ha az osztályok száma a lehető leglassabban nő, azaz egyesével. Tehát tegyük föl, hogy $|\overset{0}{\sim}| \geq 1, |\overset{1}{\sim}| \geq 2, \ldots, |\overset{i_0}{\sim}| \geq i_0 + 1$, ahonnan $i_0 + 1 \leq |A|$. Tehát $i_0 \leq |A| - 1$.

Következmény:

$\sim = \overset{|A|-1}{\sim}$.

3.1.5. Reguláris nyelvek

3.33. definíció. *Reguláris nyelveknek nevezzük az alábbi három tulajdonsággal definiált* $\mathcal{L}_{\mathrm{REG}}$ *nyelvosztály elemeit:*

(i) $\mathcal{L}_{\mathrm{REG}}$ *tartalmazza az elemi nyelveket:* $\emptyset, \{\varepsilon\}, \{t\}$, *ahol* $t \in U$,

(ii) $\mathcal{L}_{\mathrm{REG}}$ *zárt az unió, konkatenáció és a lezárás műveletekre,*

(iii) $\mathcal{L}_{\mathrm{REG}}$ *a legszűkebb olyan nyelvosztály, mely az (i), (ii) feltételeknek megfelel.*

3.34. tétel. Kleene tétele: $\mathcal{L}_{\mathrm{REG}} = \mathcal{L}_3$.

Bizonyítás. Itt is a kétirányú tartalmazást kell belátni:

„$\subseteq$”:

Elég belátni, hogy $\mathcal{L}_3$ rendelkezik az (i), (ii) tulajdonságokkal.

(i) Tudunk adni 3-as típusú nyelvtant az elemi nyelvekhez:

— $\mathcal{P} = \{S \longrightarrow t\}$, ha $L = \{t\}$,

— $\mathcal{P} = \{S \longrightarrow \varepsilon\}$, ha $L = \{\varepsilon\}$,

— $\mathcal{P} = \emptyset$, ha $L = \emptyset$.

(ii) korábban már láttuk, hogy $\mathcal{L}_3$ zárt a reguláris műveletekre.

„$\supseteq$”

Tudjuk, hogy $\mathcal{L}_3 = \mathcal{L}_{\mathrm{DA}}$. Elegendő tetszőleges $\mathcal{A}$ véges, determinisztikus automata esetén belátni, hogy $L(\mathcal{A}) \in \mathcal{L}_{\mathrm{REG}}$.

Számozzuk meg az automata állapotait a következőképpen:

$$\mathcal{A} = \left\langle \{a_1, \ldots, a_n\}, T, \delta, a_1, \left\{a_{i_j}\right\}_{j=1}^{s} \right\rangle.$$

Vezessük be a következő jelölést, ahol $1 \leq i, j \leq n$ és $k \geq 0$:

$L_{i,j}^k := \{u \in T^* \mid [a_i, u] \overset{*}{\underset{\mathcal{A}}{\vdash}} [a_j, \varepsilon]$, és a közben érintett állapotok indexe nem nagyobb k-nál $\}$.

A definíció alapján világos, hogy $L_{i,j}^n = L_{i,j}^{n+1} = \ldots$, hiszen nincs valódi megkötés az érintett állapotokra. A végállapotokra véve az uniót:

$$L(\mathcal{A}) = \bigcup_{j=1}^{s} L_{1,i_j}^n.$$

Ha minden i, j, k-ra igaz lenne a regularitás $\left(L_{i,j}^k \in \mathcal{L}_{\mathrm{REG}}\right)$, akkor $L(\mathcal{A}) \in \mathcal{L}_{\mathrm{REG}}$ igaz volna, hiszen $\mathcal{L}_{\mathrm{REG}}$ zárt az únió műveletére. Az $L_{i,j}^k$-k regularitását külön lemmaként igazoljuk, ezzel téve teljessé a tétel bizonyítását. □

3.35. lemma. *Minden i, j, k esetén $L_{i,j}^k \in \mathcal{L}_{\mathrm{REG}}$.*

Bizonyítás. Teljes indukcióval k-ra vonatkozóan, minden i, j-re párhuzamosan.
$k = 0$: Ekkor nyilván $L_{i,j}^0 = \{t \in T \mid \delta(a_i, t) = a_j\} \cup \Delta_{ij}$, ahol

$$\Delta_{ij} = \begin{cases} \{\varepsilon\}, & \text{ha } i = j, \\ \emptyset, & \text{ha } i \neq j. \end{cases}$$

Következésképp $L_{i,j}^0$ reguláris, mert elemi nyelvek uniója.

$k + 1$: Tegyük fel, hogy k-ig igaz az állítás, tehát $L_{i,j}^k$ reguláris bármely i, j-re. Vizsgáljuk meg $(k + 1)$-re. Két eset lehetséges:

$k \geq n$:

Ebből viszont következik, hogy $L_{i,j}^{k+1} = L_{i,j}^k$, tehát $L_{i,j}^{k+1}$ is reguláris.

$k < n$:

Legyen $u \in L_{i,j}^{k+1}$ tetszőleges. Ekkor az automata a_i állapotából a_j állapotába kerül az u szó hatására úgy, hogy a köztesen érintett állapotok A-beli sorszáma maximum $k + 1$. Két eset lehetséges:

a) Köztesen nem fordul elő az a_{k+1} állapot. Ekkor nyilván $u \in L_{i,j}^k$ is fennáll.

b) $l+1 \geq 1$-szer előfordult köztesen az a_{k+1} állapot. Ekkor osszuk föl az u szót úgy, hogy ezeknél az a_{k+1} állapotoknál állapítjuk meg a részszavak határát. Tehát:

$$a_i \overbrace{\ldots}^{u_1} a_{k+1} \overbrace{\ldots}^{v_1} a_{k+1} \overbrace{\ldots}^{v_2} \ldots \overbrace{\ldots}^{v_l} a_{k+1} \overbrace{\ldots}^{u_2} a_j \,.$$
$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{u}$$

Tudjuk, hogy minden közbülső szakaszon az állapotok sorszáma legfeljebb k, vagyis

$$u_1 \in L^k_{i,k+1}, \quad v_d \in L^k_{k+1,k+1}, \quad u_2 \in L^k_{k+1,j} (d = 1, 2, \ldots, l).$$

Ebből már következik, hogy

$$u = u_1 v_1 \ldots v_l u_2 \in L^k_{i,k+1} \big(L^k_{k+1,k+1}\big)^* L^k_{k+1,j}.$$

Az a) és b) eseteket összefoglalva azt kaptuk, hogy

$$L^{k+1}_{i,j} \subseteq L^k_{i,j} \cup L^k_{i,k+1} \big(L^k_{k+1,k+1}\big)^* L^k_{k+1,j}.$$

A másik irányú „$\supseteq$” tartalmazás már nyilvánvaló, tehát a két nyelv azonos.

$$L^k_{i,j} = L^k_{i,j} \cup L^k_{i,k+1} \big(L^k_{k+1,k+1}\big)^* L^k_{k+1,j}.$$

Az indukciós feltétel szerint a jobb oldalon álló nyelvek regulárisak, így $L^{k+1}_{i,j}$ a reguláris műveletekre való zártság miatt reguláris. Ezzel a lemma bizonyítását befejeztük.

□

3.1.6. A 3-as típusú nyelvek néhány további zártsági tulajdonsága

Tegyük fel, hogy adott egy $\mathcal{A} = \langle A, T, \delta, a_0, F \rangle$ véges, determinisztikus automata. Ez kiterjeszthető tetszőleges $T' \supseteq T$ ábécére. Bővítsük ki az állapothalmazt egy új d zsákutca állapottal, és definiáljuk erre δ-t a következőképp:

$\delta(a, t') := d$ minden $t' \in T' \setminus T$-re és $a \in A$-ra,

$\delta(d, t') := d$ minden $t' \in T'$-re.

A kezdő- és végállapotokat változatlanul hagyva az így kapott $\mathcal{A}'$ automatára nyilván $L(\mathcal{A}') = L(\mathcal{A})$.

3.36. tétel. *Az $\mathcal{L}_3$ nyelvosztály zárt a komplementer, a metszet, a különbség és a szimmetrikus differencia műveletekre.*

Bizonyítás.

Elég belátni az állítást a véges, determinisztikus automaták által felismert nyelvek osztályára.

Komplementer:

A standard jelöléssel adott $\mathcal{A}$ véges, determinisztikus automatához konstruálunk egy olyan $\overline{\mathcal{A}}$ automatát, amelyre $L(\overline{\mathcal{A}}) = \overline{L(\mathcal{A})}$. Legyen $\overline{\mathcal{A}} = \langle A, T, \delta, a_0, A \setminus F \rangle$, és erre teljesül az állítás.

Metszet, különbség, szimmetrikus differencia:

Legyenek $\mathcal{A}_i = \langle A_i, T, \delta_i, a_0^{(i)}, F_i \rangle$, véges, determinisztikus automaták, $i = 1, 2$, ahol feltettük, hogy az input ábécéik azonosak (hiszen különben kiterjeszthetők mindketten az unió ábécére). Legyen $\odot$ tetszőleges művelet a $\cap, \setminus, \triangle$ műveletek közül. Konstruálni kell egy $\mathcal{A}_\odot$ automatát, melyre fennáll, hogy

$$L(\mathcal{A}_\odot) = L(\mathcal{A}_1) \odot L(\mathcal{A}_2).$$

Legyen $\mathcal{A}_\odot = \langle A_1 \times A_2, T, \delta_1 \times \delta_2, (a_0^{(1)}, a_0^{(2)}), F_\odot \rangle$ ún. direkt szorzat automata. $\mathcal{A}_\odot$ működése egyszerű, komponensenként párhuzamosan történik, amit a $\delta_1 \times \delta_2$ jelöléssel fejezünk ki. Formálisan: $(\delta_1 \times \delta_2)\big((a_1, a_2), t\big) = \big(\delta_1(a_1, t), \delta_2(a_2, t)\big)$. Definiálni kell még a végállapotok halmazát:

$$F_\cap := F_1 \times F_2,$$

$$F_\setminus := F_1 \times (A_2 \setminus F_2),$$

$$F_\triangle := \big(F_1 \times (A_2 \setminus F_2)\big) \cup \big((A_1 \setminus F_1) \times F_2\big).$$

□

4. Algoritmikusan eldönthető problémák 3-as típusú nyelveken

Egy adott probléma algoritmikusan eldönthető, ha létezik olyan algoritmus, mely *igen* válasszal áll le, ha a problémának van megoldása, különben pedig *nem* választ ad.

Eldöntési problémák:
I) $u \in L$?, ahol u input szó, és $L \in \mathcal{L}_3$.

Tegyük fel, hogy L egy véges, determinisztikus $\mathcal{A}$ automatájával van megadva. (Ez L egy lehetséges véges leírása.) A döntési eljárás:

1) ha $u \notin T(\mathcal{A})^*$, akkor „nem" a válasz,

2) ha $u \in T(\mathcal{A})^*$, akkor az u szót az $\mathcal{A}$ automata inputjára tesszük, és elindítjuk $\mathcal{A}$-t. Ha az $\mathcal{A}$ automata elfogadja az u szót, akkor a döntési eljárás is igent válaszol, egyébként nemet.

Itt az automatát elegendő „fekete doboz"-ként megadni, ugyanis a döntési eljárás nem használja explicit módon az automata belső szerkezetét, hanem azt mintegy orákulumként működteti.

Az alábbi három feltétel teljesülését viszont elvárjuk az automatánktól:

1) ismert legyen az ábécéje,

2) ismert legyen az állapotainak száma $\big(n(\mathcal{A})$, ez a későbbiekben lesz szükséges$\big)$,

3) legyen egy RESET kapcsoló, amivel alapállapotba tudjuk hozni.

Ezekkel a feltételekkel jónéhány további eldöntési feladat megoldható feketedoboz automatákkal.

II) $L \neq \emptyset$?

Egy kézenfekvő (direkt) módszer, hogy minden $T(\mathcal{A})^*$-ban szereplő szóra megoldjuk az I) eldöntési problémát. Ha valamikor *igen* választ kapunk, akkor *igen* válasszal jelezzük, hogy a nyelv nem üres. Természetesen ez az út nem járható, mert $T(\mathcal{A})^*$ nem véges, s az algoritmus *nem* válasz helyett végtelen ideig működne.

4.1. állítás. $L \neq \emptyset \iff L \cap T(\mathcal{A})^{<n(\mathcal{A})} \neq \emptyset$.

Bizonyítás. „$\Longleftarrow$” irány triviális. „$\Longrightarrow$” irány bizonyítása a kis Bar-Hillel-lemmával történik. Legyen u az L nyelv egy minimális hosszúságú szava. Azt kellene belátni, hogy $l(u) < n(\mathcal{A})$.

Indirekt módon tegyük fel, hogy $l(u) \geq n(\mathcal{A})$. Mivel $L \in \mathcal{L}_3$, ezért igaz a kis Bar-Hillel-lemma, ahol konstansként választhatjuk $n(\mathcal{A})$-t, hiszen bármely L-et felismerő automata állapotszáma megfelelő. Mivel $l(u) \geq n(\mathcal{A})$, ezért létezik $v \neq \varepsilon$ beiterálható részszava. Iteráljuk ezt $i = 0$-szor és jelölje az így kapott L-beli szót u'. Világos, hogy $u' \in L$ és $l(u') = l(u) - l(v) < l(u)$, ami ellentmond az u választásának.

□

Ezzel beláttuk, hogy II) visszavezethető az I)-es eldöntési problémára úgy, hogy az összes $u \in T(\mathcal{A})^{<n(\mathcal{A})}$ szóra döntjük el, hogy $u \in L$ teljesül-e. Ha valamely u-ra I) *igen* választ ad, II)-ben is *igen* választ adunk, egyébként *nemet*.

III) $|L| = \infty$?

Ha ez eldönthető, akkor nyilván $|L| < \infty$ is eldönthető.

4.2. állítás. $|L| = \infty \iff L \cap T(\mathcal{A})^{n(\mathcal{A}) \leq l < 2n(\mathcal{A})} \neq \emptyset$.

Bizonyítás. Az előző állításhoz hasonlóan a kis Bar-Hillel-lemmával történhet, amit az olvasóra bízunk.

