Formális Nyelvek és Automaták

Dömösi, Pál

Falucskai, János

Horváth, Géza

Mecsei, Zoltán

Nagy, Benedek

Formális Nyelvek és Automaták

Dömösi, Pál
Falucskai, János
Horváth, Géza
Mecsei, Zoltán
Nagy, Benedek
Lektorálta: Vaszil, György

Tartalom

1. Bevezetés 2
2. Formális nyelvek 6
    2.1. Ábécé, szó, formális nyelv, szabad monoid, szabad félcsoport 6
    2.2. Nyelvműveletek 8
3. Formális rendszer és néhány főbb típusa 11
    3.1. Markov-féle normál algoritmus 15
    3.2. Generatív nyelvtan, Chomsky-féle nyelvosztályok 21
        3.2.1. Chomsky-féle nyelvosztályok 24
        3.2.2. A Chomsky hierarchia 31
    3.3. L-rendszerek 35
    3.4. Problémakörök 37
        3.4.1. Normálformák (Normálalakok) 37
        3.4.2. Levezetések szerkezete 38
        3.4.3. A szóprobléma és a szintaktikai elemzés 38
        3.4.4. Nyelvosztályok és automataosztályok kapcsolata 39
        3.4.5. Nyelvosztályok tulajdonságai 39
        3.4.6. Speciális alosztályok 39
4. A véges automaták elméletének alapjai 40
    4.1. Az automata fogalma és főbb típusai 40
        4.1.1. Mealy automata 40
        4.1.2. Moore automata 41
        4.1.3. Kimenőjel nélküli automata 41
        4.1.4. Iniciális automata 42
        4.1.5. Parciális és teljesen definiált automata 42
        4.1.6. Nemdeterminisztikus és determinisztikus automata 42
        4.1.7. Sztochasztikus automata 43
        4.1.8. Rabin-Scott automata 43
    4.2. Az automaták megadása 45
        4.2.1. Véges automaták megadása Cayley táblázattal 45
        4.2.2. Véges automaták megadása gráfokkal 47
    4.3. Az automata, mint algebrai struktúra: részautomata, homomorfizmus, izomorfizmus, kompatibilis osztályozás 49
        4.3.1. Mealy automata, mint algebrai struktúra 49
        4.3.2. Moore automata, mint algebrai struktúra 51
        4.3.3. Kimenőjel nélküli automata, mint algebrai struktúra 53
    4.4. Az automaták által indukált leképezések 53
    4.5. Redukált automata. Véges determinisztikus automaták minimalizálása 58
        4.5.1. Aufenkamp-Hohn-féle minimalizációs algoritmus 58
        4.5.2. Kiegészítés az Aufenkamp-Hohn minimalizációs algoritmushoz 59
        4.5.3. Aufenkamp-Hohn féle minimalizációs algoritmus Moore-automatákhoz adaptált változata 65
        4.5.4. Kiegészítés az Aufenkamp-Hohn-féle minimalizációs algoritmus Moore-automatákhoz adaptált változatához 65
        4.5.5. Minimalizációs algoritmus kimenőjel nélküli automatákra 69
        4.5.6. Kiegészítés a minimalizációs algoritmus kimenőjel nélküli automatákra ismertetett változatához 70
    4.6. Automaták ekvivalenciája, Gill tétele 70
    4.7. Automaták analízise és szintézise 72
    4.8. Egyéb véges automaták 72
        4.8.1. Irányítható automaták 72
        4.8.2. Automata több kezdőállapottal 73
        4.8.3. Átlátszóbetűs felismerő automata 73
    4.9. Irodalmi megjegyzések 74
5. Reguláris nyelvek 75
    5.1. Normálformák a reguláris nyelvtanokhoz 75

5.2. Reguláris kifejezések 77
5.2.1. Reguláris kifejezések ekvivalenciája 79
5.2.2. A kifejezésfa 81
5.2.3. Unió-normálforma reguláris kifejezésekhez 81
5.3. Egyszerű szintaxis gráfok 82
5.4. Véges elfogadó automaták (Rabin-Scott automaták) 83
5.4.1. Véges determinisztikus és nemdeterminisztikus automaták ekvivalenciája 87
5.4.2. Véges determinisztikus elfogadó automaták minimalizálása 92
5.4.3. Véges automaták és reguláris nyelvtanok ekvivalenciája 96
5.5. Nyelvek előállítása automatákban 103
5.6. Véges automaták analízise 106
5.7. Véges automaták szintézise 110
5.8. Véges automaták által indukálható leképezések 113
5.9. Szóprobléma 115
5.10. Iterációs lemma reguláris nyelvekre 116
5.11. Zártsági tulajdonságok 117
5.12. Irodalmi megjegyzések 121
6. Lineáris nyelvek 123
6.1. Normálforma 123
6.2. 2-fejű (véges állapotú) automata 129
6.3. A szóprobléma megoldása 133
6.4. A lineáris nyelvek tulajdonságai 136
6.4.1. Iterációs lemma 136
6.4.2. Zártsági tulajdonságok 137
6.5. Irodalmi megjegyzések 138
7. Környezetfüggetlen nyelvek 140
7.1. Programnyelvek szintaktikájának leírása 140
7.1.1. A BNF megadási mód 140
7.1.2. A COBOL-szerű megadási mód 141
7.1.3. A szintaxis gráf 141
7.1.4. A Hibrid megadási mód 142
7.2. Levezetési fa 142
7.2.1. Többalakúság 143
7.3. Normálformák a környezetfüggetlen nyelvtanokhoz 144
7.3.1. Chomsky-féle normálalak 144
7.3.2. Greibach normálforma 148
7.4. A Bar-Hillel lemma 151
7.5. Veremautomaták 156
7.6. Zártsági tulajdonságok 174
7.7. A szóprobléma megoldása - szintaktikai elemzés 175
7.7.1. A CYK algoritmus 176
7.7.2. Az Earley-féle algoritmus 182
7.8. Irodalmi megjegyzések 188
8. Környezetfüggő nyelvek 190
8.1. Szóhossznemcsökkentő (monoton) nyelvtanok 190
8.2. Normálformák a környezetfüggő nyelvtanokhoz, a monoton és a környezetfüggő nyelvtanok ekvivalenciája 190
8.3. Permutációs nyelvek 196
8.4. Levezetési gráfok, levezetési fák a környezetfüggő nyelvtanokhoz 197
8.4.1. Legbaloldalibb levezetés és átfogalmazása 200
8.5. Árnyék-veremautomata 200
8.6. Lineárisan korlátozott automata 202
8.7. A környezetfüggő nyelvek tulajdonságai 203
8.7.1. A szóprobléma 203
8.7.2. Zártsági tulajdonságok 204
8.8. Növekvő környezetfüggő nyelvek 205
8.9. Irodalmi megjegyzések 205
9. Rekurzívan felsorolható nyelvek és Turing-gépek 206

9.1. A Turing-gép 206
9.2. A Turing-gépek megállási problémája és az algoritmikusan eldönthetetlen feladatosztályok kapcsolata 215
9.2.1. Univerzális Turing-gép 216
9.3. A szóprobléma - rekurzív és rekurzívan felsorolható nyelvek 218
9.3.1. Bonyolultsági osztályok 220
9.4. Normálformák a mondatszerkezetű nyelvtanokhoz 222
9.4.1. Révész-féle normálalak 223
9.4.2. Geffert-féle normálformák 223
9.5. Zártsági tulajdonságok 224
9.6. Irodalmi megjegyzések 225
10. Nyelvtanrendszerek (Grammatikarendszerek) 226
10.1. Nyelvtanok kooperatív elosztott rendszerei – CD nyelvtanrendszerek 226
10.2. Nyelvtanok párhuzamos kommunikáló rendszerei – PC nyelvtanrendszerek 228
10.3. Irodalmi megjegyzések 230
11. Irodalomjegyzék 231

Colophon

A tananyag a TÁMOP-4.1.2-08/1/A-2009-0046 számú Kelet-magyarországi Informatika Tananyag Tárház projekt keretében készült. A tananyagfejlesztés az Európai Unió támogatásával és az Európai Szociális Alap társfinanszírozásával valósult meg.

1. fejezet - Bevezetés

Az algoritmikus szemlélet kialakulására nagy hatással volt az 1900-ban Párizsban rendezett első matematikai világkongresszus, ahol is többek között elhangzott David Hilbert híres előadása, melynek hatására tág teret kapott az absztrakt gondolkodásmód széleskörű alkalmazása.

Hilbert még úgy vélte, hogy létezik olyan univerzális módszer, melynek segítségével minden matematikai kérdés igaz vagy hamis volta eldönthető.

A modern elméleti számítástudomány egyik fontos fogalma a formális rendszer, melynek bevezetése Axel Thue (ejtsd: tyué) nevéhez fűződik, aki az 1910-es évek elején kezdte tanulmányozni az adatsorozatokkal megadott utasítások tulajdonságainak vizsgálatát.

A fejlődésnek e téren is újabb lökést adtak a fiatal Kurt Gödel felfedezései, aki a 20-as, illetve 30-as években kimutatta, hogy létezik olyan precízen megfogalmazható matematikai állítás, melynek sem igaz, sem hamis volta nem bizonyítható. Ez a korszakalkotó felfedezés alapjaiban rengette meg a matematikát. Ezután már senki sem lehetett biztos abban, hogy egy tetszőlegesen kiválasztott matematikai állítás igaz vagy hamis voltának eldöntése lehetséges. Erre csattanós példa volt az ugyancsak ifjú Jurij V. Matijaszevics esete, aki a 70-es évek elején kimutatta Hilbert már említett híres előadásában szereplő egyik kérdésről, hogy eldönthetetlen. (Hilbert 10. problémája.)

A 30-as évek második felében egy másik fiatal óriás, Alan Turing (ejtsd: tyuring) vizsgálataival párhuzamosan, Alonzo Church (ejtsd: csörcs) kapott hasonló módon meglepő eredményt. Nevezetesen, hogy nem létezik univerzális feladatmegoldó módszer.

A gépies bizonyítások vizsgálata elvezetett az automaták matematikai elméletének 50-es évekre kialakult megalapozásához, az idegen szövegek gépi fordításának vizsgálata pedig ugyancsak az 50-es években a formális rendszerek egy fontos típusa, a formális nyelvek matematikai elméletének kezdeteihez. Ezen a területen Noam Chomsky (ejtsd: csomszki) alapvető felfedezései jelentettek nagy lépéseket.

Végül, de nem utolsósorban meg kell említenünk, hogy az automaták matematikai elméletének megalapozásában nagy szerepet játszott a pályafutásának nagy részét az Egyesült Államokban töltő magyar származású nagy tudós, Neumann János, aki John von Neumann néven szerte a világon mint a számítástudomány atyja ismeretes. 1960-ban bekövetkezett korai halála sajnos megakadályozta abban, hogy az e téren (is) végzett úttörő kutatásait teljeskörűen megalapozza.

Már Neumann János is látta, hogy a bonyolultság és az azzal kapcsolatos kérdések nagy szerepet fognak játszani a számítástudományban. Valóban, a számítógépek 60-as évektől bekövetkezett széleskörű elterjedésével mind az elméleti, mind pedig a gyakorlati szakemberek azt tapasztalták, hogy bizonyos kérdéseket még a legmodernebb számítástechnikai eszközök felhasználásával sem képesek kezelni. Ezek a problémák nemcsak elméleti, hanem nagyon fontos gyakorlati feladatokkal kapcsolatban is felléptek. Hamarosan kiderült, hogy ez a gond nem fog megoldódni a számítógépek rohamos fejlődésével sem, mivel valószínűleg elvi akadályokról van szó. A kérdéses feladatok túl bonyolultnak látszanak ahhoz hogy kezelni tudjuk őket. A 70-es évek elején ezen kérdések kezelésére Stephen A. Cook (ejtsd: kuk), Richard M. Karp és Leonid A. Levin egy új elmélet megalapozását kezdeményezték. Megszületett az algoritmusok bonyolultságelmélete. Az eltelt több mint 30 év alatt azonban sajnos ez az elmélet nem adott választ a feltett legégetőbb kérdésekre. Ezek közül is a legfontosabb, az úgynevezett P = NP? probléma, melynek megoldása választ adna arra a nagyon fontos kérdésre, hogy egyes, a gyakorlatban is lépten-nyomon felmerülő feladatokhoz létezik-e hatékony megoldó algoritmus. Ez a több mint 30 éves nagy kérdés valószínűleg még sokáig megoldatlan marad.

A bonyolultságelmélet által felvetett problémák a 80-as évek közepére-végére elvezettek egy új terület, az úgynevezett kooperatív rendszerek megszületéséhez. Kezdetben csak arról volt szó, hogy olyan számítástechnikai eszközök születtek, melyek egyes feladatok egyidejű, párhuzamos végrehajtására voltak képesek. Ma már a kooperatív rendszerek széleskörűen elterjedtek és mind elméleti, mind pedig gyakorlati szempontból egyre nagyobb teret kapnak olyan berendezések és módszerek, amik eddig

nem, vagy csak csekély mértékben voltak használatosak. Talán ezen az úton haladva fogunk eredményt elérni a még mindig megközelíthetetlennek látszó problémák kezelésében.

A formális nyelvek és automaták elmélete kiváló eszköznek bizonyult ezen kérdések absztrakt szintű vizsgálatában. Jelen jegyzet célja ezen elmélet legalapvetőbb kérdéseinek megismertetése. Az ismertetett módszerek ma már lépten-nyomon használatosak nemcsak elméleti problémák vizsgálatában, hanem egyes gyakorlati számítástechnikai feladatok megoldásában is. A formális nyelvek és automaták elmélete nemcsak olyan kézenfekvő területeken nyert alkalmazást, mint a számítógépes nyelvészet, illetve a fordítóprogramok területe, hanem ezen keresztül a programozási nyelvek, operációs rendszerek témakörében is. Például a sztringalgoritmusokon keresztül keresőalgoritmusokban, bioinformatikában, mintaillesztésben, adatbányászatban stb. is alkalmazzák. A biológiailag motivált számítások, a DNS és membrán számítások formális modelljei szintén erősen kötődnek a formális nyelvek és automaták klasszikus elméletéhez. A képfeldolgozásban, adatsűrítésben, rendszermodellezésben szintén felhasználhatjuk az itt tanultakat. Helyenként nagy figyelmet fordítunk annak igazolására is, hogy bizonyos struktúrák és módszerek nem léteznek, illetve nem lehetségesek. Ez a fáradtság a gyakorlati számítástechnikai szakember számára feleslegesnek tűnik. Jelen jegyzet szerzői, akik egyike több mint másfél évtizedig gyakorlati számítástechnikai szakemberként dolgozott, másképp gondolják. Ha ismerjük a leggyakoribb számítástechnikai zsákutcákat, akkor könnyebben ki tudjuk kerülni azokat. Ha nem, akkor úgy járhatunk mint ahogy a jegyzet nevezett szerzője is járt nem egy esetben: esetleg olyat próbálunk keresni, amiről már kiderült, hogy nincs.

Jelen jegyzet elméleti fejezeteit Dömösi Pál (pl. Automataelmélet) és Nagy Benedek (pl. Lineáris nyelvek, Környezetfüggő nyelvek, Nyelvtanrendszerek) írták (a többi fejezetet együttesen). A feladatok döntő többségét és az animációkat Falucskai János, Horváth Géza és Mecsei Zoltán készítették.

A feladatok, példák és definíciók végét a ★ jellel jelöljük. A bizonyítások végét pedig ∎-el zárjuk.

A jegyzet egyes főbb fejezeteinek felépítése a következő összefoglaló segítségével is követhető:

Nyelvosztály	Nyelvtan	Normálformák	Automataosztály	Szóprobléma	Zártsági tulajdonságok	Egyéb
Reguláris nyelvek Chomsky 3. típus pl: egész számok halmaza Alosztályok: - véges nyelvek - uniómentes nyelvek	- jobb-lineáris : $A \to uB$ $A \to u$ - bal-lineáris: $A \to Bu$ $A \to u$ $A,B \in N$ $u \in T^*$	- gyenge [75]: $A \to aB$ $A \to a$ $A \to B$ $A \to \lambda$ - erős [76]: $A \to aB$ $A \to a$ $a \in T$ $A,B \in N$	véges automaták: - nemdeterminisztikus üresszóátmenetes [83] - nemdeterminisztikus [83] - determinisztikus parciális [83] - determinisztikus teljesen definiált [83] Minimális determinisztikus automata: Myhill-Nerode tétel [105] Kapcsolódó terület: Automataelmélet	determinisztikusan, lineáris időben megoldható	zárt a reguláris- és a halmazműveletekre	alternatív leírás: - reguláris kifejezés - szintaxis gráf pumpáló lemma [116]
Lineáris nyelvek pl: palindrómák nyelve Alosztályok: - Determinisztikus automatával: 2detLin detLin - Fix fokú lineáris: k-fokú lineáris bal-, ill. jobb-lineáris [129] Kiterjesztés: - metalineáris nyelvek	$A \to uBv$ $A \to u$ $A,B \in N$ $u,v \in T^*$	- gyenge [123]: $A \to aB$ $A \to Ba$ $A \to a$ $A \to B$ $A \to \lambda$ - erős [123]: $A \to aB$ $A \to Ba$ $A \to a$	- nemdeterminisztikus 2-fejű véges automaták - nemdeterminsztikus egyfordulós veremautomaták [164] (determinisztikus verziók alosztályokat definiálnak)	- determinisztikusan négyzetes algoritmus - nemdeterminisztikusan lineáris	- zárt az unióképzésre - nem zárt a konkatenáció, Kleene csillag, metszet, komplementerképzés műveletekre	pumpáló lemma
Környezetfüggetlen Chomsky 2. típusú pl: Zárójelek nyelve [160] Alosztályok: - determinisztikus CF [164] - számlálónyelvek Kiterjesztés: pl. CD és PC rendszerek	$A \to p$ $A \in N$ $p \in (N \cup T)^*$	- Chomsky: $A \to BC$ $A \to a$ - Greibach: $A \to aq$ $A,B,C \in N$ $a \in T$ $q \in N^*$	nemdeterminisztikus veremautomata - üres veremmel [158] - végállapottal - üresveremmel és végállapottal - állapotnélküli (üres veremmel) [161]	- CYK, Earley algoritmusok determinisztikus köbös időben - nemdeterminisztikusan lineáris időben (Greibach NF)	-zárt: reguláris műveletekre (unió, konkatenáció, Kleene-csillag) -nem zárt: metszetre és komplementerre	pumpáló lemma (Bar-Hillel lemma) Levezetési fa fogalma Legbaloldalibb levezetés [143] alternatív leírás: BNF, szintaxis gráf

Nyelvosztály	**Nyelvtan**	**Normálformák**	**Automataosztály**	**Szóprobléma**	**Zártsági tulajdonságok**	**Egyéb**
Környezetfüggő Chomsky 1. típusú pl: prímszámhosszú szavak nyelve [203] Alternatív definíció: Monoton Alosztályok: - Permutációs - Növekvő környezetfüggő	$pAq \to prq$ $p,q \in (N \cup T)^*$ $A \in N$ $r \in (N \cup T)^+$ kivéve $S \to \lambda$, de ekkor S nem szerepelhet egyetlen szabály jobb oldalán sem monoton: $u \to v$ $\lvert u \rvert \leq \lvert v \rvert$ kivéve $S \to \lambda$ a fenti feltétellel	Kuroda [190]: $AB \to CD$ $A \to BC$ $A \to B$, $A \to a$ Révész [195]: $AB \to AC$ $AB \to CB$ $A \to BC$ $A \to B$, $A \to a$ Penttonen [195]: $AB \to AC$ $A \to BC$ $A \to a$	nemdeterminisztikus árnyékveremautomata [200] nemdeterminisztikus lineárisan korlátolt automata	megoldható (PSPACE-teljes, nincs hatékony algoritmus)	zárt a reguláris és a halmaz-műveletekre is: konkatenáció, Kleene csillag unió, metszet, komplementer	Üresszó lemma Pumpáló lemma nincs. Levezetési gráf, levezetési fa és legbaloldalibb levezetés általánosítása.
Rekurzívan felsorolható (van olyan algoritmus, amivel a nyelv összes szava felsorolható.) Chomsky 0. típus Alosztály: Rekurzív [219] (Egy nyelv rekurzív, ha L és L komplementere is rekurzívan felsorolható.)	generatív nyelvtan	Révész: $AB \to AC$ $AB \to CB$ $A \to BC$ $A \to B$ $A \to a$ $AB \to B$ Penttonen [222] $AB \to AC$ $A \to BC$ $A \to a$ $A \to \lambda$ Geffert, pl: $S \to p$ $AB \to \lambda$ $CD \to \lambda$ $S \to p$ $ABC \to \lambda$	Turing gép - determinisztikus [208] - nemdeterminisztikus (egyéb változatok: több szalagos [209], csak egyirányba végtelen szalag) univerzális Turing gép kapcsolódó terület: kiszámíthatóság és bonyolultságelmélet, a Neumann-féle számítógép elméleti modellje	algoritmikusan nem megoldható (megállási probléma)	zárt a reguláris műveletekre: konkatenáció, Kleene csillag, unió. zárt a metszet képzésre nem zárt a komplementer képzésre	tárhely tétel [223], Markov algoritmus

2. fejezet - Formális nyelvek

2.1. Ábécé, szó, formális nyelv, szabad monoid, szabad félcsoport

Szimbólumok tetszőleges nemüres, véges halmazát *ábécé*nek nevezzük, és V-vel jelöljük. A V elemeit az ábécé *betű*inek mondjuk.

Jelölje V^+ (ejtsd: véplusz) a V-beli betűkből felírható $p=a_1a_2\ldots a_k$ ($a_1,\ldots,a_k \in V$) alakú véges hosszúságú sorozatok, az úgynevezett V feletti nem üres szavak halmazát. Egy p szó hosszát $|p|$-al (ejtsd p hossza) jelöljük és rajta a p-beli betűk számát értjük (esetleges többszöri előfordulással együtt). Így ha $p=a_1a_2\ldots a_k$ ($a_1,\ldots,a_k \in V$), akkor $|p| = k$.

2.1. példa - Szó hossza

Legyen a V ábécé két betűje a és b. Ekkor $bab \in V^+$ és $|bab| = 3$ (és nem 2, mert a b betű kétszer is előfordul, és a többszöri előfordulást figyelembe kell venni.) ★

Szokás beszélni az úgynevezett *üresszó*ról is, ami egy matematikai absztrakció, ugyanis olyan szót jelent, melynek egyetlen betűje sincs, vagyis az egyetlen betűt sem tartalmazó betűsorozatot. Speciálisan, jelölje λ a továbbiakban az üresszót. Itt jegyezzük meg hogy bár az automataelméleti szakirodalomban az ε-t (is) szokás az üresszó jelölésére használni, ebben a jegyzetben mi végig a λ jelet fogjuk használni. Tehát $|\lambda| = 0$. A $V^+\cup\{\lambda\}$ halmazt a *V feletti (összes) szavak halmazának* hívjuk, s V^*-al (ejtsd vécsillag) jelöljük. Speciálisan, ha valamely $a \in V$-re $V = \{a\}$ azaz V egyelemű halmaz, s egyetlen eleme egy bizonyos a betű, akkor gyakran írunk $\{a\}^*$ helyett a^*-ot, illetve $\{a\}^+$ helyett a^+-t. Valamely V^*-beli $p=a_1\ldots a_m$ és $q=b_1\ldots b_n$ ($a_1,\ldots,a_m,b_1,\ldots,b_n \in V$) szavakat pontosan akkor tekintjük egyenlőknek, ha $m = n$ és minden $i = 1,\ldots,n$-re $a_i = b_i$. Ezt a tényt ki szokás úgy is fejezni, hogy V^*-ban csak *grafikus* (betűről - betűre megegyező) *egyenlőségek* léteznek.

A V ábécé feletti szavak egy tetszőleges L halmazát a V ábécéből alkotott *(formális) nyelv*nek nevezzük, vagyis a V^* halmaz részhalmazait *V feletti formális nyelveknek*, vagy röviden *V feletti nyelveknek*, vagy csak egyszerűen *nyelveknek* hívjuk. Valamely $L \subseteq V^*$ nyelvet *üresnek, végesnek vagy végtelennek hívunk* ha az L (mint halmaz) üres, véges, illetve végtelen.

Azt a nyelvet, amelynek egyetlen szava sincs, *üres nyelv*nek nevezzük. Jelölés: $\emptyset$. Nem tévesztendő össze a $\{\lambda\}$ nyelvvel, amely egyedül az üresszót tartalmazza.

Az így definiált nyelvfogalom túl általános, magában foglalja mind a mesterséges, mind a természetes (írott) nyelvek összességét. A kérdés viszont az, hogyan lehet ténylegesen megadni egy konkrét nyelvet. Az egyszerű tulajdonságokkal rendelkező nyelveket máris megadhatjuk a halmazok megadásának különböző módjai szerint. Legyen például a véges ábécénk mindössze két elemű: $V=\{0,1\}$. Ekkor az

- $L_1 = \{\lambda\}$,
- $L_2 = \{1, 10, 0010, 111\}$,
- $L_3 = \{1^i \mid i \text{ prím}\}$

halmazok mindegyike egy-egy nyelv a fenti definíció értelmében. Szükségünk van azonban olyan eszközökre, amelyekkel a fentieknél lényegesen összetettebb nyelveket is definiálhatunk. Ebből a célból vezetjük be a generatív nyelvtan fogalmát.

Tehát a továbbiakban olyan nyelveket fogunk csak vizsgálni, melyek véges sok adat segítségével speciális módon, az úgynevezett generatív nyelvtanokkal megadhatók. Megjegyezzük azt is, hogy az általunk használt szófogalom nem esik egybe a természetes nyelvek szófogalmával, hisz egy

természetes nyelvet úgy szokás tekinteni, mint az adott nyelv összes mondatainak halmazát. De tekinthetünk egy természetes nyelvet úgy is mint a nyelv összes véges hosszú szövegeinek halmazát. A szóközt és egyéb írásjeleket is felvéve az ábécébe, az általunk definiált szófogalom amellett, hogy magában foglalja a szokásos szófogalmat, magában foglalja mind a mondat fogalmát, mind pedig a tetszőleges véges hosszúságú szöveg fogalmát is.

2.2. példa - Szavak egyenlősége

Legyen $V = \{ 1, 2, + \}$. Ekkor (V^*- ban !) fennáll, hogy $1+1 \neq 2$, mivel az 1+1 szó nem egyezik meg "betűről - betűre", azaz grafikusan a 2 szóval. ★

2.3. példa - Véges és végtelen nyelvek

Legyen $V = \{ 0,1,\ldots,9\}$. A (magyar) történelmi dátumok { 1514, 1526, 1606, 1711, 1849, … } halmaza ekkor egy V feletti véges nyelv. A (tízes számrendszerbeli) páros számok halmaza egy V feletti végtelen nyelv. Temészetesen az *üres halmaz* is egy V feletti (üres) nyelv. Ugyancsak V feletti (véges) nyelv az egy elemű $\{ \lambda \}$ halmaz is. ★

A V^* halmazon (és a V^+ halmazon is) szokás bevezetni egy (nagyon egyszerű tulajdonságú) műveletet, melyet *szorzásnak* nevezünk.

A $p = a_1 \ldots a_m$ és $q = b_1 \ldots b_n$ ($a_1, \ldots, a_m, b_1, \ldots, b_n \in V$) szavak *szorzatán* a $pq = a_1\, a_2 \ldots a_m\, b_1\, b_2 \ldots b_n$ szót értjük. Két V^*- beli (vagy két V^+- beli) szót tehát úgy szorzunk össze, hogy e szavakat (megfelelő sorrendben) egymás mellé (után) írjuk. E műveletet *konkatenációnak* vagy *összefűzésnek* is szokás hívni. Természetesen ez a szorzásfajta általában *nem kommutatív*, azaz általában nem teljesül minden $p, q \in V^*$ párra a $pq = qp$ egyenlőség. Amennyiben $p = p_1 \ldots p_k$ ($\in V^*$ $k = 1,2,\ldots$) továbbá $p_1 = p_2 = \ldots = p_k = q$ úgy alkalmazni fogjuk a $p = q^k$ jelölést, s ezesetben p- t a q szó *k-adik hatványának* is nevezzük. Tehát a q szó k- adik *hatvány*án az önmagával vett k- szoros konkatenációját értjük. Továbbá megállapodunk abban, hogy minden szó *nulladik hatványa* az üresszó (jelekben: $\forall\, q \in V^* : q^0 = \lambda$.) Röviden úgy is mondhatjuk, hogy egy szó k- adik hatványa nem más mint k- szoros ismétlődése (beleértve a nullaszoros ismétlődést is).

2.4. példa - Szavak konkatenációja

Legyen ismét $V = \{ ab\}$. Ekkor az *abba* V^*- beli szó *baba* V^*- beli szóval való szorzata *abbababa* lesz (ami persze nem egyezik meg a *babaabba* szóval. A szorzás tehát valóban, általában nem kommutatív). Igaz továbbá (definíció szerint), hogy $baba = (ba)^2$. ★

2.5. példa - Szavak konkatenációja egyelemű ábécé felett

Legyen most $V = \{ a\}$, azaz álljon ábécénk egyetlen betűből. Mutassuk meg hogy ebben a kivételes esetben a szorzás kommutatív. ★

A V^*-ban ezen szorzás műveletre nézve a λ üresszó egységelem lesz, hisz minden $p \in V^*$-ra $p\lambda = \lambda p = p$ (annak megfelelően, hogy ha egy szó - beleértve az üresszót is - elé vagy mögé nem írunk egyetlen betűt sem, azaz az üresszót "írjuk", akkor marad az eredeti szó). Nyilvánvaló továbbá, hogy minden $p,q \in V^*$ párra $| pq | = | p | + | q |$.

Az előbbiekben definiált szorzás műveletével ellátott V^* halmazt a *V által generált egységelemes szabad félcsoportnak*, más néven *V feletti egységelemes szabad félcsoportnak*, vagy röviden, *V feletti szabad monoidnak* hívjuk, s rá ugyancsak a V^* jelölést használjuk.

Hasonlóan, a szorzás műveletével ellátott V^+ halmazt a *V által generált szabad félcsoportnak*, más néven *V feletti szabad félcsoportnak* hívjuk, s rá ugyancsak a V^+ jelölést használjuk. (V^* tehát egyidejűleg jelöli az összes V feletti szavak halmazát és a V feletti szabad monoidot, míg V^+ az összes V feletti nem üres szavak halmazát és a V feletti szabad félcsoportot. Amennyiben tehát algebrai struktúraként tekintünk a V^*-ra (V^+-ra), akkor a konkatenáció az alapértelmezett művelet az elemei közt.)

Nyilvánvaló, hogy V^+ zárt marad a szorzás, azaz az összefűzes műveletére, hisz két nem üresszót egymás után írva, azaz összefűzve, nem kaphatunk üresszót.

Legyen p és q tetszőleges két szó V^*-ban. Azt mondjuk, hogy p *kezdőszelete (prefixe)* q-nak, ha van olyan $r \in V^*$, hogy $q = pr$. Pontosabban, ha $q = pr$ mellett $|p| = k$ akkor a p szót a q szó k hosszúságú kezdőszeletének nevezzük (ekkor, ha $0 < k < |q|$, akkor p a q *valódi kezdőszelete*). Hasonlóan, ha van olyan $s \in V^*$ szó, hogy $q = sp$, akkor azt mondjuk hogy p a q-nak *végződése (szuffixe)* és ha $|p| = m$ akkor m hosszúságú végződésről beszélünk (ekkor ha $p \neq \lambda$ és $p \neq q$ akkor p valódi végződése a q-nak). Végül, a $p \in V^*$ szót a $q \in V^*$ szó *részszavának* mondjuk, ha van olyan $r, s \in V^*$, hogy $q = rps$. Amennyiben $p \neq qp \neq \lambda$, *valódi rész-szóról* beszélünk. Tehát, egy szó önmagától különböző nemüres kezdőszeleteit, végződéseit, illetve részszavait valódi kezdőszeletnek, valódi végződésnek, illetve valódi részszónak hívjuk.

Egy $p \in V^+$ szó utolsó betűjére használni fogjuk a $\gg p$ jelölést.

2.6. példa - Szó részszava

Legyen $V = \{ a, b\}$, $p = abbababa$. Ekkor p-nek 4 hosszúságú kezdő szelete *abba*, 4 hosszúságú végződése pedig *baba* lesz. Ugyanekkor p-nek például a *bab* szó (valódi) rész-szava. Végül, $\gg p = a$ (*abbababa* utolsó betűje). ★

2.2. Nyelvműveletek

Két nyelv között akkor értelmezhetünk valamilyen műveletet, ha ugyanazon V ábécé felett vannak értelmezve. (Ha ez nem teljesül, akkor először a nyelveket kell átdefiniálni formálisan egy közös ábécé, pl. a két ábécé uniója felettire.)

Formális nyelvekre, mint szóhalmazokra közvetlenül értelmezhetők a halmazelméleti alapműveletek:

unió (egyesítés, összeadás): $L_1 \cup L_2 = \{ p \mid p \in L_1 \text{ vagy } p \in L_2\}$,

metszet: $L_1 \cap L_2 = \{ p \mid p \in L_1 \text{ és } p \in L_2\}$,

különbség: $L_1 \setminus L_2 = \{ p \mid p \in L_1 \text{ és } p \notin L_2\}$,

komplementer: $\overline{L_1} = V^* \setminus L_1$.

Amint látjuk a komplementerképzésnél nagyon fontos az alaphalmaz, vagyis az ábécé ismerete, hiszen pl. az $L = \{ ab, aa, ba, bb \}$ nyelv komplementere telejsen más, ha L- et a $V = \{ a, b\}$, vagy a $V' = \{ a, b, c \}$ ábécé felettinek definiáltuk. Ennek megfelelően a továbbiakban is mindig úgy tekintünk egy nyelvre, mint egy adott V ábécé feletti nyelvre (akkor is ha ezt a tömörség kedvéért nem írjuk oda).

A nyelveken a halmazműveleteken kívül a konkatenációt és az iterációt alapműveleteknek tekintjük: Két nyelv *konkatenáció*ján a következő nyelvet értjük: $L_1 \cdot L_2 = \{ pq \mid p \in L_1 \text{ és } q \in L_2\}$. Szokásos módon a konkatenáció jelét sokszor elhagyjuk, pl. $L_1 L_2$. Legyen $i = 1, 2, \ldots$. Ekkor egy L nyelv i-edik *hatvány*án a nyelv i-szer egymás utáni, önmagával való konkatenációját értjük. Jelölés: L^i. Megállapodás szerint $L^0 = \{ \lambda \}$.

A *konkatenáció lezárás*át (Kleene iterációt) (ejtsd: klíni) az $L^* = \bigcup_{i=0}^{\infty} L^i$ összefüggéssel értelmezzük.

Az előző Kleene-csillag művelet mellett szokás még az $L^+ = \bigcup_{i=1}^{\infty} L^i$ iteráció, a Kleene-plusz használata is.

A kétféle iteráció között az $L^0 = \{ \lambda \}$ jelenthet különbéget: A definiáló egyenlőségek alaján belátható, hogy $L^+ = L^*$ pontosan akkor, ha $\lambda \in L$. Továbbá $L^+ = L^* \setminus \{ \lambda \}$ pontosan akkor, ha $\lambda \notin L$.

Mejegyzés. Mivel a V ábécé mint halmaz egyben tekinthető egy formális nyelvnek is, a korábbi V^* és a V^+ jelölés éppen egybeesik a most definiált iteráció műveletek V-re való alkalmazásának eredményével.

Legyen V ábécé rögzített, ekkor igazak az alábbi összefüggések (tetszőleges $L_1, L_2, L_3 \subseteq V^*$ esetén):

- $L_1 \cup L_2 = L_2 \cup L_1$ (az unió kommutatív),
- $(L_1 \cup L_2) \cup L_3 = L_1 \cup (L_2 \cup L_3)$ (az unió asszociatív),
- $L_1 \cup L_1 = L_1$ (az unió idempotens),
- $L_1 \cap L_2 = L_2 \cap L_1$ (a metszet kommutatív),
- $(L_1 \cap L_2) \cap L_3 = L_1 \cap (L_2 \cap L_3)$ (a metszet asszociatív),
- $L_1 \cap L_1 = L_1$ (a metszet idempotens),
- $(L_1 \cdot L_2) \cdot L_3 = L_1 \cdot (L_2 \cdot L_3)$ (a konkatenáció asszociatív),
- $\overline{(\overline{L_1})} = L_1$ (a komplementerképzés involúciós tulajdonsága),
- $L_1 \cdot \emptyset = \emptyset \cdot L_1 = \emptyset$,
- $L_1 \cdot \{ \lambda \} = \{ \lambda \} \cdot L_1 = L_1$ (a konkatenáció egységeleme a $\{ \lambda \}$ nyelv),
- $L_1 \cup \emptyset = \emptyset \cup L_1 = L_1$ (az unió egységeleme az $\emptyset$ nyelv),
- $L_1 \cap V^* = V^* \cap L_1 = L_1$ (a metszet egységeleme a V^* univerzális nyelv),
- $L_1^+ = L_1 \cdot L_1^* = L_1^* \cdot L_1$,
- $L_1^* = L_1^+ \cup \{ \lambda \}$,
- $(L_1^*)^* = L_1^*$ (az iteráció idempotens tulajdonsága),
- $(L_1^+)^+ = L_1^+$ (a + művelet idempotens tulajdonsága),
- $(L_1^*)^+ = (L_1^+)^* = L_1^*$.

Az unió, metszet, illetve konkatenáció asszociativitása miatt általában nem is szoktuk zárójelekkel jelezni a műveleti sorrendjüket, így egyszerűen pl. $L_1 \cup L_2 \cup L_3$ alakot használunk.

További zárójeleket hagyhatunk el megtartva az egyértelmű jelentést a következő precedencia reláció bevezetésével. Az egyargumentumú műveletek (komplementer, (Kleene-)csillag és (Kleene-)plusz) precedenciája nagyobb, mint a kétargumentumúaké. A szorzás (konkatenáció) precedenciája nagyobb, mint az unió és metszet műveleteké.

Legyen adva két véges ábécé, V_1 és V_2. A V_1^*-nak a V_2^*-ba való h leképezését *homomorfizmus*nak nevezzük, ha:

- injektív, vagyis az értelmezési tartomány minden egyes eleméhez az értékkészletnek pontosan egy eleme van hozzárendelve,
- és művelettartó, azaz $h (p\, q) = h (p)\, h (q)$, tetszőleges $p, q \in V_1^*$-ra.

A fenti két tulajdonságból rögtön következik, hogy az üresszó képe az üresszó lesz, ugyanis $h (p) = h (p\, \lambda) = h (p)\, h (\lambda)$ minden $p \in V_1^*$ szóra.

Egy homomorf leképezést λ- *mentes*nek nevezünk, ha $h (p) = \lambda$ esetén $p = \lambda$.

2.7. példa - Nyelvműveletek - Gyakorló feladat

Legyen $V = \{ a, b, c \}$, $L_1 = \{ a, c, bb, aba \}$, $L_2 = \{ a, abba, baba, caba, abbaba, babaabba \}$. Adjuk meg az $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1\, L_2$, $L_1\, L_1$ halmazokat. ★

2.8. példa - Nyelvek konkatenációja

Adjunk példát olyan L_1 és L_2 V ábécé feletti nyelvekre, amelyekre $L_1\ L_2 = L_2\ L_1$. Keressünk nem triviális megoldást is.

Triviális megoldások:

- $L_1 = \emptyset$, $L_1 = \{\ \lambda\ \}$ vagy a szimmetria miatt L_2-re teljesül az előző esetek egyike.
- $L_1 = L_2$.
- V ábécé egyelemű.
- Az egyik nyelvben benne szerepel λ, a másik nyelv pedig a V^* (univerzális nyelv).