□

Ezzel III)-t visszavezettük I)-re úgy, hogy az összes $u \in T(\mathcal{A})$, ahol $n(\mathcal{A}) \leq l(u) < 2n(\mathcal{A})$ szóra döntjük el, $u \in L$ teljesül-e. Ha valamely u-ra *igen* választ ad, III)-ban is *igen* választ adunk, egyébként *nemet*.

IV) $L_1 \subseteq L_2$?

Világos, hogy $L_1 \subseteq L_2 \iff L_1 \setminus L_2 = \emptyset$. Ezzel a feladatot visszavezethetjük II)-re. (A zártsági tételt felhasználva $L_1 \setminus L_2 \in \mathcal{L}_3$.)

Probléma: Az $\mathcal{A}_1, \mathcal{A}_2$ automaták külön-külön adottak, most pedig a különbségautomatára van szükségünk. Nyilván az előző zártsági tételt felhasználva megkonstruálható a direkt szorzat automata: $\mathcal{A}_\setminus$. Ennek végállapotai: $F_1 \times (A_2 \setminus F_2)$ és $n(\mathcal{A}_\setminus) = n(\mathcal{A}_1)n(\mathcal{A}_2)$.

Kérdés: hogyan lehetne $\mathcal{A}_\setminus$-t mint fekete dobozt előállítani az $\mathcal{A}_1, \mathcal{A}_2$ fekete dobozokból? Szimulálással. Az $\mathcal{A}_1, \mathcal{A}_2$-t egyszerre indítjuk el ugyanazzal az input szóval, és ha mindkettő leállt, megvizsgáljuk az eredményt. Ha $\mathcal{A}_1$ igent adott, $\mathcal{A}_2$ pedig nemet, akkor *igent* adjon vissza a szimulált különbség automata, egyébként *nemet*.

V) $L_1 = L_2$?

Ez is visszavezethető II)-re, ugyanis $L_1 = L_2 \iff L_1 \triangle L_2 = \emptyset$. Az $\mathcal{A}_\triangle$ automata az előző feladathoz hasonló módon szimulálható.

5. 2-es típusú nyelvek

5.1. definíció. *Legyen* $G = \langle T, N, \mathcal{P}, S \rangle$ *tetszőleges kiterjesztett 2-es típusú nyelvtan. A* t *nemüres fát* G *feletti szintaxisfának nevezzük, ha megfelel a következő tulajdonságoknak:*

1) Pontjai $T \cup N \cup \{\varepsilon\}$ elemeivel vannak címkézve.

2) Belső pontjai N elemeivel vannak címkézve.

3) Ha egy belső pont címkéje X, a közvetlen leszármazottjainak címkéi pedig balról jobbra olvasva $X_1, X_2, \ldots, X_k$, akkor $X \longrightarrow X_1 X_2 \ldots X_k \in \mathcal{P}$.

4) Az ε-nal címkézett pontoknak nincs testvére.

Szintaxisfákkal a levezetések szerkezetét ábrázoljuk. Jelölje egy adott t szintaxisfa leveleinek balról jobbra való összeolvasását $\mathrm{front}(t)$, a fa gyökerét pedig $\mathrm{gy}(t)$.

5.2. lemma. *Ha* t G *feletti szintaxisfa, akkor* $\mathrm{gy}(t) \xrightarrow[G]{*} \mathrm{front}(t)$.

Bizonyítás. A fa magasságára vonatkozó indukcióval.

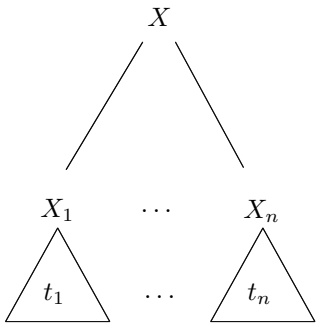
$h = 0$ magasság:

Ekkor a t szintaxisfa csak egy pontból állhat, mely esetben $\mathrm{gy}(t) = \mathrm{front}(t)$, a levezetés pedig reflexív reláció.

$h + 1$ magasság:

Tegyük fel, hogy legfeljebb h magasságra igaz az állítás. Lássuk be, hogy $h + 1$ magasságra is igaz.

Legyen $h(t) = h + 1$. Ekkor biztosan van legalább két szintje. Legyen a gyökér $X \in N$-nel, míg közvetlen leszármazottai balról jobbra $X_1, \ldots, X_n$-nel címkézve. Ha t G feletti szintaxisfa volt, akkor az $X_1, \ldots, X_n$ címkéjű pontok, mint gyökérpontok által meghatározott $t_1, t_2, \ldots, t_n$ részfák is G feletti szintaxisfák, melyekre $h(t_1), \ldots, h(t_n) \leq h$. Alkalmazzuk az indukciós feltételt a t_i részfákra, tehát $X_i \xrightarrow[G]{*} \mathrm{front}(t_i)$, minden $i = 1, \ldots, n$ esetén.



Továbbá tudjuk a G feletti szintaxisfa definíciójából, hogy $X \longrightarrow X_1 \dots X_n \in \mathcal{P}$, s így megkonstruálhatjuk a következő levezetést:

$$\mathrm{gy}(t) = X \underset{G}{\longrightarrow} X_1 \dots X_n \overset{*}{\underset{G}{\longrightarrow}} \mathrm{front}(t_1) X_2 \dots X_n \overset{*}{\underset{G}{\longrightarrow}} \dots$$

$$\dots \overset{*}{\underset{G}{\longrightarrow}} \mathrm{front}(t_1) \dots \mathrm{front}(t_n) = \mathrm{front}(t).$$

□

Az előbbiekben a t G feletti szintaxisfához konstruált levezetés speciális tulajdonságú. Ha ugyanis a levezetés folyamán valamely pozición (a mondatforma i. betűjén) helyettesítés történik, akkor a korábbi pozíciókat $(1, ..., i-1)$ a levezetés már nem érinti, azok változatlanul maradnak.

5.3. definíció. *Az előbb leírt tulajdonságú levezetéseket legbal levezetésnek hívjuk. Analóg módon adhatjuk meg a legjobb levezetés fogalmát is.*

Jelölés: A legbal, illetve legjobb levezetésekre a továbbiakban a
$A \overset{*}{\underset{G,\mathrm{lb}}{\longrightarrow}} \alpha$, illetve $A \overset{*}{\underset{G,\mathrm{lj}}{\longrightarrow}} \alpha$ jelöléseket fogjuk használni.

Világos, hogy terminális szót eredményező legbal, illetve legjobb levezetés esetében a helyettesítés helye minden mondatformában csak annak első (legbal), illetve utolsó (legjobb) nyelvtani jelénél lehet.

A terminális szavakat eredményező legbal, illetve a legjobb levezetésekhez kapcsolódva bevezetjük az elérhető mondatformák két speciális esetét, a *legbal*, illetve *legjobb* mondatformát.

5.4. definíció. *Valamely $L(G)$-beli szó legbal (legjobb) levezetése során előforduló mondatformákat legbal (legjobb) mondatformának nevezzük.*

Az előbb definiált fogalmakra az elemzési módszerek tárgyalásánál lesz szükségünk.

Következmény:

Minden G feletti t szintaxisfa esetén $\mathrm{gy}(t) \xrightarrow[G,\mathrm{lb}]{*} \mathrm{front}(t)$. (Természetesen legjobb levezetés is létezik.)

5.5. lemma. *Legyen $X \in T \cup N \cup \{\varepsilon\}$ tetszőleges. Ha $X \xrightarrow[G]{*} \alpha$, akkor létezik G feletti t szintaxisfa, amelyre* $\mathrm{gy}(t) = X$, *és* $\mathrm{front}(t) = \alpha$.

Bizonyítás. Levezetés hossza szerinti indukcióval.

$h = 0$ hosszú levezetésre:

Ekkor tehát $X \xrightarrow[G]{(0)} \alpha$, s ez definíció szerint azt jelenti, hogy $X = \alpha$, és ekkor az X-szel címkézett egyetlen pontból álló fa megfelelő.

$h + 1$ hosszú levezetésre:

Tegyük fel, hogy a legfeljebb h hosszú levezetésekre igaz az állítás és bizonyítsuk be $(h+1)$-re. Legyen $X \xrightarrow[G]{(h+1)} \alpha$. Ekkor természetesen létezik legelső lépés ($h \geq 0$). Írjuk ki ezt a legelső lépést:

$$X \xrightarrow[G]{} X_1 X_2 \dots X_k \xrightarrow[G]{(h)} \alpha.$$

Mivel a G nyelvtan 2-es típusú, ezért a függetlenségi lemma alapján α felbontható az $\alpha = \alpha_1 \dots \alpha_k$ alakra úgy, hogy $X_i \xrightarrow[G]{(\leq h)} \alpha_i, i = 1, \dots, k$. Alkalmazva az indukciós feltételt ezekhez léteznek t_i szintaxisfák, amikre $\mathrm{gy}(t_i) = X_i$ és $\mathrm{front}(t_i) = \alpha_i$. Tehát megkonstruálható a következő fa:

a) $k = 0$ esetén az $X \longrightarrow \varepsilon$ szabályt alkalmaztuk, tehát a keresett szintaxisfa az X nyelvtani jellel címkézett gyökérből és egy ε-nal címkézett levélből áll.

b) $k \geq 1$ esetén létrehozunk egy X jellel címkézett csúcsot, és ehhez kapcsoljuk hozzá balról jobbra a $t_1, \dots, t_k$ szintaxisfák gyökereit. Ezt megtehetjük, mert $X \longrightarrow X_1 X_2 \dots X_k \in \mathcal{P}$.

Világos,hogy mindkét esetben $\mathrm{gy}(t) = X$ és $\mathrm{front}(t) = \alpha$.

□

Következmény:

$X \xrightarrow[G]{*} \alpha \iff X \xrightarrow[G,\mathrm{lb}]{*} \alpha \iff X \xrightarrow[G,\mathrm{lj}]{*} \alpha \iff$ létezik G feletti t szintaxisfa, amelyre $\mathrm{gy}(t) = X$ és $\mathrm{front}(t) = \alpha$.

Ezt felhasználva az $L(G)$-re vonatkozó szóproblémát úgy is el lehet dönteni, hogy megpróbálunk megkonstruálni egy olyan G feletti t szintaxisfát, melyre $\mathrm{gy}(t) = S$, és $\mathrm{front}(t) = u$. Ezt a folyamatot nevezzük elemzésnek, míg t-t az u szintaxisfájának. A szóprobléma eldöntésének a szintaxisfa konstrukcióján alapuló módszere jól használható a programnyelvek elemzéséhez, ugyanis az eljárás így az elemzendő szó szerkezetét is megadja.

A gyakorlati problémák során a szavaknak jelentést tulajdonítunk, amit a levezetés szerkezete alapján adunk meg. Ehhez kapcsolódóan fontos kérdés azt vizsgálni, hogy egy adott szónak hány lényegesen különböző szerkezetű levezetése, szintaxisfája lehet. Ha ugyanis több, egymástól különböző szintaxisfát lehet konstruálni egy adott szóhoz, akkor esetleg több különböző jelentést is tulajdoníthatnánk neki, ami például egy program esetén általában nem kívánatos.

5.6. definíció. *Egy adott $G \in \mathcal{G}_{\text{kit2}}$ nyelvtan egyértelmű, ha minden $u \in L(G)$ szónak pontosan egy szintaxisfája létezik.*

5.7. definíció. *Az $L \in \mathcal{L}_2$ nyelv egyértelmű, ha van hozzá olyan $G \in \mathcal{G}_{\text{kit2}}$ nyelvtan, mely egyértelmű.*

5.8. definíció. *Az $L \in \mathcal{L}_2$ nyelv lényegesen nem egyértelmű, ha az adott nyelvhez nem létezik egyértelmű nyelvtan.*

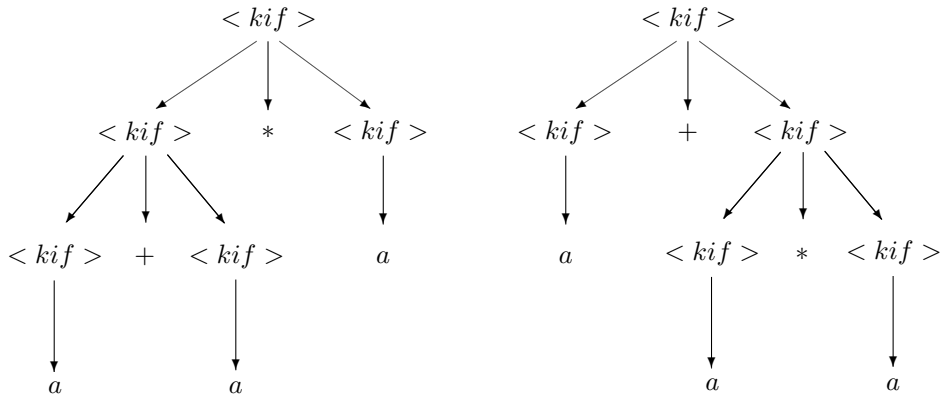
1. példa: Nem egyértelmű nyelvtanra: Legyen a az azonosítókat jelölő terminális jel, és G_1 pedig az alábbi nyelvtan:

$T := \{a, +, *\}$,

$N := \{< kif >\}$,

$\mathcal{P} := \{< kif > \longrightarrow < kif > + < kif > | < kif > * < kif > | \ a\}$.

Nézzük meg az $u = a + a * a$ szót. Ehhez két különböző szintaxisfa is adható:



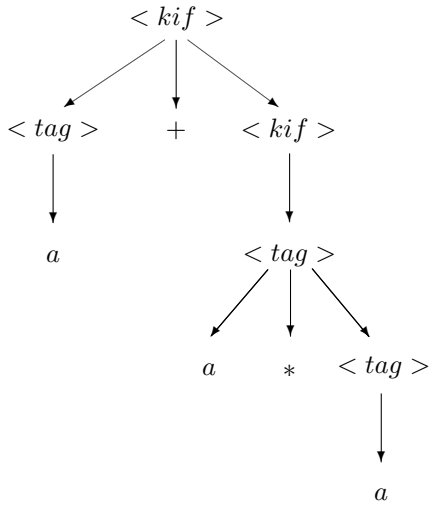
$a + a * a$ két különböző szintaxisfája $\mathcal{G}_1$-ben

Tehát a nyelvtan nem egyértelmű. Ugyanakkor az általa generált nyelv mégis egyértelmű, mert tudunk adni hozzá egyértelmű nyelvtant. Legyen G_2 nyelvtan szabályrendszere a következő:

$< kif > \longrightarrow < tag > | < tag > + < kif >$

$< tag > \longrightarrow a \ | \ a* < tag >$

Ugyanezen $u = a + a * a$ szóhoz már csak egy szintaxisfa létezik:



$a + a * a$ egyértelmű $\mathcal{G}_2$ feletti szintaxisfája

2. példa: Lényegesen nem egyértelmű nyelvre:

$$L := \{a^n b^n c^m \mid n, m \geq 0\} \cup \{a^m b^n c^n \mid n, m \geq 0\}.$$

Könnyű 2-es típusú egyértelmű nyelvtant adni az unió mindkét tagjára külön-külön, de magára az unióra már nem, mert az $a^n b^n c^n$ alakú szavak többségének minden L-hez készített nyelvtanban bizonyíthatóan lesz két különböző szintaxisfája. (A bizonyítás terjedelmi okok miatt meghaladja a könyvünk kereteit.)

5.9. tétel. Nagy Bar-Hillel-lemma: *Minden $L \in \mathcal{L}_2$ esetén léteznek $p, q > 0$ nyelvfüggő egész konstansok ($p = p(L), q = q(L)$), amelyekre ha $u \in L$, és $l(u) > p$, akkor u-nak létezik $u = xyzvw$ felbontása, ahol $l(yv) > 0$, $l(yzv) \leq q$ és minden $i \geq 0$ egészre $xy^i zv^i w \in L$.*

Kevésbé formálisan a lényeget a következőképpen fejezhetjük ki: L minden elég hosszú szavában van két, egymáshoz közel lévő, nem triviális, párhuzamosan beiterálható részszó.

Bizonyítás.

Mivel $L \in \mathcal{L}_2$, ezért létezik hozzá G Chomsky-normálformájú nyelvtan, melyre $L(G) = L$.

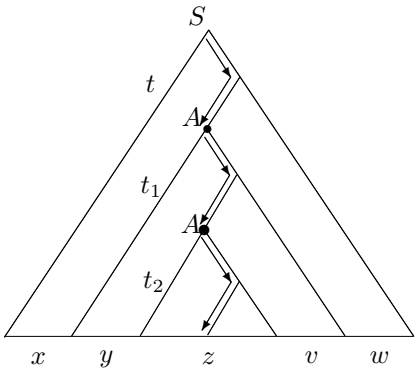
Legyen $G = \langle T, N, \mathcal{P}, S \rangle$ ez a Chomsky-nyelvtan. (Szabályai $A \longrightarrow BC, A \longrightarrow a, S \longrightarrow \varepsilon$ alakúak.) A konstansokat definiáljuk a következő módon: $p := 2^{|N|-1}$ és $q := 2^{|N|}$. Ezek a konstansok függnek a nyelvtől, de a nyelv szavaitól már nem.

Legyen $u \in L$ olyan, hogy $l(u) > p$. Tudjuk, hogy u-hoz létezik olyan G feletti t szintaxisfa, melyre $\mathrm{gy}(t) = S$ és $\mathrm{front}(t) = u$. Tudjuk továbbá, hogy t bináris fa, mert G Chomsky-normálformájú. Ha elhagyjuk leveleit, a maradék fának ugyanannyi levele lesz. (Az $A \longrightarrow a$ alakú szabályok miatt.) Jelöljük ezt a maradék fát $\mathrm{red}(t)$-vel.

Becsüljük meg t magasságát $\mathrm{front}(t)$ hossza segítségével:

$$h(t) = 1 + h(\mathrm{red}(t)) \geq 1 + \log_2 l\big(\mathrm{front}(\mathrm{red}(t))\big) = 1 + \log_2 l(\mathrm{front}(t)) = 1 + \log_2 l(u).$$

De $\log_2 l(u) > \log_2 2^{|N|-1} = |N| - 1$. Ebből az következik, hogy $h(t) > |N|$.



A szintaxisfa egyik leghosszabb útja

Ezért t-ben a leghosszabb úton legalább $|N| + 2$ darab pont van. Ebből egy terminális jellel, a maradék $|N| + 1$ pont nyelvtani jellel van címkézve. Tehát biztosan van legalább egy ismétlődő nyelvtani jel. Válasszuk ki a leghosszabb úton alulról az először ismétlődő nyelvtani jel párt. Legyen ez az ismétlődő nyelvtani jel A és vegyük azt a két részfát, melyekben az A nyelvtani jel a gyökér (az ábrán a felső A-hoz tartozó fát t_1-gyel, az alsó A-hoz tartozó részfát t_2-vel jelöltük). Ezek front(t)-t, tehát u-t, öt részre osztják szét, melyeket rendre x, y, z, v, w-vel jelöltünk. Ekkor:

a) $S \xrightarrow[G]{*} xAw$, a $t - t_1$ szintaxisfából,

b) $A \xrightarrow[G]{*} yAv$, a $t_1 - t_2$ szintaxisfából,

c) $A \xrightarrow[G]{*} z$, a t_2 szintaxisfából.

Alkalmazzuk egyszer az a) levezetést, majd i-szer a b) levezetést, végül ismét egyszer a c) levezetést:

$S \xrightarrow[G]{*} xAw \xrightarrow[G]{*} xy^iAv^iw \xrightarrow[G]{*} xy^izv^iw$.

Ez pedig valóban azt jelenti, hogy $xy^izv^iw \in L(G)$.

Be kell még látnunk, hogy $l(yv) > 0$. Nézzük meg a b) levezetést. Ha $l(yv) = 0$ lenne, az azt jelentené, hogy a G nyelvtanban $A \xrightarrow[G]{(k)} A$ teljesülne valamely $k > 0$-ra. Ez viszont a Chomsky-tulajdonságnak ellentmond, azaz $l(yv) > 0$.

Bizonyítani kell még, hogy $l(yzv) \leq q$. Világos, hogy a t fában leghosszabb út t_1-re eső szakasza t_1-ben a leghosszabb, ezért t_1 szintszámára a következő becslést adhatjuk:

$$\text{sz}(t_1) \leq |N| + 1 + 1,$$

ahol sz(t_1) a t_1 fa szintjeinek száma, $|N|$ pedig a különböző nyelvtani jelek száma. (Az első $+1$ az A ismétlése miatt, a második $+1$ pedig az út végén levő terminális jel miatt szerepel a felső becslésben. Tehát

$h(t_1) \leq |N| + 1.$

Ebből következik, hogy

$$|N| \geq h(t_1) - 1 = h(\mathrm{red}(t_1)) \geq \log_2 l\big(\mathrm{front}(\mathrm{red}(t_1))\big) = \log_2 l(\mathrm{front}(t_1)) = \log_2 l(yzv).$$

Tehát összegezve megkaptuk, hogy

$$q = 2^{|N|} \geq l(yzv).$$

□

Mivel a nagy Bar-Hillel-lemma csak egy szükséges feltételt ad a 2. típusú nyelvekre, ezért arra használhatjuk, hogy kimutassuk vele, hogy valamely adott nyelv nem 2-es típusú.

Példa:

Legyen T olyan ábécé, amelyre $|T| \geq 2$, és két különböző jele legyen t_1, t_2. Legyen D az úgynevezett dadogós szavak nyelve, tehát $D = \{uu \mid u \in T(\mathcal{A})^*\}$. (Ez a nyelv a programozási nyelvek deklaráció fogalmának modellje, ahol az első u a deklarációt, a második u a használatot jelenti.)

5.10. állítás. $D \notin \mathcal{L}_2$.

Bizonyítás. Indirekt módon. Tegyük fel, hogy $D \in \mathcal{L}_2$. Ekkor a nagy Bar-Hillel lemma alapján léteznek a p, q nyelvfüggő konstansok. Legyen $M := \max\{p, q\}$. Vizsgáljuk az $\alpha = t_1^M t_2^M t_1^M t_2^M \in D$ szót. Nyilván $l(\alpha) > p$, tehát léteznie kell két, párhuzamosan beiterálható, együtt nemüres részszónak, amire a kapott új szavak ismét benne vannak a nyelvben.

Végezzük el az $i = 0$-ra az iterációt, azaz hagyjuk el a megfelelő részszavakat. Legyen az iteráció eredménye $t_1^{m_1} t_2^{m_2} t_1^{m_3} t_2^{m_4} \in D$. De ekkor valamely $1 \leq j \leq 4$-re $m_j < M$, mert nemüres részszót hagytunk el. Tegyük fel, hogy t_1-ből hagytunk el (a bizonyítás hasonló módon történik t_2-re is). Ekkor viszont, $t_1^{m_1} t_2^{m_2} t_1^{m_3} t_2^{m_4} \in D$ miatt $m_1 = m_3 < M$. Ebből következik, hogy az yzv részszó mindenképpen lefedi az első t_2-es blokkot, és belemetsz mindkét t_1-be, legalább egy-egy jellel. Emiatt $l(yzv) \geq M + 2$ lenne, ami ellentmond a nagy Bar-Hillel-lemma alapján fennálló $l(yzv) \leq q \leq M$ becslésnek.

□

Következmény:

A programozási nyelvek szintaxisa nem írható le tisztán 2-es típusú nyelvtanok segítségével, hiszen a deklarációknak az előbb adott, dadogós szavakat tartalmazó legegyszerűbb modellje sem írható már le 2-es típusú nyelvtannal.

5.1. Algoritmikusan eldönthető problémák 2-es típusú nyelvekre

5.11. tétel. *Legyen L egy 2-es típusú nyelv. Az alábbi problémák algoritmikusan eldönthetők:*

I) $u \in L$,

II) $L \neq \emptyset$,

III) $|L| = \infty$.

Bizonyítás.
I) Beláttuk 1-es típusú nyelvekre, hogy $\mathcal{L}_1 \subseteq \mathcal{L}_{\text{Rek}}$ és $\mathcal{L}_2 \subseteq \mathcal{L}_1$.

II) Ez visszavezethető a nagy Bar-Hillel-lemmával könnyen belátható ($L \neq \emptyset \iff L \cap T(L)^{\leq p} \neq \emptyset$) összefüggés alapján az I) problémára, ahol p a nagy Bar-Hillel-lemma megfelelő konstansa.

III) Ez is visszavezethető az ($|L| = \infty \iff L \cap T(L)^{p<i\leq p+q} \neq \emptyset$) összefüggés alapján az I) problémára, ahol a p, q konstansok a nagy Bar-Hillel-lemma konstansai. □

Kérdés: Miért nem oldható meg a 3. típusnál látott módon az $L_1 \subseteq L_2$, illetve az $L_1 = L_2$ probléma $\mathcal{L}_2$-n? Erre ad választ a következő tétel.

5.12. tétel. *$\mathcal{L}_2$ nem zárt a komplementer, a metszet, a különbség és a szimmetrikus differencia műveletekre.*

Bizonyítás.
Metszet:

Vizsgáljuk meg a következő nyelveket:

$$L_1 = \{a^n b^n c^m \mid n, m > 0\},$$

$$L_2 = \{a^m b^n c^n \mid n, m > 0\}.$$

Könnyen látszik, hogy $L_1, L_2 \in \mathcal{L}_2$, de ugyanakkor

$$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\} \notin \mathcal{L}_2.$$

Ezt a nagy Bar-Hillel-lemma segítségével láthatjuk be, hasonlóan ahhoz, ahogy azt a dadogós szavak esetében tettük.

Komplementer:

Indirekt módon. Tegyük fel, hogy $\mathcal{L}_2$ zárt a komplementerre. A De Morgan-azonosság alapján $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. Mivel $\mathcal{L}_2$ az unióra zárt, ezért a komplementerre való zártságból következne a metszetre való zártsága, amiről az imént kimutattuk, hogy nem teljesül.

Különbség, szimmetrikus differencia:

Indirekt módon. Ha különbségre, illetve szimmetrikus differenciára zárt lenne, akkor tetszőleges L mellett tartalmazná $T(L)^* \setminus L$-et, illetve $T(L)^* \triangle L$-et, de ezek mindegyike $\overline{L}$, s ezért zárt lenne a komplementerre is.

□

5.2. A veremautomaták és 2-es típusú nyelvek kapcsolata

A továbbiakban 1-vermekkel fogunk foglalkozni, ezért a vermek számát mutató előtagot elhagyva csak veremautomatákról beszélünk. Legyen $\mathcal{V} = \langle A, T, \Sigma, \delta, a_0, \sigma_0, F \rangle$ veremautomata. Emlékeztetünk a végállapottal, illetve üres veremmel elfogadott nyelv definíciójára.

A $\mathcal{V}$ veremautomata által végállapottal elfogadott nyelv:

$$L^{\mathrm{Fin}}(\mathcal{V}) = \{u \in T^* \mid [a_0, u, \sigma_0] \overset{*}{\underset{\mathcal{V}}{\vdash}} [f, \varepsilon, \alpha] \text{ valamely } f \in F\text{-re}\}.$$

A $\mathcal{V}$ veremautomata által üres veremmel elfogadott nyelv:

$$L^{\varepsilon}(\mathcal{V}) = \{u \in T^* \mid [a_0, u, \sigma_0] \overset{*}{\underset{\mathcal{V}}{\vdash}} [b, \varepsilon, \varepsilon] \text{ valamely } b \in A\text{-re}\}.$$

Üres veremmel történő elfogadásnál az automata definíciójában a végállapotok halmazát nem szokták jelölni.

Jelölések: $\mathcal{L}_{\mathcal{V}}^{\mathrm{Fin}}$ a végállapottal elfogadott nyelvek osztálya. $\mathcal{L}_{\mathcal{V}}^{\varepsilon}$ az üres veremmel elfogadott nyelvek osztálya.

5.13. lemma. *Tetszőleges $\mathcal{V}$ veremautomatához létezik olyan $\tilde{\mathcal{V}}$ veremautomata, melyre $L^{\varepsilon}(\tilde{\mathcal{V}}) = L^{\mathrm{Fin}}(\mathcal{V})$.*

Bizonyítás.