Egy nem triviális megoldás:

legyen $V = \{\ a, b\}$, $L_1 = \{\ \lambda, a\}$, L_2 pedig legyen azon V feletti szavak halmaza, amelyekben pontosan egy b szerepel. Ekkor $L_1\ L_2 = L_2\ L_1 = L_2$. ★

2.9. példa - Nyelvek számossága - Gyakorló feladat

Adottak L_1 és L_2 véges nyelvek V ábécé felett, hogy $|\ L_1\ | = n$, $|\ L_2\ | = m$. Mennyi lehet a számossága az $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1\ L_2$ nyelvműveletekkel előálló nyelveknek? Adjunk meg alsó felső korlátot, és példákat. ★

2.10. példa - Formális nyelvek, nyelvműveletek 1.feladat

Igazoljuk vagy cáfoljuk, hogy $(L_1 \cup L_2\ )^* = L_1^* \cup L_2^*$!

Megoldás: Az állítás hamis. Vegyük a következő ellenpéldát: Legyen $L_1 = \{\ a\}$ és $L_2 = \{\ b\}$, ekkor $(\ L_1 \cup L_2\ )^*$ az akárhány a-t és b-t tartalmazó szavak halmaza, még $L_1^* \cup L_2^*$ a csupa a-t és csupa b-t tartalmazó szavak nyelve lesz. ★

2.11. példa - Formális nyelvek, nyelvműveletek 2.feladat

Mivel egyenlő L^2, ha

$L = \{\ a^n\ b^n | n > 0\ \}$?

Megoldás: $L^2 = \{\ a^n\ b^n\ a^m\ b^m \mid n, m > 0\}$. ★

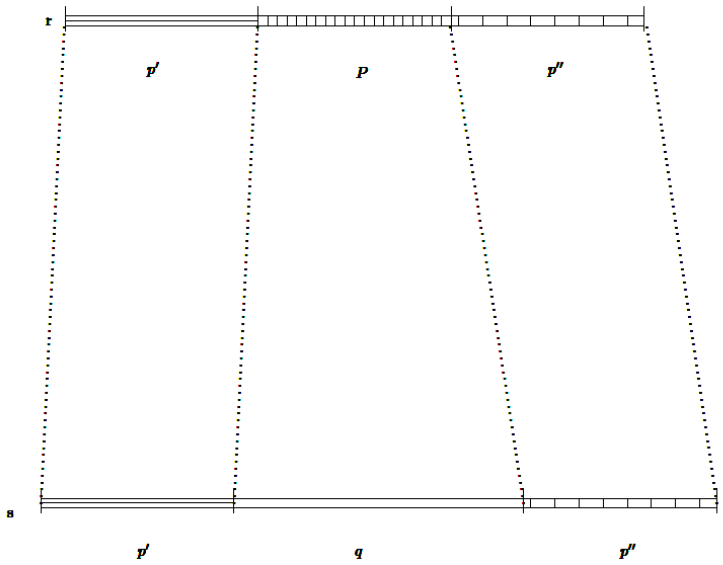
3. fejezet - Formális rendszer és néhány főbb típusa

Formális rendszernek (vagy átírási rendszernek) nevezünk minden olyan $W = (V, H)$ párt, amelyben V egy ábécé, H pedig a $V^* \times V^*$ direkt szorzat egy véges részhalmaza. A H elemeit *helyettesítési szabályoknak* hívjuk és ha $(p , q) \in H$, akkor a $p \to q$ (ejtsd: p nyíl q) jelölést használjuk.

3.1. példa - Átírási rendszer

$W = (\{ a, b , d, e, é , f, k, l , ő, s, v , z \}, \{ aked \to le, leves \to főzelék \})$. ★

Legyen $r, s \in V^*$. Akkor mondjuk, hogy r - ből az s *közvetlenül* (*vagy egy lépésben*) *levezethető* W-ben, jelekben: $r \Rightarrow_W s$, vagy ha nem vezet félreértéshez, W elhagyásával $r \Rightarrow s$, ha léteznek olyan $p '$, $p, p '' , q \in V^*$ szavak, hogy $r = p ' pp ''$, $s = p ' qp ''$ és $p \to q \in H$. (Természetesen megengedett, hogy akár a $p '$, akár a $p ''$, vagy akár mindkettő az üresszó legyen.) Szemléletesen, $r \Rightarrow_W s$ azt jelenti, hogy az s szó megkapható az r szóból úgy, hogy r-ben valamely H-beli szabály baloldalán álló p rész-szó helyébe az e szabály jobboldalán álló q szót írjuk (lásd ábra).



Azt mondjuk, hogy a p - ből a q levezethető W-ben, jelekben $p \Rightarrow_W^* q$, vagy ha nem vezet félreértéshez, W elhagyásával: $p \Rightarrow^* q$ ha léteznek olyan $p_0 , p_1, \dots , p_k \in V^*$ szavak, hogy $p_0 = p , p_k = q$ és $p_i \Rightarrow_W p_{i+1}$ ($i = 0, \dots, k$-1). Emellett, amint szokásos, megállapodunk abban, hogy minden $p \in V^*$ szóra $p \Rightarrow_W^* p$ fennáll. Használni fogjuk még a $p \Rightarrow_W^+ q$ jelölést is abban az esetben, ha azt akarjuk hangsúlyozni, hogy léteznek olyan $p_0 , p_1, \dots , p_k \in V^*$ szavak, hogy $p_0 = p , p_k = q$ és $p_i \Rightarrow_W p_{i+1}$ ($i = 0, \dots, k$-1). Világos, hogy a két jelölés $p = q$ esetén kap eltérő értelmet.

3.2. példa - Levezetés átírási rendszerben

Tekintsük a fenti példában megadott formális rendszert és a *babakedves* szót. Ekkor a *babakedves* $\Rightarrow$ *bableves* $\Rightarrow$ *babfőzelék* levezetés alkalmazásával látjuk, hogy például a W-beli *babakedves* szóból a *bableves* közvetlenül levezethető, a *babfőzelék* pedig levezethető. ★

Amennyiben olyan formális rendszereket tekintünk, melyekben a helyettesítési szabályok szimmetrikusak, eljutunk az *asszociatív kalkulus* fogalmához. Ha tehát valamely asszociatív kalkulus esetén egy p szó helyettesíthető q - val, akkor a q szó is helyettesíthető p - vel. Emiatt nem is helyettesítési szabályokról, hanem *megengedett cserékről* szokás beszélni. Továbbá, ha egy p szóból egy q szó levezethető az asszociatív kalkulusban, akkor azt mondjuk hogy p ekvivalens q - val.

Legyen $W = (\{ a, c , k, m, ó , u, s, t , y \}, \{ acs - ó , ka - kus \})$. Asszociatív kalkulus esetén a szabályok bal és jobboldalát $\rightarrow$ helyett - - el (kötőjellel) szokás elválasztani, ezzel is hangsúlyozva, hogy a szabályok szimmetrikusak. Példánknál maradva, az *acs* helyettesíthető az *ó* - val és az *ó* is helyettesíthető *acs* - csal. Ugyanígy, a *ka* helyettesíthető *kus* - sal és a *kus* a *ka* - val. Tekintsük a *m acska* $\Rightarrow$ *mó ka* $\Rightarrow$ *mókus* levezetést. Ekkor a *macska* ekvivalens *W*-ben a *mókus* - sal. Ehhez $\Rightarrow_W^*$ helyett $\simeq_W$ - t, vagy ha nem okoz félreértést, akkor egyszerűen (az ekvivalencia egyik szokásos jelét,) $\simeq$-t szokás használni, azaz *macska* $\simeq$ *mókus*. Vizsgáljuk meg most ezt az $\simeq$-t mint relációt. Ez az $\simeq$reláció *reflexív,* azaz minden V^*-beli p szóra $p \simeq p$. Például, *macska* $\simeq$ *macska*. Az $\simeq$*szimmetrikus,* azaz minden V^*-beli p, q szópárra $p \simeq q$ akkor és csak akkor, ha $q \simeq p$. Például, *macska* $\simeq$ *mókus* következménye *mókus* $\simeq$ *macska* és viszont. Ugyanígy, példánkban *kutya* $\not\simeq$ *macska* következménye *macska* $\not\simeq$ *kutya* és viszont. Az $\simeq$ *tranzitív,* azaz minden V^*-beli p, q, r szó-hármasra $p \simeq q$ és $q \simeq r$ esetén $p \simeq r$. Példánkban *macska* $\simeq$ *móka* és *móka* $\simeq$ *mókus* miatt *macska* $\simeq$ *mókus*. (De ugyanígy igaz, hogy *macska* $\simeq$ *mókus* és *mókus* $\simeq$ *móka* miatt *macska* $\simeq$ *móka*.)

1. *Megjegyzés.* a reflexív és tranzitív tulajdonság közvetlenül következik a levezethetőség definíciójából. A szimmetria a szabályok szimmetrikus volta miatt, továbbá a tranzitivitás miatt áll fenn, melynek igazolását az olvasóra bízzuk.)

Emlékeztető. Egy M halmaz önmagával való Descartes szorzatának (jelekben: $M \times M$ - nek) részhalmazait M *feletti bináris relációnak,* vagy röviden, *relációnak* szokás nevezni. Speciálisan, $M \times M$ véges részhalmazait *véges relációknak* is szokás hívni (M felett).

Felhasználva az előző példánkban tárgyaltakat, vegyük észre, hogy ha egy adott $W = (V, H)$ formális rendszerbeli közvetlen levezethetőséget mint a V^* halmaz feletti (véges) bináris relációt tekintjük, úgy a belőle származtatható levezethetőség nem más mint a közvetlen levezethetőség reflexív és tranzitív lezárása, vagyis az a legszűkebb reflexív és tranzitív reláció, melynek a tekintett levezethetőség részrelációja. Asszociatív kalkulus esetén, mint példánkban már említettük, ez a levezethetőség (mint reláció) ekvivalencia reláció lesz, ugyanis a reflexív és tranzitív tulajdonság mellett a szimmetrikus tulajdonsággal is rendelkezni fog. Emiatt szokás egy $W = (V, H)$ asszociatív kalkulusban *ekvivalensnek* hívni a $p, q \in V^*$ szavakat, ha $p \Rightarrow^* q$ azaz ha $p \simeq q$.

Akkor mondjuk, hogy a $W = (V, H)$ asszociatív kalkulusban a szóprobléma *algoritmikusan megoldható,* ha létezik olyan algoritmus, melynek segítségével tetszőleges p, q párra eldönthető, hogy p és q ekvivalens-e (azaz $p \simeq q$ fennáll-e) és ha igen, akkor az algoritmus egy $p \Rightarrow p_1 \Rightarrow p_2 \Rightarrow \ldots \Rightarrow p_{n-1} \Rightarrow q$ levezetést is szolgáltat. Ezen fogalomnak azért van különös jelentősége, mert kimutatható, hogy nem minden asszociatív kalkulusban oldható meg a szóprobléma algoritmikusan (ami egyúttal azt is igazolja, hogy nincs olyan univerzális módszer, ami minden matematikai problémát képes megoldani). Később még vissza fogunk térni erre az eldönthetőségi kérdésre.

Egy $W = (V, H)$ formális rendszert *generatív rendszernek* nevezünk, ha ki van tüntetve V^* egy nem üres és véges részhalmaza, amelyet W *axiómarendszerének* nevezünk (és rá többnyire az Ax jelölést használjuk). Egy ilyen W generatív rendszert $W = (V, Ax , H)$ alakban szokás megadni (aholis V az ábécé, Ax az axiómák, H pedig a szabályok nem üres és véges halmaza). W - hez hozzárendelünk egy $L_g (W)$ halmazt az $L_g (W) = \{ p \in V^* \mid \exists s \in Ax : s \Rightarrow_W^* p \}$ definícióval, s ezt a halmazt a W *generatív rendszer által generált nyelvnek* hívjuk. $L_g (W)$ tehát tartalmazza mindazon V^*-beli szavakat, melyek legalább az egyik axiómából levezethetők.

Egy $W = (V, Ax , H)$ generatív rendszert úgy is szokás interpretálni (azaz értelmezni), hogy V bizonyos fogalmak rendszere, Ax az axiómarendszer (vagyis az alapigazságok, vagy igaznak tartott alapvető állítások gyűjteménye), V^* elemei a V fogalomkörben megfogalmazható (értelmes vagy értelmetlen) formális mondatok halmaza, H a W-ben megengedett elemi bizonyítási lépések halmaza, $p \Rightarrow p_1 \Rightarrow \ldots \Rightarrow p_n \Rightarrow q$ a q formális mondat p - ből való bizonyítása, $L_g (W)$ pedig a V fogalomkörben megfogalmazható és az Ax axiómarendszerből W-ben bebizonyítható formális mondatok (kijelentések) halmaza.

Analóg módon, tekinthetjük az összes olyan szavak halmazát is, melyekből az axiómák közül legalább az egyik levezethető. Ebben az esetben W - t (a generatív rendszer elnevezés helyett) *analitikus rendszernek,* az $L_a (W) = \{ p \in V^* \mid \exists s \in Ax : p \Rightarrow_W^* s \}$ halmazt pedig a W *analitikus rendszer által*

acceptált, vagy más szóval, a *W analitikus rendszer által elfogadott nyelvnek,* vagy még másképp a *W analitikus rendszer által felismert nyelvnek* hívjuk.

Az analitikus rendszer által elfogadott nyelvet a későbbiekben fogjuk interpretálni (azaz értelmezni).

3.3. példa - Generatív rendszer

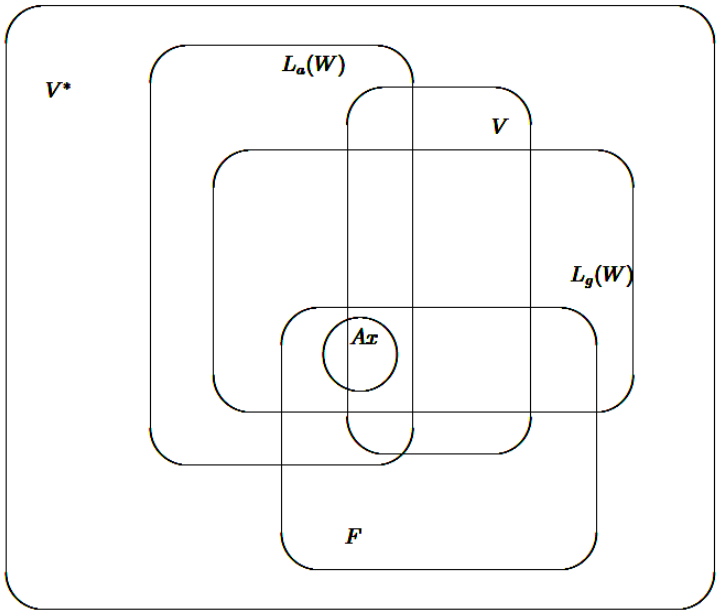
Vegyük a $W = (\{ 1, 2, +, = \}, \{ 1 + 1 = 2 \}, \{ = \rightarrow + 1 = 1 + \})$ generatív rendszert. Fogalmaink az 1, 2 természetes számok, továbbá az összeadás és az egyenlőség (jele). Egyetlen axiómánk van, 1 + 1 = 2. Az 1 + 1 + 1 + 1 = 1 + 1 + 2 kijelentés (vagy "tétel"-nek is lehetne mondani) bebizonyítható *W*-ben. Ime a "bizonyítás:" (Egyetlen) axiómánk 1 + 1 = 2. Ebből indulunk ki :

$1 + 1 = 2 \Rightarrow 1 + 1 + 1 = 1 + 2 \Rightarrow 1 + 1 + 1 + 1 = 1 + 1 + 2.$ ★

3.4. példa - Analitikus rendszer

Tekintsük a $W = (\{ <, >, 0, 1, 2, 3, 4, 5, 6 , 7, 8, 9 \}, \{ < 0 >, < 5 > \}, \{ 00 \rightarrow 0, 10 \rightarrow 0, 20 \rightarrow 0, 30 \rightarrow 0, 40 \rightarrow 0, 50 \rightarrow 0, 60 \rightarrow 0, 70 \rightarrow 0 , 80 \rightarrow 0, 90 \rightarrow 0, 05 \rightarrow 5, 15 \rightarrow 5, 25 \rightarrow 5, 35 \rightarrow 5, 45 \rightarrow 5, 55 \rightarrow 5, 65 \rightarrow 5, 75 \rightarrow 5, 85 \rightarrow 5 , 95 \rightarrow 5 \})$ analitikus rendszert. Az olvasóra bízzuk annak igazolását, hogy $L_a (W)$ épp az öttel osztható, $< a_1 \ldots a_n > (a_1, \ldots, a_n \in \{ 0, \ldots, 9 \})$ alakban megadott nemnegatív egészek halmaza. *W* tehát "felismeri" az 5 - tel osztható nemnegatív egészeket. ★

Egy $W = (V, Ax , H)$ generatív rendszert *szemi-Thue* rendszernek (ejtsd: tyué) nevezünk (az angol semi[ejtsd:szemi] = félig-, fél szó alapján), ha ki van tüntetve V^* egy olyan *F* részhalmaza, melyre $Ax \subseteq F$. Egy ilyen szemi-Thue rendszert $W = (V, Ax , H, F)$ alakban adunk meg, ahol *F* a *formulák* halmaza, és a *W* - nek a generatív rendszerhez hasonló értelmezése esetén mondhatjuk még azt is, hogy *F* az igaz (vagy igaznak tartott) állítások (kijelentések, mondatok, tételek) halmaza (lásd ábra).



Képezzünk egy szemi-Thue rendszert oly módon, hogy a 3.3. Példa (Generatív rendszer)-beli generatív rendszert kiegészítjük formulákkal. Ezzel összhangban tekintsük a következő példát.

3.5. példa - Szemi-Thue rendszer

$W = (\{ 1, 2, +, = \}, \{ 1 + 1 = 2 \}, \{ = \rightarrow + 1 = 1 + \}, \{ 1 + 1 = 2, 1 + 1 + 1 + 1 = 2 + 2, 1 + 1 (+ 1)^n = (1 +)^n + 2 , n = 1, 2, \ldots \}).$

Amint szokásos, $(+ 1)^n$ itt azt jelenti, hogy + 1 *n*-szer van írva. Például: $(+ 1)^3 = + 1 + 1 + 1.$ Hasonlóan, $(1 +)^n$ jelentése az, hogy 1 + van *n* - szer írva. Például: $(1 +)^2 = 1 + 1 + .$ A nyilvánvalóan igaz 1 + 1 + 1 + 1 = 2 + 2 összefüggés a példabeli szemi-Thue rendszerben nem bizonyítható. Mint már

a bevezetésben elmondtuk, ezen a játékos példán bemutatott probléma a matematika egyes nehezebb fejezeteiben is fennáll. Nevezetesen, vannak precízen megfogalmazható, de nem bizonyítható és nem is cáfolható matematikai kijelentések. ★

Legyen $W = (V, Ax , H, F)$ tetszőleges szemi-Thue rendszer, s jelölje $Ax \Rightarrow^* p$ röviden azt a tényt, hogy létezik olyan $p_0 \in Ax$, melyre $p_0 \Rightarrow^* p$. Ezt a jelölést használva $L(W) = \{ p \in V^* \mid Ax \Rightarrow^* p \}$ és W-ben absztrakt módon az F halmaz mint elmélet (= igaz vagy igaznak tartott állítások halmaza, rendszere, vagy gyűjteménye) axiomatizálásának tipikus kérdései a következőképp fogalmazhatók meg:

a.) *teljes-e az elmélet,* vagyis az Ax axiómarendszerből minden igaz állítás bebizonyítható-e. Jelöléssel: fennáll-e az $F \subseteq L_g (W)$ tartalmazás;

b.) *ellentmondásmentes* (*azaz helyes*)*-e az elmélet,* vagyis igaz-e, hogy az axiómarendszerből nem vezethető le egyidejűleg igaz és hamis állítás is. Képletben: igaz-e, hogy akárhogy is adjuk meg a $p \in F$ és $q \notin F$ párokat, $Ax \Rightarrow^* p$ mellett $Ax \Rightarrow^* q$ nem állhat fenn egyidejűleg. (Vegyük észre, hogy ha $Ax \Rightarrow^* q$ és $q \notin F$, akkor felhasználva $Ax \Rightarrow^* q$ jelentését, lenne olyan $p_0 \in Ax$, hogy $p_0 \Rightarrow^* q$, ami $Ax \subseteq F$ és $p_0 \Rightarrow^* p_0$ miatt azt is jelentené, hogy $p_0 \in F$ és $q \notin F$ mellett $Ax \Rightarrow^* p_0$ és $Ax \Rightarrow^* q$. Az ellentmondásmentesség feltétele tehát tulajdonképp azt jelenti, hogy hamis kijelentés nem vezethető le az axiómarendszerből.)

c.) *kategórikus* (= *konzekvens*)*-e az elmélet,* vagyis az Ax axiómarendszerből minden igaz állítás, de csakis az igaz állítások bebizonyíthatók-e. Jelöléssel: $F = L_g (W)$ fennáll-e; (Vegyük észre, hogy egy elmélet pont akkor konzekvens, ha egyidejűleg teljes és helyes is.)

d.) *minimális-e az elmélet,* azaz igaz-e, hogy egyik axióma sem bizonyítható be a másikból. Képletben: igaz-e, hogy ha $p, q \in Ax$ és $p \Rightarrow^* q$, akkor szükségképp $p = q$.

3.6. példa - Elmélet teljessége, ellentmondásmentessége

Az előző példabeli "elméletünket" vizsgálva megállapíthatjuk, hogy nem teljes, mert az $F = \{ 1 + 1 = 2, 1 + 1 + 1 + 1 = 2 + 2, 1 + 1 (+ 1)^n = (1 +)^n + 2 , n = 1, 2, \dots \}$ elmélet nem minden kijelentése bizonyítható W-ben (pl. az $1 + 1 + 1 + 1 = 2 + 2$ nem). Emiatt "elméletünk" nem is kategórikus. Mivel "hamisnak tartott", azaz F - en kívüli állítás $1 + 1 = 2$ - ből (egyetlen axiómánkból) nem vezethető le (ugyanis minden belőle levezethető "állítás" vagy maga az $1 + 1 = 2$ lesz, vagy $1 + 1 (+ 1)^n = (1 +)^n + 2$ alakú lesz, ahol n tetszőleges természetes szám), "elméletünk" ellentmondásmentes. Végül, lévén csak egyetlen axiómánk, "elméletünk" nyilvánvalóan minimális. ★

Végül megjegyezzük, hogy lehetett volna a formális rendszer és a belőle származtatott különféle speciális formális rendszer típusok fogalmának kialakításakor a közvetlen levezethetőséget és a levezethetőséget másképp is definiálni. Erre az egyik legnevezetesebb példa a Post-féle normál rendszer.

A *Post-féle normál rendszer* (hasonlóképp a generatív rendszerhez) egy olyan V ábécéből, Ax axiómarendszerből és H helyettesítési szabályokból felépülő $W = (V, Ax , H)$ hármas (Ax itt is nem üres és véges részhalmaza V^*-nak, továbbá H ezesetben is nem üres és véges részhalmaza $V^* \times V^*$-nak), melyben H elemei $pa \rightarrow aq$ alakúak, aholis $a \in V$ egy olyan betű, mely sem p-ben, sem q-ban nem fordul elő. Akkor mondjuk, hogy a V^*-beli p_1 szóból a V^*-beli q_1 szó *közvetlenül levezethető* (jelekben, mint korábban : $p_1 \Rightarrow_W q_1$, vagy ha nem vezet félreértéshez, egyszerűen csak $p_1 \Rightarrow q_1$), ha található olyan V-beli a betű, továbbá találhatók olyan $(V \setminus \{ a \})^*$-beli p, q, r szavak, hogy $p_1 = pr , q_1 = rq$, és $pa \rightarrow aq \in H$. Ezenkívül, mint korábban, azt mondjuk, hogy p - ből a q *levezethető* W-ben, jelekben: $p \Rightarrow_W^* q$, vagy ha nem vezet féreértéshez, W elhagyásával: $p \Rightarrow^* q$ ha léteznek olyan $p_0 , p_1, \dots , p_k \in V^*$ szavak, hogy $p_0 = p , p_k = q$ és $p_i \Rightarrow_W p_{i+1} (i = 0, \dots, k\text{-}1)$. Emellett, amint szokásos, itt is megállapodunk abban, hogy minden $p \in V^*$ szóra $p \Rightarrow_W^* p$ fennáll. (Másként mondva, a levezethetőség mint V^* feletti reláció, ezesetben is reflexív és tranzitív lezárása a közvetlen levezethetőségnek.) Ugyanúgy mint a generatív rendszer esetén, a Post-féle normál rendszerhez is szokás tekinteni az $L_g (W) = \{ p \in V^* \mid \exists s \in Ax : s \Rightarrow_W^* p \}$ halmazt, amit a W Post-féle normál rendszer *által generált nyelvnek* hívunk. Természetesen mint a generatív rendszerből "képzett" analitikus rendszer esetén, itt is lehet tekinteni az $L_a (W) = \{ p \in V^* \mid \exists s \in Ax : p \Rightarrow_W^* s \}$ halmazt mint a W *által elfogadott nyelvet.* (Emlékeztetőül:

szerkezetileg a generatív rendszer ugyanaz mint az analitikus rendszer, csak a hozzájuk rendelt nyelvek szerkezete más.)

A Post-féle normál rendszer fogalmának kis módosításával jutunk el a Post-féle tag (tag [ejtsd: teg] = toldalék, vmit vmihez hozzáfűző eszköz vége) rendszer fogalmához. A *Post-féle tag rendszer* egy olyan $W = (V, s , H)$ hármas, ahol V egy ábécé, H helyettesítési szabályok egy halmaza (azaz $V^* \times V^*$ egy nem üres és véges részhalmaza), s pedig egy tetszőleges, rögzített eleme V^*-nak. Erre az s elemre a *startszó* elnevezés használatos. Akkor mondjuk, hogy a V^*-beli p szóból az ugyancsak V^*-beli q szó *közvetlenül* (*vagy egy lépésben*) *levezethető* W-ben, jelekben: $p \Rightarrow_W q$, vagy ha nem vezet félreértéshez, W elhagyásával $p \Rightarrow q$, ha léteznek olyan $p_1 , q_1, r \in V^*$ szavak, hogy $p = p_1 r$, $q = rq_1$ és $(p_1 , q_1) \in H$. (Természetesen megengedett, hogy az r az üresszó legyen.) A levezethetőségre (is) ugyanazt a jelölést használjuk mint korábban, továbbá a levezethetőség mint V^* feletti reláció, ezesetben is reflexív és tranzitív lezárása a közvetlen levezethetőségnek. Ugyancsak összhangban az előzőekkel, a W *által generált nyelv:* $L_g (W) = \{ p \in V^* \mid s \Rightarrow_W^* p. \}$ Definiálhatjuk a W *által elfogadott nyelvet is:* $L_a (W) = \{ p \in V^* \mid p \Rightarrow_W^* s \}$.

3.1. Markov-féle normál algoritmus

Az algoritmus intiutív fogalmát körülbelül a következőképp szokás megfogalmazni: A fegyelmezett, egyértelműen előírt elemi lépésekből egyértelműen felépíthető olyan feladatmegoldás modellje, melynek eredménye végrehajtójától függetlenül ugyanazon feladatokra akárhányszor is megismételve ugyanazon eredményt szolgáltatja. Segítségével egy adott feladatosztály minden feladata véges számú lépésben megoldható. (A legáltalánosabb értelemben természetesen egy algoritmus bármilyen emberi vagy nem emberi tevékenységre vonatkozhat, nemcsak matematikai avagy számítástechnikai feladatok megoldására.)

Fontos tulajdonságai közé tartozik tehát a *függetlenség,* ami azt jelenti, hogy ugyanazon feladatra ugyanaz az eredménye függetlenül attól hogy ki (vagy mi) a végrehajtója, az *egyértelműség,* ami azt jelenti, hogy ugyanazon feladatra akárhányszor is alkalmazzuk az algoritmust, mindig ugyanazt az eredményt szolgáltatja, az *elemi lépésekre bonthatóság,* ahol persze ezen elemi lépések, továbbá ezen lépések sorrendje is egyértelműen meg van határozva, valamint a *végesség*, vagyis az, hogy a feladatosztály minden egyes feladatára mindig véges sok lépésben befejeződik. Amennyiben a véges lépésben történő befejeződés követelményétől eltekintünk, az *eljárás* fogalmához jutunk. Ezen és ehhez hasonló intiutív megfogalmazások után születtek meg a 30-as évek második felétől a különféle matematikailag is precíz algoritmusfogalmak (tulajdonképpen szigorúan tekintve eljárásfogalmak), melyek közös érdekessége, hogy ekvivalensek. Ez azt jelenti, hogy ha egy algoritmust az egyik algoritmusfogalom segítségével megadtuk, akkor ugyanez az algoritmus a másik algoritmusfogalom segítségével is megadható. Ezen fogalmak egyike a Markov-féle normál algoritmus.

Egy $W = (V, H)$ formális rendszert (*Markov-féle*) *normál algoritmusnak* nevezünk, ha H rendezett halmaz és ki van tüntetve benne egy H_1 részhalmaz (megengedve, hogy esetleg $H_1 = \emptyset$, azaz üres halmaz legyen). A H_1 elemeit *záróhelyettesítéseknek* hívjuk és $p \rightarrow q \in H_1$ esetén használni fogjuk a $p \rightarrow . q$ jelölést is. Egy ilyen normál algoritmust $W = (V, H , H_1)$ alakban adunk meg, ahol a $H = \{ p_1 \rightarrow q_1, \dots , p_m \rightarrow q_m \}$ megadás, azaz H elemeinek felsorolása feltételezésünk szerint egyben a H - beli rendezést is mutatja. Legyen $p \in V^*$. Azt mondjuk, hogy a p szóra a $W = (V, H , H_1)$ algoritmus *alkalmazható,* ha van olyan 1 és m közötti i ($1 \leq i \leq m$) index és olyan $p', p'' \in V^*$ szópár, hogy $p = p' p_i p''$. Legyen i a legkisebb ilyen index és válasszuk meg a p szó p' kezdőszeletét úgy, hogy p_i a $p' p_i$ szóban rész-szóként csak egyszer forduljon elő. Rendeljük hozzá p - hez a $q = p' q_i p''$ szót és ezt a hozzárendelést jelöljük a következőképpen: legyen $p \Rightarrow_W q$ ha $p_i \rightarrow q_i \notin H_1$ és legyen $p \Rightarrow_W . q$ ha $p_i \rightarrow q_i \in H_1$. E hozzárendelést, mely a formális rendszer fogalmánál bevezetett közvetlen levezethetőség fogalmának a H - beli rendezést is figyelembe vevő módosítása, *elemi helyettesítésnek* szokás nevezni. (Az elemi helyettesítés nem tévesztendő össze a helyettesítéssel, azaz a helyettesítési szabályok halmazának egy elemével.) $p \Rightarrow_W . q$ esetén elemi záróhelyettesítésről beszélünk.

Ha záróhelyettesítést hajtottunk végre az algoritmus megáll, egyébként az algoritmus helyettesítő lépése újra lefut az imént kapott szóra. Az algoritmus tehát mindaddig fut, amíg vagy záróhelyettesítés be nem következik, vagy nincs alkalmazható helyettesítési szabálya. Itt térünk ki arra, hogy a Markov-

féle normál algoritmus a szó eredeti értelmében nem algoritmus, hanem eljárás, mivel - ahogy majd példát is mutatunk rá- nem feltétlenül fejeződik be, tehát a végesség feltétele nem teljesül.

3.7. példa - Elemi helyettesítés

Vegyük a $W = (\ \{\ a, b\ , e, i, k\ , o, r, s\ , t, ú, z\ \}\ ,\ \{\ szoba \rightarrow kerti\ ,\ bab \rightarrow szob\ \},\ \{\ szoba \rightarrow kerti\ \}\ )$ normál algoritmust és a *bababútor* szót. Az $i = 2$, hiszen a második szabály alkalmazható rá. Vigyázni kell viszont, hisz a *bab* szó rész-szóként kétszer is előfordul. p' thehát nem *ba* lesz (mert ekkor p' $p_1 = babab$, aminek első három és utolsó három betűje is a *bab* szót alkotja). Így $p' = \lambda$, azaz az üresszó. Ennek megfelelően a bab*abútor* $\Rightarrow_W$ szob*abútor* $\Rightarrow_W$. *kertibútor* levezetéshez jutunk. Itt a *bababútor* $\Rightarrow_W$ *szobabútor* egy elemi helyettesítés, *szobabútor* $\Rightarrow_W$. *kertibútor* pedig egy elemi záróhelyettesítés. Igaz, hogy a *szobabútor* szóban is benne van a *bab* rész-szó, de mindig az első olyan szabályt kell alkalmazni, amit lehet. Emiatt a *szobabútor* szó után a *kertibútor* szó következik, nem a *szoszobútor*. ★

Azt mondjuk, hogy a $p \in V^*$ szóból kiindulva egy $q \in V^*$ szó eleme a W algoritmus *levezetési láncának*, ha teljesülnek az alábbi feltételek: $q = p$, vagy

(i) W alkalmazható p - re és

(ii) léteznek olyan $r_0, \ldots, r_k \in V^*$ szavak, hogy $r_0 = p$, $r_k = q$, $r_i \Rightarrow_W r_{i+1}$ ($i = 0, \ldots, k\text{-}1$), ahol az $r_0 \Rightarrow_W \ldots \Rightarrow_W r_{k-1}$ lánc nem tartalmaz záróhelyettesítést.

Azt mondjuk, hogy a $p \in V^*$ szóra egy $q \in V^*$ szó a W algoritmus *eredménye* (jelekben: $p \Rightarrow_W^* q$), ha q eleme a W p - ből induló ($p = r_0, \ldots, r_k = q$) levezetési láncának és vagy $r_{k-1} \Rightarrow_W . r_k$, vagy pedig $r_{k-1} \Rightarrow_W r_k$, és W az r_k ($= q$) szóra már nem alkalmazható.

Emellett megállapodás, hogy ha W a p szóra nem alkalmazható, akkor $p \Rightarrow_W^* p$.

Ha $p \Rightarrow_W^* q$, akkor szokás mondani, hogy q a p szóval megadott adatsorozaton a W normál algoritmussal végrehajtott számítás eredménye. Eszerint, egy $W = (\ V, H\ , H_1\ )$ normál algoritmus esetén egy $p \in V^*$ (input) szóra a következő esetek állhatnak fenn:

(I) W a p - re nem alkalmazható. Ekkor a W - vel p - n végzett számítás eredményének magát a p adatot tekintik.

(II) p - ből kiindulva létezik záróhelyettesítéssel vegződő (azaz $p \Rightarrow_W r_1$, $r_1 \Rightarrow_W r_2$, …, $r_{k-2} \Rightarrow_W r_{k-1}$, $r_{k-1} \Rightarrow_W . r_k$ alakú), vagy tovább nem folytatható helyettesítési lánc. (Ez utóbbi azt jelenti tehát, hogy a $p \Rightarrow_W r_1$, $r_1 \Rightarrow_W r_2$, …, $r_{k-1} \Rightarrow_W r_k$ láncban r_k - ra W már nem alkalmazható.) Ekkor a p adaton a W - vel végzett számítás eredménye az $r_k = q$ szó.

(III) p - ből olyan helyettesítési lánc indul ki, mely soha nem fejeződik be. Ilyenkor azt mondjuk, hogy W a p - re nem áll meg, W a p adatból eredményt nem szolgáltat.

3.8. példa - Unáris számok összeadása

Tekintsük a $W = (\ \{\ 1, +\ \},\ \{\ 1 + \rightarrow + 1, + + \rightarrow +, + \rightarrow \lambda\ \},\ \{\ + \rightarrow \lambda\ \}\ )$ normál algoritmust (ahol λ az üresszót jelöli). Ekkor $1 + 1 + 1 + 1 \Rightarrow_W + 11 + 1 + 1 \Rightarrow_W + 1 + 11 + 1 \Rightarrow_W + + 111 + 1 \Rightarrow_W + + 11 + 11 \Rightarrow_W + + 1 + 111 \Rightarrow_W + + + 1111 \Rightarrow_W + + 1111 \Rightarrow_W + 1111 \Rightarrow_W . 1111$.

Vegyük észre, hogy általában is, $1^m + 1^n \Rightarrow_W^* 1^{m+n}$, vagyis a példabeli algoritmus az "egyes számrendszerben" való "összeadás" algoritmusa. Ahány 1 - est "adunk össze", annyi 1 - esből álló sorozatot kapunk a végén záróhelyettesítéssel. Nincs alkalmazható szabálya az algoritmusnak például a 11 szóra. Ekkor eredménynek a korábbi definícióval összhangban magát a 11 - t tekintjük. ★

3.9. példa - Végtelen helyettesítési lánc

Egészítsük ki az előbbi példát az $1 \to 11$ szabállyal, méghozzá úgy, hogy ez legyen az első szabályunk. Az így kapott újabb $W = (\{ 1, + \}, \{ 1 \to 11, 1 + \to + 1, + + \to +, + \to \lambda \}, \{ + \to \lambda \})$ algoritmus már nem fog befejeződni az 1 + 1 + 1 + 1 szóra (és sok más szóra sem): $1 + 1 + 1 + 1 \Rightarrow_W 11 + 1 + 1 + 1 \Rightarrow_W 111 + 1 + 1 + 1 \Rightarrow_W 1111 + 1 + 1 + 1 \Rightarrow_W 11111 + 1 + 1 + 1 \ldots$ A módosított algoritmus tehát az 1 + 1 + 1 + 1 szóra nem áll meg, eredményt nem szolgáltat. ★

3.10. példa - Unáris számok szorzása

Adjon meg olyan Markov algoritmust, amely az 1^m*1^n bemenetre 1^{mn} kimenettel áll meg!

Megoldás:

$W=(\{1,*,a,b\},H,H_1)$, ahol H a következő:

1. $*11 \to a*1$, 2. $*1 \to a$, 3. $1a \to a1b$, 4. $ba \to ab$,

5. $b1 \to 1b$, 6. $a1 \to a$, 7. $ab \to b$, 8. $b \to 1$. $H_1=\emptyset$. ★

3.11. példa - Feladat - Legnagyobb közös osztó előállítása

Legyen $W = (\{ a, b, c, d, e, \& \}, \{ ac \to ca, a \& a \to c \&, a \& \to \& d, d \to a, c \to e, e \to a, \& \to \lambda \}, \emptyset)$ (azaz H_1 ebben a feladatban is üres halmaz) normál algoritmus. Bizonyítsuk be, hogy tetszőleges i, j pozitív egész számpárra $a^i \& a^j \Rightarrow_W a^k$, ahol k a legnagyobb közös osztója i - nek és j - nek. ★

3.12. példa - Bináris szó rendezése

Adjon meg olyan Markov algoritmust, amely egy bemenő bináris szó esetén
annyi 1-est, majd annyi 0-ást ír, amennyi a bemenő szóban van!
Például: 001011011 bemenetre 111110000 szót adja eredményül.

Megoldás:

Egyik lehetséges megoldás a következő:

$W=(\{0,1\},\{01 \to 10\}, \emptyset)$. Az algoritmus a buborékrendezést valósítja meg. Ugyanez a példa három számjegy esetén: $W=(\{0,1,2\},\{01 \to 10, 02 \to 20, 12 \to 21\}, \emptyset)$. ★

3.13. példa - Bináris szó tükrözése

Adjon meg olyan Markov algoritmust, amely egy bemenő bináris szó tükörképét adja eredményül! Például: 0110011 bemenetre 1100110 végeredményt ad.