$\tilde{\mathcal{V}}$-nek szimulálnia kell $\mathcal{V}$-t, továbbá el kell kerülnie, hogy elutasítandó szavakra idő előtt ürüljön ki a verem, valamint $\mathcal{V}$ végállapotba kerülése esetén (és csak akkor) ki kell ürítenie a vermet.

Legyen

$$\mathcal{V} = \langle A, T, \Sigma, \delta, a_0, \sigma_0, F \rangle,$$

standard jelöléssel adott veremautomata. Az őt szimuláló, üres veremmel elfogadó $\tilde{\mathcal{V}}$ veremautomata formálisan a következő alakú:

$$\tilde{\mathcal{V}} = \langle A \cup \{a_0', d\}, T, \Sigma \cup \{\sigma_0'\}, \delta', a_0', \sigma_0', \ \rangle, \text{ ahol}$$

a_0' új kezdőállapot, σ_0' új veremkezdő szimbólum, mely $\tilde{\mathcal{V}}$ utolsó lépéséig a verem alján marad, d az ürítőállapot, δ' megegyezik δ-val, az alábbi kiegészítő tevékenységekkel:

— $\delta'(a_0', \varepsilon, \sigma_0') = (a_0, \sigma_0\sigma_0')$,

— minden $f \in F, \sigma \in \Sigma \cup \{\sigma_0'\}$ esetén $(d, \varepsilon) \in \delta'(f, \varepsilon, \sigma)$, és minden $\sigma \in \Sigma \cup \{\sigma_0'\}$-re $\delta'(d, \varepsilon, \sigma) = (d, \varepsilon)$.

Ekkor nyilván $L^{\varepsilon}(\tilde{\mathcal{V}}) = L^{\mathrm{Fin}}(\mathcal{V})$.

□

5.14. lemma. *Tetszőleges $\mathcal{V}$ veremhez létezik olyan $\tilde{\mathcal{V}}$ veremautomata, melyre $L^{\mathrm{Fin}}(\tilde{\mathcal{V}}) = L^{\varepsilon}(\mathcal{V})$.*

Bizonyítás.

Ez előző lemma megfordítása, hasonlóan is bizonyítjuk. Konstruálunk egy $\tilde{\mathcal{V}}$ veremautomatát, mely alapvetően $\mathcal{V}$-t szimulálja. Fel kell venni egy új veremkezdő jelet, hogy $\mathcal{V}$ elfogadó működése esetén $\tilde{\mathcal{V}}$ verme ki ne ürüljön, továbbá kell még két új állapot, az új kezdőállapot, illetve az új végállapot. Amikor $\mathcal{V}$ elfogad egy szót (eredetileg kiürült a verem),

akkor $\tilde{\mathcal{V}}$-nek az új veremkezdő jel hatására végállapotba kell mennie. δ' tehát itt is megegyezik δ-val az alábbi kiegész´ő tevékenységekkel:

$\delta'(a'_0, \varepsilon, \sigma'_0) = (a_0, \sigma_0\sigma'_0)$,

$\delta'(b, \varepsilon, \sigma'_0) = (a_{\text{vég}}, \sigma'_0)$, minden $b \in A$ esetén.

A végállapotok halmaza $F' = \{a_{\text{vég}}\}$.

□

Az előző két lemmát összefoglalva az alábbi tételt kapjuk.

5.15. tétel. $\mathcal{L}_{\mathcal{V}}^{Fin} = \mathcal{L}_{\mathcal{V}}^{\varepsilon}$.

Mivel az új elfogadási mód nem bővítette a felismert nyelvek körét, ezért a továbbiakban $\mathcal{L}_{\mathcal{V}}$-t fogunk használni mindkettőjükre.

5.16. definíció. *Legyen $G \in \mathcal{G}_{\text{kit2}}$ tetszőleges nyelvtan. Legbal, kezdőszelet-egyeztetéses elemzésnek hívjuk azt az elemzési módszert, mely során egy $u \in T^*$ szó egy S-ből induló legbal levezetését konstruáljuk meg a levezetés során kialakult mondatforma terminális jelekből álló kezdőszeletének az eredeti u szó kezdőszeleteivel való egyeztetésével. (Természetesen ez csak az $u \in L(G)$ szavak esetén vezethet sikerre.)*

Példa:

$$\text{S} \longrightarrow \text{SS} \mid a\text{S}b \mid b\text{S}a \mid ab \mid ba.$$

Legyen $u = abba$. Ekkor egy levezetés például $\text{S} \longrightarrow \text{SS} \longrightarrow ab\text{S}$. Itt az ab szó megegyezik u szó kezdőszeletével, tehát ez egy legbal, kezdőszelet-egyeztetéses elemzés kezdete lehet. Ha a továbbiakban nincs egyezés, más alternatívát kell keresni. Világos, hogy $u \in L(G) \iff$ ha u-nak van legbal, kezdőszelet egyeztetéses elemzése G-ben.

5.17. tétel. $\mathcal{L}_{\mathcal{V}} = \mathcal{L}_2$.

Bizonyítás. „$\supseteq$”:

Ehhez elég belátni, hogy tetszőleges $G = \langle T, N, \mathcal{P}, S \rangle \in \mathcal{G}_{\text{kit2}}$ esetén létezik olyan $\mathcal{V}$ verem, melyre $L^{\varepsilon}(\mathcal{V}) = L(G)$.

Olyan $\mathcal{V}$ veremautomatát készítünk, mely a veremben egy legbal, kezdőszelet-egyeztetéses elemzést végez. Legyen $\mathcal{V} = \langle \{a_0\}, T, T \cup N, \delta, a_0, S \rangle$. Kétféle szabálytípus lesz. Az első típus ε-mozgást végző szabályok összessége, melyek a legbal levezetést végzik:

$$(a_0, p) \in \delta(a_0, \varepsilon, B) \iff B \longrightarrow p \in \mathcal{P} \quad \big(p \in (T \cup N)^*\big).$$

(A verem teteje a legbal levezetés első nyelvtani jele, ezt helyettesítjük a szabály jobb oldalával.)

A másik szabálytípus a kezdőszelet-egyeztetést végzi:

$\delta(a_0, t, t) = (a_0, \varepsilon)$, minden $t \in T$-re.

Tulajdonképpen ez az az összehasonlítás, mely a levezetett terminális jelet összveti az input szó következő, még nem egyeztetett jelével.

Erre a $\mathcal{V}$-re nyilván $L^{\varepsilon}(\mathcal{V}) = L(G)$.

Példánkban:

$$[a_0, abba, S] \underset{\mathcal{V}}{\dashv} [a_0, abba, SS] \underset{\mathcal{V}}{\dashv} [a_0, abba, abS] \underset{\mathcal{V}}{\dashv} [a_0, bba, bS] \underset{\mathcal{V}}{\dashv} [a_0, ba, S] \underset{\mathcal{V}}{\dashv}$$
$$\underset{\mathcal{V}}{\dashv} [a_0, ba, ba] \underset{\mathcal{V}}{\dashv} [a_0, a, a] \underset{\mathcal{V}}{\dashv} [a_0, \varepsilon, \varepsilon].$$

„$\subseteq$”:

Azt kell belátni, hogy tetszőleges $\mathcal{V} = \langle A, T, \Sigma, \delta, a_0, \sigma_0 \rangle$ veremautomatához létezik olyan $G \in \mathcal{G}_{\text{kit2}}$, melyre $L(G) = L^{\varepsilon}(\mathcal{V})$.

Tudjuk, hogy $\mathcal{V}$ akkor és csak akkor fogadja el az u szót, ha $[a_0, u, \sigma_0] \overset{*}{\underset{\mathcal{V}}{\dashv}} [b, \varepsilon, \varepsilon]$ valamely $b \in A$-ra. Ez azt jelenti, hogy miközben $\mathcal{V}$ elolvasta az inputot, és átment az a_0 állapotból a b állapotba, a veremben „feldolgozta” a σ_0 jelet.

5.18. definíció. *Az* $[a, u, \sigma] \overset{*}{\underset{\mathcal{V}}{\dashv}} [b, \varepsilon, \varepsilon]$ *alakú konfiguráció átmeneteket,* σ *feldolgozásának nevezzük. Egy feldolgozás adatai:*

- u *input, a vezérlés,*
- a *kiinduló állapot,*
- b *befejező állapot.*

G nyelvtani jelei feldolgozások lesznek. Egy nyelvtani jel három információt tartalmaz: honnan indult $\mathcal{V}$, hová érkezett, és mit dolgozott fel. Az $[a, u, \sigma] \overset{*}{\underset{\mathcal{V}}{\dashv}} [b, \varepsilon, \varepsilon]$ feldolgozásnak tehát az (a, σ, b) nyelvtani jelet feleltetjük meg úgy, hogy a következőket várjuk tőle:

$$(a, \sigma, b) \xrightarrow[G]{*} u \iff [a, u, \sigma] \overset{*}{\underset{\mathcal{V}}{\dashv}} [b, \varepsilon, \varepsilon].$$

Kiegészül még a nyelvtani jelek halmaza az S kezdőszimbólummal.

$$N := \{(a, \sigma, b) \mid a, b \in A, \sigma \in \Sigma\} \cup \{S\}.$$

Milyenek legyenek a nyelvtan szabályai?

1) $S \longrightarrow (a_0, \sigma_0, b)$ minden $b \in A$ esetén, az elfogadáshoz tartozó feldolgozások megkezdését szimulálja.

2) Egylépéses feldolgozás esetén $u = t, t \in T \cup \{\varepsilon\}$, tehát t hatására σ-t a veremből kivesszük, amit az alábbi szabály szimulál:

$$(a, \sigma, b) \longrightarrow t \in \mathcal{P} \iff \delta(a, t, \sigma) \ni (b, \varepsilon), t \in T \cup \{\varepsilon\}.$$

3) Nem egylépéses feldolgozás esetén a következő szabályokat vesszük fel:

$(a, \sigma, b) \longrightarrow t(c_0, \sigma_1, c_1)(c_1, \sigma_2, c_2) \dots (c_{k-1}, \sigma_k, c_k) \in \mathcal{P}$, ahol $c_k = b$,
$(c_0, \sigma_1 \dots \sigma_k) \in \delta(a, t, \sigma)$, és $c_1, \dots, c_{k-1} \in A$ tetszőlegesek, melyek σ feldolgozását visszavezetik a verembe helyezett $\sigma_1, \dots, \sigma_k$ elemeinek feldolgozására.

Bizonyítanunk kell, hogy ez tényleg jó nyelvtan lesz, azaz $L(G) = L^{\varepsilon}(\mathcal{V})$, ahol $G = \langle N, T, \mathcal{P}, S \rangle$ és $\mathcal{P}$ az előbb definiált szabályokból áll.

5.19. állítás. *Tetszőleges* $u \in T^*$ *esetén* $(a,\sigma,b)\xrightarrow[G]{*}u$ *akkor és csak akkor, ha* $[a,u,\sigma] \overset{*}{\underset{\mathcal{V}}{\vdash}} [b,\varepsilon,\varepsilon]$.

Bizonyítás. Teljes indukcióval látjuk be.

„$\Longrightarrow$”:

A levezetés hossza szerint mutatjuk meg, hogy minden levezetés megfelel egy feldolgozásnak.

Mivel $u \in T^*$, ezért a legrövidebb levezetés 1 hosszúságú. Ekkor az alkalmazott szabály 2) típusú és $u \in T \cup \{\varepsilon\}$,

$$(a,\sigma,b)\xrightarrow[G]{(1)}u \in T \cup \{\varepsilon\} \implies (b,\varepsilon) \in \delta(a,u,\sigma) \implies [a,u,\sigma] \overset{(1)}{\underset{\mathcal{V}}{\vdash}} [b,\varepsilon,\varepsilon].$$

Most tegyük fel, hogy a legfeljebb h hosszú levezetésekre ($h \leq 1$) az állítás igaz, és vizsgáljuk meg az $(a,\sigma,b)\xrightarrow[G]{(h+1)}u \in T^*$ levezetést $u \in T^*$ esetén. Nyilvánvaló, hogy van egy legelső lépés. Ez a lépés csak a 3) pontban definiált valamely szabállyal lehetséges, tehát:

$$(a,\sigma,b)\xrightarrow[G]{(1)}t(c_0,\sigma_1,c_1)\dots(c_{k-1},\sigma_k,c_k), \qquad (*)$$

ahol $c_k = b$, továbbá $(c_0,\sigma_1\sigma_2\dots\sigma_k) \in \delta(a,t,\sigma)$.

Mivel G 2-es típusú nyelvtan, ezért az u szó a függetlenségi lemma alapján felbontható $u = tv_1v_2\dots v_k$ alakra úgy, hogy

$$(c_0,\sigma_1,c_1)\xrightarrow[G]{\leq h}v_1,\ \dots\ ,(c_{k-1},\sigma_k,c_k)\xrightarrow[G]{\leq h}v_k.$$

Ezekre alkalmazzuk az indukciós feltételt, azaz

$$[c_0,v_1,\sigma_1] \overset{*}{\underset{\mathcal{V}}{\vdash}} [c_1,\varepsilon,\varepsilon],\dots,[c_{k-1},v_k,\sigma_k] \overset{*}{\underset{\mathcal{V}}{\vdash}} [c_k,\varepsilon,\varepsilon].$$

Most „rakjuk össze” ezeket a működéseket, így azt kapjuk, hogy

$$[c_0,v_1\dots v_k,\sigma_1\sigma_2\dots\sigma_k] \overset{*}{\underset{\mathcal{V}}{\vdash}} [c_k,\varepsilon,\varepsilon].$$

A (*) lépésnek megfelelő működéssel kiegészítve ezt kapjuk, hogy

$$[a,tv_1\dots v_k] \overset{(1)}{\underset{\mathcal{V}}{\vdash}} [c_0,v_1\dots v_k,\sigma_1\dots\sigma_k] \overset{*}{\underset{\mathcal{V}}{\vdash}} [c_k,\varepsilon,\varepsilon],$$

és mivel $c_k = b$, az állítást beláttuk.

„$\Longleftarrow$”:

Most feltesszük, hogy $[a,u,\sigma] \overset{*}{\underset{\mathcal{V}}{\vdash}} [b,\varepsilon,\varepsilon]$, és a feldolgozás hossza szerinti indukcióval bizonyítjuk, hogy az (a,σ,b) nyelvtani jelből levezethető az $u \in T^*$ szó.