Megoldás: Egy lehetséges algoritmus a következő: $W=(\{0,1,X,Y,U,V,Z\},H,H_1)$,
ahol H elemei:
1. $Y0 \to 0Y$, 2. $Y1 \to 1Y$, 3. $Y \to Z$, 4. $UZ \to Z0$, 5. $VZ \to Z1$,
6. $U0 \to 0U$, 7. $U1 \to 1U$, 8. $V0 \to 0V$, 9. $V1 \to 1V$, 10. $X0 \to XU$,
11. $X1 \to XV$, 12. $XZ \to \lambda$, 13. $\lambda \to XY$.
$H_1=\{XZ \to \lambda\}$.
Az algoritmus a következőképpen működik:

1. A bemenő szó elejére írja XY-t.

2. Y-t elvisszük a szó végére és átírjuk Z-re.

3. Ha az X utáni első számjegy 1, akkor V-t írunk helyette.

4. Ha az X utáni első számjegy 0, akkor U-t írunk helyette.

5. U-t vagy V-t elvisszük a szó végére.

6. Ha a Z elé U kerül, akkor 0-t írunk Z után, és ha van még szám X és Z közt, akkor vissza a 3-as pontra.

7. Ha a Z elé V kerül, akkor 1-t írunk Z után, és ha van még szám X és Z közt, akkor vissza a 3-as pontra.

8. XZ törlése.

★

3.14. példa - Bináris szó megegyező végeinek törlése

Adjon meg olyan Markov algoritmust, amely egy bemenő bináris szó első és utolsó karakterét addig törli, ameddig azok megegyeznek!
Például: 10110001 esetén a kimenet 1100.

Megoldás: Egy algoritmus lehet ez: $W=(\{0,1,X,Y,U,V,W\},H,H_1)$, ahol H elemei:
1. $0Y \to Y0$, 2. $1Y \to Y1$, 3. $0W \to W0$, 4. $1W \to W1$, 5. $XY0 \to X$,
6. $XY1 \to X$, 7. $XW0 \to 0$, 8. $XW1 \to 1$, 9. $U0 \to 0U$, 10. $U1 \to 1U$,
11. $V0 \to 0V$, 12. $V1 \to 1V$, 13. $0U \to Y$, 14. $1U \to W1$, 15. $1V \to Y$,
16. $0V \to W0$, 17. $X1 \to X1W$, 18. $X0 \to X0U$, 19. $\lambda \to X$
$H_1=\{XW0 \to 0,\ XW1 \to 1\ \}$.
Az algoritmus a következőképpen működik:

1. A bemenő szó elejére ír egy X-et.

2. Ha az első számjegy 0, akkor U-t írunk utána.

3. Ha az első számjegy 1, akkor V-t írunk utána.

4. U-t vagy V-t elvisszük a szó végére.

5. Ha az utolsó jegy 0 és U került mellé, vagy ha 1 és V, akkor ezek helyére Y-t írunk.

6. Ha az utolsó jegy 1 és U került mellé, vagy ha 0 és V, akkor ezek helyére W-t írunk és visszaírjuk az utolsó jegyet.

7. Y-t vagy W-t az elejére visszük.

8. Ha az X után Y kerül, akkor az Y-t és az első számjegyet töröljük, és vissza a 2. lépésre.

9. Ha az X után W kerül akkor töröljük XW-t és megállunk.

★

3.15. példa - Bináris szám növelése 1-gyel

Adjon meg olyan Markov algoritmust, amely egy bemenő bináris szám esetén eggyel nagyobb bináris számot szolgáltat eredményül!

Megoldás: $W=(\{0,1,X,Y\},H,H_1)$, ahol H elemei:
1. $X1 \to 1X$, 2. $X0 \to 0X$, 3. $X \to Y$, 4. $0Y \to 1$,
5. $1Y \to Y0$, 6. $Y \to 1$, 7. $1 \to X1$.
$H_1=\{0Y \to 1,\ Y \to 1\}$.
Az algoritmus működése:

1. A szó elejére X-et írunk.

2. X-et a szó végére visszük és Y-ra cseréljük.

3. Y előtt 0 van, 1-esre írjuk és kész vagyunk.

4. Y előtt 1 van, 0-ra írjuk és Y-t írunk elé. Vissza a 3-as pontra.

5. Ha Y előtt nincs semmi, akkor 1-esre írjuk át.

★

3.16. példa - Unáris szám bináris alakja

Adjon meg olyan Markov algoritmust, amely 1^n bemenetre az n szám bináris alakját adja eredményül!

Megoldás:$W=(\{0,1,X,Y\},H,H_1)$, ahol H elemei:
1. $0Y \rightarrow 1$, 2. $1Y \rightarrow Y0$, 3. $Y \rightarrow 1$, 4. $X1 \rightarrow YX$, 5. $X \rightarrow \lambda$,
6. $\lambda \rightarrow 0X$
$H_1=\{X \rightarrow \lambda\}$
Az algoritmus lépései:

1. Az 1-esek elé $0X$-et ír.

2. $X1$-et YX-re cseréli.

3. Y előtti bináris számot növeli eggyel és Y-t törli.

4. Ha van X után 1-es vissza 2-es pontra.

5. Törli X-et.

★

3.17. példa - Bináris szám csökkentése 1-gyel

Adjon meg olyan Markov algoritmust, amely egy bemenő bináris szám esetén eggyel kisebb bináris számot ad eredményül!

Megoldás: $W=(\{0,1,X,Y\},H,H_1)$, ahol H elemei:
1. $X1 \rightarrow 1X$, 2. $X0 \rightarrow 0X$, 3. $X \rightarrow Y$, 4. $1Y \rightarrow 0$,
5. $0Y \rightarrow Y1$, 6. $Y \rightarrow \lambda$, 7. $\lambda \rightarrow X$, és a záróhelyettesítés:
$H_1=\{Y \rightarrow \lambda\}$.
Az algoritmus működése:

1. A szó elejére X-et írunk.

2. X-et a szó végére visszük és Y-ra cseréljük.

3. Y előtt 1 van, 0-ra írjuk és kész vagyunk.

4. Y előtt 0 van, 1-re írjuk, és Y-t írunk elé.

5. Ha Y előtt nincs semmi, akkor töröljük Y-t, és kész vagyunk.

★

3.18. példa - Bináris szám unáris alakja

Adjon meg olyan Markov algoritmust, amely egy bemenő bináris szám unáris alakját adja vissza!

Megoldás: $W=(\{0,1,X,Y,Z,V\},H,H_1)$, ahol H elemei:
1. $X1 \to 1X$, 2. $X0 \to 0X$, 3. $X \to Y$, 4. $1Y \to 0Y1$, 5. $0Y \to Z1V1$, 6. $0Z \to Z1$,
7. $1Z \to \lambda$, 8. $Z \to \lambda$, 9. $V \to Y$, 10. $Y \to \lambda$, 11. $\lambda \to X$, és
$H_1=\{Y \to \lambda\}$ záróhelyettesítés.
Az algoritmus lépései:

1. A szó elé X-et ír.
2. X-et a végére viszi és Y-ra cseréli.
3. Y előtti bináris számot csökkenti, utáni unárist növeli eggyel.
4. Ha van Y előtt számjegy, akkor vissza 2-ra.
5. Törli Y-t.

(A feladatot úgy is meg lehetett volna oldani, hogy a bináris szóból balról jobbra haladva törlünk egy elemet, majd egy elszeparáló szimbólum után lévő 1-eseket duplázzuk, és a törölt elemtől függően vagy adunk hozzá egyet vagy nem, amíg a bináris szám el nem fogy.) ★

Amint láttuk a formális rendszereket algoritusok megadására is használhatjuk, ha egyértelműen megadjuk hogy adott lépésben melyik szabályt és hol kell alkalmazni. Ezzel szemben a generatív rendszerek teljesen nemdeterminisztikusak, általában nem csak az telejsül, hogy több alkalmazható szabály közül választunk, de az alkalmazás helye is lehet többféle. A következő fejezetben a generatív rendszerekből származtatott generatív nyelvtan fogalmát ismertetjük, mely e jegyzet egyik legfontosabb központi fogalma.

3.2. Generatív nyelvtan, Chomsky-féle nyelvosztályok

Egy $W=(V, Ax, H)$ generatív rendszerből úgy kapunk generatív nyelvtant, ha a V ábécét felbontjuk közös elemet nem tartalmazó nem üres (és V végessége miatt véges) N és T részhalmazokra ($V=N\cup T$, $N\cap T=\emptyset$, $N\neq\emptyset$, $T\neq\emptyset$), az Ax pedig egyetlen, egybetűs, N- beli (általában S- el jelölt) elemből áll, s minden H- beli szabály baloldala legalább egy N- beli betűt tartalmaz. Formálisan tehát a generatív grammatika (vagy generatív nyelvan) definíciója:

1. Definíció. A $G=(N, T, S, H)$ rendezett négyest *generatív nyelvtan*nak (vagy *generatív grammatikának*) nevezzük, ha N és T diszjunkt véges ábécék, $S\in N$, $H\subset(N\cup T)^*N(N\cup T)^*\times(N\cup T)^*$. Az N elemeit *nemterminális jel*eknek vagy változó szimbólumoknak nevezzük, és általában nagybetűkkel (S, A, B, C, …) jelöljük. A T elemeit *terminális jel*eknek, vagy konstansoknak nevezzük, és általában kisbetűkkel (a, b, c, …) jelöljük. A H elemeit képező (p, q) rendezett párokat (mint korábban is) *helyettesítési szabály*oknak nevezzük, és általában $p \to q$ alakban írjuk. Az S egy kitüntetett nemterminális jel, amely a G nyelvtanban a generálás kiinduló vagy kezdő eleme, másnéven *mondatszimbólum*a (startszava). ★

Most definiáljuk, hogy hogyan állítunk elő egy nyelvet egy generatív nyelvtan segítségével, először most is (közvetlen) levezethetőséget kell definiálnunk, ez tulajdonképpen megegyezik a generatív rendszerekben meghatározott azonos fogalmakkal:

2. Definíció. Egy G generatív nyelvtanban az $r\in(N\cup T)^*$ szóból *egy lépésben, vagy közvetlenül levezethető* a $t\in(N\cup T)^*$ szó, ha van olyan $p \to q$ helyettesítési szabály a H- ban és $p_1, p_2\in(N\cup T)^*$ úgy, hogy $r=p_1pp_2$ és $t=p_1qp_2$. Jelölés: $r\Rightarrow_G t$, vagy ha egyértelműen meghatározható a G, akkor $r\Rightarrow t$.

Egy G generatív nyelvtanban az r szóból *levezethető* a t szó, ha van olyan $r_0, r_1, \ldots, r_n$ véges szósorozat a $(N\cup T)^*$- ban, amelyre teljesül, hogy $r_0=r$, $r_n=t$ és $r_i\Rightarrow r_{i+1}$ (i=0, 1, …, n-1). Ezt a relációt $r\Rightarrow^* t$- vel jelöljük. Megegyezés szerint minden $r\in(N\cup T)^*$ szóra $r\Rightarrow^* r$ teljesül.

A $G=(N, T, S, H)$ *generatív nyelvtan által generált nyelven* a T^*- beli szavak következő halmazát értjük: $L(G)=\{w|S\Rightarrow^* w, w\in T^*\}$. ★

Úgy is mondhattuk volna, hogy $L(G)=L_g(G)\cap T^*$, ahol $L_g(G)$ a G mint generatív rendszer, pontosabban a $(V, \{S\}, H)$ generatív rendszer által generált nyelvet jelöli. (Tehát vigyázat: Nem tévesztendő össze a G generatív nyelvtan és a G mint generatív rendszer által generált nyelv, hiszen $L(G)\subseteq L_g(G)$, vagyis a két nyelv nem esik egybe: $S\in L_g(G)$, de $S\notin L(G)$!) $L_g(G)$ elemeit, vagyis azon $(N\cup T)$ feletti szavakat amelyek az S modatszimbólumból levezethetőek, *mondatformáknak,* N^* elemeit *nemterminális szavaknak,* T^* elemeit pedig *terminális szavaknak,* vagy a nyelvtani analógia miatt *mondatoknak* is hívjuk. Egy adott nyelvtan esetén a levezetés során mondatformák szereplenek, így a levezetett terminális szót sokszor egyszerűen csak levezetett szónak fogjuk hívni (amennyiben ez nem zavaró).

Chomsky nyelvészként a természetes nyelvek leírását szerette volna elérni generatív nyelvtanok segítségével, így a fogalom megfelel annak, hogy a változók a nyelvi fogalmak (ige, főnév, stb.), a konstansok pedig a szótári szavak elemeinek absztrakciói. Habár a természetes nyelvek teljes formális leírása ilyen formán a mai napig nem született meg, a generatív nyelvtanok szerepe igen fontossá vált a mesterséges nyelvek, így a számítógépek fejlődésével, pl. a programnyelvek formális leírásakor.

Első példánkban kezdetleges számítógépes nyelvészeti leírással próbálkozunk.

3.19. példa - Generatív nyelvtan természetes nyelv "leírására"

Legyen $G=(N, T, S, H)$, ahol $N=\{S, (ige), (főnév), (melléknév)\}$, $T=\{$ magyar ábécé betűi $\}$, $H=\{S \rightarrow (ige), (ige) \rightarrow (főnév)(ige), (főnév) \rightarrow (melléknév)(főnév), (melléknév) \rightarrow az$ *(melléknév)*, $(ige) \rightarrow$ *tanul*, *(főnév)* $\rightarrow$ *diák*, *(melléknév)* $\rightarrow$ *engedelmes*$\}$.

Tekintsük a következő levezetést.

$S\Rightarrow(ige)\Rightarrow(főnév)(ige)\Rightarrow(melléknév)(főnév)(ige)\Rightarrow az\ (melléknév)(főnév)(ige)$

$\Rightarrow az\ engedelmes\ (főnév)(ige)\Rightarrow az\ engedelmes\ diák\ (ige)\Rightarrow az\ engedelmes\ diák\ tanul.$

Ugyanígy levezethető "nyelvtanunkkal" például: *az az engedelmes diák tanul*, stb. ★

A generatív nyelvtan persze nemcsak nyelvtani elemzésre alkalmas eszköz. Mint a következő (M. Soittola -tól, illetve M. Penttonen-től származó) példák is mutatják, segítségével elő tudjuk állítani az összes négyzetszámot, illetve a prímszámokat.

3.20. példa - Unáris négyzetszámok

$G=(\{S, X, Y, Z, X_1, X_2, Y_1, Y_2\}, \{a\}, S, \{S \rightarrow a, S \rightarrow aXX_2Z, X_2Z \rightarrow aa, Xa \rightarrow aa, Ya \rightarrow aa, X_2Z \rightarrow Y_1YXZ, XX_1 \rightarrow X_1YX, YX_1 \rightarrow Y_1YX, XY_1 \rightarrow X_1Y, YY_1 \rightarrow Y_1Y, aX_1 \rightarrow aXXYX_2, X_2Y \rightarrow XY_2, Y_2Y \rightarrow YY_2, Y_2X \rightarrow YX_2\}$.

Az összes négyzetszámot a következőképp állíthatjuk elő:

Először bizonyítsuk be hogy a generált terminális szavak hossza eleme lesz az 1, 1+3, 1+3+5, …, 1+3+…+(2n-1), … sorozatnak:

Csoportosítsuk a szabályokat az alábbi módon.

(1) $S \rightarrow a, S \rightarrow aXX_2Z$,

(2) $X_2Z \rightarrow aa$,

(3) $Xa \rightarrow aa, Ya \rightarrow aa$,

(4) $X_2Z \rightarrow Y_1YXZ$,

(5) $XX_1 \to X_1YX$, $YX_1 \to Y_1YX$,

(6) $XY_1 \to X_1Y$, $YY_1 \to Y_1Y$,

(7) $aX_1 \to aXXYX_2$,

(8) $X_2Y \to XY_2$, $Y_2Y \to YY_2$, $Y_2X \to YX_2$.

A továbbiakban (1)-(8) a megfelelő csoport valamelyik szabályát jelöli. (1) (mely a mondatszimbólumból kiinduló szabályokat mutatja) az a és aXX_2Z szavak valamelyikét eredményezi. Tekintsünk a $p_i=aXq_iX_2Z$, $q_i\in\{X, Y\}^*$ szóból terminális szavakhoz vezető levezetéseket. Először csak (2) vagy (4) alkalmazható. Ha (2) került alkalmazásra, akkor a folytatás egyedüli módja (3) alkalmazása egész addig, míg egy $1+3+|q_i|$ hosszú terminális szót nem kapunk. (Mivel a terminális ábécé egy betűből áll, minden szó meg van határozva a hossza által.) Ha (4) kerül alkalmazásra, az aXq_iY_1YXZ szót kapjuk. A folytatás egyedüli módja az "1" index balra léptetése (5)- el és (6) egész addig, amíg (7) alkalmazhatóvá válik. (7) alkalmazása után egy $aXXYX_2q'_iYYXZ$ szóhoz jutunk, ahol q'_i- t úgy kapjuk q_i- ból, hogy X minden előfordulásakor Y- t írunk. A "2" index jobbra léptetésének egyedüli lehetősége a (8), mely a $P_{i+1}=aXQ_{i+1}X_2Z$, $Q_{i+1}=XYQ'_iYY$ szavakhoz vezet. (Megjegyezzük, hogy (8) eleget tesz a "2" index jobbra léptetési követelményének, hisz $X_2X \to XX_2$ nem szükséges, mivel nem fordul elő két egymást követő X.) Egy, a kiinduló p_i szavunkkal megegyező formájú szóhoz jutottunk, s kezdődhet egy új ciklus (2) vagy (4) alkalmazásával. Következésképp, $|q_{i+1}|=|q_i|+q_i[X]+5$, ahol $q_i[X]$ jelöli az X betű előfordulásainak számát q_i- ben. Mivel világos, hogy $q_{i+1}[X]=q_i[X]+2$, kapjuk, hogy a generált terminális szavak hosszai elemei az 1, 1+3, 1+3+5, …, 1+3+…+(2n-1), … sorozatnak.

Másrészt könnyen igazolható, és közismert, hogy $n^2=1+3+\ldots+(2n-1)$(n=1, 2, …). Így $L(G)=\{a^{n^2}|n=1, 2, \ldots\}$. ★

3.21. példa - Unáris prímszámok

$G=(\{S, A, B, C, D, E, X_1, X_2, X_3, Y_A, Y_B, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8, Y_9\}, \{a\}, S, \{S \to a^2, S \to a^3, S \to a^5, S \to a^7, S \to Y_1A^6X_3, Y_1 \to Y_1A^2, Y_1A^3 \to X_1A^3X_2, iAX_2A \to DY_2Y_iE, iDY_2 \to DY_2i, X_1DY_2 \to X_1BY_3, Y_3i \to iY_3, Y_3Y_iE \to Y_iY_3E, Y_3Ei \to iY_3E, Y_3EX_3 \to Y_4CX_3, iY_4 \to Y_4i, Y_iY_4 \to iX_2, (i\in\{A, B\}), X_1B \to X_1Y_5, Y_5B \to BY_5, Y_5X_2A \to Y_6X_2A, BY_6 \to Y_6A, XY_6 \to X_1A, CX_3 \to Y_7X_3, CY_7 \to Y_7A, BA^jX_2Y_7 \to Y_8A^{j+1}X_2, (j\in\{0, 1, 2\}), A^4X_2Y_7 \to Y_8A^4X_2, AY_8 \to Y_8A, BY_8 \to Y_8A, X_1Y_8 \to X_1A, X_2X_3 \to Y_9a, AY_9 \to Y_9a, X_1Y_9 \to aa\})$. Igazolható hogy $L(G)=\{a^p|p$ *-prímszám*$\}$.

Az összes prímszámot a következőképp állíthatjuk elő: Csoportosítsuk a szabályokat a következő módon.

(1) $S \to a^2$, $S \to a^3$, $S \to a^5$, $S \to a^7$,

(2) $S \to Y_1A^6X_3$, $Y_1 \to Y_1A^2$, $Y_1A^3 \to X_1A^3X_2$,

(3) $iAX_2A \to DY_2Y_iE$, $iDY_2 \to DY_2i$, $X_1DY_2 \to X_1BY_3$, $Y_3i \to iY_3$, $Y_3Y_iE \to Y_iY_3E$, $Y_3Ei \to iY_3E$, $Y_3EX_3 \to Y_4CX_3$, $iY_4 \to Y_4i$, $Y_iY_4 \to iX_2$, $i\in\{A, B\}$,

(4) $X_1B \to X_1Y_5$, $Y_5B \to BY_5$, $Y_5X_2A \to Y_6X_2A$, $BY_6 \to Y_6A$, $XY_6 \to X_1A$,

(5) $CX_3 \to Y_7X_3$, $CY_7 \to Y_7A$, $BA^jX_2Y_7 \to Y_8A^{j+1}X_2$, $A^4X_2Y_7 \to Y_8A^4X_2$, $AY_8 \to Y_8A$, $BY_8 \to Y_8A$, $X_1Y_8 \to X_1A$, $j\in\{0, 1, 2\}$

(6) $X_2X_3 \to Y_9a$, $AY_9 \to Y_9a$, $X_1Y_9 \to aa$.

Az (1)- beli szabályok generálják minden 10 alatti prím n- re az a^n alakú terminális szavakat. A (2)- beli szabályok generálják minden 9- nél nem kisebb n- re az $X_1A^3X_2A^{n-6}X_3$ alakú szavakat. A (3)-(5) szabályok tesztelik, hogy n osztható-e 3- al, s ha nem, akkor eljutunk az $X_1A^4X_2A^{n-7}X_3$ szóhoz. Ezután következik a 4- el oszthatóság tesztje. Negatív eredmény esetén X_2 egyet lép jobbra. A pozitív

válasz olyan levezetést eredményez, mely nem vezet terminális szóhoz. Ha az oszthatósági teszt eredménytelen, eljutunk az $X_1A^{n-3}X_2X_3$ szóhoz, s (6) válik alkalmazhatóvá, mely elvezet a^n- hez. A részletesebb bizonyítást mellőzzük. ★

A generatív nyelvtan duális fogalma az *analitikus nyelvtan,* amit a generatív nyelvtanhoz hasonlóan, $G=(N, T, S, H)$ alakban adunk meg, benne N a *változó*(*szimbólumo*)*k,* vagy még más néven, a *nemterminálisok* halmaza, T a *konstansok* vagy *terminálisok* halmaza, S a *mondatszimbólum, H* pedig (mint korábban) a szabályok halmaza. E fogalomnál is használjuk a $V=N\cup T$ jelölést , N^* elemeit *nemterminális szavaknak,* T^* elemeit pedig *terminális szavaknak,* vagy a nyelvtani analógia miatt *mondatoknak* hívjuk (ugyanúgy mint a generatív nyelvtannál). Szemben a generatív nyelvtannal, itt azt tételezzük fel, hogy minden H- beli szabály jobboldala legalább egy N- beli betűt tartalmaz. A *G analitikus nyelvtan által acceptált,* vagy *elfogadott,* vagy még más néven, *felismert nyelven* a T^*- beli szavak következő halmazát értjük: $L(G)=\{w\in T^* | w\Rightarrow_G^* S\}$. Itt is mondhattuk volna, hogy $L(G)=L_a(G)\cap T^*$, ahol $L_a(G)$ a G mint analitikus rendszer, pontosabban az $(N\cup T, \{S\}, H)$ analitikus rendszer által felismert nyelvet jelöli. (Tehát vigyázat: Hasonlóan a generatív változatokhoz, nem tévesztendő össze a G analitikus nyelvtan és a G mint analitikus rendszer által felismert nyelv, hiszen $L(G)\subsetneq L_a(G)$, vagyis a két nyelv soha nem esik egybe!)

Érvényes a következő tétel, melynek bizonyítását az olvasóra bízzuk.

1. Tétel. Tekintsünk egy tetszőleges $G=(N, T, S, H)$ tetszőleges generatív (analitikus) nyelvtant. Alkossuk meg hozzá az összes olyan $q \rightarrow p$ szabályok H' halmazát, melyekre $p \rightarrow q\in H$. Ekkor a $G'=(N, T, S, H')$ analitikus (generatív) nyelvtanra $L(G')=L(G)$.

3.22. példa - Gyakorló feladat

Az 1. Tétel. felhasználásával, továbbá az előző két példa segítségével készítsünk analitikus nyelvtanokat, melyek által felismert nyelvek $L(G)=\{a^{n^2} | n=1, 2, \ldots\}$, illetve $L(G)=\{a^p | p$ - prímszám $\}$. ★

A 1. Tétel. alapján minden (generatív vagy analitikus) nyelvtan befoglalható egy vele ekvivalens duális nyelvtannal alkotott párba. (Nevezetesen, minden generatív nyelvtanhoz tartozik egy vele ekvivalens analitikus nyelvtan, s megfordítva, minden analitikus nyelvtanhoz tartozik egy vele ekvivalens generatív nyelvtan.) A továbbiakban a "nyelvtan" elnevezést fenntartjuk a "generatív nyelvtan" elnevezés rövidítésére. Nyelvtan alatt tehát mindig generatív nyelvtant fogunk érteni (míg analitikus nyelvtan esetén mindig kitesszük az "analitikus" jelzőt).

3.2.1. Chomsky-féle nyelvosztályok

A nyelvtan és az általa generált nyelv definíciója szerint minden nyelvtanhoz egy egyértelműen meghatározott nyelv tartozik, de megfordítva, egy nyelvet nem csak egy nyelvtannal generálhatunk.

Két nyelvtant (*gyengén*) *ekvivalens*nek nevezünk, ha ugyanazt a nyelvet generálják, vagy az általuk generált nyelv legfeljebb az üresszóban tér el. Tehát, formálisan, G_1 és G_2 (gyengén) ekvivalens, ha $L(G_1)\setminus\{\lambda\}=L(G_2)\setminus\{\lambda\}$.

Az ekvivalencia fogalmának ismeretében kézenfekvőnek látszik a különböző nyelvtanokat bizonyos formai tulajdonságok alapján osztályokba sorolni. Az osztályozás alapját a helyettesítési szabályok alakjára vonatkozó megszorítások képezik abban a hierarchiában, amelyet az elmélet egyik megalapozója, N. Chomsky vezetett be, és amelyet alább ismertetünk.

3. Definíció. A $G=(N, T, S, H)$- t *i-típusú nyelvtan*nak (i=0,1,2,3) nevezzük, ha az alábbi megszorítások közül az i- edik teljesül:

(0) *Mondatszerkezetű nyelvtan,* (*Phrase-structure*). A generatív nyelvtan korábban már ismertetett általános definíciójánál feltettük, hogy H olyan $(N\cup T)^*$- beli párokból áll, melyek első eleme (azaz a szabály baloldala) tartalmaz legalább egy nemterminálist. H- ra ebben az esetben (azaz i=0 esetén) ezen kívül nem rovunk ki külön feltételt.

(1a) *Szóhossznemcsökkentő, vagy monoton nyelvtan,* (*Monotone*). Minden H- beli $p \to q$ szabályra $|p| \le |q|$ teljesül, vagy $S \to \lambda$ alakú, de ez esetben, vagyis $S \to \lambda \in H$ előfordulása esetén S nem lép fel egyetlen H- beli szabály jobboldalán sem.

(1) *Környezetfüggő nyelvtan,* (*Context-Sensitive*). Minden H- beli szabály vagy $p_1Ap_2 \to p_1qp_2$ alakú, aholis $p_1, p_2 \in (N \cup T)^*$, $A \in N$, $q \in (N \cup T)^+$, (emlékeztetőül: $(N \cup T)^+ = (N \cup T)^* \setminus \{\lambda\}$) vagy pedig $S \to \lambda$ alakú. Utóbbi esetben, azaz $S \to \lambda \in H$ előfordulása esetén S nem lép fel egyetlen H- beli szabály jobboldalán sem.

(2) *Környezetfüggetlen nyelvtan,* (*Context-Free*). Minden H- beli szabály $A \to p$ alakú, aholis $A \in N$, $p \in (N \cup T)^*$.

(2.5) *Lineáris nyelvtan,* (*Linear*). Minden H- beli szabály $A \to uBv$ vagy $A \to u$ alakú, ahol A, $B \in N$, u, $v \in T^*$.

(3) *Reguláris nyelvtan,* (*Regular*). Minden H- beli szabály vagy $A \to uB$, vagy pedig $A \to u$ alakú, ahol A, $B \in N$, $u \in T^*$. ★

A 0-típusú nyelvtanokat *mondatszerkezetű nyelvtanoknak* is szokás nevezni, utalva azok nyelvészeti eredetére. Az 1a-típusnál a monoton, illetve szóhossznemcsökkentő név önmagáért beszél, itt a levezetés során nem csökkenhet a mondatforma hossza (az egyedüli $S \Rightarrow \lambda$ lehetséges levezetést kivéve). Az 1-típusúakat *környezetfüggőnek* nevezzük, az elnevezés itt arra mutat, hogy egy szabály egyetlen A nemterminális helyébe egy nem üresszónak a beírását jelenti, de csak akkor ha a nemterminális közvetlen környezete egy adott p_1, p_2 szópár. (Természetesen megengedett az is, hogy p_1, p_2 bármelyike, vagy akár mind a kettő üres legyen.) Üres szavat csakis a mondatszimbólum helyébe tehetünk, s azt is csak egy levezetés legelső lépésében teszi lehetővé az 1-típusú nyelvtan. Ezt biztosítjuk azon megszorítással, hogy ha egy 1-típusú nyelvtanban egy $p_1Ap_2 \to p_1qp_2$ szabályban $q=\lambda$ csak úgy lehessen, ha $A=S$, $p_1=p_2=\lambda$ (azaz $S \to \lambda$ szabályról van szó). S hogy egy levezetés során az S helyébe csak az első lépésben lehessen üresszót tenni (a többi nemterminálisra ez a lehetőség is ki van zárva), az $S \to \lambda$ szabály előfordulása esetén kizárjuk annak lehetőségét, hogy S szabályok jobboldalán szerepeljen. Utalva arra, hogy a 2-típusú nyelvtanok esetén egy szabály baloldalán álló nemterminális bármilyen környezetben helyettesíthető a szabály jobboldalán álló szóval, ezesetben *környezetfüggetlen nyelvtanokról* beszélünk. Ez nem áll szemben a környezetfüggéssel, hiszen ott az üresszó, mint környezet is megengedett, mint látni fogjuk inkább arról van szó, hogy a környezetfüggetlen nyelvtanok a környezetfüggőknek speciális esetei. Ha egy 2-típusú nyelvtan minden szabálya legfeljebb egy nemterminálist tartalmaz a jobb oldalán, akkor eljutunk a lineáris grammatika fogalmához. A lineáris nyelvtanok, mint látni fogjuk, a hierarchiában a 2- és a 3-típus között helyezkednek el. Ha egy lineáris nyelvtanban speciálisan az összes $A \to uBv$ alakú szabályban $v=\lambda$, akkor 3-típusú nyelvtanhoz jutunk, amik a *jobb-lineáris nyelvtan* elnevezést innen kapták. Hasonló módon, ha az összes $A \to uBv$ alakú szabályban $u=\lambda$, akkor *bal-lineáris nyelvtanokról* beszélünk.

4. Definíció. Egy nyelvről azt mondjuk, hogy mondatszerkezetű (RE) / monoton / környezetfüggő (CS) / környezetfüggetlen (CF) / lineáris (LIN) / jobblineáris / ballineáris / reguláris (REG), ha mondatszerkezetű / monoton / környezetfüggő / környezetfüggetlen / lineáris / jobblineáris / ballineáris / reguláris nyelvtannal generálható. Továbbá az i=0, 1, 2, 3 értékek esetén azt mondjuk, hogy egy *nyelv i-típusú*, ha van olyan i-típusú nyelvtan, amely azt generálja. ★

Egyszerűen bizonyíthatók a következő tételek.

2. Tétel. Minden G i-típusú (i=0,1,2,3) nyelvtanhoz létezik egy vele ekvivalens G' i-típusú nyelvtan, amelynek szabályai jobb oldalán a mondatszimbólum nem lép fel.

Bizonyítás. Valóban, ha $G=(N, T, S, H)$ i-típusú nyelvtan, akkor egy ilyen G' nyelvtant megadhatunk $G'=(N \cup \{S'\}, T, S', H')$ alakban, ahol $S' \notin N$ és $H' = H \cup \{S' \to p | S \to p \in H\}$. ∎

3. Tétel. Ha az L nyelv i-típusú, akkor $L \cup \{\lambda\}$ is.

Bizonyítás. Legyen $G=(N, T, S, H)$ egy i-típusú nyelvtan melyre $L=L(G)$ és melyben az előző tétel értelmében a mondatszimbólum nem fordul elő egyik szabály jobboldalán sem. Ha L tartalmazza az

üresszót, $L \cup \{\lambda\} = L$ i-típusú volta automatikusan teljesül. Ha nem tartalmazza, akkor tekintsünk egy G' nyelvtant, ahol $G' = (N, T, S, H')$ és $H' = H \cup \{S \rightarrow \lambda\}$. Ekkor a G' i-típusú nyelvtan generálja $L \cup \{\lambda\}$-t. ∎

3.23. példa - Grammatikák főbb típusai 1. példa

Az alábbi szabályok hányas típusú grammatika elemei lehetnek, ha a kisbetűk terminálist, a nagybetűk nemterminálist jelölnek?

$A \rightarrow B$
3-as típusú: ez a szabály $X \rightarrow xY$ alakú, $A = X \in N$ és $x = \lambda \in T^*$, és $Y = B \in N$.
2-es típusú: $X \rightarrow p$ alakú, $X = A \in N$, $p = B \in V^*$.
1-es típusú: $P_1PP_2 \rightarrow P_1QP_2$ alakú, ahol $P_1 = P_2 = \lambda \in V^*$, $P = A \in N$, $Q = B \in V^+$.
0-ás típusú, ahol a szabályokra nincs megkötés azon túl, hogy minden szabály baloldala tartalmaz nemterminálist.
(ha nemcsak a 0,1,2,3 típusokat vizsgáljuk, akkor lineáris és monoton is)

$A \rightarrow BC$
2-es típusú $X \rightarrow p$ alakú, $X = A \in N$, és $p = BC \in V^*$.
1-es típusú: $P_1PP_2 \rightarrow P_1QP_2$ alakú, ahol $P_1 = P_2 = \lambda \in V^*$, $P = A \in N$, $Q = BC \in V^+$.
0-ás típusú, ahol a szabályokra nincs megkötés azon túl, hogy minden szabály baloldala tartalmaz nemterminálist.

$XY \rightarrow XaB$
1-es típusú, $P_1PP_2 \rightarrow P_1QP_2$ alakú, ahol $P_1 = X \in V^*$, $P = Y \in N$, $Q = aB \in V^+$, $P_2 = \lambda \in V^*$.
0-ás típusú, ahol a szabályokra nincs megkötés azon túl, hogy minden szabály baloldala tartalmaz nemterminálist.

$ABC \rightarrow ACB$
0-ás típusú lehet csak, ahol a szabályokra nincs megkötés azon túl, hogy minden szabály baloldala tartalmaz nemterminálist.
(egyébként, ha nemcsak a 0,1,2,3 típusokat tekintjük, akkor monoton is)

$A \rightarrow \lambda$
3-as típusú: ez a szabály $X \rightarrow x$ alakú, $A = X \in N$ és $x = \lambda \in T^*$.
2-es típusú: $X \rightarrow p$ alakú, $X = A \in N$, $p = \lambda \in V^*$.
0-ás típusú, ahol a szabályokra nincs megkötés azon túl, hogy minden szabály baloldala tartalmaz nemterminálist.
(ha nemcsak a 0,1,2,3 típusokat tekintjük, akkor lineáris is. Ha $A = S$ (azaz A a mondatszimbólum), akkor egyéb feltételek teljesülése mellett szerepelhet 1-típusú, vagy monoton nyelvtanban is). ★

3.24. példa - Grammatikák főbb típusai 2. példa

Milyen típusú a $G = (\{S, A, B\}, \{x, y\}, S, H)$ grammatika, ahol H szabályai:
$S \rightarrow AB$, $A \rightarrow BSB$,
$A \rightarrow BB$, $B \rightarrow xAy$,
$B \rightarrow \lambda$, $B \rightarrow x$, $B \rightarrow y$.
Megoldás:
$S \rightarrow AB$; $A \rightarrow BSB$; $A \rightarrow BB$; $B \rightarrow xAy$ megfelel a
0-ás, 1-es és 2-es típusú grammatika követelményeinek, de a 3-asénak nem.
$B \rightarrow x$; $B \rightarrow y$ megfelel a 0-ás, 1-es 2-es és 3-as grammatika követelményeinek.
$B \rightarrow \lambda$ pedig a 0-ás 2-es és 3-as grammatika követelményeinek.
Tehát a grammatikánk minden szabálya megfelel a 0-ás és 2-es típusú grammatika követelményeinek is. Ilyenkor azt szokták nézni, hogy melyik a legnagyobb index, amelyhez tartozó követelményeinek a teljes szabályhalmaz eleget tesz. Ez alapján a nyelvtanunk 2-es típusú. ★

3.25. példa - Grammatikák főbb típusai 3. példa

Milyen típusú a következő grammatika?
G=({S, A, B},{a}, S, H), ahol H szabályai a következők:
$S \rightarrow ABa$, $AB \rightarrow AaBB$,
$B \rightarrow aaa$, $S \rightarrow AS$,
$AAS \rightarrow ABS$.
Megoldás:
$S \rightarrow ABa$ (0-ás, 1-es, 2-es)
$AB \rightarrow AaBB$ (0-ás, 1-es)
$B \rightarrow aaa$ (0-ás, 1-es, 2-es, 3-as)
$S \rightarrow AS$ (0-ás, 1-es, 2-es)
$AAS \rightarrow ABS$ (0-ás, 1-es)
Grammatikánk eleget tesz a 0-ás és az 1-es típus követelményeinek,
ezért a grammatikánk 1-es típusú lesz. ★

3.26. példa - Grammatikák főbb típusai 4. példa

Hányas típusú a következő grammatika?
G=({ S, A, B }, { a, b, c }, S, H), ahol H szabályai a következők:
$S \rightarrow ABc$, $A \rightarrow aB$,
$A \rightarrow Bc$, $B \rightarrow aAc$,
$B \rightarrow bc$.
Adjuk meg az összes hatbetűs szót a grammatika által generált nyelvben!

Megoldás:
Az összes szabály eleget tesz a 0-ás, 1-es és a 2-es típus követelményeinek,
ezért 2-es típusú a grammatika.

Most határozzuk meg a hatbetűs szavakat!
Először is azt kell megállapítsuk, hogy a grammatika szabályai közül a baloldal mindenhol rövidebb, mint a jobboldal, vagyis egy szóból sehol sem kapunk rövidebbet (sőt, mindig szigorúan hosszabb mondatformát kapunk), tehát a levezetéseket elég hatbetűs szavakig nézni, és azok közül azok lesznek a grammatika által generált nyelvben, melyek csak terminálist tartalmaznak.
A levezetés első lépése mindenképpen az $S \rightarrow ABc$ lesz.
Az A helyére vagy Bc vagy aB kerülhet csak.
Mindkét esetben lesz egy négybetűs szavunk, melyben lesz két terminális
és két B nemterminális. B helyére csak bc kerülhet, mert különben úgy kapunk hatbetűs szót, hogy még marad benne nemterminális.
Ezért a levezetéseink a következők lesznek:
$S \Rightarrow ABc \Rightarrow aBBc \Rightarrow abcBc \Rightarrow abcbcc$,
$S \Rightarrow ABc \Rightarrow BcBc \Rightarrow bccBc \Rightarrow bccbcc$.
Természetesen van még ezen kívül több levezetés is, de az
csak a nemterminálisok helyettesítésének sorrendjében tér el. ★

3.27. példa - Grammatikák főbb típusai 5. példa

Milyen típusú a G=({S, A, B}, {0, 1}, S, {$S \rightarrow AB$, $A \rightarrow 0B1$, $0B \rightarrow 011$, $1B \rightarrow 11BS$, $S \rightarrow \lambda$})?

Megoldás: A $0B \rightarrow 011$ és $1B \rightarrow 11BS$ alakú szabályok miatt csak 1- és 0-típusú lehet. Az $S \rightarrow \lambda$ szabály miatt lehetne 1-típusú a nyelvtan, ha S nem szerepelne egyik szabály jobboldalán sem, de mivel $1B \rightarrow 11BS$ szabályra ez nem teljesül, a nyelvtan 0-típusú.

3.28. példa - Grammatikák főbb típusai 6. példa

Hányas típusú a következő grammatika?
$G=(\{S\},\{x, y, +, *,), (\}, S, H)$, ahol H szabályai a következők:
$S \rightarrow S+S$, $S \rightarrow S*S$, $S \rightarrow (S)$, $S \rightarrow x$, $S \rightarrow y$.
Adjunk meg az $y*(x+y)$ szó egy levezetését!

Megoldás:
$S \rightarrow S+S$; $S \rightarrow S*S$; $S \rightarrow (S)$ lehetnek 0-ás, 1-es és 2-es típusú szabályok.
$S \rightarrow x$; $S \rightarrow y$ pedig 0-ás, 1-es, 2-es és 3-as típusú szabályok, vagyis minden szabály eleget tesz a 0-ás 1-es és a 2-es típus követelményeinek, tehát a grammatika 2-es típusú lesz.

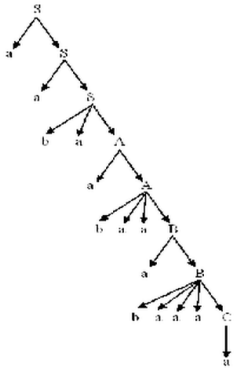
Az $y*(x+y)$ szó egyik levezetése a következő:
$S \Rightarrow S*S \Rightarrow y*S \Rightarrow y*(S) \Rightarrow y*(S+S) \Rightarrow y*(x+S) \Rightarrow y*(x+y)$ ★

3.29. példa - Grammatikák főbb típusai 7. példa

Adott a $G=(\{S,A,B,C\},\{a,b\},S, H)$ nyelvtan, ahol H szabályai:
$S \rightarrow aS$, $S \rightarrow baA$, $A \rightarrow aA$, $A \rightarrow baaB$,
$B \rightarrow aB$, $B \rightarrow b$, $B \rightarrow baaaC$, $C \rightarrow aC$, $C \rightarrow a$.
Milyen típusú? Adjuk meg az *aabaabaaabaaaa* szó levezetését!
Megoldás:
A nyelvtan szabályait megnézve, láthatjuk, hogy az összes szabály eleget tesz a 0-ás, 1-es, 2-es és 3-as típusú nyelvtanok követelményeinek, tehát a nyelvtan 3-as típusú.

A generálás az első két szabály valamelyikével kezdődhet.
Mivel a keresett szó a-val kezdődik, ezért az 1. szabályt kell használnunk először.
Az első szabály n-szeri (n tetszőleges nem-negatív egész) alkalmazásával a^nS-et kapunk.
Ahhoz, hogy az S szimbólum eltűnjön a mondatformából a második szabály használatára van szükség.
Ennek alkalmazása után a^nbaA-hoz jutunk.
Ezután olyan A-ból kiinduló szabályt kell keresni, ahol a következik a nyíl után.
Ezért a 3. szabály m-szeri alaklmazásával a^nbaa^mA-t kapunk.
Majd a negyedik szabállyal: a^nbaa^mbaaB-t.
Ezután három alkalmazható szabály van.
Az ötödik k-szori alkalmazása az $a^nbaa^mbaaa^kB$ alakhoz vezet.
Ekkor a levezetés befejezhető a $B \rightarrow b$ szabállyal: $a^nba\ a^mbaaa^kb$.
Alternatívaként a $B \rightarrow baaaC$ szabállyal folytatható a levezetés: $a^nbaa^mbaaa^kbaaaC$.
Ekkor a $C \rightarrow aC$ j-szeri alkalmazása az $a^nbaa^mbaa\ a^kbaaa\ a^jC$ formához vezet, a levezetés ekkor az utolsó szabállyal fejezhető be: $a^nbaa^mbaaa^kbaaaa^ja$ eredménnyel.

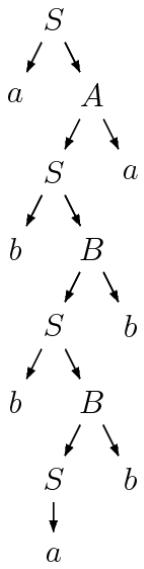
A levezetéseket gráf segítségével is szemléltethetjük. Ebben az esetben az *aabaabaaabaaaa* szólevezetését a következő fa reprezentálja:


S
a
S
a
S
b
a
A
a
A
b
a
a
B
a
B
b
a
a
a
C
a

★

3.30. példa - Grammatikák főbb típusai 8. példa

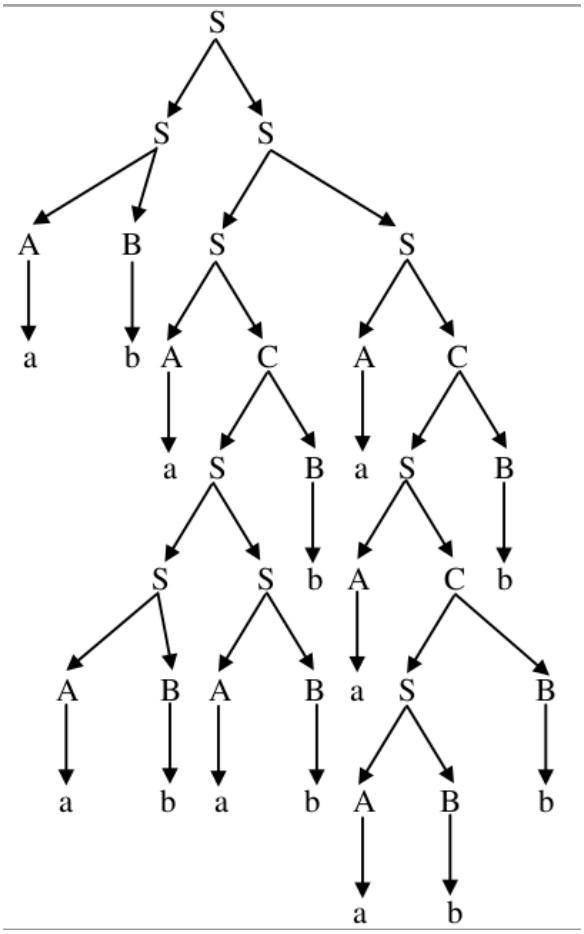
Adott a G=({ S,A,B }, { a,b }, S, H) nyelvtan, ahol H={$S \to a\,A$, $A \to Sa$, $A \to a$, $S \to bB$, $B \to Sb$, $B \to b$, $S \to a$, $S \to b$ }.
Milyen típusú? Mely nyelvet generálja?
Adjuk meg az *abbabba* szó levezetését fa alakban!
Megoldás:
Amint látjuk a szabályok közt van jobb-, illetve bal-lineáris alakú is, ezért a megadott grammatika nem reguláris. Az viszont minden szabályra teljesül, hogy a nyíl után legfeljebb egy nemterminális betű szerepel. Tehát a nyelvtan lineáris.
A nyelvtan szabályait csoportokban vizsgálhatjuk.
Az első szabály alkalmazása után a második vagy harmadik szabályt is fel kell használnunk a levezetésben.
Az első után a harmadik szabállyal a levezetés befejeződik, az S szimbólum helyére aa kerül így a levezetésben.
Az első és a második szabályok az eredeti S-t aSa-val helyettesítik.
Hasonlóan a következő három szabály az S szimbólumot a bSb, illetve a bb szavakkal helyettesítheti.
Ezek alapján könnyen belátható, hogy a generált szó az elejéről olvasva éppen ugyanaz, mint a végéről visszafelé olvasva.
Az is látszik, hogy az $\{a,b\}$ ábécé felett minden ilyen szót előállít a nyelvtanunk.
Az utolsó két szabály a páratlan hosszúságú ilyen tulajdonságú szavak levezetésének befejezesében játszik szerepet.
Ezt a nyelvet egyébként a kétbetűs ábécé feletti palindrom nyelvnek nevezzük.
Az ábra az *abbabba* szó levezetési fáját mutatja.



★

3.31. példa - Grammatikák főbb típusai 9. példa

Legyen adott $G=(\{S, A,B,C\},\{a,b\}, S, H)$ generatív nyelvtan, ahol $H=\{ S \to SS, S \to AB, S \to AC, S \to SB, A \to a, B \to b\}$.
Milyen típusú ez a nyelvtan? Mely nyelvet generálja?
Adjuk meg az *abaababbaaabbb* szó levezetési fáját!
Megoldás:
A nyelvtan környezetfüggetlen.
Az utolsó két szabály alapján az *A*, illetve a *B* szimbólumok az *a* és *b* terminálisoknak felelnek meg.
A második, illetve a harmadik és negyedik szabályok használata egy-egy *A* és *B* szimbólumot vezet be, mégpedig úgy, hogy az *A* a *B* előtt szerepel. A nyelv minden szavában megegyezik tehát az *a* és *b* betűk száma.
Másrészt az is igaz, hogy a nyelv egy szavának minden kezdőszeletében legalább annyi *a* van, mint ahány *b*.
(Ez a nyelv egyébként a kétbetűs ábécé fölötti Dyck-nyelv, vagyis a zárójelek nyelve.
Zárójelei: *a* a nyitó; *b* pedig a záró zárójeleket jelenti.)
Az ábra mutatja az *abaababbaaabbb* szó levezetési fáját.



★

3.2.2. A Chomsky hierarchia

Definíció szerint azonnal látható, hogy minden 3-típusú nyelvtan egyben lineáris, és minden lineáris nyelvtan egyben 2-típusú is, valamint minden 1-típusú nyelvtan monoton és minden monoton nyelvtan egyben 0-típusú is. Az is nyilvánvaló, hogy minden 2-típusú, lineáris vagy 3-típusú nyelvtan egyben 0-típusú is. Nézzük meg az 1-típusú és a 2-típusú nyelvtanok közötti kapcsolatot.

4. Tétel. (Üresszó-lemma). Minden környezetfüggetlen G grammatikához megadható olyan G' környezetfüggetlen nyelvtan, hogy $L(G)=L(G')$ (azaz az általuk generált nyelv ugyanaz), s ha $\lambda \notin L(G)$, akkor a G'- beli szabályok jobboldalán λ nem fordul elő. Ha viszont $\lambda \in L(G)$, akkor az egyetlen G'- beli szabály aminek jobboldala az üresszó $S' \to \lambda$, ahol S' a G' mondatszimbólumát jelöli. Ezesetben, azaz $\lambda \in L(G')$ fennállása esetén viszont S' (azaz a G' mondatszimbóluma) nem fordulhat elő egyetlen G'- beli szabály jobboldalán sem. Ennek megfelelően, tahát G' nemcsak környezetfüggetlen, de egyben a környezetfüggő definíciónak is eleget tesz.

Bizonyítás. Legyen $G=(N, T, S, H)$ 2-típusú nyelvtan. Megszerkesztünk hozzá egy $G_1=(N, T, S, H_1)$ grammatikát a következő módon. Definiáljuk az U_i halmazokat (i=0,1,...) a következő módon: először az $U_1=\{A | A \to \lambda \in H\}$, majd az $U_{i+1}=U_i \cup \{A | A \to p \in H, p \in U_i^*\}$, $i \geq 1$ halmazokat. Mivel $U_1 \subseteq U_2 \subseteq \ldots U_i \subseteq \ldots \subseteq N$, és N véges, azért létezik olyan k, hogy $U_k=U_{k+1}$. Az U_i (i=1, 2, …) halmazok konstrukciója alapján ekkor látható, hogy minden m pozitív egészre $U_k=U_{k+m}$. Jelöljük ezt U- val, tehát $U=U_k$. Erre az U- ra ekkor $A \Rightarrow^* \lambda$ akkor és csak akkor ha $A \in U$ ($A \in N$), azaz $\lambda \in L(G)$ akkor és csak akkor ha $S \in U$. Tekintsük most azt a $G_1=(N, T, S, H_1)$ grammatikát, amelyre $A \to p_1 \in H_1$ pontosan akkor, ha $p_1 \neq \lambda$ és van olyan $A \to p \in H$, hogy $p_1=p$, vagy p_1 előállíthtó p- ből úgy, hogy p- ből U- beli elemet vagy elemeket hagyunk el. p- ből tehát egy ilyen p_1 szót úgy származtatunk, hogy alkalmas U- beli (nem feltétlen egymástól páronként különböző) A_1, …, A_m nemterminális betűkre és $(N \cup T)^*$- beli q_1, $\ldots q_{m+1}$ szavakra $p=q_1A_1 \ldots q_mA_mq_{m+1}$ mellett $p_1=q_1 \ldots q_{m+1}$ fennálljon. Ekkor nyilvánvaló, hogy $L(G_1) \subseteq L(G) \setminus \{\lambda\}$. De megfordítva, $L(G) \setminus \{\lambda\} \subseteq L(G_1)$ is fennáll, hisz ha $S \Rightarrow^* p$ és $p \neq \lambda$, akkor fennáll $p \in L(G')$ is, mivel a p- nek egy G- beli levezetése esetén minden $A \to \lambda$ alakú szabály alkalmazása helyett vehetjük a megfelelő H_1- beli szabályokat. Így azt kaptuk, hogy $L(G) \setminus \{\lambda\}=L(G_1)$. Ha tehát $\lambda \notin L(G)$, akkor G'- ként G_1- et választhatjuk. Ha pedig $\lambda \in L(G)$, akkor egy új S' ($\notin N \cup T$) szimbólumot választva vegyük a $G'=(N \cup \{S'\}, T, S', H_1 \cup \{S' \to S, S' \to \lambda\})$ nyelvtant, mely ezesetben nyilvánvalóan eleget tesz a tételbeli tulajdonságoknak, így a tétel bizonyítást nyert. ∎

Láthatjuk tehát, hogy a környezetfüggetlen nyelvek generálhatóak olyan (környezetfüggő) nyelvtanokkal, melyekben minden szabály bal oldalának hossza egy.

A fenti tételünk alapján könnyen látható, a Chomsky-féle nyelvosztályok alábbi hierarchiája:

$REG \subseteq LIN \subseteq CF \subseteq CS \subseteq RE$.

Ez a hierarchia valójában élesebb, de ezt a későbbiekben az adott nyelvosztályok vizsgálatakor fogjuk bizonyítani.

A jegyzet során e hirarchia nyelvosztályait is sorra vesszük, és különböző fontos tulajdonságaikat viszgáljuk, ezeket a problémaköröket ismertetjük a Problémakörök alfejezetben.

3.32. példa - Üresszó-lemma 1. feladat

G=({S, A, B},{x, y}, S, H), ahol H a következő szabályokból áll:
$S \to AB$, $A \to BSB$, $A \to BB$, $B \to xAy$,
$B \to \lambda$, $B \to x$, $B \to y$.

Először nézzük meg, hogy az üresszó benne van-e a G által generált nyelvben!

U_1:={B} - azok a nemterminálisok, amelyekből λ közvetlenül megkapható.
U_2:={A,B} - azok a nemterminálisok, amelyekből U_1^*-beli szavak közvetlenül megkaphatóak!
U_3:={S, A, B} - azok a nemterminálisok, amelyekből U_2^*-beli szavak közvetlenül megkaphatóak.

Mivel U_3 a nyelvtan összes nemterminálisát tartalmazza, ezért U:=U_3=U_4=U_5=...
Tehát U-beli nemterminálisokból kapható meg az üresszó.
Mivel $S \in U$, ezért az üresszó benne van a G által generált nyelvben.
Készítsük el a G'=({S', S, A, B},{x, y}, S', H') grammatikát, ahol H' a következő szabályokból áll:

I. $S' \to S$, $S' \to \lambda$. Ez a két szabály azért került be a H' szabályai közé, mert $\lambda \in L(G)$, ezért $\lambda \in L(G')$-nek is teljesülnie kell!

II. Most vegyük azokat a H-beli szabályokat, amelyekben nem a λ szerepel a jobboldalon. Ezek a következők:
$S \to AB$, $A \to BSB$, $A \to BB$, $B \to xAy$, $B \to x$, $B \to y$.

III. És vegyük még azokat a szabályokat, amelyek H-ban nem szerepeltek, és amelyeket úgy kapunk, hogy H-beli szabályok jobb oldaláról az összes lehetséges módon
elhagyjuk U halmaz nemterminálisait, figyelve azonban arra, hogy a jobb oldalon λ ne maradjon magában:
$S \to A$, $S \to B$,
$A \to BS$, $A \to SB$, $A \to B$, $A \to S$,
$B \to xy$.

★

3.33. példa - Üresszó-lemma 2. feladat

Készítsen a következő grammatikával ekvivalens 1-es típusú grammatikát!
G=({S, A, B, C, D}, {a, b}, S,H), ahol H szabályai:
$S \to AB$, $A \to CB$, $A \to SS$, $A \to a$, $B \to \lambda$,
$C \to D$, $D \to \lambda$, $D \to b$.

Megoldás:
U={S,A,B,C,D}
$S \in U$, ezért $\lambda \in L(G)$, vagyis új kezdőszimbólumot kell bevezetni.
G'=({S',S, A, B, C, D}, {a, b}, S', H'), ahol H' szabályai:

I. $S' \to S$ *és* $S' \to \lambda$, mert $\lambda \in L(G)$

II. $S \to AB$, $A \to CB$, $C \to D$, $A \to SS$, $A \to a$, $D \to b$ lesznek azok a szabályok H-ból melyekben nem λ van a jobboldalon.

III. $S \to A$, $S \to B$, $A \to C$, $A \to B$, $A \to S$ lesznek az új szabályok, amelyeket H-beli szabályok jobb oldaláról U-beli nemterminálisok elhagyásával kapunk.

Ez a grammatika ekvivalens a G grammatikával, de már 1-es típusú. ★

3.34. példa - Üresszó-lemma 3. feladat

Küszöbölje ki a törlő szabályokat a $G=(\{S, A, B\}, \{a, b\}, S,H)$ grammatikából!
$H=\{S \to \lambda, S \to AB, A \to SAB, A \to a, B \to S, B \to b\}$.

Megoldás:
A feladat az Üresszó-lemma alkalmazásával oldható meg:
$U=\{S, B\}$
$S \in U$, ezért $\lambda \in L(G)$, vagyis új kezdőszimbólumot kell bevezetni.
$G'=(\{S',S, A, B\}, \{a, b\}, S', H')$, ahol H' szabályai:

I. $S' \to S$ *és* $S' \to \lambda$, mert $\lambda \in L(G)$.

II. $S \to AB, A \to SAB, A \to a, B \to S, B \to b$ lesznek azok a szabályok H-ból, amelyekben nem λ van a jobboldalon.

III. $S \to A, A \to SA, A \to AB$ lesznek az új szabályok, amelyeket H-beli szabályok jobb oldaláról U-beli nemterminálisok elhagyásával kapunk.
(Az $A \to A$ alakú szabályt nyugodtan elhagyhatjuk, mivel levezetés szempontjából nincs hatása.)

★

3.35. példa - Üresszó-lemma 4. feladat

Küszöbölje ki a törlő szabályokat a $G=(\{S, A, B, C, D, E\},\{a, d\}, S,H)$ grammatikából!
$H=\{S \to SA, S \to BS, A \to a, B \to CDE, C \to DddE, C \to d, C \to \lambda, D \to E, D \to d,$
$E \to \lambda\}$.

Megoldás:
A feladat az Üresszó-lemma alkalmazásával oldható meg:
$U=\{B, C, D, E\}$
$S \notin U$, ezért $\lambda \notin L(G)$, vagyis nem kell új kezdőszimbólumot bevezetni.
$G'=(\{S, A, B, C, D, E\}, \{a, d\}, S, H'\})$, ahol H' szabályai:

I. Azok a szabályok H-ból melyekben a jobboldal nem λ:
$S \to SA, S \to BS, A \to a, B \to CDE, C \to DddE, C \to d, D \to E, D \to d$

II. Az új szabályok, melyeket U-beli nemterminálisok elhagyásával kapunk:
$B \to CD, B \to CE, B \to DE, B \to C, B \to D, B \to E,$
$C \to ddE, C \to Ddd, C \to dd$ ($S \to S$ alakú szabálynak nincs jelentősége.)

★

3.36. példa - Üresszó-lemma 5. feladat

Hozza a $G=(\{S, A, B, C\}, \{a, b\}, S, H)$ grammatikát 1-es típusúra!
$H=\{S \to ASC, S \to BA, B \to ACB, B \to AA, C \to bB, A \to \lambda, A \to a, B \to ba\}$.

Megoldás:
$U=\{S, A, B\}$
$S \in U$, ezért $\lambda \in L(G)$, vagyis új kezdőszimbólumot kell bevezetni.
$G'=(\{S', S, A, B, C\}, \{a, b\}, S', H')$, ahol H' szabályai:

I. $S' \to S$ *és* $S' \to \lambda$, mert $\lambda \in L(G)$.

II. $S \to ASC, S \to BA, B \to ACB, B \to AA, C \to bB, A \to a, B \to ba$
lesznek azok a szabályok H-ból amelyekben nem λ a jobboldal.

III. $S \to AC, S \to SC, S \to C, S \to A, S \to B, B \to CB, B \to AC, B \to C, B \to A, C \to b$
lesznek az új szabályok, amelyeket U-beli nemterminálisok elhagyásával kapunk.

★

3.3. L-rendszerek

A Markov-féle normál algoritmus esetén minden lehetséges lépésben egyértelműen meg volt fogalmazva, hogy melyik szabályt és hol kell alkalmazni (illetve, ha kiszámoltuk a megoldást, és nem folytatódik a számítás). Ezzel szemben a generatív nyelvtanoknál sem az hogy melyik szabályt (több alkalmazható is lehet egyszerre), sem az hogy hol (egy adott szabály az aktuális mondatforma több helyén is alkalmazható lehet ugyanakkor) kell alkalmazni nincs egyértelműen megadva, vagyis a nyelvtanok nemdeterminisztikusak. Viszont minden eddigi korábban ismertetett rendszerre teljesült, hogy minden időpillanatban pontosan egy szabályt pontosan egy helyen alkalmaztunk, vagyis az eddig tárgyalt rendszerek szekvenciális működésűek. Ebben a fejezetben egy olyan modellt vizsgálunk meg, amely alapvetően párhuzamos.

Aristid Lindenmayer (1925-1989), a Fasori Gimnáziumba (Budapesti Evangélikus Gimnázium) járt, hasonlóan több magyar Nobel-díjashoz (pl. Wigner Jenő), vagy Neumann Jánoshoz. Később Hollandiában tevékenykedő biológusként először bizonyos algafajok növekedési mintázatainak leírására alkalmazott formális rendszereket, majd ezeket a matematikai eszközöket magasabb szintű növények fraktálszerkezetének leírására alkalmazták.

5. Definíció. Egy L-rendszer (Lindenmayer-system, L-system) egy párhuzamos átíró rendszer $LS=(T, s, H)$, ahol T egy véges ábécé, $s \in T^*$ (axióma), H pedig $a \to r$ alakú ($a \in T$, $r \in T^*$) átíró szabályok halmaza.

Az alap L-rendszerekben minden $a \in T$ betűhöz pontosan egy átíró szabály (esetleg $a \to a$ alakú) létezik. Egy levezetési lépésben az adott mondatforma/szó (axióma) minden betűjét helyettesítjük a megadott átírási szabályok alapján, így létrehozva a következő mondatformát/szót.

Az LS rendszer által generált nyelv tartalmazza az összes olyan szót (beleértve az axiómát is) amely véges sok lépésben generálható az axiómából.

3.37. példa - Fibonacci szavak

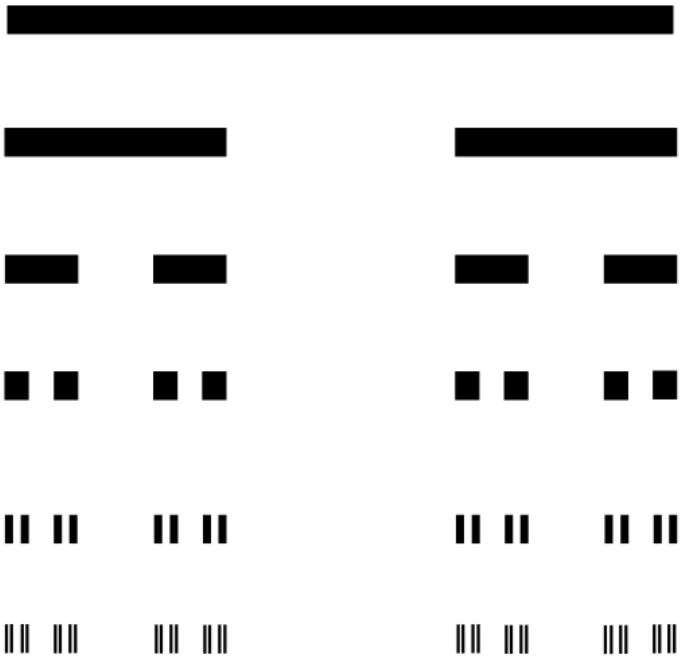
Legyen $(\{a, b\}, a, \{a \to b, b \to ba\})$ L-rendszer. Ekkor könnyen ellenőrizhető, hogy az ezzel a rendszerrel generált nyelv első néhány szava: a, b, ba, bab, $babba$, $babbabab$, … Ez a Fibonacci szavak nyelve.

Figyeljük meg, hogy az egymást követő szavak hosszai éppen a Fibonacci számok (az $f(0)=1$, $f(1)=1$, $f(i)=f(i\text{-}1)+f(i\text{-}2)$, ($i>1$- re) rekurzív képlettel definiált sorozat). Másrészt a szavak felépítése is hasonlít a számsorozatot előállító képlet alkalmazására: $w(0)=a$, $w(1)=$b, $w(i)=w(i\text{-}1)w(i\text{-}2)$. ★

3.38. példa - Fraktálgenerálás L-rendszerrel

A Cantor-halmaz egyike a jól ismert fraktáloknak, ennek egy előállítása történhet a következő L-rendszer segítségével: ({0,1}, 1, {1 → 101, 0 → 000}).

A generálás folyamata: 1, 101, 101000101, 10100010100000000101000101 jobban követhető a következő képen, ahol 1 jelzi a szakaszokat, 0 pedig azok hiányát:

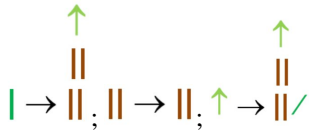


A levezetést a végtelenségig folytatva (határsetben) megkapjuk a Cantor által definiált fraktált. ★

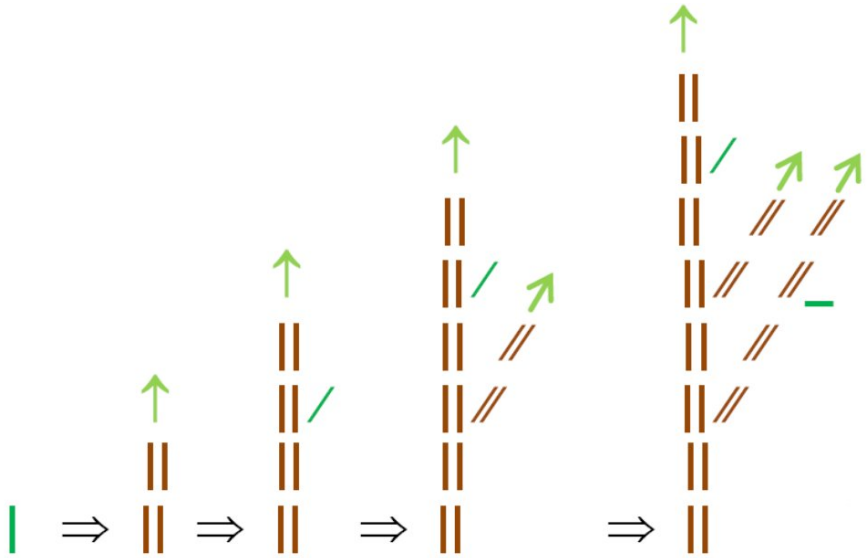
Növények formáját, hasonlóan, néhány egyszerű átíró szabállyal tudjuk kódolni, valószínűleg a természet is hasonlóképpen kódolja a kifejlett növény felépítését a növény génállományában. A következőkben egy egyszerű ilyen jellegű példát mutatunk.

3.39. példa - Kétdimenziós rajz L-rendszerrel

({ , , }, , *H*), ahol a *H* szabályhalmaz elemei:



A generálás első néhány lépése:



★

Természetesen rengeteg változata van az L-rendszereknek, itt csak a legalapvetőbb változatot ismertettük röviden.

3.4. Problémakörök

Az alábbiakban áttekintjük, hogy a jegyzetben a különböző típusú nyelvekkel kapcsolatban mi is érdekel minket. Általában adott nyelvosztályra illusztratív példát is adunk, valamint bizonyítjuk a Chomsky-féle hierarchia szigorúságát. Egyes nyelvosztályok tulajdonságai erősen különbözhetnek egymástól. Ebben a jegyzetben általában a következő tulajdonságokat fogjuk vizsgálni.

3.4.1. Normálformák (Normálalakok)

A monoton és a 0-típusú nyelvtanokban a terminális szimbólumokat is átírhatjuk, itt a terminális és nemterminális szimbólumoknak a megkülönböztetése inkább csak azért fontos, hogy lássuk, készen vagyunk-e a levezetéssel (csak terminálisokból áll az aktuális modatforma), vagy van benne nemterminális, ahol még szabályt kell alkalmaznunk ha be akarjuk fejezni a levezetést (az más kérdés, hogy nem feltétlenül lehet befejezeni minden levezetést). A továbbiakban belátjuk, hogy minden nyelvtannal van olyan ekvivalens, amiben terminális szimbólumokat nem írunk át, vagyis a terminális szimbólumok tényleg terminálisak, véglegesen ott maradnak a levezetésben, míg a változókat természetesen át kell írnunk egy terminálló levezetés során.

Azt mondjuk, hogy egy nyelvtan (*terminális*) *normális alak*ban van, ha a helyettesítési szabályokban terminális jelek csak $A \to a$ ($A \in N$, $a \in T$) alakú szabályokban fordulnak elő.

5. Tétel. Minden $G=(N, T, S, H)$ nyelvtanhoz megadható egy vele ekvivalens $G'=(N', T, S, H')$ nyelvtan úgy, hogy az általuk generált nyelvek megegyeznek: $L(G)=L(G')$, továbbá minden H'-

beli szabály, amely terminális jelet tartalmaz, $A \rightarrow a$ alakú, ahol $A \in N$, $a \in T$. Emellett G' nyelvtan ugyanolyan típusú, mint G, kivéve, ha G reguláris, vagy lineáris.

Bizonyítás. Minden egyes $a \in T$ terminális jelhez vezessünk be új $D_a \notin N$ nemterminálist, és legyen $N' = N \cup \{D_a | a \in T\}$. A H'- beli szabályokat adjuk meg úgy, hogy a H szabályaiban minden nem $A \rightarrow a$ ($A \in N$, $a \in T$) alakú szabályban az a terminális helyére a megfelelő D_a változót írjuk. Ezenkívül a H'- beli szabályok közé még felvesszük az $D_a \rightarrow a$ alakú új szabályokat (minden $a \in T$ esetén). Az így kapott H' nyilván rendelkezik a kívánt tulajdonsággal.

Az ekvivalencia bizonyításához először megmutatjuk, hogy $L(G) \subseteq L(G')$. Legyen $w = a_1 a_2 \ldots a_k \in L(G)$. Ekkor a G' nyelvtanban levezethető a megfelelő $D_1 D_2 \ldots D_k$ nemterminális szó, és az $D_a \rightarrow a$ szabályok segítségével ebből a w- t megkaphatjuk. Ha továbbá $\lambda \in L(G)$, akkor a G- ben a megfelelő szabályok alkalmazásával ugyancsak megkapjuk λ- t.

Most lássuk be, hogy $L(G') \subseteq L(G)$. Legyen $h:(N' \cup T)^* \rightarrow (N \cup T)^*$ homomorf leképezés a következő módon értelmezve:

- $h(D_a) = a$, (minden $a \in T$ esetén) illetve

- $h(r) = r$, $r \in (N \cup T)$.

Ekkor bármely két $p, q \in (N' \cup T)^*$ szóra a $p \Rightarrow_{G'} q$ relációból következik, hogy vagy $h(p) = h(q)$, vagy $h(p) \Rightarrow_G h(q)$. Ha ugyanis a G nyelvtanban a p- ből a q úgy adódik, hogy valamelyik $D_a \rightarrow a$ szabályt alkalmazzuk, akkor $h(p) = h(q)$. Ha pedig H'- nek egy olyan szabályát alkalmazzuk, amelyet egy H- beli szabályból nyertünk, akkor nyilván $h(p) \Rightarrow_G h(q)$. Tehát mindkét esetben $p \Rightarrow_{G'} q$ relációból a $h(p) \Rightarrow_G h(q)$ következik. Legyen most $p \in L(G')$, akkor $S \Rightarrow_G p$, mert $S = h(S) \Rightarrow_{G'} h(p) = p$, amivel a tételt bebizonyítottuk. ∎

A továbbiakban különböző típusú nyelvtanokhoz ennél jóval erősebb megkötéseket tartalmazó normálformákat is fogunk bemutatni. Az egyik fontos kérdés az lesz, hogy mennyi korlátozást tehetünk adott nyelvtanosztály szabályaira, ahhoz, hogy továbbra is generálhassuk a nyelvosztály üresszómentes nyelveit, azaz minden adott típusú nyelvtannal legyen ekvivalens amire a korlátozás fennáll.

3.4.2. Levezetések szerkezete

Mivel a levezetés központi fogalom egy nyelvtan által generált nyelv előállításában, érdemes megvizsgálnunk a lehetséges levezetések szerkezetét. A levezetéseket gráfokkal fogjuk ábrázolni, a levezetési gráf alakja (pl. fa) nemcsak szemléletes, de elméleti és gyakorlati fontossággal is bír. Mindez szorosan összefügg a következőkben ismertetendő szóproblémával is.

3.4.3. A szóprobléma és a szintaktikai elemzés

Ha adott egy nyelv (pl. nyelvtan segítségével definiálva), akkor annak eldöntését, hogy egy adott w szó benne van-e a (generált) nyelvben, az adott nyelvhez (nyeltanhoz) tartozó szóproblémának hívjuk. Általában nemcsak egy konkrét nyelv érdekel minket, hanem az hogy adott nyelvosztály esetén hogyan oldható meg (illetve megoldható-e egyáltalán) a szóprobléma. Ezt eldönteni, illetve erre hatékony algoritmust megadni érdekes és fontos része a formális nyelvek elméletének.

Ha nem csak igen/nem választ várunk el, hanem igen válasz esetén azt is hogy egy lehetséges levezetést is megadjon az algoritmus, akkor szintaktikai elemzésről beszélünk.

A mesterséges intelligenciában szokásos módon állapottér gráffal szemléltethetjük az összes lehetséges levezetést egy adott nyelvtanban: a gyökér csúcs címkéje az S mondatszimbólum, és bármely csúcsból úgy származtathatjuk annak "utódait", hogy a csúcs címkéjében levő mondatformára valamely levezetési szabályt alkalmazzuk valamely alkalmas helyen. Ilyenkor grafikusan irányított élt húzunk az adott csúcsból abba amelyben, mint címke, az új mondatforma található. Mivel bármely (véges) mondatforma esetén csak véges sok szabály és véges sok helyen alkalmazható,

így felsorolhatjuk az összes olyan mondatformát, amaly létrejöhet a már meglevő mondatformákból egy lépésben. Ez alapján a gráf alapján kereshetünk választ a formális nyelvek elméletének egyik legfontosabb problémájára, a szóproblémára: Ezt a problémát a következő formában is megfogalmazhatjuk: egy adott L nyelv és egy adott w szó esetén milyen feltételek mellett dönthető el algoritmikusan az, hogy $w \in L$ reláció teljesül-e vagy sem. Könnyű észrevenni, hogy végtelen nyelvek esetén a gráfnak végtelen sok levél eleme van, sőt azok mélysége (vagyis a legrövidebb, a gyökércsúcsból induló és az adott levélcsúcshoz tartó irányított út, vagyis a legrövidebb levezetés, hossza) sem korlátos. A szóprobléma eldöntése ezeknek a levélelemeknek az egyenkénti megvizsgálását jelentené, ami általában algoritmikusan nem megoldható. Azonban speciális esetekben, például a szó hosszát nem csökkentő nyelvtanok (monoton nyelvtanok) esetén a levezetési gráfoknak csak egy véges részgráfját kell a vizsgálatunkba bevonni, ami garantálja a probléma algoritmikus eldönthetőségét.

3.4.4. Nyelvosztályok és automataosztályok kapcsolata

Egyik központi feladatunk a különböző nyelvosztályok elfogadására alkalmas automataosztályok feltérképezése, vagyis, hogy adott nyelvosztályba tartozó nyelveket milyen automatákkal lehet elfogadni. Néhány nyelvosztály esetén egyéb speciális, alternatív, a nyelvosztálynak megfelelő leírást is fogunk adni a benne szereplő nyelvekre.

3.4.5. Nyelvosztályok tulajdonságai

Az, hogy egy adott formális nyelv milyen nyelvosztályba tartozik, nem dönthető el minden egyes nyelv esetén könnyen. Néhány nyelvosztály esetén segítséget jelenthet, ha olyan tulajdonságot tudunk, ami a nyelvosztály minden egyes nyelvére teljesül. Ha sikerül bizonyítani, hogy az adott nyelvre ez a tulajdonság nem teljesül, akkor nem tartozhat az adott nyelvcsaládhoz.

Egy adott nyelvosztály tulajdonságait vizsgálva érdekes információt hordoznak a zártsági tulajdonságok.

Azt mondjuk, hogy egy nyelvosztály zárt egy nyelvműveletre nézve, vagyis a művelet nem vezet ki az adott nyelvosztályból, ha bármely a nyelvosztályhoz tartozó (ugyanazon ábécé felett értelmezett) nyelvekre a műveletet elvégezve az így létrejött (új) nyelv ugyancsak az adott nyelvcsaládhoz tartozik. Ha vannak olyan nyelvei az adott nyelvosztálynak, amikre a létrejövő nyelv már nem eleme az adott osztálynak, akkor a nyelvosztály nem zárt az adott műveletre nézve, úgy is mondhatjuk, hogy az adott művelet kivezet az adott nyelvosztályból. Fontos kérdés, hogy egy adott nyelvosztály zárt-e egyes nyelvműveletekre nézve.

Minden nyelvosztálynál fogjuk vizsgálni a konkatenáció, Kleene-csillag, unió, metszet és komplementerképzés műveleteket.

3.4.6. Speciális alosztályok

Több fontos nyelvosztály esetén speciális alosztályokat is be fogunk mutatni, amelyek általában speciálisan megszorított nyelvtanokkal generálhatóak, vagy speciálisan megszorított automatákkal fogadtathatóak el.

4. fejezet - A véges automaták elméletének alapjai

Ebben a fejezetben a véges automaták elméletébe nyújtunk betekintést.

4.1. Az automata fogalma és főbb típusai

Automatán egy olyan absztrakt rendszert fogunk érteni, mely egy diszkrétnek képzelt időskála időpillanataiban érkezett ingerek hatására ezen időpillanatokban válasszal reagál, miközben belső állapotát megadott szabályok szerint változtatja a külső ingerek hatására. Az ingerekre adott válasz függ mind az ingerektől mind pedig a pillanatnyi belső állapottól. Ebben az értelemben tehát nemcsak a gépek, hanem bármiféle élő vagy élettelen objektumok tekinthetők automatának, ha ezen séma szerint vizsgáljuk őket, azaz ilyenfajta működést tulajdonítunk nekik.