Egy jel feldolgozásra került, ezért világos, hogy a feldolgozás legalább 1 hosszú. Vizsgáljuk tehát a $h = 1$ legrövidebb feldolgozási hosszt. Ekkor tehát $[a, u, \sigma] \overset{(1)}{\underset{\mathcal{V}}{\vdash}} [b, \varepsilon, \varepsilon]$, és $u \in T \cup \{\varepsilon\}$, amiből nyilván következik, hogy $(b, \varepsilon) \in \delta(a, \sigma, b)$. Viszont a 2)-es szabálydefiníció alapján $(a, \sigma, b) \longrightarrow u \in \mathcal{P}$ és így levezethető u.
$h + 1$ hossz: Most tegyük fel, hogy a legfeljebb h hosszú feldolgozásokra van levezetés, és lássuk be $(h+1)$-re. Legyen a feldolgozás $[a, u, \sigma] \overset{(h+1)}{\underset{\mathcal{V}}{\vdash}} [b, \varepsilon, \varepsilon]$. Mivel h legalább 1 volt, ezért ez nem egy közvetlen konfigurációátmenet, tehát létezik egy legelső lépése. Ezek alapján tehát az u szó felbontható $u = tv$ alakban, ahol $t \in T \cup \{\varepsilon\}$, és $v \in T^*$. Ekkor a legelső lépést külön kiírva:

$$[a, tv, \sigma] \overset{(1)}{\underset{\mathcal{V}}{\vdash}} [c_0, v, \sigma_1 \dots \sigma_k] \overset{\leq h}{\underset{\mathcal{V}}{\vdash}} [b, \varepsilon, \varepsilon]. \qquad (**)$$

A $\sigma_1 \dots \sigma_k$ jelek teljesen feldolgozásra kerülnek, így létezik egy legelső időpont, amikor σ_1 eltűnik. Legyen az automata ebben az időpontban a c_1 állapotban. (Ekkor nyilván σ_2 van a verem tetején.) Folytassuk ezt a jelölést, és legyen az automata c_{k-1} állapotban abban az időpontban, amikor már csak σ_k van a verem tetején. Végül vezessük be a $c_k = b$ jelölést is. Jelölje $v_1, \dots, v_k$ azokat a részszavakat, amiket a $c_0, c_1, \dots, c_k$ állapotok között olvastunk. Ezen jelölések alapján nyilván igaz, hogy

$$[c_0, v_1, \sigma_1] \overset{\leq h}{\underset{\mathcal{V}}{\vdash}} [c_1, \varepsilon, \varepsilon], \ \dots \ , [c_{k-1}, v_k, \sigma_k] \overset{\leq h}{\underset{\mathcal{V}}{\vdash}} [c_k, \varepsilon, \varepsilon].$$

Ezekre már alkalmazhatjuk az indukciós feltételt, miszerint

$$\forall i = 1, \dots, k : (c_{i-1}, \sigma_i, c_i) \xrightarrow[G]{*} v_i.$$

Viszont a legelső konfigurációátmeneti lépés $(**)$ miatt igaz, hogy

$$(a, \sigma, b) \longrightarrow t(c_0, \sigma_1, c_1) \dots (c_{k-1}, \sigma_k, c_k) \in \mathcal{P}.$$

Innen nyilván már következik az állítás, mivel $u = tv_1v_2 \dots v_k$. □

Most térjünk vissza a tétel bizonyításához. Az előbbi állítás és az 1)-es szabálydefiníció alapján könnyen látszik, hogy

$$S \xrightarrow[G]{*} u \iff \exists b \in A : S \xrightarrow[G]{(1)} (a_0, \sigma_0, b) \xrightarrow[G]{*} u \iff \exists b \in A : [a_0, u, \sigma_0] \overset{*}{\underset{\mathcal{V}}{\vdash}} [b, \varepsilon, \varepsilon] \iff u \in L^{\varepsilon}(\mathcal{V}).$$

A $\mathcal{V}$ veremautomatához tehát tényleg tudunk adni olyan $G \in \mathcal{G}_{\text{kit2}}$ nyelvtant, melyre $L(G) = L^{\varepsilon}(\mathcal{V})$.

□

5.3. Determinisztikus 1-vermek

Láttuk, hogy a determinisztikus 0-vermek ugyanazt tudják, mint a nemdeterminisztikus 0-vermek. Kérdés az, hogy mit mondhatunk 1-vermek esetén.

Emlékeztető:
Az 1-verem determinisztikus, ha minden konfigurációjának legfeljebb egy rákövetkezője van. Az állapotfüggvánnyel ez a következő képpen írható le:

a) $\forall a \in A, \sigma \in \Sigma, t \in T \cup \{\varepsilon\} : |\delta(a,t,\sigma)| \leq 1.$

b) $\forall a \in A, \sigma \in \Sigma : |\delta(a,\varepsilon,\sigma)| \neq 0 \implies \forall t \in T : |\delta(a,t,\sigma)| = 0.$

A következőkben kimutatjuk, hogy a determinisztikus 1 vermek kevesebbet tudnak, mint a nemdeterminisztikusak. Ehhez szükségünk lesz néhány új fogalomra.

5.3.1. Veremautomaták normálformái

5.20. definíció. *A $\mathcal{V}$ 1-verem* **1-es normálformájú**, *ha $(b,\gamma) \in \delta(a,t,\sigma)$ esetén $\gamma \in \{\varepsilon\} \cup \{\sigma\} \cup \Sigma\{\sigma\}$.*

Megjegyzés: Ez a normálforma tulajdonképpen a klasszikus verem megvalósítója, ugyanis

$\gamma = \varepsilon$ a *pop* művelet,

$\gamma = \sigma$ a *top* művelet,

$\gamma \in \big(\Sigma\{\sigma\}\big)$ a *push* művelet.

5.21. definíció. *A $\mathcal{V}$ 1-verem* **2-es normálformájú**, *ha 1-es normálformájú, és bármely $[a,u,\alpha]$ terminális konfiguráció esetén $u = \varepsilon$. (Azaz ha az automata megáll, akkor biztosan végigolvasta az inputot.)*

5.22. definíció. *A $\mathcal{V}$ 1-verem* **3-as normálformájú**, *ha 2-es normálformájú, és minden bemenetre megáll.*

A fejezet további részében csak a végállapottal történő elfogadással foglalkozunk. A $\mathcal{V}$ 1-vermet a standard jelöléssel adjuk meg.

5.23. lemma. *Tetszőleges $\mathcal{V}$ 1-veremautomatához létezik 1-es normálformájú $\tilde{\mathcal{V}}$ 1-veremautomata, amire $L(\tilde{\mathcal{V}}) = L(\mathcal{V})$.*

Bizonyítás. δ' konstrukciójával célunk, hogy a normálforma szempontjából rossz átmeneteket megfelelőkkel szimuláljuk. Hogy néznek ki ezek a rossz átmenetek? Nyilván így:

$(b, \sigma_k\sigma_{k-1}\ldots\sigma_1) \in \delta(a,t,\sigma)$.

Ennek szimulációja 1-es normálformájú működésekkel, ahol $c_0, c_1, \ldots, c_{k-1}$ új állapotok:

$\delta'(a,t,\sigma) = (c_0, \varepsilon)$,

$\delta'(c_0,\varepsilon,\sigma') = (c_1, \sigma_1\sigma')$ minden $\sigma' \in \Sigma \cup \{\sigma_0'\}$ esetén,

$\delta'(c_1,\varepsilon,\sigma_1) = (c_2, \sigma_2\sigma_1)$,

$\vdots$

$\delta'(c_{k-1}, \varepsilon, \sigma_{k-1}) = (b, \sigma_k\sigma_{k-1})$.

Egy új σ'_0 veremkezdő jelet is beveszünk a $\delta'(a, t, \sigma) = (c_0, \varepsilon)$ hatására történő veremkiürülés elleni védekezésül a $\delta'(a'_0, \varepsilon, \sigma'_0) = (a_0, \sigma_0\sigma'_0)$ átmenettel, ahol a'_0 új kezdőállapot.

□

5.24. lemma. *Tetszőleges $\mathcal{V}$ 1-veremautomatához létezik $\tilde{\mathcal{V}}$ 2-es normálformájú 1-veremautomata, melyre $L(\tilde{\mathcal{V}}) = L(\mathcal{V})$.*

Bizonyítás.

Az előző lemma értelmében feltehető, hogy $\mathcal{V}$ már 1-es normálformában adott. Ekkor a gond az, hogy egy $[a, u, \alpha]$ konfiguráció úgy is lehet termináló, hogy $u \neq \varepsilon$. Azt szeretnénk, hogy ebből a helyzetből $\tilde{\mathcal{V}}$ tovább tudjon lépni. Ha $\alpha = \varepsilon$, tehát a verem kiürült, akkor ezen új kezdőállapottal és veremkezdő jellel a már korábban is látott módon tudunk segíteni. Feltehető tehát, hogy $\alpha \neq \varepsilon$. Ilyenkor a $\mathcal{V}$ azért nem tud működni, mert $u = tv$ és $\alpha = \sigma\alpha'$ első jeleire $\delta(a, t, \sigma)$ és $\delta(a, \varepsilon, \sigma)$ mindegyike üres.

Megoldásként bevezetünk egy új d állapotot, és minden ilyen $a \in A, t \in T$ és $\sigma \in \Sigma$ esetén felvesszük a következő átmeneteket:

$\delta'(a, t, \sigma) := (d, \sigma)$.

A d új állapot arra szolgál, hogy segítségével végig tudjuk olvasni az inputot.

$\delta'(d, t, \sigma') := (d, \sigma')$ minden $t \in T$-re és $\sigma' \in \Sigma$-ra.

Természetesen d-t a végállapotok közé nem vesszük be.

Tehát kaptunk egy $\tilde{\mathcal{V}}$ 2-es normálformájú veremautomatát $\mathcal{V}$-ből, melyre $L(\tilde{\mathcal{V}}) = L(\mathcal{V})$.

□

Megjegyzés: Mindkét lemma esetében nyilvánvaló, hogy ha a $\mathcal{V}$ veremautomata determinisztikus volt, akkor $\tilde{\mathcal{V}}$ is az lesz.

5.25. lemma. *Tetszőleges $\mathcal{V}$ determinisztikus 1-veremautomatához létezik $\tilde{\mathcal{V}}$ 3-as normálformájú determinisztikus 1-veremautomata, melyre $L(\tilde{\mathcal{V}}) = L(\mathcal{V})$.*

Bizonyítás.

Az előző lemma értelmében feltehető, hogy a $\mathcal{V}$ 1-verem 2-es normálformában adott. Ha $\mathcal{V}$ nem 3-as normálformájú, akkor létezik $\mathcal{V}$-ben végtelen konfigurációátmenet-lánc. Célunk az, hogy az ilyen láncokat felismerjük és $\tilde{\mathcal{V}}$ - megelőzve őket - álljon meg. A végtelen konfigurációátmenet-láncokra vezessük be a $[a, u, \alpha] \overset{*}{\underset{\mathcal{V}}{\vdash}} \infty$ jelölést.

Tegyük fel, hogy $[a, u, \alpha] \overset{*}{\underset{\mathcal{V}}{\vdash}} \infty$. Ekkor tudunk találni a működés során olyan állapotot, ahol már vagy elolvasta az input szót, vagy már nem olvas többet. A nem olvasott inputot elhagyva így egy $[b, \varepsilon, \beta] \overset{*}{\underset{\mathcal{V}}{\vdash}} \infty$ alakú konfigurációátmenetet kapunk. Tegyük fel, hogy ekkor $\beta := \sigma_1 \ldots \sigma_k$. Mivel $\mathcal{V}$ 2-es normálformájú, ezért 1-es is, tehát σ_k sose kerül ki a veremből. (Ugyanis *pop* tudná kivenni a veremből, de akkor az automata üres veremmel leállna.)

Legyen j_0 a legkisebb indexű olyan σ_j, mely a működés során végig a veremben van, azaz $j_0 := \min\{j \mid \sigma_j \text{ mindig a veremben van}\}$. Ekkor nyilván létezik egy olyan időpillanat, amikor ez a σ_{j_0} lesz a verem tetején. Legyen ennek az időpillanatnak a konfigurációja

$[c_0, \varepsilon, \sigma_{j_0}\sigma_{j_0+1}\dots\sigma_k]$. Tudjuk, hogy σ_{j_0} sose kerül ki a veremből, tehát a veremben takart $\sigma_{j_0+1}, \dots, \sigma_k$ jeleket el is hagyhatjuk, így egy

$[c_0, \varepsilon, \sigma_{j_0}] \overset{*}{\underset{\mathcal{V}}{\vdash}} \infty$, végtelen konfigurációátmenetet kapunk.

Ha $\tilde{\mathcal{V}}$-ben az ilyen típusú végtelen konfigurációátmenetek lehetőségét a δ átdefiniálásával ki tudjuk zárni, készen vagyunk. Legyen

$$\delta'(a,\varepsilon,\sigma) := \begin{cases} \delta(a,\varepsilon,\sigma), & \text{ha } [a,\varepsilon,\sigma]\text{-ből indulva nem végtelen a konfigurációátmenetlánc,} \\ (d,\sigma), & \text{ha } [a,\varepsilon,\sigma] \overset{*}{\underset{\mathcal{V}}{\vdash}} \infty, \text{ és közben nem érint végállapotot,} \\ (f,\sigma), & \text{ha } [a,\varepsilon,\sigma] \overset{*}{\underset{\mathcal{V}}{\vdash}} \infty, \text{ és közben érint végállapotot.} \end{cases}$$

Itt d, f új állapotok, és $F' := F \cup \{f\}$. Ekkor d ismét csak az input szó végigolvasását végzi el.

$\delta'(d,t,\sigma') := (d,\sigma')$ minden $\sigma' \in \Sigma$-ra.

Egyetlen probléma még az f végállapotnál maradt, ugyanis lehet, hogy itt még nem olvastuk végig az input szót. Ezért ha van még jel az inputon, azt a szót se fogadhatjuk el, azaz

$\delta'(f,t,\sigma') := (d,\sigma')$ minden $\sigma \in \Sigma, t \in T$-re.