4.1. példa - Automaták

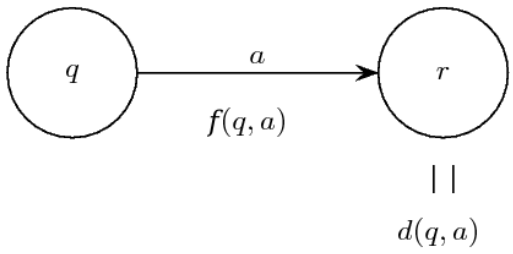
A legkülönbözőbb létező vagy nem létező dolgok tekinthetők automatának. Automatának tekinthető a lakásunk ajtaja, ablaka. Bizonyos értelemben automatának tekinthető a kedvenc macskánk, a számítógépünk, vagy a lakáscsengőnk is. Vizsgálhatjuk a főnökünket is mint automatát, hisz minden bizonnyal különféleképp reagál arra, hogy az elvárásainak megfelelően cselekszünk-e vagy sem. (És az is valószínű, hogy ezek a dolgok a főnök állapotát is befolyásolják.) De automatának tekinthető az inkák esőistene, aki imádságra esővel, káromkodásra szárazsággal reagál. ★

Azon automatákat amelyek egy inputszóhoz egy output szót rendelnek a működésük folyamán, átalakítóknak, transzduszereknek is szokás nevezni. A következőkben főleg ilyenekkel fogunk foglalkozni, szemben a későbbi részekben előtérbe kerülő elfogadó automatákkal.

4.1.1. Mealy automata

Az absztrakt automaták egyik nevezetes típusa a Mealy (ejtsd: míli) automata. *Mealy automatán* fogunk érteni egy $A=(Q, T, V, d, f)$ ötöst, aholis Q a (belső) állapotok nem üres halmaza, T a bemenő jelek nem üres halmaza, V a kimenőjelek nem üres halmaza, $d:Q\times T \rightarrow Q$ az átmeneti függvény és $f:Q\times T \rightarrow V$ a kimeneti függvény.

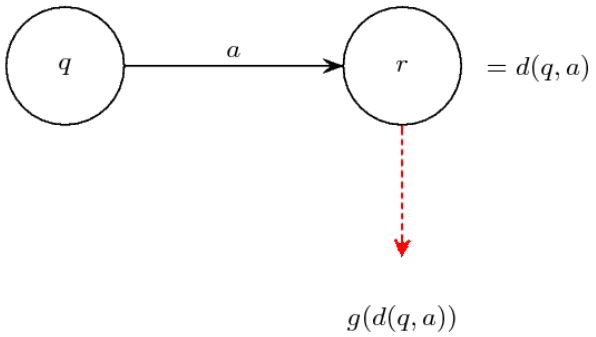
Úgy képzeljük, hogy a Mealy automata diszkrét időskála mentén működik, s annak minden egyes időpillanatában egy-egy jól meghatározott állapotban van. Ha valamely időpillanatban egy $A=(Q, T, V, d, f)$ Mealy automata egy $q\in Q$ állapotában az $a\in T$ bemenő jelet kapja, akkor ugyanezen időpillanatban a $f(q, a)$ kimenőjellel reagál, majd a következő időpillanatra átmegy a $d(q, a)$ állapotba.



A Mealy-automata működési sémája

4.1.2. Moore automata

Amennyiben az $A=(Q, T, V, d, f)$ Mealy-automatához létezik olyan $g:Q \to V$ függvény, hogy tetszőleges $q \in Q$ állapota és $a \in T$ bemenő jele esetén teljesül a $f(q, a)=g(d(q, a))$ egyenlőség, akkor *Moore-automatáról* beszélünk. A Moore-automatát $A=(Q, T, V, d, g)$ alakban szokás megadni, ahol $g:Q \to V$ a Moore-automata *jelfüggvénye*. Tetszőleges $q \in Q$ állapot esetén azt mondjuk, hogy $g(q)$ a $q \in Q$ *állapotjele*.



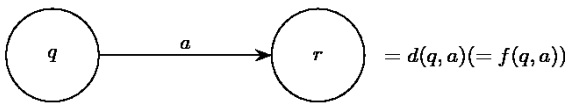
A Moore-automata működési sémája

A Moore automata a diszkrét időskála minden egyes időpillanatában egy-egy jól meghatározott $q \in Q$ állapotban van (amikor az állapotjele $g(q)$). Ha egy adott időpillanatban ezen $q \in Q$ állapotában az $a \in T$ bemenő jelet kapja, akkor a következő időpillanatban $d(q, a)$ lesz az állapota, s állapotjele pedig $g(d(q, a))$ lesz. Ily módon a Moore-automata egy adott időpillanatban kapott bemenő jel hatására a következő időpillanatra átmegy ezen bemenő jel és a belső állapota által egyértelműen meghatározott állapotba, s ezzel egyidejűleg kiadja az új állapot által egyértelműen meghatározott állapotjelet.

Képletesen szólva, amíg a Mealy-automata az olyan embert is modellezheti, aki először cselekszik, azután gondolkodik, addig a Moore-automata az olyan embert modellezi, aki először gondolkodik azután cselekszik.

4.1.3. Kimenőjel nélküli automata

A Moore-féle automata a Mealy-féle automata speciális eseteként adódik. A Moore-féle automata további specializálásával jutunk el a kimenőjel nélküli automata fogalmához a következő módon. Amennyiben egy $A=(Q, T, V, d, g)$ Moore-automatára $Q=V$ és $g:Q \to Q$ egy identikus leképezés (azaz minden $q \in Q$- ra $g(q)=q$), akkor a *kimenőjel nélküli automata* fogalmához jutunk. Figyelembe véve azt a tényt, hogy a Moore-automatát egy olyan $A=(Q, T, V, d, f)$ Mealy-automatából származtatjuk, melyhez található olyan $g:Q \to V$ függvény, hogy egy tetszőleges $q \in Q$, $a \in T$ pár esetén $f(q, a)=g(d(q, a))$, továbbá figyelembe véve, hogy a kimenőjel nélküli automata olyan Moore-automata, melynek a jelfüggvénye identikus leképezés, azt is mondhatjuk, hogy a kimenőjel nélküli automata egy olyan $A=(Q, T, V, d, f)$ Mealy-automata, melyre $Q=V$ és $d=f$. Ezen okok miatt a kimenőjel nélküli automatát $A=(Q, T, d)$ alakban szokás megadni.



A kimenő jel nélküli automata működési sémája

Egy $A=(Q, T, d)$ kimenőjel nélküli automata esetén a T bemenő jelhalmaz miden $a \in T$ eleme egy olyan $a:Q \rightarrow Q$ egy változós műveletnek (vagyis Q önmagába történő leképezésének) is tekinthető, mely az állapothalmaz tetszőleges $q \in Q$ eleméhez az $a(q)=d(q, a)$ elemét rendeli. Univerzális algebrai kifejezéssel élve tehát a kimenőjel nélküli automaták unoidok, vagy más szóval unáris algebrák. Ez az egyszerű felismerés lehetővé teszi számunkra a modern algebra módszereinek automataelméleti alkalmazását.

Amint láttuk, a Moore-féle automata speciális Mealy-automataként definiálható. Később látni fogjuk, hogy ez a specializáció látszólagos, ugyanis információ átalakítás szempontjából a két fogalom ekvivalens. (Az információ átalakításán azt értjük, hogy az automata tetszőleges bemenő információ hatására valamilyen "kimenő" információval reagál.) Nevezetesen, absztrakt szempontból a Mealy és a Moore-féle automaták ekvivalensek egymással abban az értelemben, hogy már a speciálisabb Moore-automatákkal előállíthatók azok az információ átalakítások, amelyek Mealy automatákkal megvalósíthatók. Tehát a Moore-automata ebből a szempontból csak látszólag speciálisabb a Mealy-automatánál.

Később látni fogjuk azt is, hogy az elmélet kiépítésénél sok esetben elegendő kimenőjel nélküli automatákra szorítkozni.

Az említett három automata típus mindegyike esetén szokás *véges automatáról* beszélni, ha az állapothalmaz, a bemenő jelhalmaz, s a kimenő jelhalmaz végesek. Szokás véges állapotú automatáról vagy Q- véges automatáról beszélni, ha az állapothalmaz véges. Hasonló értelemben beszélünk véges bemenetű vagy T- véges, illetve véges kimenetű vagy V- véges automatáról, valamint (Q, T)-, (Q, V)-, illetve (T, V)- véges automatáról.

4.1.4. Iniciális automata

Amennyiben az említett automata-típusok valamelyikénél kijelölünk egy q_0 iniciális-, vagy más néven kezdőállapotot, s feltételezzük, hogy az automata működésének van egy kezdő időpontja, amikor az automata ebben az állapotban van, akkor iniciális automatáról beszélünk. Az iniciális Mealy-féle automatát $A=(Q, T, V, q_0, d, f)$ alakban, az iniciális Moore-féle automatát $A=(Q, T, V, q_0, d, g)$ alakban, illetve az iniciális kimenőjel nélküli automatát $A=(Q, T, q_0, d)$ alakban szokás megadni, ahol mindhárom esetben q_0 az iniciális állapotot jelöli.

Az említett automatáknak szokás beszélni az alábbi általánosításairól is.

4.1.5. Parciális és teljesen definiált automata

Ha az átmeneti, illetve kimeneti függvény lehet parciális is, azaz nem teljesen definiált, akkor *parciális automatáról* van szó. Teljesen definiált függvényértékek esetén időnként szokás *teljesen definiált automatáról* is beszélni.

4.2. példa - Parciális automata

Az ajtó szigorú értelemben egy parciális kimenőjel nélküli automatának tekinthető. Bemenő jelei a csukás és a nyitás, s ennek megfelelően két állapota van, nevezetesen csukott és nyitott állapot. Csukott állapotból nyitással lehet nyitott állapotba hozni, nyitott állapotból pedig csukással lehet csukott állapotba hozni. De az ajtó valóban parciális automata, hisz nyitott ajtót kinyitni, vagy csukott ajtót becsukni nem lehetséges. ★

4.1.6. Nemdeterminisztikus és determinisztikus automata

Amennyiben az átmeneti és a kimeneti függvények (illetve Moore-automata esetén az átmeneti és a jelfüggvények) nem egyértelműen definiáltak, *nemdeterminisztikus automatáról* van szó. Ekkor valójában ezek nem is tekinthetőek függvénynek a hagyományos értelemben. Ahhoz, hogy

matematikai értelemben mégis függvényekkel dolgozzunk úgy tekintjük őket, mintha nem az állapot-, illetve a kimenő jelhalmazba képeznének, hanem ezen halmazok összes részhalmazainak halmazaiba. Egy nemdeterminisztikus $A=(Q, T, V, d, f)$ Mealy-automata esetén tehát az átmeneti függvény $d:Q\times T \rightarrow 2^Q$, a kimeneti függvény pedig $f:Q\times T \rightarrow 2^V$ alakú, ahol 2^Q, illetve 2^V az állapothalmaz, illetve a kimenő jelhalmaz részhalmazainak halmazát jelöli. Értelemszerűen, egy nemdeterminisztikus $A=(Q, T, V, d, g)$ Moore-automata esetén az átmeneti függvény $d:Q\times T \rightarrow 2^Q$, a jelfüggvény $g:Q \rightarrow 2^V$ alakú, míg egy nemdeterminisztikus kimenőjel nélküli $A=(Q, T, d)$ automatánál az átmeneti függvény formája $d:Q\times T \rightarrow 2^Q$.

4.3. példa - Nemdeterminisztikus automata

Nemdeterminisztikus kimenőjel nélküli automatának tekinthető a dobókocka, melynek egyetlen bemenő jele a feldobás. Ezen bemenő jel, azaz a feldobás hatására a dobókocka a hat lehetséges állapotából átmehet a hat lehetséges állapot bármelyikébe annak megfelelően, hogy a feldobás után éppen melyik lapjára esik. ★

További változata a nemdeterminisztikus automatáknak, ha megengedjük, hogy az automata bemenő jel nélkül is állapotot váltson: $d:Q\times(T\cup\{\lambda\}) \rightarrow 2^Q$. Ezeket szokás *üresszóátmenetes* (*nemdeterminisztikus*) *automatáknak* is nevezni.

A nemdeterminisztikus automata ellentéteként beszélünk determinisztikus automatáról is. Determinisztikus automata esetén tehát a szóban forgó függvényertékek mindig pontosan egy (parciális automata esetén maximum egy) meghatározott értéket vesznek fel. (A nemdeterminisztikus terminológiával pedig a függvények értékkészlete csak egyelemű, illetve maximum egyelemű részhalmazokat tartalmaz.) Determinisztikus automatáknál nem fordulhat elő üresszóátmenet.

Ha például egy nemdeterminisztikus $A=(Q, T, V, d, f)$ Mealy-automata esetén a $d(q, a)$ függvényérték a Q- nak egy hat elemű részhalmaza, $f(q, a)$ pedig a V egy két elemű részhalmaza, akkor az A Mealy-automata a q állapotából az a bemenő jel hatására ezen hat elemű részhalmaz bármelyik elemébe átmehet, s kimenőjelként pedig az említett két elemű halmaz báremlyik elemét kiadhatja. S tekintettel arra, hogy egy halmaznak az üres halmaz is részhalmaza, az is előfordulhat, hogy valamely $q\in Q$ állapotra és $a\in T$ bemenő jelre $d(q, a)$, vagy éppen $f(q, a)$ értéke az üres halmaz. Ha $d(q, a)$ az üres halmaz, ez annak felel meg, hogy erre az állapota és bemenő jelre nincs értelmezve egyetlen állapot sem amibe átmenet történhet, ha pedig $f(q, a)$ az üres halmaz, akkor ez azt jelenti, hogy erre az állapota és bemenő jelre nincs értelmezve egyetlen kimenőjel sem.

Ezek szerint a parciális automata olyan nemdeterminisztikus automatának tekinthető, ahol a megfelelő függvényértékek vagy egy elemű halmazokat, vagy pedig az üres halmazt szolgáltatják, a teljesen definiált determinisztikus automata pedig egy olyan nemdeterminisztikus automata, ahol ezek a függvényértékek mindig egy elemű halmazok.

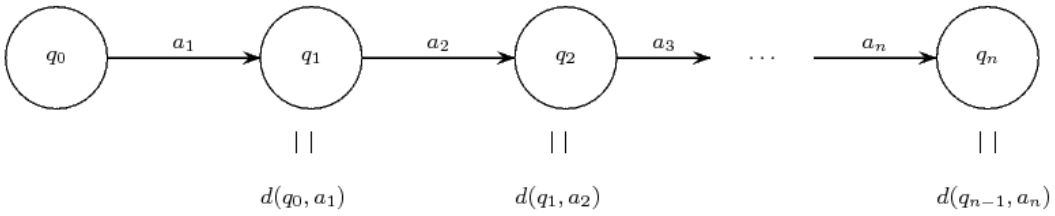
4.1.7. Sztochasztikus automata

Fontos általánosítás a *valószínűségi vagy sztochasztikus automata,* mikoris egy $P((r, b)|(q, a))$ feltételes valószínűség adja meg, hogy mi a valószínűsége annak, hogy az automata a r állapotba megy át és a b kimenőjelet adja ki azon feltétel mellett, hogy miközben a q állapotban volt, az a bemenő jelet kapta. Tehát egy valószínűségi automata $A=(Q, T, V, P)$ alakban adható meg, ahol Q az állapotok nem üres halmata, T a bemenő jelek nem üres halmaza, V a kimenőjelek nem üres halmaza, P pedig az említett feltételes valószínűség.

4.1.8. Rabin-Scott automata

Az automaták egy fontos osztályát képezik a *Rabin-Scott féle automaták* (ejtsd rabinszkott), melyeket *felismerő* vagy *elfogadó automatáknak* is hívnak. A Rabin-Scott féle automata egy $A=(Q, T, q_0, d, F)$ ötös, ahol Q a nem üres állapothalmaz, $q_0\in Q$ a kezdőállapot, T a nem üres bemenő jelhalmaz, $d:Q\times T \rightarrow Q$ az átmeneti függvény, $F\subseteq Q$ pedig a végállapotok nem üres halmaza. ($q_0\in F$ megengedett, azaz előfordulhat, hogy a kezdőállapot egyúttal végállapot is.)

A bemenő jelekből felépülő véges hosszúságú láncokat *bemenő szavaknak* hívjuk. Bemenő szónak tekintjük a λ üresszót is, mely nem tartalmaz egyetlen betűt sem. Egy Rabin-Scott automata a λ üresszót definíció szerint akkor *ismeri fel,* ha $q_0 \in F$. Egy nem üres, a_1, …, a_n (nem feltétlenül különböző) bemenő jelekből álló $a_1 \cdots a_n$ bemenő szó esetén akkor mondjuk, hogy a tekintett $A=(Q, T, q_0, d, F)$ Rabin-Scott féle automata *felismeri,* ha alkalmas q_1, …, q_n állapotaira $q_1=d(q_0, a_1)$, …, $q_n=d(q_{n-1}, a_n)$ teljesülése mellett $q_n \in F$.



A Rabin-Scott automata működési vázlata

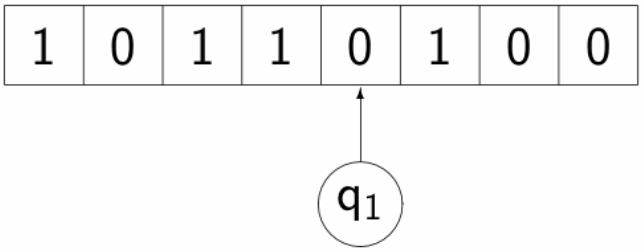
Az *A Rabin-Scott automata által felismert* L(A) *nyelvnek* hívjuk mindazon bemenő szavak halmazát, melyeket az automata felismer.

Értelmezhetjük a nemdeterminisztikus felismerő automatát is. Ekkor az automata által elfogadott nyelv alatt azon *w* szavak halmazát értjük, amelyekre az automatának van olyan lehetséges állapotlánca, amely a kezdőállapotból indul és végállapottal végződik, valamint az átmenetek bemenő jeleit összeolvasva éppen a *w* szót kapjuk.

Itt jegyezzük meg, hogy az átmenetfüggvényt szokás a *d* helyett a görög δ betűvel is jelölni.

4.4. példa - Rabin-Scott automata működés közben

Páros bináris számok elfogadása



A vges automata a páros bináris számokat fogadja el.

$$A = (\{q_0, q_1, q_v, q_w\}, \{0, 1\}, q_0, \delta, \{q_v, q_w\}),$$

$\delta(q_0, 0) = q_w,$
$\delta(q_0, 1) = q_1,$
$\delta(q_1, 1) = q_1,$
$\delta(q_1, 0) = q_v,$
$\delta(q_v, 1) = q_1,$
$\delta(q_v, 0) = q_v.$

★

A véges elfogadó automatákra a következő részben, a reguláris nyelvek kapcsán még visszatérünk.

4.2. Az automaták megadása

Egy automatát akkor tekintünk adottnak, ha a hozzá tartozó halmazok és függvények adottak. Egy automatát tehát úgy lehet megadni, hogy megadjuk a hozzá tartozó halmazokat és függvényeket. Ezen halmazok és függvények minden olyan típusú megadása lehetséges, ami a halmazok és függvények megadásánál szokásos.

4.2.1. Véges automaták megadása Cayley táblázattal

Véges halmazok és függvények megadásánál szokásos a művelettáblával történő megadás. Automaták művelettáblás megadását automaták Cayley táblázatának (ejtsd: kéjli) is hívjuk Cayley francia matematikus emlékére és tiszteletére, aki véges csoportok művelettábláinak leírására vezette be ezt a táblázatos módszert.

4.2.1.1. Véges Mealy-automata Cayley táblája

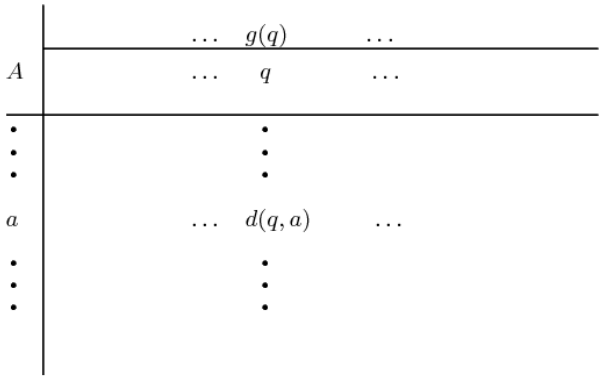
A táblázat bal felső sarkába írjuk az automata nevét, első sorában felsoroljuk az állapotait, első oszlopában pedig a bemenő jeleit. A táblázat (*i*+1)- edik sorában és (*j*+1)- edik oszlopában szerepel egy két dimenziós vektor, melynek első tagja azt mondja meg hogy az automata a *j*- edik állapotból az *i*- edik bemenő jel hatására melyik állapotába megy át, a másik tagja pedig azt mutatja, hogy a *j*- edik állapot az *i*- edik bemenő jel hatására milyen kimenőjelet ad ki.

A	$\dots$	q	$\dots$
$\vdots$		$\vdots$	
a	$\dots$	$(d(q,a), f(q,a))$	$\dots$
$\vdots$		$\vdots$	

Véges Mealy-automata művelet táblázata

4.2.1.2. Véges Moore-automata Cayley táblája

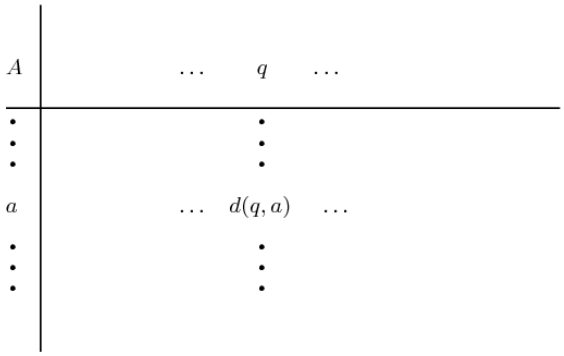
A táblázat bal felső sarkába írjuk az automata nevét, első sorában felsoroljuk az állapotait, első oszlopában pedig a bemenő jeleit. Minden állapot fölé beírjuk az állapotjelét. Így az első sor két részsorra oszlik. A táblázat (*i*+1)- edik sorában és (*j*+1)- edik oszlopában szerepel az az állapot, amibe az automata a *j*- edik állapotból az *i*- edik bemenő jel hatására átmegy.



Véges Moore-automata művelet táblázata

4.2.1.3. Véges kimenőjel nélküli automata Cayley táblája

A táblázat bal felső sarkába írjuk az automata nevét, első sorában felsoroljuk az állapotait, első oszlopában pedig a bemenő jeleit. (Ezesetben az állapotjel maga az állapot, így nem kell az első sorban levő állapotok fölé írni az állapotjelet mint az előző esetben.) Itt is a táblázat (i+1)- edik sorában és (j+1)- edik oszlopában szerepel az az állapot, amibe az automata a j- edik állapotból az i- edik bemenő jel hatására átmegy.



Véges kimenő jel nélküli automata művelet táblázata

4.5. példa - A csengő, mint automata

Vegyünk egy villamos csengőt. a_1 és a_2 jelöljék azon helyzeteket, mikoris nyomjuk vagy nem nyomjuk a csengőt. A csengő kezdetben az q_0 kezdőállapotban van, ami annak felel meg, hogy nem cseng. Az a_1 jel hatására, vagyis a csengő megnyomására átmegy a csengő a q_0 állapotból a q_1 állapotba. Megszakítva a csengő nyomását, vagyis az a_2 jel hatására a csengő átmegy nem csengő, azaz az q_0 állapotba. Leírásunkat táblázatba foglalva jutunk el a csengő következő absztrakt modelljéhez:

A	***q₀***	***q₁***
a₁	q_1	q_1
a₂	q_0	q_0

A táblázat mutatja, hogy mely jel hatására mely állapotból mely állapotba megy át az automata. Például a 3. sor 3. oszlopában levő q_0 azt jelenti, hogy a_2 hatására a q_1 állapotból az q_0 állapotba megy át az automata. Táblázatos megadásnál iniciális automata esetén rendszerint az első oszlop jelzi a kezdőállapot oszlopát (azaz annak megadását, hogy különféle bemenő jelek hatására a kezdőállapotból mely állapotokba megy át az automata). ★

Megjegyezzük, hogy némely feladatoknál célszerű a táblázatos megadásban a sorok és oszlopok szerepeinek felcserélése, vagyis ekkor az oszlopok a bemenőjeleket (illetve a λ- t, ha az

automata bemenőjel nélkül is állapotot válthat), míg a sorok az automata állapotait reprezentálják. Mindenképpen célszerű jelezni a táblázat bal felső sarkában, hogy a bemenőjelek, illetve az állapotok hol találhatóak. Elfogadó automaták esetén a végállapotokat is meg kell jelölni, pl. bekeretezéssel. A biztonság kedvéért a kezdőállapotot külön is jelezhetjük, pl. egy nyilacskával.

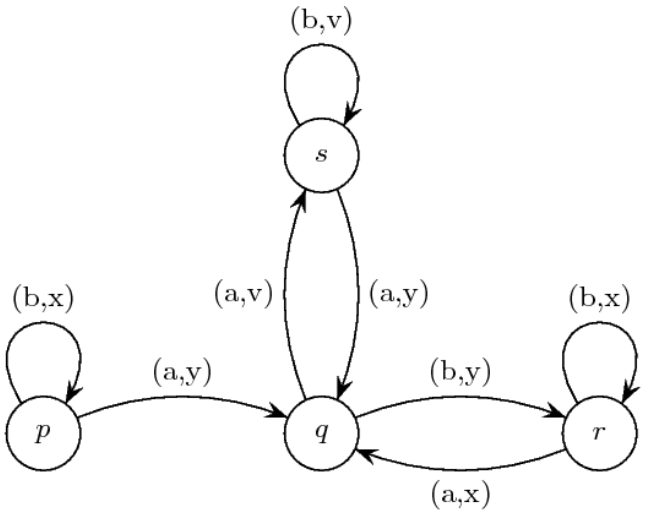
Parciális automata esetén a táblázat néhány helye üresen maradhat, amit ∅ jelölhet. Itt jegyezzük meg, hogy nem-determinisztikus automaták esetén a táblázat cellái az állapotok-, illetve a kimenőjelek halmazának részhalmazait tatralmazzák.

4.2.2. Véges automaták megadása gráfokkal

Véges automaták megadásának egy másik szokásos módja a címkézett irányított gráffal történő megadás.

4.2.2.1. Véges Mealy-automata megadása gráffal

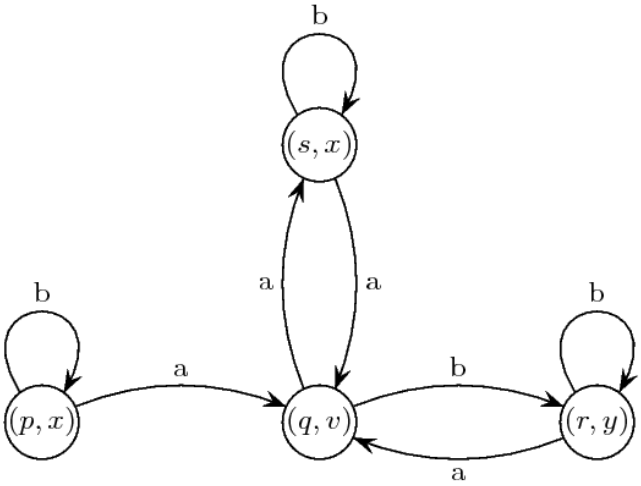
A gráf minden egyes csúcsa meg van címkézve egy állapottal, a csúcsokból kivezető minden irányított él (mely hurokél is lehet) pedig meg van címkézve egy két dimenziós vektorral. Ezen vektor első komponense egy bemenő jel, a második pedig egy kimenőjel. Determinisztikus esetben minden csúcsból pont annyi él vezet ki, ahány bemenő jel van és az élek, valamint azok címkéi adják meg, hogy egy adott állapotból egy adott bemenő jel hatására az automata milyen kimenőjellel reagál és melyik állapotba megy át. Nevezetesen, a bemenőjeleket az élek címkéinek első komponense, a kimenőjeleket az élek címkéinek második komponense, az állapot átmeneteket pedig az élek kezdő- és végcsúcsainak címkéi adják meg. Egy él kezdőcsúcsában lévő állapot az él címke első (bemenő jel) komponense hatására épp abba az állapotba megy át, mellyel az él végcsúcsa van megcímkézve.



EGY MEALY-AUTOMATA GRÁFJA

4.2.2.2. Véges Moore-automata megadása gráffal

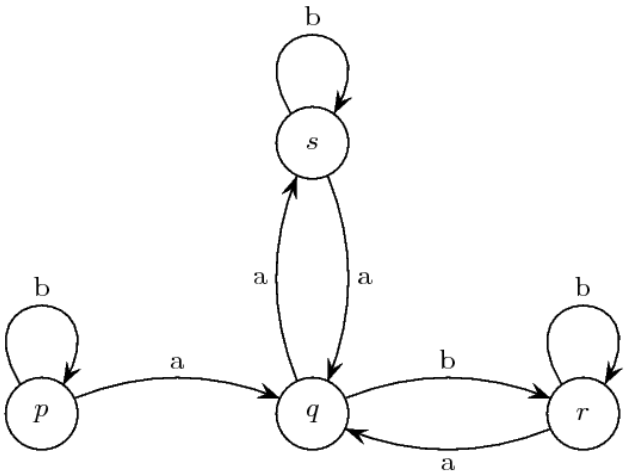
A gráf minden egyes csúcsa meg van címkézve egy két dimenziós vektorral, melynek első komponense egy állapot, a második komponense pedig ezen állapot állapotjele. A csúcsokból kivezető minden irányított él (mely hurokél is lehet) meg van címkézve egy bemenő jellel. (Determinisztikus esetben itt is) minden csúcsból pont annyi él vezet ki, ahány bemenő jel van és az élek, valamint azok címkéi adják meg, hogy egy adott állapotból egy adott bemenő jel hatására az automata melyik állapotba megy át. Nevezetesen, a bemenőjeleket az élek címkéi, az állapot átmeneteket és az állapotjeleket pedig az élek kezdő- és végcsúcsainak címkéi adják meg. Egy él kezdő csúcsában lévő címke első (állapot) komponense az él címke (bemenő jel) hatására épp abba az állapotba megy át, ami az él végcsúcsában levő címke első (állapot) komponense. Az átmenet utáni állapotjel pedig az él végcsúcsában levő címke második (kimenőjel) komponense.



EGY MOORE-AUTOMATA GRÁFJA

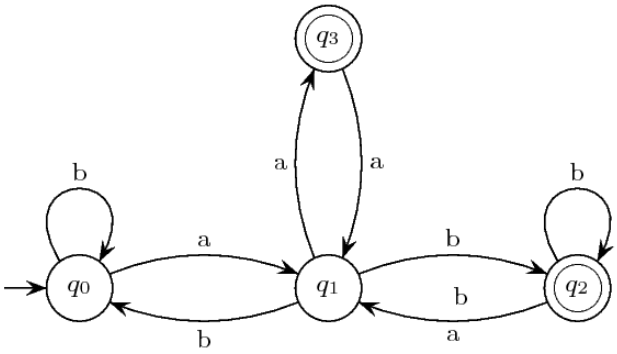
4.2.2.3. Véges kimenőjel nélküli automata megadása gráffal

Csaknem olyan szerkezetű gráffal történik a megadás mint a Moore-automata esetén. Az egyedüli lényeges különbség, hogy a csúcsok itt az állapotokkal vannak megcímkézve (és nem két dimenziós vektorokkal mint a Moore-automata esetén). Tehát a gráf minden egyes csúcsa meg van címkézve egy állapottal, a csúcsokból kivezető minden irányított él (mely hurokél is lehet) pedig meg van címkézve egy bemenő jellel. Most is igaz a (teljesen definiált) determinisztikus esetre, hogy minden csúcsból pont annyi él vezet ki, ahány bemenő jel van és az élek, valamint azok címkéi adják meg, hogy egy adott állapotból egy adott bemenő jel hatására az automata melyik állapotba megy át. Nevezetesen, a bemenőjeleket az élek címkéi, az állapot átmeneteket pedig az élek kezdő- és végcsúcsainak címkéi adják meg. Egy él kezdőcsúcsában lévő állapot címke az él címke (bemenő jel) hatására épp abba az állapotba megy át, ami az él végcsúcsában levő állapot címke értéke.



EGY KIMENŐ JEL NÉLKÜLI AUTOMATA GRÁFJA

Itt jegyezzük meg, hogy elfogadó automata esetén szokás a végállapotokat pl. dupla körrel rajzolni, míg a kezdőállapot jelölésére szokás az adott állapotba egy plusz bemenő nyilat rajzolni.



EGY KEZDŐ ÉS VÉGÁLLAPOTOKKAL RENDELKEZŐ NEM-DETERMINISZTIKUS AUTOMATA GRÁFJA

4.3. Az automata, mint algebrai struktúra: részautomata, homomorfizmus, izomorfizmus, kompatibilis osztályozás

Az absztrakt automatákat vizsgálhatjuk algebrai struktúráként is. Mint ahogy beszélni szoktunk algebrai struktúrák rész-struktúráiról, homomorfizmusairól, izomorfizmusairól, ugyanígy szokás beszélni ezekről automaták esetén is. Először Mealy-féle automatákra fogjuk megadni a megfelelő részautomata-, homomorfizmus- és izomorfizmus-fogalmakat.

4.3.1. Mealy automata, mint algebrai struktúra

Egy $A=(Q, T, V, d, f)$ Mealy-automata *részautomatája* alatt értjük azt az $A'=(Q', T', V', d', f')$ Mealy-automatát, melyre $Q'\subseteq Q$, $T'\subseteq T$, $V'\subseteq V$ és $d'=d|_{Q'\times T'}$, illetve $f'=f|_{Q'\times T'}$. (Szavakban, d' a d restrikciója, azaz megszorítása $Q'\times T'$- re, f' pedig az f restrikciója, azaz megszorítása $Q'\times T'$- re.) Másként mondva, d' és f' úgy van definiálva, hogy alakjuk $d':Q'\times T' \rightarrow Q'$, illetve $f':Q'\times T' \rightarrow V'$, s teljesül minden $q\in Q'$, $a\in T'$ pár esetén a $d'(q, a)=d(q, a)$, illetve az $f'(q, a)=f(q, a)$ egyenlőség. Amint látjuk, a d és az f nem minden $Q'\times T'$- re vett restrikciója alkalmas a részautomata átmeneti, illetve kimeneti függvényének. Kell még az is, hogy a tekintett d' restrikció a Q'- be, illetve hogy a tekintett f' restrikció a V'- be képezzen. Azt a tényt, hogy az A' automata részautomatája az A automatának, időnként $A'\subseteq A$- val jelöljük.

Amennyiben az $Q'\subseteq Q$, $T'\subseteq T$, $V'\subseteq V$ tartalmazások valamelyike valódi tartalmazás, azaz $Q'\subsetneq Q$, $T'\subsetneq T$, $V'\subsetneq V$ legalább egyike fennáll, úgy azt is mondjuk, hogy A' valódi részautomatája A- nak, s ezt $A'\subsetneq A$- val is jelöljük.

Ha $Q'\subseteq Q$ és $T=T'$, $V=V'$, akkor az A'- t az A (valódi vagy nem valódi) állapot-, vagy Q- részautomatájának hívjuk (jelekben: $A'\subseteq_Q A$). Ha $T'\subseteq T$ és $Q=Q'$, $V'=V$, akkor A' bemenő jel részautomatája vagy T- részautomatája (jelekben: $A'\subseteq_T A$), ha pedig $V'\subseteq V$, továbbá $Q'=Q$, $T'=T$, akkor az A' kimenőjel részautomatája, vagy V- részautomatája A- nak (jelekben: $A'\subseteq_V A$). Hasonló értelemben beszélünk (Q, T)-, (Q, V)-, (T, V)- részautomatákról.

További speciális automata-típus az iniciális részautomata, mely ugyancsak definiálható az említett részautomata-típusok bármelyikére. Akkor mondjuk, hogy egy A iniciális automatának egy A' iniciális automata iniciális részautomatája, ha mindamellett, hogy A' az A- nak részautomatája, az is teljesül, hogy az A' kezdőállapota épp az A kezdőállapota lesz. Hasonló értelemben mint a korábbiakban, beszélünk iniciális Q- részautomatáról, iniciális T- részautomatáról, iniciális V- részautomatáról, illetve iniciális (Q, T)-, (Q, V)-, (T, V)- részautomatáról. Minden esetben tehát a megfelelő részautomata-tulajdonság mellett még azt is elvárjuk, hogy a megfelelő részautomata-típus kezdőállapota épp az őt tartalmazó automata kezdőállapota legyen.

Azt mondjuk, hogy az $A'=(Q', T', V', d', f')$ Mealy-automata homomorf képe az $A=(Q, T, V, d, f)$ Mealy-automatának (jelekben $A \sim A'$), ha megadható olyan szürjektív (azaz "ra" típusú, vagy más néven ráképező) $\psi_1:Q \rightarrow Q'$, $\psi_2:T \rightarrow T'$, $\psi_3:V \rightarrow V'$ leképezésekből álló $\psi=(\psi_1, \psi_2, \psi_3)$ leképezés-hármas, hogy teljesülnek rá az úgynevezett művelettartó tulajdonságok. Más szóval, képletben kifejeve, tetszőleges $q \in Q$, $a \in T$ pár esetén

$$\psi_1(d(q, a))=d'(\psi_1(q), \psi_2(a)),$$

illetve

$$\psi_3(f(q, a))=f'(\psi_1(q), \psi_2(a)).$$

Világos, hogy a homomorfia mint reláció reflexív ($A \sim A$) és tranzitív ($A \sim A'$, $A' \sim A'' \Rightarrow A \sim A''$). Mint ahogy beszélünk speciális részautomatákról, ugyanúgy beszélünk speciális homomorfizmusokról. Azt mondjuk, hogy az $A'=(Q', T', V', d', f')$ Mealy-automata állapothomomorf képe vagy Q- homomorf képe az $A=(Q, T, V, d, f)$ Mealy-automatának (jelekben $A \sim_Q A'$), ha az előbb definiált ψ_1, ψ_2, ψ_3 leképezés-hármasban a ψ_2 és a ψ_3 választható identikus leképezésnek. Ezen feltétel mellett tehát az előbbi egyenlőségeink tetszőleges $q \in Q$, $a \in T$ pár esetén a következő alakúak lesznek: $\psi_1(d(q, a))=d'(\psi_1(q), a)$, illetve $f(q, a)=f'(\psi_1(q), a)$. Ebben az értelemben beszélünk tehát $\psi_1:Q \rightarrow Q'$ Q-homomorfizmusról. Ilyenkor nem egy leképezes-hármas, hanem egyetlen ψ_1 leképezés képviseli az állapothomomorfizmust (mert eltekintünk az identikus $\psi_2:T \rightarrow T'$ és $\psi_3:V \rightarrow V'$ leképezések szerepeltetésétől). Hasonló értelemben beszélünk $\psi_2:T \rightarrow T'$ bemenőjel-, vagy T- homomorfizmusról, $\psi_3:Y \rightarrow Y'$ kimenőjel-, vagy Y- homomorfizmusról, illetve $\psi'=\{\psi_1, \psi_2\}$ (Q, T)- homomorfizmusról (ahol $\psi_1:Q \rightarrow Q'$, $\psi_2:T \rightarrow T'$), $\psi''=\{\psi_1, \psi_3\}$ (Q, V)- homomorfizmusról (ahol $\psi_1:Q \rightarrow Q'$, $\psi_3:V \rightarrow V'$), valamint $\psi'''=\{\psi_2, \psi_3\}$ (T, V)- homomorfizmusról (ahol $\psi_2:T \rightarrow T'$, $\psi_3:V \rightarrow V'$).