Az így definiált $\tilde{\mathcal{V}}$ veremautomára tényleg $L(\tilde{\mathcal{V}}) = L(\mathcal{V})$, és ez a $\tilde{\mathcal{V}}$ is determinisztikus lesz.

□

Bár a 3. normálforma létezését bizonyítottuk, de az egy további kérdés, hogy a kapott konstrukció algoritmikus-e, azaz megállapítható-e algoritmikusan, hogy valamely $a \in A$ és $\sigma \in \Sigma^*$ mellett $[a,\varepsilon,\sigma] \overset{*}{\underset{\mathcal{V}}{\vdash}} \infty$ vagy sem. Ha az $[a,\varepsilon,\sigma]$ konfigurációból indított determinisztikus működés során találunk két olyan $[c_1,\varepsilon,\alpha_1] \overset{*}{\underset{\mathcal{V}}{\vdash}} [c_2,\varepsilon,\alpha_2]$ konfigurációt, melyre $c_1 = c_2$ és

1) $\alpha_1 = \alpha_2$ vagy
2) α_1 és α_2 ugyanazzal a $\hat{\sigma}$ jellel kezdődik és az átmenet során α_1 végig a veremben marad,

akkor az automata ettől kezdve az ezen két konfiguráció közötti működést ismétli periódikusan, tehát az átmenet biztosan végtelen. (A második esetben az $\alpha_1 = \hat{\sigma}\beta$, $\alpha_2 = \hat{\sigma}\gamma\hat{\sigma}\beta$ és a $c_1 = c_2 = c$ jelölést alkalmazva a $[c,\varepsilon,\hat{\sigma}] \overset{*}{\underset{\mathcal{V}}{\vdash}} [c,\varepsilon,\hat{\sigma}\gamma\hat{\sigma}]$ működés ismétlődik, egyre újabb $\hat{\sigma}\gamma$-val hízlalva a vermet.)

Igaz-e, hogy az $[a,\varepsilon,\sigma] \overset{*}{\underset{\mathcal{V}}{\vdash}} \infty$ átmenet az előbb említett két típusú periódikusság valamelyike miatt végtelen? Ha $[a,\varepsilon,\sigma] \overset{*}{\underset{\mathcal{V}}{\vdash}} \infty$ során a verem mérete korlátos marad, akkor csak véges számú konfiguráció fordulhat elő benne, tehát valamelyik ismétlődni fog. Az ilyen esetben az átmenet az első típus szerint periódikus.

Ha $[a,\varepsilon,\sigma] \overset{*}{\underset{\mathcal{V}}{\vdash}} \infty$ során a verem mérete nem korlátos, akkor minden $i \geq 1$-re léteznek olyan $[c_i,\varepsilon,\alpha_i]$ közbülső konfigurációk, melyektől kezdve a verem mérete már mindig legalább i hosszúságú. Az első normálforma miatt ez csak úgy lehetséges, hogy a $[c_i,\varepsilon,\alpha_i]$-t követő konfigurációk során α_i már mindig a veremben marad (hiszen például α_i első elemét onnan törölni csak a *pop* művelettel lehetne az első normálforma tulajdonsága miatt, de akkor a *pop* után a verem mérete $i-1$ lenne, ellentétben c_i és α_i választásával). Ennek alapján léteznek $\sigma_1,\sigma_2,\ldots$ jelek, hogy $\alpha_i = \sigma_i\sigma_{i-1}\ldots\sigma_1$. De ekkor a (c_i,σ_i) párok között biztosan van ismétlődő, például az $i_1 < i_2$ indexpárhoz tartozó. De akkor $[c_{i_1},\varepsilon,\alpha_{i_1}]$ és $[c_{i_2},\varepsilon,\alpha_{i_2}]$ pontosan megfelel a második típusú periódikusságot eredményező feltételnek.

Hány lépésen belül találjuk meg egy végtelen láncban a periódikusságot bizonyító konfiguráció-párt? A $[c_i,\varepsilon,\alpha_i]$ konfiguráció előtt, ha közöttük nem volt ismétlődő, legfeljebb $|A|\cdot|\Sigma^{\leq i}|$ konfiguráció lehet. (Ha volt ismétlődés, már az 1. típusú periódikusságot felismertük.) A különböző állapot és veremtető párok száma $|A||\Sigma|$, tehát $i_0 = |A||\Sigma|+1$ mellett már biztosan megtalálunk egy 2. típusú periódikusságot garantáló párt. Ez előtt tehát legfeljebb $|A|\cdot|\Sigma^{|A||\Sigma|+1}|$ konfiguráció lehet, azaz legfeljebb ennyi lépésen belül megállapítható, hogy az $[a,\varepsilon,\sigma]$-ból induló működés véges-e vagy végtelen. Ha végtelen a lánc, akkor a végállapot detektálása a periódus végéig terjedő működés vizsgálatával, tehát szintén véges időn belül történhet meg.

5.26. definíció. *A $\mathcal{V}$ veremautomatát inputterminátoros veremautomatának nevezzük, ha az inputja végén van egy vége jel (EOF), amit # jelöl, és δ erre a # jelre is definiálva van. Ekkor*

$$L^{\mathrm{Fin}}(\mathcal{V}) = \{u \in T^* \mid [a_0,u\#,\sigma_0] \overset{*}{\underset{\mathcal{V}}{\vdash}} [f,\varepsilon,\alpha] \text{ valamely } f \in F\text{-re}\}.$$

Jelölése: $\mathcal{L}_{\mathrm{It1}\mathcal{V}} = \{L^{\mathrm{Fin}}(\mathcal{V}) \mid \mathcal{V}$ *inputterminátoros 1-verem* $\}$.

Megjegyzés: ha $\mathcal{V}$ determinisztikus, akkor jelölésben $\mathcal{L}_{\mathrm{DIt1}\mathcal{V}}$-et használunk.

Könnyen belátható, hogy $\mathcal{L}_{\mathrm{It1}\mathcal{V}}$=$\mathcal{L}_{1\mathcal{V}}$.

Tegyük fel, hogy $\mathcal{V}$ egy inputterminátoros veremautomata. A $\tilde{\mathcal{V}}$ őt szimuláló (nem inputterminátoros) veremautomata a # olvasása helyett ε-mozgást végezzen, tehát ha $(b,\alpha) \in \delta(a,\#,\sigma)$, akkor $(b,\alpha) \in \delta'(a,\varepsilon,\sigma)$. $\tilde{\mathcal{V}}$ már nem feltétlenül lesz determinisztikus még akkor sem, ha a $\mathcal{V}$ az volt.

Egy $\mathcal{V}$ végállapottal elfogadó veremautomatából szintén könnyen készíthetünk $\tilde{\mathcal{V}}$ inputterminátoros veremautomatát a $\delta'(f,\#,\sigma) := (f^{(2)},\sigma)$ új átmenetek segítségével (minden $f \in F$ és $\sigma \in \Sigma$-ra), ahol $f^{(2)}$ új végállapot. Determinisztikus $\mathcal{V}$ esetén $\tilde{\mathcal{V}}$ is az lesz.

Azt is kimutattuk tehát, hogy $\mathcal{L}_{D1\mathcal{V}} \subseteq \mathcal{L}_{\mathrm{DIt1}\mathcal{V}}$. Összefoglalva:

$$\mathcal{L}_{D1\mathcal{V}} \subseteq \mathcal{L}_{\mathrm{DIt1}\mathcal{V}} \subseteq \mathcal{L}_{\mathrm{It1}\mathcal{V}} = \mathcal{L}_{1\mathcal{V}} = \mathcal{L}_2.$$

Célunk az, hogy kimutassuk, $\mathcal{L}_{\mathrm{DIt1}\mathcal{V}}$ zárt a komplementer képzésre. Ugyanis amiatt, hogy $\mathcal{L}_2$ nem zárt a komplementer képzésre, következményként megkapnánk az $\mathcal{L}_{D1\mathcal{V}} \subset \mathcal{L}_2$ valódi tartalmazást.

5.27. lemma. $\mathcal{L}_{\mathrm{DIt1}\mathcal{V}}$ *zárt a komplementer műveletre.*

Bizonyítás. Legyen $L \in \mathcal{L}_{\mathrm{DIt1}\mathcal{V}}$ tetszőleges nyelv. Be kell látni, hogy ekkor $\overline{L} \in \mathcal{L}_{\mathrm{DIt1}\mathcal{V}}$. Legyen $\mathcal{V} = \langle A, T, \Sigma, \delta, a_0, \sigma_0, F, \# \rangle$ az L nyelvet elfogadó inputterminátoros determinisztikus 1-verem, melyre $L(\mathcal{V}) = L$.

A korábbi lemmák alapján feltehető, hogy $\mathcal{V}$ 3-as normálformában adott, azaz tetszőleges $u\#$ inputot végigolvas és megáll. Ez általában úgy történik, hogy a # olvasása után még következik néhány ε-lépés.

Egy tetszőleges $u \notin L(\mathcal{V})$ szó feldolgozása során $\mathcal{V}$ a # elolvasását követő ε-lépések során nem érint végállapotot (legyen ez a ' tulajdonság), míg $u \in L(\mathcal{V})$ esetén a # elolvasása után az ε-lépésekben érint végállapotot (legyen ez a " tulajdonság).

Most próbáljuk megkonstruálni a $\overline{\mathcal{V}}$ determinisztikus, inputterminátoros 1-vermet. Ez lényegét tekintve $\mathcal{V}$-vel azonos, eltérés csak a # jel elolvasása után van.

Legyen $\overline{\mathcal{V}} = \langle A \cup A' \cup A'' \cup \{a_{\text{vég}}\}, T, \Sigma, \delta', a_0, \sigma_0, \{a_{\text{vég}}\}, \# \rangle$, ahol $A' = \{b' \mid b \in A\}$ és $A'' = \{b'' \mid b \in A\}$. Ekkor # elolvasására δ' definíciója:

$$\delta'(a, \#, \sigma) := \begin{cases} (b', \alpha), & \text{ha } \delta(a, \#, \sigma) = (b, \alpha), \text{ és } b \notin F, \\ (b'', \alpha), & \text{ha } \delta(a, \#, \sigma) = (b, \alpha), \text{ és } b \in F \end{cases}.$$

Valamint

$$\delta'(a', \varepsilon, \sigma) := \begin{cases} (b', \alpha), & \text{ha } \delta(a, \varepsilon, \sigma) = (b, \alpha) \text{ és } b \notin F, \\ (b'', \alpha), & \text{ha } \delta(a, \varepsilon, \sigma) = (b, \alpha) \text{ és } b \in F \end{cases}.$$

Tehát ha megérkezünk $\mathcal{V}$-ben az első végállapotba, akkor $\overline{\mathcal{V}}$ "-ős állapotba megy át, amit meg is tart:

$$\delta'(a'', \varepsilon, \sigma) := (b'', \alpha), \text{ ha } \delta(a, \varepsilon, \sigma) = (b, \alpha).$$

Ha $\mathcal{V}$ '-s állapotban áll meg, akkor nyilván $u \notin L(\mathcal{V})$, és ha egy "-s állapotban, akkor $u \in L(\mathcal{V})$. De a megállás során kialakult eredeti (vesszőtlen) állapotra az eredeti δ a megállás miatt nincs definiálva, tehát δ'-t definiálhatjuk így:

$$\delta'(a', \varepsilon, \sigma) := (a_{\text{vég}}, \sigma), \text{ ha } \delta(a, \varepsilon, \sigma) = \emptyset.$$

Az így megadott $\overline{\mathcal{V}}$ 1-veremre nyilván $L(\overline{\mathcal{V}}) = \overline{L(\mathcal{V})} = \overline{L}$. □

Foglalkozhatnánk még a 2-vermekkel, illetve az ezeknél nagyobb veremszámú automatákkal. Ezek már minden $\mathcal{L}_0$-beli nyelvet el tudnak fogadni, azaz az 1-vermekhez képest már két osztálynyi az ugrás. Továbbá igaz, hogy az 1-vermekkel ellentétben a determinisztikus 2-vermek ugyanazt a nyelvosztályt adják, mint a nemdeterminisztikus 2-vermek.

6. Elemzési módszerek

A most következő fejezetben a fordítóprogramok készítsének elméleti hátteréül szolgáló elemzési módszerekkel foglalkozunk. Nem célunk, hogy teljes részletességgel mutassuk be az egyes algoritmusokat, hiszen ez már túlmutatna egy bevezető automaták és formális nyelvek kurzus keretein.

6.1. Az elemzési módszerek célja

A fordítóprogramok az inputként kapott karaktersorozatot az adott programozási nyelv szabályai alapján szintaktikusan ellenőrzik és a helyesnek bizonyuló programokból elkészítik a tárgykódot. A fordítás általában több lépésben történik, melyre már volt néhány utalás az előző fejezetekben. Az első lépés, a lexikális elemzés során a fordító felismeri a programnyelv lexikális egységeit (azonosítók, alapszavak, ...) és azokat táblázatban rögzítve a felsőbb szintek számára érthető kódjukkal helyettesíti. A lexikális analízis egy 3-as típusú (reguláris) nyelv szavainak felismerésén alapszik, mely a nyelvet felismerő véges determinisztikus automata programnyelvi eszközökkel történő szimulációjával viszonyleg könnyen megvalósítható (ezzel a továbbiakban nem foglalkozunk). A szintaktikus elemző a lexikálisan elemzett, átkódolt szimbólumsorozatot szerkezetileg ellenőrzi a nyelvet leíró 2-es típusú nyelvtan felhasználásával. Ennek az elemzési fázisnak egy szintaxisfa a kimenete. A szintaktikus analízis utolsó lépésében a fordító szemantikusan ellenőrzi a felépített szintaxisfát, majd a kódgenerálás fázisában a fa alapján elkészíti a tárgykódot. Ez utóbbi két lépést sem tesszük vizsgálat tárgyává könyvünk keretein belül.

Nézzük tehát, mi a szűkebb értelemben vett szintaktikus elemzők feladata. Legyen adva egy $G = \langle T, N, \mathcal{P}, S \rangle \in \mathcal{G}_{\text{kit2}}$ második típusú nyelvtan és egy $u \in T^*$ szó. Az elemzési algoritmusok feladata azt eldönteni, hogy u szó eleme-e $L(G)$-nek és ha igen, akkor felépíteni u egy G feletti szintaxisfáját.