További speciális homomorfizmus típus az iniciális homomorfizmus, mely ugyancsak definiálható az említett speciális homomorfizmus-típusok (iniciális Q- homomorfizmus, iniciális T- homomorfizmus, stb.) bármelyikére is. Akkor mondjuk, hogy egy A iniciális automatának egy A' iniciális automata iniciális homomorf képe (iniciális Q- homomorf képe, iniciális T- homomorf képe, stb.), ha mindamellett, hogy A' a A- nak homomorf képe, az is teljesül, hogy az A' kezdőállapota épp az A kezdőállapotának homomorf képe (Q- homomorf képe, T- homomorf képe, stb.) lesz. Hasonló értelemben beszélünk tehát iniciális Q- homomorfizmusról, iniciális T- homomorfizmusról, iniciális V- homomorfizmusról, illetve iniciális (Q, T)-, (Q, V)-, (T, V)- homomorfizmusról. (Minden esetben a megfelelő homomorfia-tulajdonság mellett még azt is elvárjuk, hogy a homomorf kép kezdőállapota épp a ráképező automata kezdőállapota legyen.)

Amennyiben a homomorfizmus olyan, hogy az összes szereplő ráképezések bijekciók (azaz kölcsönösen egyértelmű ráképezések), akkor izomorfizmusról beszélünk. Így minden egyes speciális homomorfizmus-típusnak van egy megfelelő izomorfizmus-típusa (Q- izomorfizmus, iniciális Q-izomorfizmus, stb.). Nyilvánvaló, hogy ha egy A Mealy-automatának egy A' Mealy-automata izomorf képe (képletben $A \simeq A'$), s a $\varphi=(\varphi_1, \varphi_2, \varphi_3)$ leképezés-hármas az A izomorfizmusa A'- re, akkor a $\varphi^{-1}=(\varphi_1^{-1}, \varphi_2^{-1}, \varphi_3^{-1})$ leképezés-hármas az A' izomorfizmusa lesz A- ra. Az izomorfizmus tehát mint reláció, szimmetrikus. Mindamellett mint a homomorfizmus általában (és az izomorfizmus a homomorfizmusnak speciális fajtája), az izomorfizmus mint reláció reflexív és tranzitív. Az izomorfizmus mint reláció tehát ekvivalencia reláció, hisz reflexív, szimmetrikus és tranzitív. (Ugyanez természetesen vonatkozik az összes speciális izomorfizmus-típusokra is.)

Az absztrakt algebrai vizsgálatoknál az egymással izomorf struktúrákat azonosnak szokták tekinteni. Ez az úgynevezett *izomorfia elv*. Ezt az elvet automataelméleti vizsgálatoknál sok esetben csak részlegesen szokásos elfogadni a vizsgálatok sajátosságai miatt.

Most részletesebben foglalkozunk még a Q- homomorfizmusokkal és a velük kapcsolatban álló kompatibilis osztályozásokkal. Legyen $A=(Q, T, V, d, f)$ tetszőleges Mealy-automata és legyen ϱ tetszőleges ekvivalencia reláció a Q állapothalmazon. Ismeretes, hogy minden ekvivalencia reláció egyértelműen indukál egy osztályozást azon a halmazon, amin a reláció értelmezve van. Nevezetesen, pontosan az egymással relációban lévő elemek fognak egy osztályba sorolódni. Így a ϱ reláció is indukálja a Q halmaz egy C_ϱ osztályozását: a p és q állapotokat akkor és csak akkor soroljuk egy osztályba, ha egymással relációban vannak, azaz $p \varrho q$ fennáll. A q elem által reprezentált (vagyis

a q elemet tartalmazó) osztályt $C_\varrho[q]$- val jelöljük. Jelölje $\overline{Q}$ az osztályok halmazát, azaz legyen $\overline{Q}=\{C_\varrho[q]|q\in Q\}$. Egy ilyen C_ϱ osztályozást *kompatibilis osztályozásnak* nevezünk, ha a hozzá tartozó ϱ reláció *kongruencia reláció,* vagyis ha ϱ- ra teljesül az a követelmény, hogy minden p, $q\in Q$ párra $p\varrho q$- nek tetszőleges $a\in T$ esetén következménye lesz $d(p, a)\varrho d(q, a)$ és $f(p, a)=f(q, a)$.

Tegyük fel most, hogy a Q halmaz C_ϱ osztályozása pont egy kompatibilis osztályozás. Ekkor tehát minden p, $q\in Q$ párra $p\varrho q$- nek tetszőleges $a\in T$ esetén következménye lesz $d(p, a)\varrho d(q, a)$ és $f(p, a)=f(q, a)$. Másként fogalmazva, feltételezzük, hogy ha valamely p, $q\in Q$ pár a C_ϱ osztályozás szerint egy és ugyanazon C_ϱ osztályba esik, akkor tetszőleges $a\in T$ bemenő jel esetén $d(p, a)$ és $d(q, a)$ is egy osztályba fognak esni, továbbá $f(p, a)=f(q, a)$ is fennáll. Ily módon definiálható a következő automata:

$$A/C_\varrho=(\overline{Q}, T, V, \overline{d}, \overline{f}),$$

ahol minden $q\in Q$, $a\in T$ párra

$$\overline{d}(C_\varrho([q], a)=C_\varrho[d(q, a)],\ \overline{f}(C_\varrho([q], a)=f(q, a).$$

Belátható, hogy ez az automata jól definiált és érvényes a következő:

$$A\sim_Q A/C_\varrho,$$

aholis a megfelelő Q- homomorfizmushoz úgy jutunk, hogy minden állapot állapothomomorf képeként az őt tartalmazó osztály adódik. Más szóval, egy automata faktorautomatái az automatának mindig Q- homomorf képei. A következő tétel azt mondja ki, hogy megfordítva, az A- homomorf képek mindig megszerkeszthetőek faktorautomataként. Ez a tétel az algebrában jól ismert Általános Homomorfia Tétel Mealy-automatákra vonatkozó speciális esete.

6. Tétel. (Általános Homomorfia Tétel Mealy-Automatákra Vonatkozó Speciális Esete) Tegyük fel, hogy az $A=(Q, T, V, d, f)$ Mealy-automata az $A'=(Q', T', V', d', f')$ Mealy-automatára valamely állapothomomorfizmussal leképezhető, s tekinsük a Q állapothalmaznak azt a C osztályozását, amelynél bármely két p, $q\in Q$ állapot akkor és csak akkor van egy osztályban, ha a tekintett állapothomomorfizmus szerinti képük ugyanaz. Ekkor a Q állapothalmaz ezen C osztályozása kompatibilis, s a hozzá tartozó A/C faktorautomata Q- izomorf lesz az A' automatával. Képletben,

$$A'\sim_Q A(\varphi)\Rightarrow A/C\simeq_Q A'(C[q]\ \rightarrow \varphi(q)).$$

4.3.2. Moore automata, mint algebrai struktúra

Most Moore-féle automatákra fogjuk megadni a megfelelő részautomata-, homomorfizmus-, és izomorfizmus-fogalmakat. Ekkor egy $A=(Q, T, V, d, g)$ Moore-automata valamely részautomatája alatt értünk egy olyan $A'=(Q', T', V', d', g')$ Moore-automatát, melyre $Q'\subseteq Q$, $T'\subseteq T$, $V'\subseteq V$ mellett $d'=d|_{Q'\times V'}$ és $g'=g|_{Q'}$ teljesül. Más szóval tetszőleges $q\in Q'$, $a\in T'$ pár esetén $d'(q, a)=d(q, a)$ és $g'(q)=g(q)$. Felmerül a kérdés, hogy ha egy Moore-automatát Mealy-féle automatának tekintünk, s vesszük ezen Mealy-automata egy (Mealy-automatáknál tárgyalt értelemben tekintett) részautomatáját, vajon ez a részautomata tekinthető lesz-e ugyancsak Moore-féle automatának, illetve részautomata lesz-e abban az értelemben is, ahogy azt a Moore-automaták esetén definiáltuk. A válaszunk az, hogy a két értelemben vett részautomata fogalom (Moore automatáknál) egybeesik. Legyen ugyanis $f:Q\times T\ \rightarrow V$ az a leképezés, melyre a tekintett Moore-automata, tetszőleges $q\in Q'$, $a\in T'$ pár esetén eleget tesz a $f(q, a)=g(d(q, a))$ összefüggésnek. Tegyük fel, hogy az ezen f függvénnyel definiált, A- ból nyert $B=(Q, T, V, d, f)$ Mealy-automatának a $B'=(Q', T', V', d', f')$ Mealy-automata részautomatája. Ekkor $d'=d|_{Q'\times T'}$ és $f'=f|_{Q'\times T'}$. Ily módon tetszőleges $q\in Q'$, $a\in T'$ párra $d'(q, a)=d(q, a)$ és $f'(q, a)=f(q, a)$. Viszont $f(q, a)=g(d(q, a))$, $f'(q, a)=f(q, a)$, valamint $d'(q, a)=d(q, a)$ miatt ekkor $f'(q, a)=g(d'(q, a))$. Vagyis definiálhatjuk az $A''=(Q', T', V, d', g)$ Moore-automatát, mely nyilvánvalóan (Q, T)- részautomatája lesz A- nak abban az értelemben, ahogy azt a Moore-automatáknál definiáltuk. Természetesen az is igaz, hogy V tetszőleges olyan $V''\subseteq V$ részhalmazára, melyre $\{g'(a)|q\in Q'\}\subseteq V''$, a $g''=g|_{Q''}$ feltételnek eleget tevő $A'''=(Q', T', V'', d', g'')$ Moore-automata a A automatának Moore-féle részautomatája lesz a Moore-automatáknál tekintett értelemben. Így valóban, ez a két részautomata-fogalom Moore-automatáknál egybeesik.

A Mealy-automatákra definiált speciális részautomata-fogalmak természetes módon definiálhatók Moore-féle automatákra, s az általános Moore-féle részautomata fogalomnál tárgyaltakhoz hasonló észrevételeket nyerünk, ha a Moore-automatákat speciális Mealy-automatáknak tekintve vesszük ezen Mealy-automaták megfelelő speciális (Q-, T-, V-, (Q, T)-, (Q, V)-, (T, V)-) részautomatáit. Ugyanez érvényben marad az iniciális Moore-automaták iniciális részautomatáira is és azok további speciális (iniciális Q- részautomata, iniciális T- részautomata, stb.) eseteire is.

Egy $A=(Q, T, V, d, g)$ Moore-automata valamely $A'=(Q', T', V', d', g')$ Moore-automatára történő (általános) Moore-homomorfizmusa alatt egy olyan $\psi=(\psi_1, \psi_2, \psi_3)$ szürjektív leképezésekből álló leképezés-hármast értünk, melynek tagjai $\psi_1:Q \to Q'$, $\psi_2:T \to T'$, $\psi_3:V \to V'$ formájúak, s teljesülnek rájuk minden $q\in Q$, $a\in T$ pár esetén a

$\psi_1(d(q, a))=d'(\psi_1(q), \psi_2(a))$,

illetve a

$\psi_3(g(q))=g'(\psi_1(q))$

összefüggés. Igazolható, hogy a Moore-féle homomorfizmus speciális esete a Mealy-féle automatákra értelmezett homomorfizmusnak abban az értelemben, hogy ha szereplő Moore-automatákat speciális Mealy-automatáknak tekintjük, akkor az előbb definiált Moore-féle homomorfizmus egy Mealy-féle automatákra definiált homomorfizmust fog szolgáltatni. De az is igaz, hogy lehetséges példát adni olyan Moore-automatára, melyet ha Mealy-automatának tekintünk, a Mealy-automatáknál vett értelemben homomorfan leképezhető lesz egy olyan Mealy-automatára, mely nem tekinthető Moore-automatának (lásd alábbi példában).

4.6. példa - Mealy és Moore automaták homomorfizmusa

Legyen adva egy $A=(\{p, q\}, \{a, b\}, \{x, y\}, d, g)$ Moore-automata, melyre $d(p, a)=d(q, a)=p$, $d(p, b)=d(q, b)=q$ és $g(p)=x$, $g(q)=y$. Ezt a Moore-automatát Mealy-automatának tekintve nyerjük a $B=(\{p, q\}, \{a, b\}, \{x, y\}, d, g)$ Mealy-automatát, melyre $f(p, a)=f(q, a)=x$, $f(p, b)=f(q, b)=y$. Nyilvánvalóan fennáll minden $q'\in\{p, q\}$, $a'\in\{a, b\}$ párra a $f(q', a')=g(d(q', a'))$ összefüggés. Most vegyük a $B'=(\{r\}, \{a, b\}, \{x, y\}, d', f')$ Mealy-automatát, melyre $d'(r, a)=d'(r, b)=r$ fennállása mellett $f'(r, a)=x$, $f'(r, b)=y$. Azonnal látszik, hogy a $\psi(p)=\psi(q)=r$ összefüggéssel definiált $\psi:\{p, q\} \to \{r\}$ leképezés a B egy Q- homomorfizmusa lesz B'- re. De $d'(r, a)=d'(r, b)=r$ és $f'(r, a)\neq f'(r, b)$ mellett az is látszik, hogy nincs olyan $g':\{r\} \to \{x, y\}$ leképezés, melyre $f'(r, a')=g'(d'(r, a'))$ fennállna minden $a'\in\{a, b\}$ mellett. B' tehát nem Moore-automata. ★

Ugyanúgy mint a Mealy-automatáknál, itt is tekinthetünk különféle speciális Moore-féle Q-, T-, V-, (Q, T)-, (Q, V)-, (T, V)- homomorfizmus típusokat, továbbá tekinthetjük mind az általános Moore-féle homomorfizmus-fogalom, mind pedig a speciális Moore-féle homomorfizmus-típusok iniciális változatait iniciális Moore-automatákra. Ugyanúgy, mint az általános esetben, ezekben az esetekben is lehet példát adni arra, hogy (Moore-automaták esetén) a Moore-féle speciális homomorfizmus-fogalmak (Moore-féle Q-, T-, V-, (Q, T)-, (Q, V)-, (T, V)- homomorfizmus és ezek iniciális változatai) is valódi speciális esetei a Mealy-féle változatoknak. Végül megjegyezzük, hogy ha a szereplő leképezések bijektívek (vagy ha egy ilyen leképezés van akkor ha a szereplő leképezés bijektív), akkor Moore-féle izomorfizmusról, illetve annak megfelelő speciális típusairól beszélünk. Izomorfizmusok esetén viszont a két féle (Mealy-féle, illetve Moore-féle) izomorfizmus-fogalmak egybeesnek.

Hasonlóan mint Mealy-automaták esetén, kimondható az algebrában jól ismert Általános Homomorfia Tétel Moore-automatákra vonatkozó speciális esete.

7. Tétel. (Általános Homomorfia Tétel Moore-Automatákra Vonatkozó Speciális Esete) Tegyük fel, hogy az $A=(Q, T, V, d, g)$ Moore-automata az $A'=(Q', T', V', d', g')$ Moore-automatára valamely Moore-féle állapothomomorfizmussal leképezhető, s tekinsük a Q állapothalmaznak azt a C osztályozását, amelynél bármely két $p, q\in Q$ állapot akkor és csak akkor van egy osztályban, ha a tekintett Moore-féle állapothomomorfizmus szerinti képük ugyanaz. Ekkor az Q állapothalmaz ezen C osztályozása kompatibilis, s a hozzá tartozó A/C faktorautomata Q- izomorf lesz az A' automatával. Képletben,

$A' \sim_Q A(\varphi) \Rightarrow A/C \simeq_Q A'(C[q] \rightarrow \varphi(q))$.

4.3.3. Kimenőjel nélküli automata, mint algebrai struktúra

Hátramaradt még a megfelelő részautomata-, homomorfizmus-, és izomorfizmus-fogalmak kimenőjel nélküli automatákra történő definiálása. Természetesen az is igaz, hogy némi megszorítással (a kimenő jelhalmazokra vonatkozó összefüggések figyelmen kívül hagyásával) a homomorfizmus, izomorfizmus és annak egyes speciális típusai definiálhatók kimenőjel nélküli automatákra is. Így például az $A=(Q, T, d)$ kimenőjel nélküli automatának részautomatája az $A'=(Q', T', d')$ kimenőjel nélküli automata, ha $Q' \subseteq Q$, $T' \subseteq T$, valamint $d'=d|_{Q' \times T'}$. Továbbá az $A=(Q, T, d)$ kimenőjel nélküli automatának homomorf képe az $A'=(Q', T', d')$ kimenőjel nélküli automata, ha alkalmas $\psi_1: Q \rightarrow Q'$, $\psi_2: T \rightarrow T'$ alakú, szürjektív leképezésekből álló $\psi=(\psi_1, \psi_2)$ leképezés-párra minden $q \in Q$, $a \in T$ esetén $\psi_1(d(q, a))=d'(\psi_1(q), \psi_2(a))$ fennáll. Az algebrában jól ismert Általános Homomorfia Tétel kimenőjel nélküli automatákra vonatkozó speciális esete formailag csaknem egybeesik a Mealy-féle automatákra vonatkozó speciális esettel.

8. Tétel. (Általános Homomorfia Tétel Kimenő Jel Nélküli Automatákra Vonatkozó Speciális Esete) Tegyük fel, hogy az $A=(Q, T, d)$ kimenőjel nélküli automata az $A'=(Q', T', d')$ kimenőjel nélküli automatára valamely állapothomomorfizmussal leképezhető, s tekinsük a Q állapothalmaznak azt a C osztályozását, amelynél bármely két $p, q \in Q$ állapot akkor és csak akkor van egy osztályban, ha a tekintett állapothomomorfizmus szerinti képük ugyanaz. Ekkor a Q állapothalmaz ezen C osztályozása kompatibilis, s a hozzá tartozó A/C faktorautomata Q- izomorf lesz az A' automatával. Képletben,

$A' \sim_Q A(\varphi) \Rightarrow A/C \simeq_Q A'(C[q] \rightarrow \varphi(q))$.

Az elmélet további kiépítésénél elsősorban Mealy-automatákra szorítkozunk. Így ha mást nem mondunk, a jegyzetnek ebben a részében automata alatt mindig Mealy-féle automatát fogunk érteni, azaz a Mealy-automaták esetén a "Mealy" jelzőt sok esetben elhagyjuk.

4.4. Az automaták által indukált leképezések

A korábban nem üres és véges halmazokra definiált néhány fogalmat a továbbiakban tetszőleges halmazokra is értelmezni fogjuk. Így valamely (véges vagy végtelen) V halmaz elemeiből alkotott véges láncot V- beli *szónak, V* elemeit pedig időnként *betűknek* hívjuk. Ha V az üres halmaz, technikai okokból (ahogy a Kleene itaráció definíciójából is kitűnik) V^* egy olyan egy elemű halmazt jelöl, melynek egyetlen eleme az üresszó: tehát $\varnothing^*=\{\lambda\}$. Nemcsak véges, hanem végtelen V halmazokra is (és az üres halmazra is) érvényes lesz, hogy a V^*- beli összes szavak a *konkatenációra* (egymás mellé írásra) nézve - mint műveletre - *monoidot, azaz egységelemes szabad félcsoportot* fognak alkotni. Ugyancsak igaz, nemcsak véges hanem végtelen V halmazokra is, hogy a V^+- beli összes szavak a konkatenációra (egymás mellé írásra) nézve - mint műveletre - *szabad félcsoportot* fognak alkotni. (Ha V az üres halmaz, akkor a V feletti összes nem üres szavak V^+ halmaza is nyilván üres halmaz. Márpedig egy félcsoportról fel szokás tételezni, hogy legalább egy eleme van, azaz az üres halmazt nem szokás félcsoportnak tekinteni. Így V^+- t nem tekintjük félcsoportnak ha V üres halmaz.) Egy $p \in V^*$ szó *hosszát* - akkor is ha V nem véges - $|p|$- el jelöljük, s mint korábban, a szót alkotó betűk számát értjük alatta (multiplicitásokkal együtt). A V halmaz számosságát is (akár véges, akár nem, akár üres akár nem) $|V|$- el jelöljük. Végül, ha p nem üres, mint korábban, $\gg p$ jelöli a p *utolsó betűjét.*

Legyen $A=(Q, T, V, d, f)$ tetszőleges Mealy-automata. A $d: Q \times T \rightarrow Q$ és a $f: Q \times T \rightarrow V$ függvények értelmezését kiterjesztjük $Q \times T^*$- ra a következő definícióval:

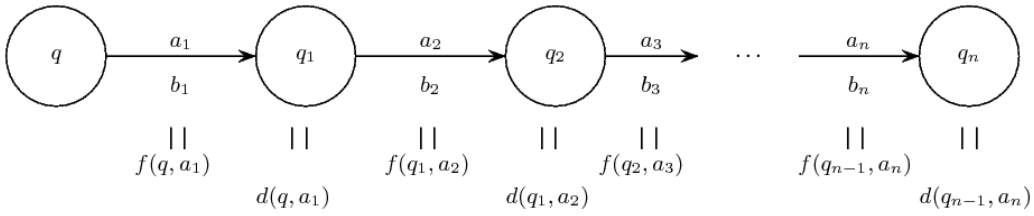
Legyenek $d: Q \times T^* \rightarrow Q^+$, $f: Q \times T^* \rightarrow V^*$ úgy definiálva, hogy tetszőleges $q \in Q$ és $a_1, \ldots, a_n \in T$ esetén álljanak fenn a

$d(q, a_1 \cdots a_n)=q_1 \cdots q_n$

és a

$f(q, a_1 \cdots a_n)=b_1 \cdots b_n$

összefüggések, ahol $q_1=d(q, a_1)$, …, $q_n=d(q_{n-1}, a_n)$, illetve $b_1=f(q, a_1)$, …, $b_n=f(q_{n-1}, a_n)$. Emellett legyen $d(q, \lambda)=q, f(q, \lambda)=\lambda$. Ez a formális definíció annak az interpretációnak felel meg, hogy bármely $q \in Q$ állapotból indulva az A automata egy bemenő jelsorozatnak egy kimenő jelsorozatot feleltet meg (az automata "szekvenciális gép"). Nevezetesen, az automata az üres bemenő szóra üres kimenő szóval reagál.



Mealy-automata mint általánosított szekvenciális gép működési vázlata

Valamely T halmaz feletti T^* szabad monoidot egy V halmaz feletti V^* szabad monoidba leképező $\alpha: T^* \to V^*$ leképezést *alfabetikus leképezésnek* hívunk. A bemenő szavak a bemenő információ, a kimenő szavak pedig a kimenő információ hordozói. Az automata az információ átalakítást alfabetikus leképezések segítségével realizálja. Így a fenti $a_1a_2 \cdots a_n$ szóhoz az automata a fenti $b_1b_2 \cdots b_n$ szót rendeli hozzá. Speciálisan, az üresszónak minden állapot az üresszót felelteti meg, hisz definícióink értelmében tetszőleges $q \in Q$ állapotra $f(q, \lambda)=\lambda$. A továbbiakban egy $A=(Q, T, V, d, f)$ Mealy-automata minden egyes $q \in Q$ állapotához hozzárendeljük a $\varphi_{q, A}(w)=f(q, w)$, $w \in T^*$ összefüggéssel definiált $\varphi_{q, A}: T^* \to V^*$ leképezést, melyet a $q \in Q$ *állapot által indukált leképezésnek* is fogunk hívni. Ha nem áll fenn a félreértés veszélye, $\varphi_{q, A}$ helyett a legtöbbször csak φ_q- t írunk. Iniciális $A=(Q, T, V, q_0, d, f)$ Mealy-automata esetén a q_0 kezdőállapottal indukált φ_{q_0} leképezést az A *iniciális automata által indukált leképezésnek*, vagy röviden az A *automata leképezésének* is mondjuk és időnként φ_A- val jelöljük. A következő tétel George N. Raneytől (ejtsd: réni) származik.

9. Tétel. (Raney tétele) Egy $\varphi: T^* \to V^*$ alfabetikus leképezés akkor és csak akkor automata leképezés, ha eleget tesz a következő két feltételnek:

(i) hossztartó, azaz tetszőleges $w \in T^*$- ra $|w|=|\varphi(w)|$,

(ii) kezdőszelet tartó (kezdőszeletet kezdőszeletbe visz át), azaz minden $w, v \in T^*$- hoz létezik olyan $u \in V^*$, hogy $\varphi(wv)=\varphi(w)u$.

Bizonyítás. A szükségesség nyilvánvaló a kiterjesztett átmeneti és kimeneti függvények tulajdonságai miatt. Valóban, legyen $A=(Q, T, V, d, f)$ tetszőleges Mealy-automata, s legyen $q \in Q$ tetszőleges állapota. Legyen $w \in T^*$ tetszőleges. Ha $w=\lambda$, azaz w az üresszó, akkor $f(q, \lambda)=\lambda$ a definíciónk szerint, azaz a $\varphi_q(\lambda)=f(q, \lambda)$ és a $f(q, \lambda)=\lambda$ összefüggések miatt ekkor $\varphi_q(\lambda)=\lambda$, amiből $|\varphi_q(\lambda)|=|\lambda|$ nyilvánvalóan következik. φ_q tehát az üresszóra teljesíti a hossztartó tulajdonságot. Most legyen $w \in T^*$ nem üresszó, azaz legyen valamely $a_1, \ldots, a_n \in T$ bemenő jelekre $w=a_1 \ldots a_n$. Ekkor, amint a d és f függvények kiterjesztésénél láttuk, alkalmas $q_1, \ldots, q_n \in Q$ állapotokra $q_1=d(q, a_1)$, $q_2=d(q_1, a_2)$, …, $q_n=d(q_{n-1}, a_n)$ teljesülése mellett $f(q, a_1 \cdots a_n)=f(q, a_1)f(q_1, a_2) \cdots f(q_{n-1}, a_n)$. Vagyis, bevezetve a $b_1=f(q, a_1)$, $b_2=f(q_1, a_2)$, …, $b_n=f(q_{n-1}, a_n)$ jelöléseket, $\varphi_q(a_1 \cdots a_n)=f(q, a_1 \cdots a_n)=b_1 \cdots b_n$. Ebből nyilvánvaló, hogy $|a_1 \cdots a_n|=|b_1 \cdots b_n|=n$, vagyis $|w|=|\varphi_q(w)|$, azaz φ_w minden T^*- beli (üres vagy nem üres) szóra hossztartó.

Az üresszó T^*- nak és V^*- nak egységeleme, azaz tetszőleges $v \in T^*$, $w \in V^*$ esetén $\lambda v=v\lambda=v$, illetve $\lambda w=w\lambda=w$. Legyen most $v \in T^*$ tetszőleges szó. Ekkor azt kapjuk, hogy $\varphi_q(\lambda v)=\varphi_q(v)=\lambda\varphi_q(v)$. Vagyis a kezdőszelet tartó tulajdonság teljesülni fog ha az üresszó a kezdőszelet, a végszelet pedig maga a

tekintett szó. (Írjunk u- t a $\varphi_q(v)$ helyébe a $\varphi_q(\lambda v)=\lambda\varphi_q(v)$ egyenlőség jobboldalán.) Hasonlóan kapjuk tetszőleges $w\in T^*$- ra a $\varphi_q(w\lambda)=\varphi_q(w)=\varphi_q(w)\lambda$ levezetést. Tehát a kezdőszelet tartó tulajdonság akkor is teljesülni fog, ha a szó kezdőszeletének magát a szót tekintjük, végszeletének pedig az üresszót. (Írjunk u- t a λ helyébe a $\varphi_q(w\lambda)=\varphi_q(w)\lambda$ egyenlőség jobboldalán.) Most válasszuk meg a w, $v\in T^*$ párt úgy, hogy egyikük se legyen üres. Ekkor valamely $a_1, \ldots, a_k, a_{k+1}, \ldots, a_n\in T$ bemenő jelekre $w=a_1\cdots a_k$, $v=a_{k+1}\cdots a_n$. Vezessük be rendre a $q_1=d(q, a_1)$, $q_2=d(q_1, a_2)$, …, $q_n=d(q_{n-1}, a_n)$ és a $b_1=f(q, a_1)$, $b_2=f(q_1, a_2)$, …, $b_n=f(q_{n-1}, a_n)$ jelöléseket. Ekkor $\varphi_q(wv)=f(q, wv)=f(q, a_1\cdots a_n)=f(q, a_1)f(q_1, a_2)\cdots f(q_{k-1}, a_k)f(q_{k+1}, a_{k+1})\cdots f(q_{n-1}, a_n)=b_1\cdots b_k b_{k+1}\cdots b_n$ és $\varphi_q(w)=f(q, w)=f(q, a_1\cdots a_k)=f(q, a_1)f(q_1, a_2)\cdots f(q_{k-1}, a_k)=b_1\cdots b_k$ fennállnak. Ebből viszont $\varphi_q(wv)=\varphi_q(w)b_{k+1}\cdots b_n$, vagyis az $u=b_{k+1}\cdots b_n$ választással kapjuk, hogy alkalmas $u\in V^*$- ra $\varphi_q(wv)=\varphi_q(w)u$ ebben az esetben is fennáll. φ_q tehát valóban kezdőszelet tartó.

Az elegendőséghez legyen $\varphi:T^* \to V^*$ tetszőleges hossz- és kezdőszelet-tartó alfabetikus leképezés. Ekkor minden w, $v\in T^*$ párhoz lesz olyan $u\in V^*$, hogy $\varphi(wv)=\varphi(w)u$.

Rögzített $w\in T^*$ esetén jelölje φ_w azt a $\varphi_w:T^* \to V^*$ alfabetikus leképezést, melyre tetszőleges $v\in T^*$ esetén a $\varphi(wv)=\varphi(w)\varphi_w(v)$ egyenlőség fog teljesülni. Először mutassuk meg, hogy a tetszőlegesen választott $w\in T^*$- hoz tekintett φ_w alfabetikus leképezés is automata leképezés. Valóban, φ hossztartó volta miatt $|wv|=|\varphi(wv)|$ és $|w|=|\varphi(w)|$. Másrészt $\varphi(wv)=\varphi(w)\varphi_w(v)$ miatt $|\varphi(w)\varphi_w(v)|=|\varphi(wv)|$, azaz $|w|+|v|=(|wv|=|\varphi(w)\varphi_w(v)|=)|\varphi(w)|+|\varphi_w(v)|$. Ebből $|w|=|\varphi(w)|$ értelmében $|v|=|\varphi_w(v)|$, azaz φ_w hossztartó. Mutassuk meg, hogy kezdőszelet tartó is. Legyen v, $z\in T^*$ tetszőleges. Ekkor $\varphi(wvz)=\varphi(w)\varphi_w(vz)$ és $\varphi(wvz)=\varphi(wv)\varphi_{wv}(z)=\varphi(w)\varphi_w(v)\varphi_{wv}(z)$ is teljesülni fognak a φ kezdőszelet tartó volta miatt. Így $\varphi(w)\varphi_w(vz)=\varphi(w)\varphi_w(v)\varphi_{wv}(z)$ is teljesül. Ám szabad monoidban a szavak egyenlősége betűről-betűre való egyenlőséget jelent, azaz a legutóbb kapott egyenlőségünkből $\varphi_w(vz)=\varphi_w(v)\varphi_{wv}(z)$ is következik. Más szóval, tetszőleges v, $z\in T^*$- hoz létezik olyan $u(=\varphi_{wv}(z))\in V^*$, hogy $\varphi_w(vz)=\varphi_w(v)u$ teljesül. Tehát φ_w is kezdőszelet tartó. Azt kaptuk tehát, hogy tetszőleges $w\in T^*$ esetén φ_w is automata leképezés. Tetszőleges $w\in T^*$ esetén a φ_w automata leképezést a továbbiakban a φ *leképezés állapotának* fogjuk hívni. Legyen $Q_\varphi=\{\varphi_w | w\in T^*\}$ és definiáljuk az $A_\varphi=(Q_\varphi, T, V, \varphi_\lambda, d_\varphi, f_\varphi)$ Mealy-automatát úgy, hogy minden $\varphi_w\in Q_\varphi$, $a\in T$ esetén $d_\varphi(\varphi_w, a)=\varphi_{wa}, f_\varphi(\varphi_w, a)=\varphi_w(a)$. A tételhez azt kell még megmutatni, hogy A_φ automata a φ leképezést indukálja.

Mivel az üresszó az egyetlen olyan szó, melynek hossza nulla, a φ hossztartó volta miatt $\varphi(\lambda)=\lambda$. Így tekintettel arra, hogy az üresszó mint bemenő szó hatására egy Mealy-automata kimenő szóként definíció szerint az üresszót adja ki, $f_\varphi(\varphi_\lambda, \lambda)=\varphi(\lambda)(=\lambda)$ teljesülni fog. Most legyen $w=a_1\cdots a_n$ egy nem üres bemenő szó, ahol $a_1, \ldots, a_n\in T$ tetszőleges bemenő jelek. Az A_φ automata definíciója értelmében, valamint amiatt, hogy tetszőleges $u\in T^*$ szóra $\lambda u=u$ fennáll, $d(\varphi_\lambda, w)=d(\varphi_\lambda, a_1\cdots a_n)=\varphi_{\lambda a_1}\varphi_{\lambda a_1 a_2}\cdots\varphi_{\lambda a_1\cdots a_n}=\varphi_{a_1}\varphi_{a_1 a_2}\cdots\varphi_{a_1\cdots a_n}$. Ekkor azonban, ugyancsak az A_φ automata definíciója alapján $\varphi_{A_\varphi}=f_\varphi(\varphi_\lambda, w)=f_\varphi(\varphi_\lambda, a_1\cdots a_n)=f_\varphi(\varphi_\lambda, a_1)f_\varphi(\varphi_{a_1}, a_2)\cdots f_\varphi(\varphi_{a_1\cdots a_{n-1}}, a_n)=\varphi_\lambda(a_1)\varphi_{a_1}(a_2)\cdots\varphi_{a_1\cdots a_{n-1}}(a_n)=$
$=\lambda\varphi_\lambda(a_1)\varphi_{a_1}(a_2)\cdots\varphi_{a_1\cdots a_{n-1}}(a_n)=\varphi(\lambda)\varphi_\lambda(a_1)\varphi_{a_1}(a_2)\cdots\varphi_{a_1\cdots a_{n-1}}(a_n)=\varphi(\lambda a_1)\varphi_{a_1}(a_2)\cdots\varphi_{a_1\cdots a_{n-1}}(a_n)=$
$=\varphi(\lambda a_1 a_2)\varphi_{a_1 a_2}(a_3)\cdots\varphi_{a_1\cdots a_{n-1}}(a_n)=\cdots=\varphi(\lambda a_1\cdots a_n)=\varphi(a_1\cdots a_n)=\varphi(w)$.

Viszont épp ezt kellett bizonyítani. ∎

Nyilvánvaló, hogy ha egy automata leképezés egy véges iniciális automatával indukálható, akkor ennek az automata leképezésnek legfeljebb annyi állapota lehet mint az őt indukáló véges automatának. Másrészt az is világos, hogy a Raney tételének bizonyításában szereplő A_φ automata véges, ha a bemenő és kimenő jelhalmazok végessége mellett a tekintett φ automata leképezés véges állapotú. Így a következő állításhoz jutunk.

1. Következmény. Egy véges halmaz feletti monoidot egy véges halmaz feletti monoidba képező automata leképezés akkor és csakis akkor indukálható véges automatában, ha állapotainak száma véges.

2. *Megjegyzés.* Egy $\varphi:T^* \to V^*$ automata leképezés indukálható a következő Mealy-automatával is: $A^\varphi=(T^*, T, V, \lambda, d^\varphi, f^\varphi)$, ahol tetszőleges $w\in T^*$, $a\in T$ párra $d^\varphi(w, a)=wa$, $f^\varphi(w, a)=\gg\varphi(wa)$ (ahol $\gg\varphi(wa)$ a $\varphi(wa)$ utolsó betűjét jelöli).

Az A_φ automatát a φ *automata leképezéshez tartozó alsó automatának,* az A^φ automatát pedig a φ *automata leképezéshez tartozó felső automatának* hívjuk. Világos, hogy a felső automata mindig végtelen állapotú.

Egy iniciális automatát *iniciálisan összefüggőnek* hívunk, ha a kezdőállapotából minden állapotba visz át bemenő szó. Képletben, az $A=(Q, T, V, q_0, d, f)$ iniciális Mealy automata iniciálisan összefüggő, ha minden $q\in Q$ állapothoz létezik olyan $w\in T^*$ bemenő szó, hogy $d(q_0, w)=q$, azaz a q_0 állapotból indulva a w szót feldolgozva az automata a q állapotba jut. (Megjegyezzük, hogy $w=\lambda$ választással ez a kezdőállapotra mindig fennáll.) Nyilvánvaló, hogy egy automata leképezéshez tartozó alsó és felső automaták iniciálisan összefüggőek. Bizonyítás nélkül megemlítjük a következő tételt.

10. Tétel. Ha A tetszőleges olyan iniciálisan összefüggő automata, amely a φ leképezést indukálja, akkor az A automata az A^φ felső automatának állapothomomorf képe, s ugyanekkor az A_φ alsó automata pedig A- nak állapothomomorf képe.

3. *Megjegyzés.* Egy φ automata leképezés előállításánál felhasznált automaták közül a hozzá tartozó alsó automata a lehető leggazdaságosabb, a hozzá tartozó felső automata pedig a lehető leggazdaságtalanabb. Így az optimális előállításnál a φ automata leképezéshez tartozó alsó automatát kell előállítani. Ez így azonban csupán elméleti eredmény, mivel adott $w, v\in T^*$ párra általában nehéz ellenőrizni, hogy fennáll-e a $\varphi_w=\varphi_v$ egyenlőség.

11. Tétel. Tetszőleges T halmazra T^* összes önmagába történő automata leképezéseinek K_T halmaza a leképezések szokásos szorzására nézve monoidot, azaz egységelemes félcsoportot alkot.

4. *Megjegyzés.* Egy $A=(Q, T, V, q_0, d, f)$ által indukált leképezésnek a $B=(Q', T', V', q'_0, d', f')$ automata által indukált leképezéssel való szorzása értelmezhető, ha $V\subseteq T'$. Ezt a szorzat-leképezést a B automata indukálja, ha $B''=(Q\times Q', T, V', (q_0, q'_0), d'', f'')$ alakú, ahol tetszőleges $(q, q')\in Q\times Q'$, $a\in T$ párra $d''((q, q'), a)=(d(q, a), d'(q', f(q, a))$ és $f''((q, q'), a)=f'(q', f(q, a))$. A B' automatát az A automata B automatával történő *soros kapcsolásának* vagy *szuperpozíciójának* hívjuk.

12. Tétel. Egy T halmaz feletti T^* monoid összes önmagába történő, véges automaták által indukálható leképezéseinek L_T halmaza a leképezések szokásos szorzására nézve részfélcsoportot alkot a K_T félcsoportban.

13. Tétel. Tetszőleges T halmazra az T^* összes önmagába történő bijektív automata leképezéseinek A_T halmaza a leképezések szokásos szorzására nézve részcsoportot alkot a K_T félcsoportban.

14. Tétel. Egy T halmaz feletti T^* monoid összes önmagába történő, véges automaták által indukálható bijektív leképezéseinek G_T halmaza a leképezések szokásos szorzására nézve részcsoportot alkot a K_T félcsoportban és az A_T csoportban.

Egy S *félcsoport generátorrendszerén* értjük S- beli elemek egy H részhalmazát, ha S minden eleme előáll H- beli elemek szorzataként. H *minimális generátorrendszere,* vagy más néven *bázisa* S- nek, ha amellett hogy S- nek generátorrendszere, tetszőleges $h\in H$ mellett $H\setminus\{h\}$ már nem generátorrendszere S- nek. Hasonlóan, egy G csoport *generátorrendszerén* értjük G- beli elemek egy H részhalmazát, ha G minden eleme előáll olyan szorzatként, melynek minden tényezője vagy egy H- beli elem, vagy pedig egy H- beli elem inverze. Ugyanúgy mint félcsoportok esetén, H *minimális generátorrendszere,* vagy más néven *bázisa* G- nek, ha amellett hogy G- nek generátorrendszere, tetszőleges $h\in H$ mellett $H\setminus\{h\}$ már nem generátorrendszere G- nek. (Az itt szereplő S és G betűk az angol *group* (=csoport), illetve *semi-group* szavak kezdőbetűi, a generatív nyelvtanoknál szereplő S, illetve G- vel való azonosságuk csak a véletlen műve!)

Nem nehéz belátni, hogy ha T egy egyelemű halmaz vagy T az üres halmaz, akkor K_T, L_T, A_T, G_T mindegyike egy elemű. Így ebben az esetben mindegyiknek önmaga a bázisa. Ha T legalább két elemű, akkor érvényes a következő állítás.

15. Tétel. Egy legalább két elemű T halmaz esetén K_T- nek és L_T- nek nincs bázisa.

Nyitott kérdés, hogy van-e bázisa A_T- nek, illetve G_T- nek egy legalább két elemű T halmaz esetén.

Végül megjegyezzük, hogy Moore-automata esetén az átmeneti és jelfüggvények kiterjesztése hasonlóképp történik mint Mealy-automata esetén. Legyen $A=(Q, T, V, d, g)$ tetszőleges Moore-automata. A $d:Q\times T \to Q$ és a $g:Q \to V$ függvények értelmezését kiterjesztjük $Q\times T^*$- ra, illetve Q^+-ra a következő definícióval:

Legyenek $d:Q\times T^* \to Q^+$, $g:Q^* \to V^*$ úgy definiálva, hogy tetszőleges $q\in Q$ és $a_1, \ldots, a_n\in T$ esetén álljanak fenn a

$d(q, a_1\cdots a_n)=q_1\cdots q_n$

és a

$g(q_1\cdots q_n)=b_1\cdots b_n$

összefüggések, ahol $q_1=d(q, a_1)$, …, $q_n=d(q_{n-1}, a_n)$, illetve $b_1=g(q_1)$, …, $b_n=g(q_n)$. Emellett legyen $d(q, \lambda)=q$, $g(\lambda)=\lambda$. Ekkor a q állapot által indukált φ_q leképezésre $\varphi_q(\lambda)=g(\lambda)=\lambda$, illetve tetszőleges $a_1, \ldots, a_n\in T$- re $\varphi_q(a_1\cdots a_n)=g(q_1\cdots q_n)$, aholis $q_1=d(q, a_1)$, $q_2=d(q_1, a_2)$, …, $q_n=d(q_{n-1}, a_n)$. Később látni fogjuk, hogy minden iniciális Mealy-automatával indukálható automata leképezés indukálható iniciális Moore-automatával is. Sőt az is igaz, hogy ha egy automata leképezés véges iniciális Mealy automatával indukálható, akkor indukálható véges iniciális Moore-automatával is. Így a fejezetben tárgyaltak Moore-automatákra ugyanígy érvényben maradnak. Az átmeneti függvény természetesen kiterjeszthető kimenőjel nélküli automata esetén is. Nevezetesen, ugyanúgy mint Moore-automata esetén, egy $A=(Q, T, d)$ tetszőleges kimenőjel nélküli automatára a $d:Q\times T \to Q$ függvény értelmezését úgy terjesztjük ki $Q\times T^*$- ra, hogy $d:Q\times T^* \to Q^+$- t a következőképp definiáljuk: Legyen $d:Q\times T^* \to Q^+$ úgy definiálva, hogy tetszőleges $q\in Q$ és $a_1, \ldots, a_n\in T$ esetén álljanak fenn a

$d(q, a_1\cdots a_n)=q_1\cdots q_n$

összefüggések, ahol $q_1=d(q, a_1)$, …, $q_n=d(q_{n-1}, a_n)$. Emellett legyen $d(q, \lambda)=q$. Ekkor a q állapot által indukált φ_q leképezésre $\varphi_q(\lambda)=\lambda$, illetve tetszőleges $a_1, \ldots, a_n\in T$- re $\varphi_q(a_1\cdots a_n)=q_1\cdots q_n$, aholis $q_1=d(q, a_1)$, $q_2=d(q_1, a_2)$, …, $q_n=d(q_{n-1}, a_n)$.

Mint korábban megjegyeztük, egy $A=(Q, T, d)$ kimenőjel nélküli automatát szokás olyan $A=(Q, T, V, d, f)$ Mealy-automatának tekinteni, ahol $V=Q$ és $f=d$. Ez a kiterjesztett átmeneti és kimeneti függvényekre csak korlátozottan érvényes. Nevezetesen, ha $w\in T^*$ nem üres, akkor a kiterjesztett kimeneti függvényt is úgy értelmezzük, hogy $f(q, w)=d(q, w)$, $q\in Q$. Az üresszó esetén viszont a kiterjesztett átmeneti és kimeneti függvényeket ebben az esetben (tehát egy $A=(Q, T, Q, d, f)$ Mealy automatának tekintett $A=(Q, T, d)$ kimenőjel nélküli automaták esetén) úgy definiáljuk, hogy $d(q, \lambda)=q$ és $f(q, \lambda)=\lambda$ minden $q\in Q$- ra. Ekkor egy $A=(Q, T, d)$ kimenőjel nélküli automata esetén egy $q\in Q$ állapot által indukált φ_q leképezés alatt értjük azt a $\varphi_q:Q^* \to Q^*$ leképezést, melyre tetszőleges $w\in T^*$ esetén

$\varphi_q(w)=d(q, w)$, ha $w\neq\lambda$,

$\varphi_q(w)=\lambda$, ha $w=\lambda$.

A kimenőjel nélküli automaták által indukált automata leképezések speciális automata leképezések, s nem minden (Mealy- vagy Moore-féle automatával indukálható) automata leképezés indukálható véges automatával.

4.7. példa - Automataleképezések

Legyen $A=(\{q_0, q\}, \{a, b\}, \{a, b\}, q_0, d, f)$ egy iniciális Mealy-automata, ahol $d(q_0, a)=d(q, a)=q_0$, $d(q_0, b)=d(q, b)=q$, $f(q_0, a)=f(q, a)=f(q_0, b)=a$, $f(q, b)=b$. Ekkor $\varphi_{q_0}(aababb)=aaaaab$, amihez akárhogy is szerkesztünk meg egy iniciális kimenőjel nélküli $B=(Q', T, p_0, d')$ automatát, $\varphi_{p_0}(aababb)\neq aaaaab$ fog teljesülni. Tehát valóban, nem minden automata leképezés indukálható kimenőjel nélküli automatával. ★

4.5. Redukált automata. Véges determinisztikus automaták minimalizálása

Az előző fejezetben láttuk, hogy tetszőleges automata alfabetikus leképezések egy sokaságát indukálja. (Minden állapot indukál egy alfabetikus leképezést, melyek közül egyesek egybeeshetnek.) Jelölje $F_A=\{\varphi_q|q\in Q\}$ egy $A=(Q, T, V, d, f)$ által indukált leképezések sokaságát. Egy ilyen sokaságot az *A automata által indukált leképezések családjának* is fogunk hívni. Előfordulhat, hogy $p, q\in Q, p\neq q$ és mégis $\varphi_p=\varphi_q$. Most azzal a kérdéssel foglalkozunk, hogy ha egy F leképezés családhoz sikerül már olyan A automatát találnunk, melyre $F_A=F$ teljesül, hogyan tudjuk helyettesíteni A- t egy ugyanilyen tulajdonságú, de minimális állapotszámú A_0 automatával. (Természetesen ennek a dolognak akkor van igazán értelme, ha az állapothalmaz véges.) Definiáljunk egy $A=(Q, T, V, d, f)$ Mealy-automata állapothalmazán egy ϱ_A relációt: $q\varrho_A p\Leftrightarrow\varphi_q=\varphi_p$ (azaz $q\varrho_Q p\Leftrightarrow f(q, w)=f(p, w)$ minden w bemenő szóra). Könnyen belátható, hogy az így definiált ϱ_A reláció kongruencia, s ráadásul a ϱ_A- hoz tartozó C_A kompatibilis osztályozás maximális abban az értelemben, hogy minden más kompatibilis osztályozás C_A- nak finomítása. Ezesetben C_A- t az A automatához tartozó maximális kompatibilis osztályozásnak is hívjuk, az A/ϱ_A faktorautomatát pedig az *A -hoz tartozó redukált automatának* mondjuk. Általában, egy $A=(Q, T, V, d, f)$ Mealy-automatát redukáltnak nevezünk, ha tetszőleges $p, q\in Q$ pár esetén $p\varrho_A q\Leftrightarrow p=q$.

Ehhez két fontos megjegyzésünk van:

5. *Megjegyzés.*

1. Egy automatához tartozó redukált automata a legkisebb állapotszámmal (illetve végtelen automaták esetén a legkisebb számosságú állapothalmazzal) bíró olyan automata, amely ugyanazt a leképezés családot állítja elő, mint az illető automata.

2. Egy automata akkor és csak akkor redukált, ha Q- izomorf saját magával.

Egy A automata minimalizálásán az A- hoz tartozó A_0 redukált automata megszerkesztését értjük. Ebben a vonatkozásban az A_0 redukált automatát *minimálisnak* hívjuk. A következő, véges automaták minimalizálására szolgáló algoritmus D. D. Aufenkamp és F. E. Hohn nevéhez fűződik.

4.5.1. Aufenkamp-Hohn-féle minimalizációs algoritmus

Egy véges $A=(Q, T, V, d, f)$ automata esetén az A_0 redukált automatához úgy jutunk el, hogy a

$$\forall p, q\in Q:(p\varrho_A q\Leftrightarrow\forall w\in T^*:f(p, w)=f(q, w))$$

(azaz minden $p, q\in Q$ pár esetén $p\varrho_A q$ akkor és csak akkor, ha $f(p, w)=f(q, w)$ bármely $w\in T^*$ esetén fennáll) relációhoz tartozó C_A osztályozást osztályozások egy $C_1, C_2, \ldots$ sorozatán keresztül szerkesztjük meg, melyeket a következőképp definiálunk:

$$(i)\forall p, q\in Q:(C_1[p]=C_1[q]\Leftrightarrow\forall a\in T:f(p, a)=f(q, a))$$

(azaz minden $p, q\in Q$ párra a C_1 osztályozás szerint p és q akkor és csak akkor esnek egy osztályba, ha ugyanazon bemenő jelre ugyanazon kimenőjellel reagálnak);

(*ii*) ha $i\geq 1 C_{i+1}[p]=C_{i+1}[q]\Leftrightarrow C_i[p]=C_i[q]\forall a\in T:C_i[d(p, a)]=C_i[d(q, a)]$.

(Azaz ha $i\geq 1$, úgy a C_{i+1} osztályozás szerint p és q akkor és csak akkor esnek egy osztályba, ha egyrészt már C_i szerint is egy osztályba esnek, másrészt pedig minden bemenő jel hatására egy és ugyanazon C_i szerinti osztályba mennek át.)

Először megszerkesztjük a Q állapothalmaz C_1 szerinti osztályait, mikoris pontosan akkor tartozik két állapot egy osztályba, amikor minden egyes bemenő jel hatására ugyanazt a kimenőjelet adják

ki. (Lásd (i).) Ezután minden egyes $m>1$- re megszerkesztjük a C_m szerinti osztályokat egészen addig, míg $C_m=C_{m+1}$ teljesül. Látni fogjuk, hogy ilyen m létezik, s nagysága legfeljebb $|Q|-1$. Igazolni fogjuk, hogy ez a C_m osztályozás épp a Q maximális kompatibilis osztályozása. Ezután a redukált automata megszerkesztése van csak hátra, melynek szerkezete $A/C_m=(C_m, T, V, d_{C_m}, f_{C_m})$, ahol minden $C_m[q]\in C_m$, $a\in T$- re $d_{C_m}(C_m[q], a)=C_m[d(q, a)]$, illetve $f_{C_m}(C_m[q], a)=f(q, a)$. Speciálisan, ha az eredeti automata iniciális volt, és a kezdő állapota q_0 volt, akkor a redukált automata is választható iniciálisnak, éspedig úgy, hogy a kezdőállapotát $C_m[q_0]$- nak választjuk.

4.5.2. Kiegészítés az Aufenkamp-Hohn minimalizációs algoritmushoz

Ha a célunk nem az automatához tartozó (redukált) minimális állapotszámú automata, hanem csupán a φ_{q_0}- t indukáló minimális iniciális automata meghatározása, akkor ehhez az $A=(Q, T, V, q_0, d, f)$ minimalizálandó automata helyett annak a kezdőállapotból elérhető állapotok által meghatározott, azaz a $Q'=\{\gg d(q_0, w)|w\in T^*\}$ állapothalmazzal rendelkező $A'=(Q', T, V, q_0, d', f')$ iniciálisan összefüggő állapot-részautomatáját minimalizáljuk. (Emlékeztetőül: a $\gg d(q_0, w)\in Q^+$ szó utolsó betűjét jelenti.) A q_0- ból el nem érhető állapotok ugyanis nem játszanak szerepet a φ_{q_0} leképezés indukálásában (az összes q_0- ból elérhető állapot viszont igen). Mivel A végessége miatt ekkor a kezdőállapotból csak azok az állapotok érhetők el, melyek legfeljebb $|Q|-1$ hosszúságú szavakkal elérhetők, a Q állapothalmaz Q' részhalmaza algoritmikusan meghatározható. Erre a következő módszer kínálkozik: Legyen először $Q_0=\{q_0\}$. Ezután minden $i\geq 1$- re legyen $Q_{i+1}=\{q|\exists q'\in Q_i, a\in T: d(q', a)=q\}$. Ha valamely $i\geq 1$- re elérjük, hogy $Q_i=Q_{i+1}$, akkor $Q'=Q_i$. Nyilván az ilyen i- re $i\leq |Q|-1$ teljesül.

Most meg fogjuk mutatni, hogy az algoritmus és a kiegészítő megjegyzésünk korrekt. Érvényesek a következő megállapítások:

1. Segédtétel. Tetszőleges $p, q\in Q$ párra és $i\geq 1$ természetes számra $C_i[p]=C_i[q]$ akkor és csak akkor áll fenn, ha minden i- nél nem hosszabb $w\in T^*$ bemenő szóra $f(p, w)=f(q, w)$.

Bizonyítás. Először azt igazoljuk, hogy tetszőleges $p, q\in Q$ párra és $i\geq 1$ természetes számra $C_i[p]=C_i[q]$- nek következménye, hogy minden i- nél nem hosszabb $w\in X^*$ bemenő szóra $f(p, w)=f(q, w)$. Állításunkat teljes indukcióval fogjuk igazolni.

Ha $i=1$, akkor egy i- nél nem hosszabb szó vagy az üresszó, vagy a bemenő jelhalmaz egy eleme. Az üresszóra $f(p)=\lambda$ minden $p\in Q$ pár esetén fennáll, tehát $f(p, \lambda)=f(q, \lambda)(=\lambda)$ akkor is igaz, ha speciálisan $C_1[p]=C_1[q]$ valamely $p, q\in Q$- ra. Amennyiben viszont $a\in T$ és $C_1[p]=C_1[q]$, akkor $f(p, a)=f(q, a)$ is teljesülni fog összhangban a C_1 osztályozás definíciójával.

Tegyük fel most, hogy valamely $i\geq 1$- re igaz az állítás. Mutassuk meg, hogy ekkor $i+1$- re is igaz. Először észrevesszük, hogy minden olyan $p, q\in Q$ párra, melyre $C_{i+1}[p]=C_{i+1}[q]$ fennáll, definíciónk értelmében $C_i[p]=C_i[q]$ is igaz. Ekkor viszont az indukciós feltevésünk miatt minden olyan $w\in T^*$- ra, melyre w nem hosszabb i- nél, $f(p, w)=f(q, w)$. Azt kell tehát csak belátnunk, hogy amennyiben egy w szó hossza $i+1$, akkor minden olyan $p, q\in Q$ párra, melyre $C_{i+1}[p]=C_{i+1}[q]$, fennáll a $d(p, w)=d(q, w)$ egyenlőség. Igen ám, de ekkor w előáll $w=av$ alakban, ahol $a\in T$, s ugyanekkor $v\in T^*$ pedig i hosszúságú. Ekkor tehát $f(p, w)=f(p, av)=f(p, a)f(d(p, a), v)$, illetve $f(q, w)=f(q, av)=f(q, a)f(d(q, a), v)$. $C_{i+1}[p]=C_{i+1}[q]$ fennállása mellett $C_1[p]=C_1[q]$ is igaz, amiből $f(p, a)=f(q, a)$ következik (lásd $i=1$ eset). Másrészt $C_{i+1}[p]=C_{i+1}[q]$ definíció szerint azt is jelenti, hogy minden $a\in T$ esetén $C_i[d(p, a)]=C_i[d(q, a)]$. Viszont az indukciós feltevésünk értelmében ekkor minden i hosszúságú $v\in T^*$ szóra $f(d(p, a), v)=f(d(q, a), v)$. Ehhez figyelembe véve az előbb megállapított $f(p, a)=f(q, a)$ egyenlőséget, fennáll az $f(p, a)f(d(p, a), v)=f(q, a)f(d(q, a), v)$ egyenlőség is, ami épp azt jelenti, hogy $f(p, av)=f(q, av)$. Ez viszont $w=av$ értelmében az $f(p, w)=f(q, w)$ egyenlőséghez vezet.

Most azt tegyük fel, hogy adott $p, q\in Q$ és $i\geq 1$ mellett $f(p, w)=f(q, w)$ teljesül minden $w\in T^*$, $|w|\leq i$ feltételnek eleget tevő bemenő szóra. Igazoljuk teljes indukcióval, hogy ekkor $C_i[p]=C_i[q]$.

Ha i=1 és minden $a \in T$- re (azaz minden $w \in T^*$, $|w|$=1- re) $f(p, a)=f(q, a)$, akkor $C_1[p]=C_1[q]$ definíció szerint teljesül. Tegyük fel ezután indukciós feltevésként, hogy valamely rögzített $i \geq 1$- re valahányszor egy $p, q \in Q$ párra $f(p, w)=f(q, w)$ minden $|w| \leq i$- nek eleget tevő $w \in T^*$ bemenő szó esetén fennáll, mindannyiszor $C_i[p]=C_i[q]$. Legyen ezután minden i+1- nél nem hosszabb nem üres $w \in T^*$ bemenő szó esetén $f(p, w)=f(q, w)$. Ekkor $w=av$, ahol $a \in T$ és $v \in T^*$, ahol $|v| \leq i$. Ez többek között azt jelenti, hogy minden $a \in T$- re és i- nél nem hosszabb $v \in T^*$- ra $f(d(p, a), v)=f(d(q, a), v)$. Indukciós feltevésünk értelmében így $C_i[d(p, a)]=C_i[d(q, a)]$ tetszőleges $a \in T$ esetén teljesül. Másrészt ha $f(p, w)=f(q, w)$ minden i+1- nél nem hosszabb w, $v \in T^*$ bemenő szóra teljesül, úgy teljesül minden i- nél nem hosszabb bemenő szóra is. Ez viszont indukciós feltevésünk miatt $C_i[p]=C_i[q]$- t eredményezi. Ezt összevetve azzal, hogy $C_i[d(p, a)]=C_i[d(q, a)]$ tetszőleges $a \in T$ esetén teljesül, definíció szerint kapjuk, hogy $C_{i+1}[p]=C_{i+1}[q]$. Ezzel az állítást igazoltuk. ∎

2. Segédtétel. Ha valamely $m \geq 1$ természetes számra $C_m=C_{m+1}$, akkor minden $j, k \geq m$ természetes számpárra $C_j=C_k$.

Bizonyítás. Állításunkhoz nyilván elég igazolni, hogy ha $C_m=C_{m+1}$ akkor $C_{m+1}=C_{m+2}$. Legyen tehát valamely $m \geq 1$ természetes számra $C_m=C_{m+1}$, s legyen valamely $p, q \in Q$ állapotpárra $C_{m+1}[p]=C_{m+1}[q]$. Ekkor viszont definíció szerint tetszőleges $a \in T$- re teljesülni fog a $C_m[d(p, a)]=C_m[d(q, a)]$ egyenlőség. Viszont $C_m=C_{m+1}$ miatt így $C_{m+1}[d(p, a)]=C_{m+1}[d(q, a)]$ is fenn fog állni. Ez viszont a feltételezett $C_{m+1}[p]=C_{m+1}[q]$ egyenlőséggel együtt azt fogja definíció szerint eredményezni, hogy $C_{m+2}[p]=C_{m+2}[q]$. Vagyis azt kaptuk, hogy ha $C_m=C_{m+1}$, akor minden olyan $p, q \in Q$ párra, melyre $C_{m+1}[p]=C_{m+1}[q]$, fennáll $C_{m+2}[p]=C_{m+2}[q]$ is. Ez pedig pontosan azt jelenti, hogy $C_{m+1}=C_{m+2}$. ∎

Most igazoljuk, hogy az Aufenkamp-Hohn féle algoritmus valóban korrekt.

16. Tétel. Az Aufenkamp-Hohn féle algoritmus véges sok lépésben minimális redukált automatát állít elő.

Bizonyítás. Ha C_1 szerint minden állapot egy osztályba esik, akkor minden állapot egy és ugyanazon kimenőjellel reagál egy és ugyanazon bemenő jelre. Ekkor $C_1=C_2$ nyilvánvaló. Ez esetben tehát a 2. Segédtétel értelmében a minimális automata egyetlen állapottal fog rendelkezni, s egy-egy bemenő jelre ez a redukált automata egy és ugynazon kimenőjelet adja ki, mint az eredeti automata bármelyik állapota.

Tegyük fel most, hogy $|C_1|>1$. Ha $C_1=C_2$, akkor m=1 és ismét a C_1 a kívánt maximális kongruencia osztályozás. Ellenkező esetben C_2- nek legalább eggyel több osztályt kell tartalmaznia mint C_1- nek, vagyis legalább három osztályt. Ezt folytatva lesz egy egyre finomodó osztályozás rendszerünk, ahol minden egyes osztályozás legalább eggyel több osztályt fog tartalmazni mint az előző. Tekintettel arra, hogy feltételeztük $|C_1|>1$- t, továbbá figyelembe véve, hogy legfeljebb $|Q|$ számú osztálya lehet egy-egy osztályozásnak (hisz egy adott osztályozásnál minden osztálynak kell legalább egy Q- beli elemet tartalmaznia), azt kapjuk, hogy lesz olyan $m \leq |Q|-1$, melyre $C_m=C_{m+1}$. Valóban, ha veszünk valamely $m>1$- re egy egyre finomodó olyan osztályozás rendszert, melyre $C_1 \supset C_2 \supset \cdots \supset C_m$, akkor ha C_1- nek legalább két, C_2- ek legalább három, …, C_m- nek legalább m+1 eleme van és összesen $|Q|$ számú állapotunk van (vagyis legfeljebb $|Q|$ számú osztálya lehet egy-egy osztályozásnak), akkor $m+1 \leq |Q|$ fennállása azt is jelenti, hogy $m \leq |Q|-1$. Így viszont valóban lesz olyan $m \leq |Q|-1$, mikoris $C_m=C_{m+1}$, ami a 2. Segédtétel értelmében azt is jelenti, hogy teszőleges $i \geq m$- re $C_i=C_m$. Ez azonban az 1. Segédtétel miatt azt is eredményezi, hogy teszőleges olyan $p, q \in Q$ állapotpárra, melyre $C_m[p]=C_m[q]$, teljesülni fog a $f(p, w)=f(q, w)$ egyenlőség, akármilyen hosszú is a tekintett $w \in T^*$ bemenő szó. Ezzel beláttuk, hogy C_m egy kongruencia osztályozás, aholis $m \leq |Q|-1$.

Be kell még látnunk, hogy C_m valóban maximális kongruencia osztályozás, vagyis ha két állapot nem tartozik C_m szerint egy osztályba, akkor az általuk indukált leképezések különbözőek. Legyen $p, q \in Q$ két állapot, mely C_m szerint nem tartozik egy osztályba. Ha már C_1 szerint sem tartoznak egy osztályba, akkor valamely $a \in T$- re $f(p, a) \neq f(q, a)$ és akkor valóban igaz, hogy $\varphi_p \neq \varphi_q$. Ellenkező esetben van egy maximális $k<m$, hogy $C_k[p]=C_k[q]$, ám $C_{k+1}[p] \neq C_{k+1}[q]$. Ekkor viszont a 1. Segédtétel alapján van

legalább egy olyan k+1 hosszúságú $w \in T^*$ szó, hogy $d(p, w) \neq d(q, w)$. (Nevezetesen, minden k+1- nél rövidebb $w \in T^*$ szóra $C_k[p]=C_k[q]$ miatt $f(p, w)=f(q, w)$, másrészről pedig $C_{k+1}[p] \neq C_{k+1}[q]$ miatt van legalább egy olyan k+1- nél nem hosszabb $w \in T^*$ szó, melyre $f(p, w) \neq f(q, w)$. A két állítás együttes fennállása azt jelenti, hogy $f(p, w) \neq f(q, w)$ legalább egy k+1 hosszú szóra teljesül.)

Be kell még látnunk azt is hogy a kiegészítő megjegyzésünk is korrekt. A fentiek alapján az is igaz, hogy ha az A automata iniciális és kezdőállapota q_0, akkor a redukált automatát is választhatjuk iniciálisnak oly módon, hogy kezdőállapotának a $C_m[q_0]$ osztályt választjuk. Ez az osztály is a φ_{q_0} leképezést fogja a redukált automatában indukálni ugyanúgy, mint ahogy a q_0 a φ_{q_0} leképezést indukálja A- ban.

Ha egy iniciális $A=(Q, T, V, q_0, d, f)$ véges automatához keressük azt a minimális automatát, mely φ_{q_0}- t indukálja, először célszerű a Q állapothalmaz $Q'=\{q' \in Q | q' = \gg d(q_0, w), w \in T^*\}$ részhalmazát meghatározni, s azután a Q' állapothalmaz által meghatározott A' iniciális állapot-részautomatát minimalizálni. Világos ugyanis, hogy ez az A' iniciális állapot-részautomata ugyancsak a φ_{q_0}- t indukálja, azaz A- ban a $Q \setminus Q'$- beli, azaz az q_0- ból el nem érhető állapotok ugyanis φ_{q_0} indukálásában nem játszanak szerepet. Jelölje B a A- hoz tartozó iniciális redukált automatát, B' pedig a A'- höz tartozó iniciális redukált automatát. Világos, hogy B' ugyanúgy a φ_{q_0}- t fogja indukálni, mint B. Mivel Q' részhalmaza a Q- nak, az is világos, hogy a B' állapothalmaza részhalmaza a B állapothalmazának. Így ha Q'_B jelöli a B', továbbá Q_B jelöli a B állapothalmazát, $|Q'_B| \leq |Q_B|$.

Esetleg lehet olyan $q \in Q \setminus Q'$ állapotunk, melyre $\varphi_q \notin \{\varphi_{q'} | q' \in Q'\}$. Ekkor azonban $|Q'_B| < |Q_B|$, vagyis ilyen esetben B nem egy minimális, φ_{q_0} leképezést indukáló automata. Tehát valóban, egy φ_{q_0} leképezést indukáló minimális automata keresésénél célszerű először a vizsgált A iniciálisan összefüggő állapot részautomatáját meghatározni, majd az Aufenkamp-Hohn féle algoritmust erre az iniciális állapot részautomatára alkalmazni.

Kérdés még, hogy a Q' meghatározásához javasolt algoritmusunk korrekt-e. Az világos, hogy minden $i \geq 1$- re a Q_i halmaz definíció szerint a q_0- ból legfeljebb i hosszúságú szavakkal elérhető állapotok halmaza. Így azt kell csak kimutatnunk, hogy ha egy állapot q_0- ból elérhető, akkor elérhető $|Q|$-1- nél nem hosszabb bemenő szóval is.

Tetszőleges $w, v, u \in T^*$ esetén $\gg d(q_0, wv) = \gg d(q_0, w)$ mellett $\gg d(q_0, wvu) = \gg d(q_0, wu)$ nyilván fennáll. Így ha valamely $q \in Q$ állapothoz van olyan w' szó, hogy $\gg d(q_0, w') = q$, akkor w' megválaszható úgy, hogy bármely két különböző w'', $w''' \in T^*$ kezdőszeletére $\gg d(q_0, w'') \neq \gg d(q_0, w''')$ teljesüljön. Ekkor viszont w' hossza legfeljebb $|Q|$-1 lehet, különben az ismétlődés elkerülhetetlen. Így megvizsgálva azt, hogy melyek azok az állapotok, melyek elérhetők a kezdőállapotból legfeljebb $|Q|$-1 hosszú szóval, megkapjuk Q'- t. Az is világos, hogy ha valamely $k < |Q|$-1- re a q_0- ból legfeljebb k hosszú szóval elérhető állapotok Q_k halmaza egybeesik a q_0- ból legfeljebb k+1 hosszú szóval elérhető állapotok halmazával, úgy a q_0- ból elérhető összes állapotok halmaza Q_k- val megegyezik. Valóban, ha a legfeljebb k hosszú bemenő szavakkal ugyanazokat az állapotokat érjük el mint a legfeljebb k+1 hosszú szavakkal, akkor minden $l \geq 0$- ra a legfeljebb k hosszú hosszú bemenő szavakkal ugyanazokat az állapotokat érjük el mint a legfeljebb $k+l$, azaz bármilyen hosszú szavakkal. Az Aufenkamp-Hohn-féle minimalizációs algoritmushoz vett kiegészítő megjegyzésünk tehát ugyancsak korrekt. ∎

4.8. példa - Mealy-automata minimalizálása 1.feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus segítségével az
$A=(\{a_1,a_2,a_3,a_4,a_5,a_6\},\{x,y\},\{u,v,w\},d,f)$
Mealy-automatához tartozó A_0 minimális állapotszámú automatát! (d és f a következő táblázattal adott):

A	a_1	a_2	a_3	a_4	a_5	a_6
x	(a_2,u)	(a_1,w)	(a_1,u)	(a_5,w)	(a_4,u)	(a_3,u)
y	(a_3,v)	(a_2,v)	(a_4,v)	(a_4,v)	(a_3,v)	(a_2,v)

Megoldás:
(I.) A feladat megoldásának első lépéseként különböző osztályokba fogjuk sorolni az A automata belső állapotait. A kiinduló oszályokban két állapot akkor és csak akkor esik egy osztályba, ha ugyanarra a bemenő jelre ugyanazzal a kimenőjellel reagálnak.

Jelen esetben:
$C_1=\{a_1,a_3,a_5,a_6\},\{a_2,a_4\}$.
Ezek után a C_{i+1}-edik osztályozás esetén két állapot akkor esik egy osztályba, ha egyrészt a C_i-edik osztályozás esetén is azonos osztályba tartoztak, másrészt pedig minden bemenő jel hatására azonos C_i-beli osztályban található állapotokba mennek át. Az osztályozás véget ér, amennyiben $C_i=C_{i+1}$ valamely $i \geq 1$ esetén.

Jelen esetben:
$C_2=\{a_1,a_5\},\{a_3,a_6\},\{a_2,a_4\}$.
$C_3=\{a_1,a_5\},\{a_3\},\{a_6\},\{a_2,a_4\}$.
$C_4=\{a_1,a_5\},\{a_3\},\{a_6\},\{a_2,a_4\}$.
Mivel $C_3=C_4$, ezért az osztályozás véget ért. A C_3 osztályait jelöljük valamely új betűvel, például b-vel:
$b_1=\{a_1,a_5\},b_2=\{a_3\},b_3=\{a_6\},b_4=\{a_2,a_4\}$.
(II.) Készítsük el az A automatával ekvivalens, minimális állapotszámú A_0 automatát, mely állapothalmazát az osztályozás és az új jelölés bevezetése után kapott b_i betűk alkotják,
- a bemenő- és kimenőjeleinek halmaza megegyezik az A automata megfelelő jeleinek halmazával,
- az átmenetfüggvényét megkapjuk úgy, hogy megnézzük, hogy az adott b_i osztálybeli állapotok az adott bemenő jel hatására mely b_j osztálybeli állapotokba mentek át az A automata esetén
- és végül az aktuális b_i osztályhoz tartozó kimenőjelet megkapjuk, ha megnézzük, hogy az adott b_i osztálybeli állapotok az adott bemenő jel hatására mely kimenőjelet szolgáltatták az A automata működése közben.
Jelen esetben:
$A_0=(\{b_1,b_2,b_3,b_4\},\{x,y\},(u,v,w),d',f')$. Ahol d' és f' a következő táblázattal adott:

A_0	b_1	b_2	b_3	b_4
x	(b_4,u)	(b_1,u)	(b_2,u)	(b_1,w)
y	(b_2,v)	(b_4,v)	(b_4,v)	(b_4,v)

★

4.9. példa - Mealy-automata minimalizálása 2.feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus segítségével az
$A=(\{a_1,a_2,a_3,a_4,a_5,a_6,a_7\},\{x,y\},\{u,v,w\},d,f)$
Mealy-automatához tartozó A_0 minimális állapotszámú automatát!
Ahol d és f a következő táblázattal adott:

A	a_1	a_2	a_3	a_4	a_5	a_6	a_7
x	(a_2,w)	(a_3,w)	(a_2,w)	(a_7,w)	(a_6,w)	(a_5,w)	(a_4,w)
y	(a_4,u)	(a_2,v)	(a_4,u)	(a_2,v)	(a_4,u)	(a_5,v)	(a_2,u)

Megoldás:
(I.)
$C_1=\{a_1,a_3,a_5,a_7\},\{a_2,a_4,a_6\}.$
$C_2=\{a_1,a_3,a_5,a_7\},\{a_2,a_4\},\{a_6\}.$
$C_3=\{a_1,a_3,a_7\},\{a_5\},\{a_2,a_4\},\{a_6\}.$
$C_4=\{a_1,a_3,a_7\},\{a_5\},\{a_2,a_4\},\{a_6\}.$
$b_1=\{a_1,a_3,a_7\},b_2=\{a_5\},b_3=\{a_2,a_4\},b_4=\{a_6\}.$
(II.)
$A_0=(\{b_1,b_2,b_3,b_4\},\{x,y\},(u,v,w),d',f').$
Ahol d' és f' a következő táblázattal adott:

A_0	b_1	b_2	b_3	b_4
x	(b_3,w)	(b_4,w)	(b_1,w)	(b_2,w)
y	(b_3,u)	(b_3,u)	(b_3,v)	(b_2,v)

★

4.10. példa - Mealy-automata minimalizálása 3.feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus segítségével az
$A=(\{a_1,a_2,a_3,a_4,a_5\},\{x,y,z\},\{u,v\},d,f)$
Mealy-automatához tartozó A_0 minimális állapotszámú automatát!

A	a_1	a_2	a_3	a_4	a_5
x	(a_2,u)	(a_2,v)	(a_2,u)	(a_5,v)	(a_4,v)
y	(a_4,u)	(a_3,u)	(a_1,u)	(a_3,u)	(a_3,u)
z	(a_5,v)	(a_5,u)	(a_1,v)	(a_5,u)	(a_2,u)

Megoldás:
(I.)
$C_1=\{a_1,a_3\},\{a_2,a_4,a_5\}$.
$C_2=\{a_1\},\{a_3\},\{a_2,a_4,a_5\}$.
$C_3=\{a_1\},\{a_3\},\{a_2,a_4,a_5\}$.
$b_1=\{a_1\},b_2=\{a_3\},b_3=\{a_2,a_4,a_5\}$.
(II.)
$A_0=(\{b_1,b_2,b_3\},\{x,y,z\},(u,v),d',f')$.

A_0	b_1	b_2	b_3
x	(b_3,u)	(b_3,u)	(b_3,v)
y	(b_3,u)	(b_1,u)	(b_2,u)
z	(b_3,v)	(b_1,v)	(b_3,u)

★

4.11. példa - Mealy-automata minimalizálása 4.feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus segítségével az
$A=(\{a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8\},\{x,y\},\{u,v\},d,f)$
Mealy-automatához tartozó A_0 minimális állapotszámú automatát!

A	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
x	(a_8,u)	(a_3,v)	(a_8,u)	(a_1,v)	(a_8,u)	(a_4,u)	(a_3,v)	(a_5,v)
y	(a_2,v)	(a_1,u)	(a_7,u)	(a_2,v)	(a_7,v)	(a_2,v)	(a_6,u)	(a_3,v)

Megoldás:
(I.)
$C_1=\{a_1,a_5,a_6\},\{a_2,a_7\},\{a_3\},\{a_4,a_8\}$.
$C_2=\{a_1,a_5,a_6\},\{a_2,a_7\},\{a_3\},\{a_4\},\{a_8\}$.
$C_3=\{a_1,a_5\},\{a_6\},\{a_2,a_7\},\{a_3\},\{a_4\},\{a_8\}$.
$C_4=\{a_1,a_5\},\{a_6\},\{a_2\},\{a_7\},\{a_3\},\{a_4\},\{a_8\}$.
$C_5=\{a_1\},\{a_5\},\{a_6\},\{a_2\},\{a_7\},\{a_3\},\{a_4\},\{a_8\}$.
Ebben az esetben - mivel a C_5 osztályozás esetén minden a_i állapot külön osztályba esik, - az A automata már minimális, így tovább nem minimalizálható, azaz $A_0=A$. ★

4.5.3. Aufenkamp-Hohn féle minimalizációs algoritmus Moore-automatákhoz adaptált változata

Megjegyezzük először, hogy ha egy Moore-automatát Mealy-automatának tekintünk, s minimalizáljuk, az eredmény nem feltétlenül lesz Moore-féle automata (lásd Mealy és Moore automaták homomorfizmusa). Ha az $A=(Q, T, V, d, g)$ Moore-automatához azt a minimális állapotszámú Moore-féle automatát akarjuk meghatározni, mely ugyanazt a leképezési családot indukálja, mint az eredeti Moore-automata, akkor az Aufenkamp-Hohn féle algoritmust egy kicsit módosítanunk kell.

Egy véges $A=(Q, T, V, d, g)$ Moore-automata esetén az A_0 redukált Moore-automatához úgy jutunk el, hogy definiálunk egy kongruencia relációt a következőképpen:

$\forall p, q\in Q:(p\varrho_A q\Leftrightarrow\forall a_1, \ldots, a_n\in T:g(p_1\cdots p_n)=g(q_1\cdots q_n))$,

$p_1=d(p, a_1), p_2=d(p_1, a_2), \ldots, p_n=d(p_{n-1}, a_n)$,

$q_1=d(q, a_1), q_2=d(q_1, a_2), \ldots, q_n=d(q_{n-1}, a_n)$

(azaz minden $p, q\in Q$ pár esetén $p\varrho_A q$ akkor és csak akkor, ha $g(p_1\cdots p_n)=g(q_1\cdots q_n)$ fennáll minden olyan $p_1, \ldots, p_n, q_1, \ldots, q_n\in Q$ esetén, melyekre $p_1=d(p, a_1), p_2=d(p_1, a_2), \ldots, p_n=d(p_{n-1}, a_n), q_1=d(q, a_1), q_2=d(q_1, a_2), \ldots, q_n=d(q_{n-1}, a_n)$ fennáll valamely $a_1, \ldots, a_n\in T$ bemenő jelek esetén).

Ekkor a ϱ_A relációhoz tartozó C_A osztályozást osztályozások egy $C_1, C_2, \ldots$ sorozatán keresztül szerkesztjük meg, melyeket a következőképp definiálunk:

(i) $\forall p, q\in Q:(C_1[p]=C_1[q]\Leftrightarrow\forall a\in T:g(p)=g(q))$

(azaz minden $p, q\in Q$ párra a C_1 osztályozás szerint p és q akkor és csak akkor esnek egy osztályba, ha ugyanaz az állapotjelük);

(ii) ha $i\geq 1 C_{i+1}[p]=C_{i+1}[q]\Leftrightarrow C_i[p]=C_i[q]\forall a\in T:C_i[d(p, a)]=C_i[d(q, a)]$.