6.2. A szintaxiselemzési módszerek osztályozásai

Az elemzés során tehát az adott $G = \langle T, N, \mathcal{P}, S \rangle$ 2-es típusú nyelvtan és $u \in T^*$ szó esetén meg kell kísérlelni felépíteni u egy szintaxisfáját. Ha ez sikerült, az elemzés sikeres volt, egyébként pedig a szó szintaktikusan hibás.

Az elemzési módszereket elsősorban annak alapján osztályozzuk, hogy a szintaxisfát honnan kezdve próbáljuk meg felépíteni. Amikor a nyelv főfogalmából elindulva, vagyis a gyökértől kezdve építkezünk, akkor felülről lefelé (*top-down*) elemzésről beszélünk, míg ha magából az u-ból, vagyis a készítendő fa leveleinek irányából indulva építkezünk, akkor alulról-felfelé (*bottom-up*) elemzővel van dolgunk. Bár mindkét módszer általánosan használható, de valamely konkrét nyelvtan esetében előfordulhat, hogy a kettő közül valamelyik sokkal hatékonyabb, mint a másik (például a későbbiekben definiált $LL(k)$ nyelvtanokat felülről lefelé elemzővel célszerű elemezni). Természetesen előfordulhat a két módszer kombinációja is, például magát a nyelvet az egyik módszerrel, míg annak egy szintaktikus egységét a másik módszerrel elemezzük (pl. ilyennek tekinthető a lexikális elemző is egy felülről-lefele elven alapuló szintaktikus elemzőben).

Az elemzők másik szempont szerinti osztályozása az input karaktersorozat elemeihez való hozzáférés lehetőségein alapul. Direkt módon elérhetők-e például az input szó karakterei, vagy csak valamelyik irányból szekvenciálisan olvashatjuk végig őket. Ha csak a valamilyen irányú végigolvasás megengedett, akkor hányszor tehetjük ezt meg.

Mivel a legfontosabb alkalmazás, a fordítóprogramok esetében az elemzendő program szövege szekvenciális input file-ban érhető el, s a nagy méret alapján nem gazdaságos a többszöri végigolvasás, ezért mi csak olyan elemzési módszerekkel foglalkozunk, ahol az input szót csak balról jobbra, egyszer lehet olvasni. Ezeket az elemzőket balról-jobbra (*left to right*) elemzőknek nevezzük.

Osztályozhatjuk az elemző algoritmusokat aszerint is, hogy determinisztikus, illetve nemdeterminisztikus algoritmusok-e. A nemdeterminisztikus algoritmusoknak a számítástudományban elméleti jelentőségük van, mert segítségükkel tömören, elegánsan oldhatók meg bizonyos keresési feladatok, mint amilyen például a szintaktikus elemzés feladata is. A gyakorlatban elemzésre természetesen csak determinisztikus algoritmusokat lehet használni. Programtranszformációk segítségével minden nemdeterminisztikus algoritmusból előállítható vele ekvivalens determinisztikus algoritmus, például úgy, hogy a nemdeterminisztikus algoritmus össszes lehetséges működéseinek fáját, az úgynevezet lefutási fát, valamilyen bejárási algoritmussal bejárjuk. Az így kapott determinisztikus algoritmusok általában nem hatékonyak, ezért a fordítóprogramok specálisan kifejlesztett determinisztikus elemzőket használnak, melyek csak bizonyos tulajdonságú nyelvtanokra alkalmazhatók (például a későbbiekben definiált $LL(k), LR(k)$) nyelvtanok), de ezekre igen gyorsan működnek.

6.3. Felülről-lefelé elemzések

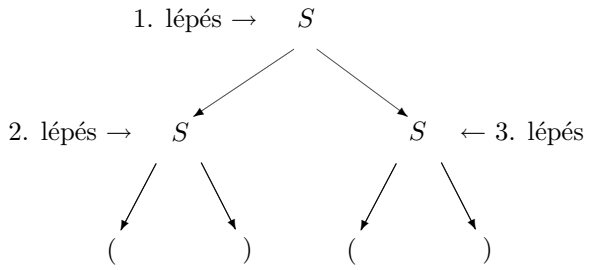
Mint már utaltunk rá, ezek a módszerek a szintaxisfát a gyökérétől (az S kezdőszimbólumtól) kezdve építik fel.

Először nézzük egy példát. Legyen $T = \{(,)\}, N = \{\mathrm{S}\}, u = ()()$, és legyen

$G_1 = \langle T, N, \mathcal{P}, \mathrm{S}\rangle$, ahol

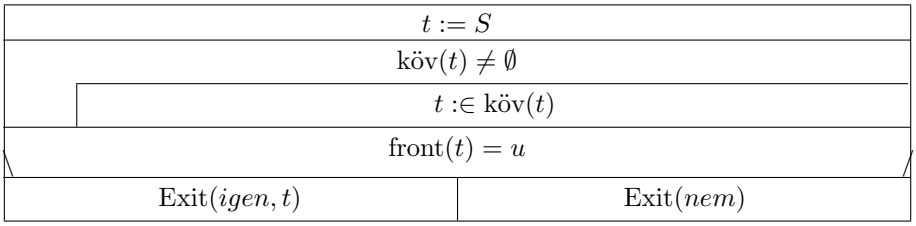
$\mathcal{P} = \{S \longrightarrow SS \mid (S) \mid ()\}$.

Ekkor a szintaxisfa felépítése S kezdőszimbólumból az S$\longrightarrow$SS$\longrightarrow$ ()S$\longrightarrow$ ()() legbal levezetést követve az alábbi:



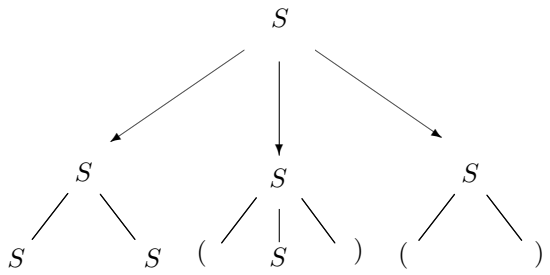
az $u = ()()$ szó szintaxisfája

Az általános, nemdeterminisztikus Top-Down elemző algoritmusa igen egyszerű és elegáns, amint azt az alábbi struktogram mutatja. Itt a t változó szintaxisfát jelent, s felhasználjuk a köv(t) függvényt, mely valamely t szintaxisfából egy szabály alkalmazásával származtatható új szintaxisfák halmazát adja eredményül.



Ez a nemdeterminisztikus algoritmus abban az értelemben oldja meg az elemzés feladatát, hogy ha a szó valóban eleme a nyelvtan által generált nyelvnek, akkor van olyan működése, melynek során ez kiderül, s ilyenkor felépül a megfelelő szintaxisfa is.

A determinisztikussá alakítás érdekében a lefutási fát bejárjuk valamilyen stratégiával keresve egy olyan működést, mely u felismeréséhez tartozik. A fa csúcspontjaiban most - az algoritmus állapotait jellemző adatként - a működés során előállított szintaxisfa áll. Ahhoz, hogy a lefutási fa alakja - s így a bejárási sorrend - egyértelmű legyen, a köv(t) halmazt, a t szintaxisfa lehetséges továbbépítéseinek halmazát, rendezzük úgy, hogy előbb legyenek a szó elejéhez közelebb eső pozíción végzett helyettesítések, míg ugyanazon pozicíón az alternatíváknak a nyelvtan megadásánál rögzített sorrendje határozza meg a levezetések sorrendjét. Nézzük meg, milyen lesz a lefutási fa előbbi nyelvtanunk esetében:



a lefutási fa első két szintje

A lefutási fában egy levél kapjon „+” cimkét, ha a pontban az adott u szó szintaxisfája van, ellenkező esetben pedig kapjon „-” cimkét. A determinisztikus algoritmusunk a lefutási fában keres „+” cimkéjű levelet, a (lefutási) fa pontjainak valamilyen szisztematikus sorrend szerinti bejárásával (amire általánosan ismert sémák vannak). Ha a bejárás során valamikor „+” cimkéjű pontot érintünk, *igen* választ adva megállunk. Ha a fa „elfogy”, illetve valahonnan megállapítható, hogy a további keresés felesleges, akkor *nem* válasszal állunk meg. Egyébként a bejárás végtelen ideig folytatódhat tovább.

Fontos kérdés, hogy milyen bejárást alkalmazzunk. Tekintsük először a szintfolytonos (szélességi) bejárást. Mivel a lefutási fában minden pontnak csak véges számú leágazása van (köv(t) mindig véges), ezért minden szintje véges. Emiatt - ha egyátalán vannak - a „+” cimkéjű leveleket a szintfolytonos bejárás biztosan megtalálja. Ha nincs „+” cimkéjű levél, akkor végtelen fák esetén a bejárás egyéb kritériumok híján soha nem áll le, míg véges lefutási fa esetén az összes pont megtekintése után *nem* válasszal megáll.

Meg kell vizsgálni, hogy mennyi lehet egy „+” cimkéjű csúcs megtalálásának műveletigénye a legrosszabb esetben. Ezt $l(u)$ függvényében fogjuk megbecsülni. Legyen nyelvtanunk kellően nemdeterminisztikus, azaz minden nyelvtani jelre legyen legalább két alkalmazható szabály, továbbá legyen K az összes szabály jobboldalát tekintve a leghosszabb helyettesítés hossza. Ekkor u levezetéséhez biztosan kell legalább $l(u)/K$ levezetési lépés (hogy elérjük az u hosszát), ezért egy „+” címkejű pontnak a lefutási fában lévő magassága ennél rövidebb nem lehet. Ekkor viszont a bejárt pontok száma legalább $2^{l(u)/K}$, mert a lefutási fa legalább bináris. A szintfolytonos bejárási stratégiánál tehát még a pozitív válasznak is lehet exponenciális időbonyolultsága, ezért ebben az általánosságában a kapott determinisztikus algoritmus gyakorlatban nem használható.

Most nézzük meg, hogy a *preorder*, másképpen *mélységi* bejárás esetén (ami tulajdonképpen a visszalépéses keresés algoritmusa), mit mondhatunk. Ez az algoritmus még a létező „+” cimkéjű leveleket sem mindig találja meg, mert előtte ráfuthat egy végtelen ágra (az előbbi példánk is ilyen). Valamely „+” címke megtalálása csak akkor garantált, ha a fának a preorder bejárás szerint a csúcsot megelőző része véges. (Ezt például elősegíthetjük azzal, hogy a balrekurzív szabályokat az alternatívák sorának végére helyezzük.) A lefutási idő az *igen* válasz esetén itt akár lehet polinomiális is, ha például a legelső ágon van a megfelelő „+” cimkéjű levél, de a nem válaszhoz (egyéb kritérium híján) az egész fát végig kell nézni.

Vizsgáljuk meg, milyen módszerekkel lehetne az előbb említett algoritmusok futási idejét csökkenteni.

— Csökkenthetjük a lefutási fa szélességét azzal, hogy a levezetésekre kikötünk bizonyos feltételeket (például csak legbal levezetésekkel foglalkozunk, korlátozzuk az alkalmazható szabályokat).

— Vágási kritériumokat határozunk meg, melyek alapján egy pontról eldönthető, hogy az alatta lévő részfában már nincs „+” csúcs. (Ilyen vágási kritérium lehet például a kezdőszeletek különbözősége, hosszúságot nem csökkentő nyelvtan esetében $l(u)$ hosszának meghaladása)

Igazán hatékony top-down elemzők így is csak bizonyos feltételeknek megfelelő nyelvtanok esetén készíthetőek, mint amilyenek például az $LL(k)$ nyelvtanok.

6.4. $LL(k)$ nyelvtanok

Vizsgáljuk most azt a nemdeterminisztikus felülről-lefelé elemzőt, mely csak legbal levezetéseket enged meg, kezdőszelet egyeztetés mellett (legbal, kezdőszelet egyezetetéses felülről-lefelé elemző). Ilyenkor csak front(t) legelső nyelvtani jelére engedünk meg helyettesítést, és azokat az ágakat már nem vizsgáljuk, ahol az eredmény terminálisokból álló maximális prefixe nem prefixe u-nak. A kezdőszelet egyezősége mellett persze még mindig lehet több alkalmazható szabály, algoritmusunk tehát továbbra is nemdeterminisztikus. Az lenne jó, ha tudnánk találni olyan algoritmikusan is jól kezelhető kritériumot, mely már csak legfeljebb egy ág továbbépítését engedi meg, hiszen ekkor algoritmusunk önmagában is determinisztikus lenne, nem volna szükség az időt rabló bejárásra. Általában nem adható az előbb említett célt elérő kritérium, de vannak olyan nyelvtanok, melyek esetében igen, például az $LL(k)$ nyelvtanoknál.

Egy nyelvtan $LL(k)$ nyelvtan, ha legbal, kezdőszelet egyezetetéses felülről-lefelé elemzést végezeve az u még nem egyezetetett részének első k jele alapján egy híján minden alternatíva kizárható. (Az első L az input balról jobbra olvasására, a második L a legbal levezetésre utal, míg a k jelentése világos.) Formálisan ez a következőképp fogalmazható meg:

6.1. definíció. *A G 2-es típusú nyelvtan $LL(k)$ nyelvtan, ha tetszőleges*
$S \xrightarrow[G,\mathrm{lb}]{*} vA\alpha_1 \xrightarrow[G,\mathrm{lb}]{} v\gamma_1\alpha_1 \xrightarrow[G]{*} vw_1$ *és*
$S \xrightarrow[G,\mathrm{lb}]{*} vA\alpha_2 \xrightarrow[G,\mathrm{lb}]{} v\gamma_2\alpha_2 \xrightarrow[G]{*} vw_2$
levezetések esetén abból, hogy $pre(w_1,k)$=$pre(w_2,k)$ következik, hogy $\gamma_1 = \gamma_2$.
($pre(\alpha,k)$ az α szó első k betűjét jelöli, ha volt legalább k betűje, egyébként pedig α.)

Világos, hogy minden kvázi-Greibach formában adott nyelvtan, melyben az azonos baloldalhoz tartozó alternatívák különböző jellel kezdődnek, $LL(1)$ nyelvtan. Az ilyen nyelvtanokat egyszerű $LL(1)$ nyelvtanoknak nevezzük.

Hasonlóan $LL(1)$ (bár nem egyszerű) az alábbi szabályokat tartalmazó nyelvtan:

S$\longrightarrow$ aBBB$c|b$C,

B$\longrightarrow$ bB$|a$C,

C$\longrightarrow$SS$|c$. A C-re vonatkozó bizonytalanságot ugyanis feloldja, hogy az SS első jeléből csak a-val, illetve b-vel kezdődő szavak vezethetők le.

Az alábbi(szabályaival megadott) nyelvtan esetében 2 jelre előre nézve tehetünk különbséget a B-re vonatkozó két alternatíva között:

S$\longrightarrow$ aBB$c|b$C,

B$\longrightarrow$CB$|a$C,

C$\longrightarrow$ abS$|c$, tehát a nyelvtan $LL(2)$.

Egy $LL(k)$ nyelvtan esetében elkészíthetők azok a táblázatok, melyek az elemzés pillanatnyi állapota, a legbal nyelvtani jel és az előre olvasott k jel alapján egyértelműen meghatározzák a használható alternatívát. Mindezek következtében $LL(k)$ nyelvtanok egyértelmű nyelvtanok, s így az általuk meghatározott nyelvek is azok.

6.2. állítás. *$LL(k)$ nyelvtan esetében a legbal, kezdőszelet egyezetetéses felülről-lefelé elemző valamely $u \in L$ szó levezetése során legfeljebb a nyelvtani jelek számánál eggyel kevesebb lépést tehet úgy, hogy ne hozzon be újabb terminális jelet a mondatforma elejére.*

Bizonyítás. Világos, hogy minden legbal levezetési lépés során egyértelmű, mely szabályt lehet egyátalán alkalmazni. Tegyük fel, hogy lenne u levezetése során olyan szituáció, amikor a nyelvtani jelek számával egyező lépést elvégezve az egyeztetett rész hossza nem nőne meg. Ekkor ezen lépések során - a skatulya elv miatt - volna két olyan mondatforma, amelyben az egyezetetett rész és a legbal nyelvtani jel egyezik, továbbá az input még nem egyeztetett részének első k jele is ugyanaz. De ekkor az $LL(k)$ tulajdonság miatt mindkét helyzetben csak ugyanazzal a szabállyal lehetne folytatni a levezetést, s ettől kezdve a levezetés további része periódikusan ismétlődő levezetés-szekvenciákból állna, s emiatt nem juthatnánk el u-hoz. □

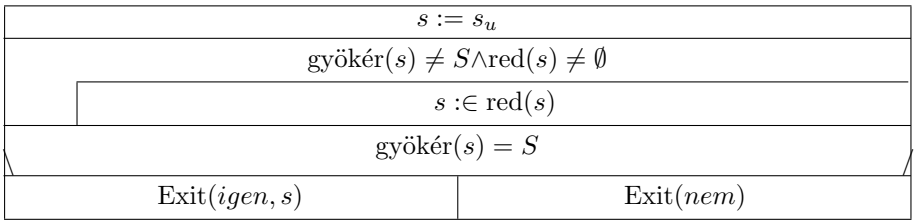
Az előző állításból következik, hogy tetszőleges $u \in L$ szó legbal levezetéséhez legfeljebb a nyelvtani jelek számaszor a szó hossza levezetési lépés kell. Emiatt ha ennyi lépésen belül nem vezettük le u-t, akkor az már nem is lesz levezethető, tehát negatív választ lehet adni.

Az előbbieket összefoglalva elmondhatjuk, hogy a legbal, kezdőszelet egyeztetéses elemző $LL(k)$ nyelvtanok esetében lineáris idejű elemzést tesz lehetővé.

A gyakorlatban az $LL(1)$ nyelvtanokat használják, melyek esetében könnyen előállíthatók azok a táblázatok, melyek valamely nyelvtani jel (mint legbal nyelvtani jel) és egy előre olvasott jel esetén megadják az alkalmazható alternatívát. Ennek vizsgálata azonban már meghaladja könyvünk kereteit.

6.5. Alulról-fölfelé módszerek

Az alulról-felfelé elemzés során a szintaxisfa rekonstrukcióját magából az elemzendő szóból kiindulva kezdjük meg. Az általános, nemdeterminisztikus bottom-up elemző algoritmus itt is nagyon egyszerű, mint azt az alábbi struktogram is mutatja. Ebben s G feletti szintaxisfák egy sorozatát jelöli (ilyen sorozatokra az erdő elnevezést használjuk a továbbiakban). Egy s erdő esetén gyökér(s) az s-beli fák gyökereinek, míg front(s) a frontjaik (s-beli sorrend szerinti) konkatenációját jelöli. s_u az az erdő, melyre front(s_u) $= u$, míg red(s) az olyan erdők halmaza, melyekben valamely fát a közvetlen leágazásaival helyettesítve s-et kapjuk (a lehetséges visszahelyettesítések eredményeinek halmaza).



Ez az algoritmus - visszafelé alkalmazva a szabályokat - valójában egy levezetés megfordítását állítja elő. A nemdeterminisztikus bottom-up elemző determinisztikussá alakítása - a lefutási fa segítségével - hasonló módon történhet, mint a top-down elemzőknél. A lefutási

fa szélességének csökkentésére most azzal próbálkozunk, hogy csak legjobb levezetéseket engedünk meg. Világos, hogy egy legjobb levezetés utolsó lépése a legtöbb esetben az eredmény elejéhez közel eső helyettesítés. Emiatt az előző algoritmust megpróbálhatnánk úgy specializálni, hogy mindig a legbaloldali visszahelyettesíthető részt helyettesítjük vissza. Sajnos azonban semmi sem garantálja, hogy az adott szó legjobb levezetésének utolsó lépése pontosan az adott szó legbaloldali visszahelyettesíthető részét adja eredményül, mint azt az alábbi példa is mutatja.

Legyen $T = \{a, b\}, N = \{\mathrm{S}\}$ és legyen

$G = \langle T, N, \mathcal{P}, \mathrm{S} \rangle$, ahol

$\mathcal{P} = \{S \longrightarrow aSb \mid a\}$.

Az $u = aaabb$ (egyetlen, tehát legjobb is) levezetésében a harmadik a betű az utolsó helyettesítéssel előálló rész, pedig a legbaloldali visszahelyettesíthető rész az első „a" lenne.

6.3. definíció. *Az olyan visszahelyettesíthető részre, mely egy legjobb mondatforma valamely legjobb levezetésének utolsó lépésében áll elő, a nyél elnevezést használjuk.*

Nem minden mondatformának van tehát nyele (csak azoknak, melyek $L(G)$-beli szó legjobb levezetése során állnak elő), míg nem egyértelmű nyelvtanokban akár több nyele is lehet ugyanannak a mondatformának.

Az alulról-felfelé elemzésnél a lefutási fa szélességét úgy szeretnénk csökkenteni, hogy gyökér(s) nyelét próbáljuk meg visszahelyettesíteni, redukálni (s így egy legjobb levezetést fordított sorrendben megtalálni). Egy mondatformából a nyele nem olvasható ki olyan könnyen, mint ahogy a legbaloldali nyelvtani jel a top-down elemzőknél. Annyi igaz ugyan, hogy egy legjobb mondatformában a nyelétől jobbra már nem lehetnek nyelvtani jelek (hiszen ekkor az eredeti levezetés már nem lehetne legjobb levezetés), de a nyél kiválasztása továbbra is nemdeterminisztikus marad, s így a lefutási fa bejárásán alapuló determinisztikus elemzők műveleti igénye (az input szó hosszának függvényében) itt is exponenciális lesz.

Ahhoz, hogy gyakorlati szempontból is használható algoritmust nyerjünk, most is kitűzhetjük célként, hogy a top-down elemzőknél látottak analógiájára kritériumokat keressünk arra, hogyan lehet a mondatformából közvetlen megállapítani, mi a nyele (ha van egyátalán). Ahogy várható, nem minden nyelvtan esetében vannak megfelelő kritériumok, de például az $LR(k)$ nyelvtanok esetében léteznek ilyenek.

6.6. $LR(k)$ nyelvtanok

Az $LR(k)$ nyelvtanok esetében tetszőleges mondatforma esetén magából a mondatformából egyértelműen megmondható, mely visszahelyettesíthető része lehet a nyele és azt mire lehet csak visszahelyettesíteni (ez utóbbi azért lényeges, mert lehetnek egyező jobboldalú szabályok). Hogy tényleg nyél volt-e ez az egyértelműen visszahelyettesítendő rész, úgy derül ki, hogy a visszahelyettesítés eredménye vagy közvetlenül S, vagy a visszahelyettesítéssel kapott mondatformára rekurzívan feltett hasonló kérdésre igen a válasz.

Magát az elemzendő u szót most is úgy képzeljük el, hogy egyszer, balról jobbra haladva olvashatjuk el. A mondatforma nyelének vége mindig a mondatforma még visszahelyettesítésekkel nem bolygatott, feldolgozatlan terminális része előtt helyezkedhet csak el (utaltunk rá, hogy a nyél után csak terminálisok lehetnek). Ha az addigi visszahelyettesítési

lépések, valamint egy adott visszahelyettesíthető rész és a mögötte lévő (még nem bolygatott) terminális sorozat első k eleme alapján egyértelműen eldönthető, hogy csak az adott rész lehet a mondatforma nyele, s hogy azt mire kell visszahelyettesíteni, akkor a nyelvtant $LR(k)$ nyelvtannak hívjuk. (Az L az input balról jobbra olvasására, az R a legjobb levezetésre utal, míg a k jelentése világos.) Világos, hogy egy $LR(k)$ nyelvtanban egyértelműen rekonstruálhatjuk tetszőleges nyelvbeli szó legjobb levezetését, emiatt minden $LR(k)$ nyelvtan egyértelmű nyelvtan is.

Formálisan az $LR(k)$ nyelvtan fogalmát következőképp definiálhatjuk:

6.4. definíció. *A G 2-es típusú nyelvtan $LR(k)$ nyelvtan, ha tetszőleges*
$S \xrightarrow[G,\mathrm{lj}]{*} \alpha_1 A v_1 \xrightarrow[G,\mathrm{lj}]{} \alpha_1\gamma_1 v_1 \xrightarrow[G,\mathrm{lj}]{*} w_1 v_1$ *és*
$S \xrightarrow[G,\mathrm{lj}]{*} \alpha_2 B v_2 \xrightarrow[G,\mathrm{lj}]{} \alpha_2\gamma_2 v_2 \xrightarrow[G,\mathrm{lj}]{*} w_2 v_2$
legjobb levezetések esetén abból, hogy $\alpha_1\gamma_1 pre(v_1,k)$ és $\alpha_2\gamma_2 pre(v_2,k)$ valamelyike kezdőszelete a másiknak, következik, hogy $\alpha_1 = \alpha_2$, $A = B$ és $\gamma_1 = \gamma_2$.

Az S$\longrightarrow a\mathrm{S}c|a$ szabályokkal adott nyelvtan nyilván $LR(1)$ nyelvtan, hiszen generált nyelvének szavai $a^{n+1}c^n, n \geq 0$ alakúak, s ebben az az a a nyél, melyet c vagy semmi sem követ. A további legjobb mondatformák nyele már egyértelműen csak $a\mathrm{S}c$ lehet.

Az S$\longrightarrow a\mathrm{S}a|a$ szabályokkal adott nyelvtan viszont nyilván nem lehet semmilyen k-ra $LR(k)$ nyelvtan, hiszen a generált nyelv a^{2n+1} alakú szavainak nyele mindig az $n+1$. (középső) a. A nyél tehát függ a szó hosszától, s ezt a hosszt rögzített számú jel előre olvasásával sehogyan sem tudjuk megállapítani.

Az $LR(k)$ nyelvtanok esetében is elkészíthetők azok a táblázatok, melyek az elemzés pillanatnyi állapota (belátható, hogy ez véges információval jellemezhető), továbbá az előre olvasott k jel alapján egyértelműen meghatározzák a visszahelyettesítések menetét. Mindezek következtében $LR(k)$ nyelvtanok egyértelmű nyelvtanok, s így az általuk meghatározott nyelvek is azok.

6.7. Kapcsolat $\mathcal{L}_{\mathrm{DIt1}\nu}$ és az $LR(k)$ nyelvek között

Az $LR(k)$ nyelvtanok definiálásának pontosan az volt a célja, hogy determinisztikus módon elemezhetők legyenek. Felmerül a kérdés, hogy mit lehet mondani $LR(k)$ nyelvtanok esetén, ha a szóproblémát veremautomata segítségével szeretnénk eldönteni. Igaz-e, hogy ilyen nyelvtanok esetében a veremautomatával történő elemzés is determinisztikusan történhet. A választ erre a kérdésre a következő, bizonyítás nélkül közölt tétel adja meg:

6.5. tétel. *Az inputterminátoros determinisztikus veremautomaták által felismert nyelvek osztálya megegyezik az $LR(k)$ nyelvtanok által generált nyelvek osztályával.*

Az $LL(k)$ és $LR(k)$ nyelvek közötti összefüggésről szól az alábbi, szintén bizonyítás nélkül közöt tétel:

6.6. tétel. *Minden $LL(k)$ nyelvtannal generálható nyelvhez létezik őt generáló $LR(k)$ nyelvtan is.*

Az $LL(k)$ és $LR(k)$ nyelvtanok részletes tanulmányozása és gyakorlati felhasználásuk bemutatása a fordítóprogramokkal foglalkozó kurzusok, illetve könyvek témája.

Irodalomjegyzék

[1] A.V. Aho, J.D. Ullman, *The Theory of Parsing, Translation, and computing*, Prentice-Hall, 1972.

[2] Arto Salomaa, *Formal Languages*, Academic Press, 1973.

[3] Révész György, *Formális nyelvek*, Tankönyvkiadó, 1977.

[4] J. Demetrovics, J. Denev, R. Pavlov, *A számítástudomány matematikai alapjai*, Tankönyvkiadó, 1995.

[5] Fülöp Zoltán, *Formális nyelvek és szintaktikus elemzésük*, Poligon, 1999.