(Azaz ha $i\geq 1$, akkor amint a Mealy-automatáknál, a C_{i+1} osztályozás szerint p és q akkor és csak akkor esnek egy osztályba, ha egyrészt már C_i szerint is egy osztályba esnek, másrészt pedig minden bemenő jel hatására egy és ugyanazon C_i szerinti osztályba mennek át.)

A továbbiakban formailag a Moore-automatákra adaptált Aufenkamp-Hohn féle eljárás csaknem teljesen megegyezik a Mealy-automatáknál tárgyaltakkal. Először megszerkesztjük a Q állapothalmaz C_1 szerinti osztályait, mikoris pontosan akkor tartozik két állapot egy osztályba, amikor ugyanaz az állapotjelük. (Lásd (i) pont, fentebb.) Ezután minden egyes $m>1$- re megszerkesztjük a C_m szerinti osztályokat egészen addig, míg $C_m=C_{m+1}$ teljesül. Ugyanúgy mint Mealy-automaták esetén, ilyen m létezik, s nagysága legfeljebb $|Q|-1$. Ez a C_m osztályozás épp a Q (Moore-változatú) maximális kompatibilis osztályozása. Ezután a Moore-féle redukált automata megszerkesztése van csak hátra, melynek szerkezete $A/C_m=(C_m, T, V, d_{C_m}, g_{C_m})$, aholis minden $C_m[q]\in C_m$, $a\in T$- re $d_{C_m}(C_m[q], a)=C_m[d(q, a)]$, illetve $g_{C_m}(C_m[q])=g(q)$. Speciálisan, ha az eredeti automata iniciális volt, és a kezdő állapota q_0 volt, akkor a redukált automata is választható iniciálisnak, éspedig úgy, hogy a kezdőállapotát $C_m[q_0]$- nak választjuk.

4.5.4. Kiegészítés az Aufenkamp-Hohn-féle minimalizációs algoritmus Moore-automatákhoz adaptált változatához

Ha a célunk nem az automatához tartozó (redukált) minimális állapotszámú Moore-automata, hanem csupán az a φ_{q_0}- t indukáló minimális iniciális Moore-automata meghatározása, akkor ehhez az $A=(Q, T, V, q_0, d, g)$ minimalizálandó automata helyett annak a kezdőállapotból elérhető állapotok által meghatározott, azaz a $Q'=\{\gg d(q_0, w)|w\in T^*\}$ állapothalmazzal rendelkező $A=(Q', T, V, q_0, d', g')$

iniciálisan összefüggő állapot-részautomatáját minimalizáljuk. A q_0- ból el nem érhető állapotok ugyanis nem játszanak szerepet a φ_{q_0} leképezés indukálásában (az összes q_0- ból elérhető állapot viszont igen). Mivel A végessége miatt ekkor a kezdőállapotból csak azok az állapotok érhetők el, melyek legfeljebb $|Q|$-1 hosszúságú szavakkal elérhetők, a Q állapothalmaz Q' részhalmaza algoritmikusan meghatározható. Erre ugyanaz a módszer kínálkozik mint Mealy-automaták esetén: Legyen először $Q_0=\{q_0\}$. Ezután minden $i \geq 1$- re legyen $Q_{i+1}=\{q|\exists q' \in Q_i,\ a \in T: d(q',\ a)=q\}$. Ha valamely $i \geq 1$- re elérjük, hogy $Q_i=Q_{i+1}$, akkor $Q'=Q_i$. Nyilván az ilyen i- re $i \leq |Q|$-1 teljesül.

4.12. példa - Moore-automata minimalizálása 1.feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus segítségével az
$A=(\{a_1,a_2,a_3,a_4,a_5,a_6\},\{x,y\},\{u,v,w\},d,g)$
Moore-automatához tartozó A_0 minimális állapotszámú automatát!
A d és g függvények a következő táblázattal adottak:

	u	***v***	***u***	***u***	***w***	***w***
A	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$	$\boldsymbol{a_5}$	$\boldsymbol{a_6}$
x	a_4	a_1	a_4	a_1	a_1	a_3
y	a_1	a_3	a_6	a_4	a_6	a_5

Megoldás:
(I.) A feladat megoldásának első lépéseként különböző osztályokba fogjuk sorolni az A automata belső állapotait. A kiinduló oszályokban két állapot akkor és csak akkor esik egy osztályba, ha ugyanarra a bemenő jelre ugyanazzal a kimenőjellel reagálnak.

Jelen esetben:
$C_1=\{a_1,a_3,a_4\},\{a_2\},\{a_5,a_6\}$.
Ezek után a C_{i+1}-edik osztályozás esetén két állapot akkor esik egy osztályba, ha egyrészt a C_i-edik osztályozás esetén is azonos osztályba tartoztak, másrészt pedig minden bemenő jel hatására azonos C_i-beli osztályban található állapotokba mennek át. Az osztályozás véget ér, amennyiben $C_i=C_{i+1}$ valamely $i \geq 1$ esetén.

Jelen esetben:
$C_2=\{a_1,a_4\},\{a_2\},\{a_3\},\{a_5,a_6\}$.
$C_3=\{a_1,a_4\},\{a_2\},\{a_3\},\{a_5\},\{a_6\}$.
$C_4=\{a_1,a_4\},\{a_2\},\{a_3\},\{a_5\},\{a_6\}$.
Mivel $C_3=C_4$, ezért az osztályozás véget ért. A C_3 osztályait jelöljük valamely új betűvel, például b-vel:
$b_1=\{a_1,a_4\},b_2=\{a_2\},b_3=\{a_3\},b_4=\{a_5\},b_5=\{a_6\}$.
(II.) Készítsük el az A automatával ekvivalens, minimális állapotszámú A_0 automatát, mely állapothalmazát az osztályozás és az új jelölés bevezetése után kapott b_i betűk alkotják,
- a bemenő- és kimenőjeleinek halmaza megegyezik az A automata megfelelő jeleinek halmazával,
- az átmenetfüggvényét megkapjuk úgy, hogy megnézzük, hogy az adott b_i osztálybeli állapotok az adott bemenő jel hatására mely b_j osztálybeli állapotokba mentek át az A automata esetén
- és végül az aktuális b_i osztályhoz tartozó kimenőjelet megkapjuk, ha megnézzük, hogy az adott b_i osztálybeli állapotok az adott bemenő jel hatására mely kimenőjelet szolgáltatták az A automata működése közben.
Jelen esetben:
$A_0=(\{b_1,b_2,b_3,b_4,b_5\},\{x,y\},(u,v,w),d',g')$.
Táblázattal:

	u	***v***	***u***	***w***	***w***
$\boldsymbol{A_0}$	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$	$\boldsymbol{b_3}$	$\boldsymbol{b_4}$	$\boldsymbol{b_5}$
x	b_1	b_1	b_1	b_1	b_3
y	b_1	b_3	b_5	b_5	b_4

★

4.13. példa - Moore-automata minimalizálása 2.feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus segítségével az
$A=(\{a_1,a_2,a_3,a_4,a_5\},\{x,y\},\{u,v\},d,g)$
Moore-automatához tartozó A_0 minimális állapotszámú automatát!

	u	***v***	***v***	***u***	***u***
A	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$	$\boldsymbol{a_5}$
x	a_2	a_4	a_4	a_2	a_3
y	a_5	a_3	a_2	a_3	a_1

Megoldás:
(I.)
$C_1=\{a_1,a_4,a_5\},\{a_2,a_3\}$.
$C_2=\{a_1,a_5\},\{a_4\},\{a_2,a_3\}$.
$C_3=\{a_1,a_5\},\{a_4\},\{a_2,a_3\}$.
$b_1=\{a_1,a_5\},b_2=\{a_4\},b_3=\{a_2,a_3\}$.
(II.)
$A_0=(\{b_1,b_2,b_3\},\{x,y\},(u,v),d',g')$.
Ahol d' és g' a következő táblázattal adott:

	u	***u***	***v***
$\boldsymbol{A_0}$	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$	$\boldsymbol{b_3}$
x	b_3	b_3	b_2
y	b_1	b_3	b_3

★

4.14. példa - Moore-automata minimalizálása 3.feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus segítségével az
$A=(\{a_1,a_2,a_3,a_4,a_5\},\{x,y,z\},\{u,v,w\},d,g)$
Moore-automatához tartozó A_0 minimális állapotszámú automatát!

	u	***v***	***v***	***w***	***w***
A	a_1	a_2	a_3	a_4	a_5
x	a_2	a_4	a_4	a_2	a_3
y	a_5	a_4	a_5	a_3	a_3
z	a_1	a_3	a_2	a_5	a_4

Megoldás:
(I.)
$C_1=\{a_1\},\{a_2,a_3\},\{a_4,a_5\}$.
$C_2=\{a_1\},\{a_2,a_3\},\{a_4,a_5\}$.
$b_1=\{a_1\},b_2=\{a_2,a_3\},b_3=\{a_4,a_5\}$.
(II.)
$A_0=(\{b_1,b_2,b_3\},\{x,y,z\},(u,v,w),d',g')$.

	u	***v***	***w***
A_0	b_1	b_2	b_3
x	b_2	b_3	b_2
y	b_3	b_3	b_2
z	b_1	b_2	b_3

★

4.5.5. Minimalizációs algoritmus kimenőjel nélküli automatákra

Tetszőleges $A=(Q, T, d)$ kimenőjel nélküli automata esetén valamely $q\in Q$ állapot és nem üres w bemenő szó esetén a kimenő szót $d(q, w)$- vel azonosítottuk. Ha a bemenő szó az üresszó, akkor viszont kimenő szónak ugyancsak az üresszót tekintettük minden állapotra vonatkozóan, mint a Mealy-automatáknál.

Ezek szerint egy egymástól különböző $p, q\in Q$ állapotpár nem tartozhat egy osztályba, ha valamely (nem feltétlen különböző) $r, s\in Q$ állapotpárra és $a, b\in T$ bemenő jelekre $d(r, a)=p$ és $d(s, b)=q$. Másképp mondva, két $p, q\in Q$ állapot csak úgy tartozhat egy osztályba, ha egyrészt minden $a\in T$- re $d(p, a)=d(q, a)$, másrészt legalább egy $r\in\{p, q\}$- ra akárhogy is adjuk meg az $s\in Q$, $a\in T$ párt, $d(s, a)\neq r$.

Így az Aufenkamp-Hohn féle minimalizációs algoritmus helyett kimenőjel nélküli automatákra a következő algoritmust alkalmazzuk:

Egy véges $A=(Q, T, d)$ kimenőjel nélküli automata esetén az A_0 redukált kimenőjel nélküli automatához is úgy jutunk el, hogy először képezzük a Q egy olyan Q_0 részhalmazát, mely tartalmaz minden olyan $q\in Q$ állapotot, melyhez van olyan $p\in Q$, $a\in T$, hogy $d(p, a)=q$. Vezessük be továbbá a $C_0=Q\setminus Q_0$ jelölést. Mindaddig, míg valamely $i\geq 0$- ra C_i üres nem lesz, hajtsuk végre a következő algoritmust:

Tegyük fel, hogy valamely $i\geq 0$- ra Q_i és C_i adottak. Legyen valamely tetszőlegesen rögzített $r\in C_i$- re $Q_{i+1}=Q_i\cup\{r\}$. Eután C_{i+1}- t C_i- ből úgy képezzük, hogy C_i elemei közül elhagyunk minden olyan $r'\in C_i$ elemet, melyhez van olyan $p\in Q_{i+1}$, hogy tetszőleges $a\in T$ mellett $d(r', a)=d(p, a)$. Ha $C_{i+1}=\emptyset$, akkor készen vagyunk, különben növeljük eggyel i értékét, s ismételjük meg az előző lépést.

Világos, hogy Q végessége miatt ez az eljárás véges lépésben valamely $i \geq 0$- ra véget ér, s az is könnyen igazolható, hogy a fenti algoritmus végén minden $q \in Q$- hoz tartozik pontosan egy olyan $p \in Q_{i+1}$, hogy tetszőleges $a \in T$ mellett $d(q, a)=d(p, a)$. Ekkor megkapjuk a $B=(Q', T, d')$ kimenőjel nélküli automatát, ahol $Q'=Q_{i+1}$ és $d'=d|_{Q' \times T}$ mellett B olyan (kimenőjel nélküli) állapot részautomatája lesz A- nak, hogy minden $q \in Q$- hoz található olyan $p \in Q'$, hogy tetszőleges $a \in T$ mellett $d(q, a)=d(p, a)$. Ez a B kimenőjel nélküli automata lesz az A kimenőjel nélküli automatához tartozó redukált automata.

4.5.6. Kiegészítés a minimalizációs algoritmus kimenőjel nélküli automatákra ismertetett változatához

Ha a célunk nem a kimenőjel nélküli automatához tartozó (redukált) minimális állapotszámú kimenőjel nélküli automata, hanem csupán az a φ_{q_0}- t indukáló minimális iniciális kimenőjel nélküli automata meghatározása, akkor ehhez az $A=(Q, T, q_0, d)$ minimalizálandó kimenőjel nélküli automata helyett annak a kezdő állapotból elérhető állapotok által meghatározott, azaz a $Q'=\{\gg d(q_0, w) | w \in T^*\}$ állapothalmazzal rendelkező $A=(Q', T, q_0, d')$ iniciálisan összefüggő (kimenőjel nélküli) állapot-részautomatáját minimalizáljuk. A q_0- ból el nem érhető állapotok ugyanis most sem játszanak szerepet a φ_{q_0} leképezés indukálásában (az összes q_0- ból elérhető állapotok viszont igen). Mivel A végessége miatt ekkor a kezdőállapotból csak azok az állapotok érhetők el, melyek legfeljebb $|Q|$-1 hosszúságú szavakkal elérhetők, a Q állapothalmaz Q' részhalmaza algoritmikusan meghatározható. Erre itt is a következő módszer kínálkozik: Legyen először $Q_0=\{q_0\}$. Ezután minden $i \geq 1$- re legyen $Q_{i+1}=\{q | \exists q' \in Q_i, a \in T: d(q', a)=q\}$. Ha valamely $i \geq 1$- re elérjük, hogy $Q_i=Q_{i+1}$, akkor $Q'=Q_i$. Nyilván az ilyen i- re $i \leq |Q|$-1 teljesül. A q_0- ból elérhető minden $q' \in Q'$ állapot nyilván rendelkezik azzal a tulajdonsággal, hogy alkalmas $p \in Q'$, $a \in T$ párra $d(p, a)=q'$. Így ha q_0 is elérhető önmagából, akkor A' egy minimális, φ_{q_0}- t indukáló automata lesz. Ugyancsak az lesz, ha minden $q \in Q' \setminus \{q_0\}$ esetén van olyan $a \in T$, melyre $d(q_0, a) \neq d(q, a)$. Ellenkező esetben valamely $q \in Q'$- re $d(q, a)=d(q_0, a)$ minden $a \in T$- re fennáll, továbbá tetszőleges (az üresszótól különböző) $w \in T^+$ bemenő szó esetén $d(q_0, w) \neq zq_0$ semmilyen $z \in Q^*$ esetén sem. Ha van olyan $w \in T^+$ (üresszótól különböző) bemenő szó, hogy $d(q, w)=zq$ valamely $z \in Q^*$- ra, akkor világos, hogy A' ismét redukált automata lesz. Ellenkező esetben amellett, hogy az $q, q_0 \in Q$ párra $d(q, a)=d(q_0, a)$, $a \in T$ fennáll, $\{\gg d(q_0, w) | w \in T^+\} \cap \{q_0, q\}=\emptyset$ és $\{\gg d(q, w) | w \in T^+\} \cap \{q_0, q\}=\emptyset$ is teljesül. Világos hogy ebben az esetben a $d''(p, a)=d(p, a)$, $p \in Q' \setminus \{q_0\}$, $a \in T$ átmeneti függvénnyel definiált $B=(Q' \setminus \{q_0\}, T, q, d'')$ egy olyan minimális állapotszámú iniciális kimenőjel nélküli automata lesz, melynek $q \in Q' \setminus \{q_0\}$ kezdőállapota pont φ_{q_0}- t indukálja.

Csak megemlítjük itt, hogy hasonló módszerrel minimalizálhatóak a Rabin-Scott (felismerő-) automaták is (részleteket lásd Véges determinisztikus elfogadó automaták minimalizálása fejezetben). Ezesetben, az Aufenkamp-Hohn algoritmus első osztályozó lépésében két csoportot képzünk, mégpedig a végállapotok, illetve a többi állapot halmazát.

4.6. Automaták ekvivalenciája, Gill tétele

A közös T, V halmazokkal bíró $A=(Q, T, V, d, f)$ és $A'=(Q', T, V, d', f')$ automaták $p \in Q$ és $q \in Q'$ állapotait egymással ekvivalens állapotoknak mondjuk (jelekben: $p \equiv q$), ha p és q az A, illetve B automatában ugyanazt a leképezést indukálja, azaz $\varphi_{p, A}=\varphi_{q, B}$. Magukat az A és B automatákat ekvivalenseknek nevezzük, ha bármelyikük bármely állapotához van a másiknak ezzel az állapottal ekvivalens állapota, azaz $F_A=F_B$. Speciálisan, két iniciális automatát akkor mondunk ekvivalensnek, ha kezdőállapotaik ekvivalensek egymással.

17. Tétel. Tetszőleges A automatával ekvivalens automaták M halmazában létezik olyan Q-izomorfiától eltekintve egyértelműen meghatározott A' automata, amely bármely M- beli automatának Q- homomorf képe. A'- ként választható bármely M- beli automatához tartozó redukált automata.

Mint már korábban láttuk, egy Moore-automata tekinthető speciális Mealy-automatának. Az elmélet kiépítése folyamán fontos szerepet tölt be Arthur Gill (ejtsd: gill, zsill) következő eredménye, mely azt igazolja, hogy információ átalakítás szempontjából a két automata fogalom ekvivalens.

18. Tétel. (Gill tétele) Bármely Mealy-automatához létezik vele ekvivalens Moore-automata. Emellett, ha a Mealy-automata véges, akkor a hozzá megkonstruált, vele ekvivalens Moore-automata is választható végesnek.

Bizonyítás. Legyen $A=(Q, T, V, d, f)$ tetszőleges Mealy-automata. Egy vele ekvivalens Moore-automatát a következőképp konstruálhatunk meg:

Legyen $A'=(Q', T, V, d', f')$, ahol $Q'=Q\cup Q\times T$, továbbá tetszőleges $q'\in Q'$, $a\in T$ párra

$d'(q', a)=(q', a)$ ha $q'\in Q$,

$d'(q', a)=(d(q, a'), a)$ ha $q'=(q, a')\in Q\times T$,

$f'(q', a)=f(q, a)$ ha $q'\in Q$,

$f'(q', a)=f(d(q, a'), a)$ ha $q'=(q, a')\in Q\times T$.

Mutassuk meg, hogy minden $q\in Q$, $w\in T^*$ esetén $f(q, w)=f'(q, w)$. Nyilván elegendő nem üres szavakra szorítkoznunk. Legyen tehát $a_1, \ldots a_n\in T$ és tekintsük a q, $q_1=d(q, a_1)$, $q_2=d(q_1, a_2)$, …, $q_{n-1}=d(q_{n-2}, a_{n-1})$ állapotokhoz a $b_1=f(q, a_1)$, $b_2=f(q_1, a_2)$, …, $b_n=f(q_{n-1}, a_n)$ kimenőjeleket. Világos, hogy ekkor definíció szerint $f(q, a_1\cdots a_n)=b_1\cdots b_n$. Másrészről, d' és f' definíciója alapján $d'(q, a_1)=(q, a_1), d'((q, a_1), a_2)=(d(q, a_1), a_2)=(q_1, a_2), d'((q_1, a_2), a_3)=(d(q_1, a_2), a_3)=(q_2, a_3)$, illetve $f'(q, a_1)=f(q, a_1)=b_1, f'((q, a_1), a_2)=f(d(q, a_1), a_2)=f(q_1, a_2)=b_2, f'((q_1, a_2), a_3)= =f(d(q_1, a_2), a_3)=f(q_2, a_3)=b_3, \ldots, f'((q_{n-2}, a_{n-1}), a_n)=f(d(q_{n-2}, a_{n-1}), a_n)=f(q_{n-1}, a_n)=b_n$. Azt kaptuk tehát, hogy minden $q\in Q$- ra és $w\in T^*$- ra $f(q, w)=f'(q, w)$. Így minden A- beli állapothoz létezik vele ekvivalens A'- beli állapot. Természetesen az is igaz, hogy A' minden A- ba eső állapotához van vele ekvivalens A- beli állapot. A és A' ekvivalenciájához azt kell még megmutatnunk, hogy az A' minden $Q\times T$- beli állapotához is van az A automatának vele ekvivalens állapota. Ehhez legyen $(q, a)\in Q\times T$. Mutassuk meg, hogy A' ezen állapota ekvivalens az A automata $d(q, a)$ állapotával, azaz $f'((q, a), w)=f(d(q, a), w)$ minden $w\in T^*$ mellett fennáll. Nyilván most is elegendő nem üres szavakra szorítkoznunk. Legyen tehát $a_1, \ldots a_n\in T$.

Ekkor, az előbb kapott $f(q, a_1\cdots a_n)=f'(q, a_1\cdots a_n)$ egyenlőség fennállásának igazolásához hasonlóan bizonyítható, hogy $f(q, aa_1\cdots a_n)=f'(q, aa_1\cdots a_n)$. Viszont $f(q, aa_1\cdots a_n)=f(q, a)f(d(q, a), a_1\cdots a_n)$ és $f'(q, aa_1\cdots a_n)==f(q, a)f'(d'(q, a), a_1\cdots a_n)$ definíció szerint fennáll. d' definíciója szerint $d'(q, a)=(q, a)$, így $f'(q, aa_1\cdots a_n)=f(q, a)f'((q, a), a_1\cdots a_n)$. Másrészt f' definíciója értelmében $f'(q, a)=f(q, a)$. Mivel két kimenő szó pontosan akkor egyezik meg, ha betűről-betűre megegyezik, $f'(q, a)=f(q, a)$ és $f(q, aa_1\cdots a_n)=f'(q, aa_1\cdots a_n)$ teljesülése ekkor $f(q, aa_1\cdots a_n)=f(q, a)f(d(q, a), a_1\cdots a_n)$ és $f'(q, aa_1\cdots a_n)=f(q, a)f'((q, a), a_1\cdots a_n)$ miatt azt jelenti, hogy $f(d(q, a), a_1\cdots a_n)=f'((q, a), a_1\cdots a_n)$. Ezzel kimutattuk, hogy az A automata $d(q, a)$ állapota ekvivalens az A' automata (q, a) állapotával. Tehát, valóban, az A automata ekvivalens az A' automatával.

A tételhez azt fogjuk még igazolni, hogy A' tekinthető Moore-automatának. Legyen $g:Q'\to V$ tetszőleges olyan leképezés, melyre minden $(q, a')\in Q\times T$ esetén $g((q, a'))=f(q, a)$. ($g(q)$ értéke $q\in Q$ mellett tehát lényegtelen azon túlmenően, hogy egy rögzített V- beli elemnek kell lennie.) Ekkor tetszőleges $q\in Q$, $a\in T$ párra $f'(q, a)=f(q, a)$, ahol $(q, a)=d'(q, a)$. Mindemellett ha $(q, a')\in Q\times T$, úgy minden $a\in T$- re $f'((q, a'), a)=f(d(q, a'), a)$, ahol $(d(q, a'), a)=d'((q, a'), a)$. Tehát $g:Q\to V$ speciális megválasztásával kaptuk, hogy minden $q\in Q'$, $a\in T$ párra $f'(q, a)=g(d'(q, a))$. A' így valóban Moore-automata. (Érdekességként megjegyezzük, hogy $g(q)$ értékére $q\in Q$ esetén annyit kellett csak kikötni, hogy egy tetszőlegesen rögzített V- beli elem legyen.)

Végül, ha A véges, akkor Q, T, V rendre véges halmazok. Ekkor viszont $Q'=Q\cup Q\times T$ is véges, ami (T és V végessége mellett) azt jelenti, hogy A' véges. Ezzel a tételt teljesen bebizonyítottuk. ∎

6. *Megjegyzés.* Gill tétele lehetővé teszi, hogy bizonyos kérdésekben csupán Moore-automatákra szorítkozzunk, hiszen a tétel azt fejezi ki, hogy már Moore-automatákkal előállíthatók mindazok a leképezések, amelyek előállíthatók az általánosabb Mealy-automatákkal. Másrészt azt is meg kell jegyeznünk, hogy a Moore-automaták osztálya számos fontos konstrukcióra nézve nem zárt (állapothomomorf kép képzése, redukált automata meghatározása, stb.). Emiatt mégsem

szorítkozhatunk minden vizsgálatnál csak a Moore-automaták osztályára. Végül megjegyezzük azt is, hogy hasonló konstrukcióval igazolható Gill tétele iniciális automatákra is.

4.7. Automaták analízise és szintézise

Az automaták és az automata leképezések kapcsolatát illetően két ellentétes kérdés fogalmazható meg:

I. Ha adva van egy iniciális automata, meghatározandó az általa indukált leképezés (analízis);

II. Ha adva van egy iniciális automata leképezés, meghatározandó legalább egy olyan iniciális automata, amely ezt a leképezést indukálja (szintézis).

A szintézis feladata nem egyértelmű ugyan, de egyértelművé tehető a minimalizálás feladatával. Az analízis és szintézis feladata csak akkor fogalmazható meg egzakt módon, ha megállapodunk abban, hogy mit értünk automata és automata leképezés megadásán. Véges automatát például táblázattal adunk meg. Véges automata leképezések megadása általában problémát jelent a végtelen értelmezési tartomány miatt. Bizonyos esetekben viszont közönséges módon is lehetséges. Így például $T=\{a\}$, $V=\{b, c\}$ mellett a $\lambda \to \lambda$, $a^n \to bc^{n-1}$, $n \geq 1$ kifejezésekkel nyilván egy automata leképezést adtunk meg. Az analízis és szintézis probléma ebben a megfogalmazásban túl általános. Ezért ezt a két feladatot csak véges automatákra fogalmazzuk meg pontosabban. Ezzel kapcsolatban egy további probléma is fellép:

0. Meg kell adnunk a véges automaták által indukálható automata leképezéseknek az automatáktól független leírását. Más szóval, keresnünk kell olyan írásmódot, amelyben el tudjuk dönteni, hogy valamely, ebben az írásmódban megadott automata leképezés indukálható-e véges automatával vagy sem. Ezek után véges automatákra az analízis és szintézis problémája így fogalmazható meg:

I'. Bármely adott véges automata esetén meg kell tudnunk adni a szóban forgó írásmódban azt a leképezést, amelyet az automata indukál;

II'. Ha ebben az írásmódban meg van adva egy véges automata által indukálható leképezés, akkor meg kell tudnunk adni legalább egy olyan automatát, mely ezt a leképezést indukálja és állapothalmaza véges.

Egy olyan leképezés megadási módszer, mely eleget tesz a 0.,I.',II.' feltételeknek, először S. C. Kleene talált 1956-ban, bevezetve a reguláris kifejezés fogalmát (lásd: Reguláris kifejezések). Ennek megfelelően az automaták szintézisét és analízisét véges elfogadó automatákra tesszük meg a reguláris nyelvek fejezetben.

4.8. Egyéb véges automaták

Az automataelméleti fejezet zárásaként röviden bemutatunk néhány további véges automata fajtát.

4.8.1. Irányítható automaták

Vegyünk egy űrszondát, ami a Hold körül kering, ennek következtében időnként elveszítjük vele a kapcsolatot. A műholdat egy véges állapotú rendszernek tekintve, ha tudjuk milyen állapotban van, akkor megfelelő input jeleket küldve neki mi irányítjuk. Amikor viszont nincs kapcsolatunk vele, nincs ellenőrzésünk alatt, így nem tudjuk hogy milyen állapotba kerül. Az ilyen esetekre kínálnak megoldást az irányítható automaták.

6. Definíció. Az $A=(Q, V, d)$ automatát irányíthatónak vagy szinkronizálhatónak neveztük, ha van olyan $u \in V$ bemenő szó és $q \in Q$ állapot, hogy minden $p \in Q$ állapot esetén $d(p, u)=zq$, ahol $z \in Q^*$. Az ilyen u szót az A automata irányító vagy szinkronizáló szavának, q- t pedig az A automata irányított vagy szinkronizált állapotának nevezzük. ★

Tehát, visszatérve az űrszondánkra, mindig amikor felvesszük vele kapcsolatot először egy irányító szót küldünk neki, és ezután tudjuk vele elvégeztetni a kívánt tevékenységeket. Természetesen az

egyik kézenfekvő megoldás, ha pl. egy adott jelre (egybetűs irányítószóra) az automata irányított állapotba kerül, másrészt viszont, ha limitált ábécével kommunikálunk, nem biztos hogy érdemes egy betűt erre a célra fenntartani.

Jan Černý (ejtsd: csernyi) 1964-ben írt dolgozatában fogalmazta meg következő sejtését az irányítható automatákról.

1. Sejtés. Ha egy n állapotú automata irányítható, akkor a legrövidebb szinkronizáló szavának hossza nem több, mint $(n\text{-}1)^2$.

A Černý sejtés az automaták algebrai elméletének egyik legrégebbi megoldatlan problémája.

4.8.2. Automata több kezdőállapottal

Az üresszavas átmeneteket is megengedve könnyen belátható, hogy az automata ugyanazt a nyelvosztályt fogadja el, akkor is ha több kezdőállapotot engedünk meg. A következő módszerrel egyszerűen konstruálhatunk egy kezdőállapottal rendelkező üresszavas átmenetes véges automatát, amely ugyanazt a nyelvet fogadja el, mint az eredeti több kezdőállapottal rendelkező automatánk.

Legyen (Q, I, T, d, F) adott, ahol Q az állapotok véges halmaza, $I \subseteq Q$ a kezdőállapotok halmaza, T az input ábécé, d az átmenetfüggvény, F pedig a végállapotok halmaza. Vegyünk fel egy új q_0 állapotot, amely nem eleme a Q- nak. Legyen $Q' = Q \cup \{q_0\}$ és legyen a d' származtatva a d- ből oly módon, hogy a d átmenetein kívül legyen benne q_0- ból átmenet az üresszó hatására minden $q \in I$ állapotba. Ekkor a (Q', q_0, T, d', F) automata megfelel az állításunknak: pontosan ugyanazt a nyelvet fogadja el, mint az eredeti és pontosan egy kezdőállapottal rendelkezik.

4.8.3. Átlátszóbetűs felismerő automata

A felismerő automatának az átlátszóbetűs változatát, melynek elfogadóereje jóval túlmutat a korábban ismertetett felismerő automatáén 2010-ben Friedrich Otto és Nagy Benedek vezették be. A hagyományos automatának ebben a kiterjesztésében minden állapotra megadhatunk a T bemenő ábécének egy olyan részhalmazát, amit az automata az adott állapotban nem lát. Az inputszalagon így az automata az első nem átlátszó betűt olvassa (törli), ami így nem biztos, hogy a legelső betű. Az olvasott betűnek megfelelő átmenet utáni állapotban más betűk lehetnek átlátszóak, így lehetőség van az előző állapotban nem látott - akár a törölt betűt megelőző - betűk elolvasására is. Az automata mindig az adott állapotban látszó legelső betűt tudja olvasni, ily módon nem feltétlenül a hagyományos balról jobbra sorrendben feldolgozva az inputot.

7. Definíció. Átlátszóbetűs (nemdeterminisztikus) felismerő automatának nevezzük az $A=(Q, I, T, \$, t, d, F)$ rendezett hetest, ahol Q az állapotok véges halmaza, $I \subseteq Q$ a kezdőállapotok halmaza, T az inputábécé, $\$ \notin T$ a szalagvége jel, $t: Q \to 2^T$ átlátszósági függvény, $d: Q \times T \to 2^Q$ az állapotátmenet-függvény, $F \subseteq Q$ pedig a végállapotok halmaza. Minden $q \in Q$ állapotra a $t(q)$- ban szereplő betűk átlátszóak, az automata ebben az állapotban ezeket a betűket nem látja. ★

Az automata működése a következő: először nemdeterminisztikusan választunk egy $q \in I$ állapotot. A kezdőkonfiguráció: $(q_0, w\$)$. Legyen $w = a_1 a_2 \ldots a_n$ ahol $n \geq 1$, és $a_1, a_2, \ldots, a_n \in T$. Ekkor megkeressük az első olyan betűt balról amely nem átlátszó az adott állapotban, legyen $w = uav$ olyan, hogy $a \notin t(q_0)$, $u \in (t(q_0))^*$. Ekkor a $d(q_0, a)$- ból nemdeterminisztikusan választunk egy q' állapotot és a következő konfiguráció $(q', uv\$)$ lesz. Ha $d(q_0, a) = \emptyset$, akkor az automata megáll anélkül hogy elfogadna. Ha $w \in (t(q_0))^*$, akkor az automata a $ szimbólumot látja és megáll. Ha a gép a $ jelet látja, és az aktuális állapot végállapot, akkor elfogadja az inputot.

4.15. példa - Átlátszóbetűs elfogadó automata

Az $\{a, b, c\}$ ábécé felett azokat a szavakat tartalmazó nyelv elfogadása, amelyekben az a- k, a b- k és a c- k száma megegyezik pl. a következő átlátszóbetűs felismerő automatával megy: $(\{q_0, q_1, q_2, q_3\}, \{q_0\}, \{a, b, c\}, \$, t, d, \{q_3\})$, ahol $t(q_0) = \{b, c\}$, $t(q_1) = \{a, c\}$, $t(q_2) = \{a, b\}$, $t(q_3) = \emptyset$ és $d(q_0, a) = \{q_1\}$, $d(q_1, b) = \{q_2\}$, $d(q_2, c) = \{q_0, q_3\}$, minden más esetben a d értéke az üres halmaz. ★

A jegyzetben a későbbiekben nemvéges (de végesen definiálható, illetve leírható) automatákról is lesz szó.

4.9. Irodalmi megjegyzések

A determinisztikus véges automaták elmélete egymástól függetlenül a [Huffman 1954], [Mealy 1955] és [Moore 1956] művekben került megalapozásra. Az elfogadóautomatákkal kapcsolatos alapmunka [Rabin, Scott 1959], itt vezették be a nemdeterminisztikus véges automatákat és bizonyították ekvivalenciájukat a determinisztikus elfogadó automatákkal. Az automaták elméletével kapcsolatos részletes jegyzet magyarul a 3 kötetes [Peák 1988,1989,1990]. Az automataelmélettel kapcsolatos további részletes jegyzet [Babcsányi 2007]. Az automaták elméletének egy jelenleg is aktív kutatási iránya az automatahálózatok vizsgálata, lásd pl. [Dömösi, Nehaniv 2005]. Az átlátszóbetűs véges automatákról és ezek kiterjesztéseiről, pl. átlátszóbetűs veremautomatáról stb., és ezek tulajdonságairól részletesen olvashatunk a [Nagy, Otto 2010a, 2010b, 2010c, 2011a, 2011b] cikkekben.

5. fejezet - Reguláris nyelvek

A reguláris nyelvek alkotják a Chomsky-féle hierarchia legkisebb osztályát. Ezzel a nyelvosztállyal kapcsolatosan számos elméleti eredmény ismert, és ezek a nyelvek a gyakorlatban is számos helyen alkalmazhatóak.

Emlékeztetőül, egy nyelvtanról akkor mondjuk, hogy reguláris, ha minden egyes szabálya $A \to uB$ vagy $A \to u$ alakú (ahol $A, B \in N$, $u \in T^*$).

Elsőként azt a bárkiben felmerülő kérdést tisztázzuk, hogy mi a véges és a reguláris nyelvek viszonya.

19. Tétel. Minden véges nyelv reguláris.

Bizonyítás. Legyen $L=\{w_1, \ldots, w_n\}$ véges nyelv (megadhatjuk az elemeinek felsorolásával) valamilyen T ábécé felett, ekkor a $G=(\{S\}, T, S, \{S \to w_i \mid w_i \in L\})$ nyelvtan pont L- et generálja. Másrészt G minden szabályára igaz, hogy a jobboldalon csak terminálisok szerepel(het)nek, így G reguláris. ∎

A következő egyszerű példa azt mutatja, hogy vannak végtelen reguláris nyelvek is:

5.1. példa - Végtelen reguláris nyelv

Legyen $G=(\{S\}, \{a\}, S, \{S \to aaS, S \to aa\})$. Ekkor a generált nyelv $L=\{a^{2n} \mid n \geq 1\}$. ★

2. *Következmény.* A véges nyelvek halmaza valódi része a reguláris nyelvek halmazának.

A reguláris nyelvtanok esetén levezetéseket nagyon egyszerű, csúcscímkézett fagráfokkal tudjuk reprezentálni: A kiinduló gyökércsúcs címkéje az S mondatszimbólum. Egy A nemterminális címkéjű csúcsból, ha arra már alkalmaztunk egy $A \to a_1 \ldots a_k B$ megfelelő levezetési szabályt ($k \geq 0$ a szabályban szereplő terminálisok száma, k=0 esetben $A \to B$ a szabály alakja), k+1 él indul, és a gyerekcsúcsok címkéje balról jobbra haladva a_1, …, a_k, valamint B. Ha a levélelemek közt van nemterminális címkéjű, akkor a fa egy még be nem fejezett levezetést ábrázol, ilyenkor pontosan egy nemterminális címkéjű levélcsúcs van. Ha nincs nemterminális címkéjű levél, akkor az ábrázolt levezetés befejezett (terminál). A terminális címkéjű csúcsok levélelemei minden levezetési fának.

A mondatformát, illetve a levezetett szót megkapjuk, ha a levelek címkéit balról jobbra haladva elolvassuk.

5.1. Normálformák a reguláris nyelvtanokhoz

A Chomsky-féle nyelvosztályok definíciójában megengedtük, hogy egy lépésben bármennyi terminális bekerülhet a mondatformába, vagyis a szabályok jobb oldalának hosszára nem adtunk korlátot. Most megmutatjuk, hogy minden reguláris nyelv generálható speciálisabb formájú nyelvtanokkal is.

8. Definíció. Egy reguláris nyelvtanról azt mondjuk, hogy gyenge normálformában van, ha minden szabályára teljesül az alábbi alakok egyike: $A \to aB$, $A \to B$, $A \to a$, $A \to \lambda$ (ahol $A, B \in N$, $a \in T$). ★

20. Tétel. Minden reguláris nyelv generálható gyenge normálformájú reguláris nyelvtannal.

Bizonyítás.. Legyen adva $G=(N, T, S, H)$ reguláris nyelvtan. Ez alapján fogjuk elészíteni a $G'=(N', T, S, H')$ nyelvtant, hogy a generált nyelv ne változzon meg, de H' csak olyan alakú szabályokat tartalmazzon, amikben maximum egy terminális szerepel. Az eredeti definícióban megengedett, hogy egynél több terminális szerepeljen egy szabály jobboldalán, ezeket kell helyettesítenünk a normálformának eleget tevő szabályok sorozataival. Vegyük sorra tehát a H szabályait:

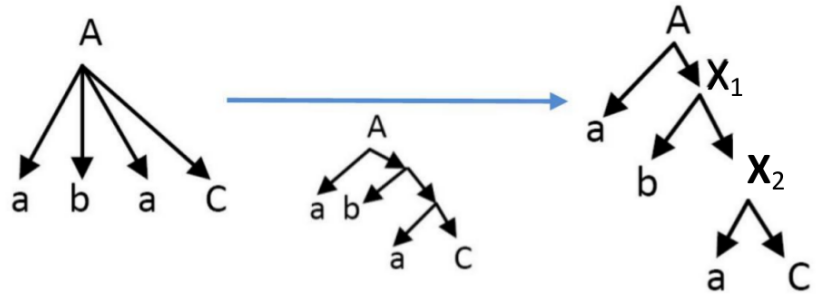
Ha az aktuális szabályra teljesül az $A \to aB$, $A \to B$, $A \to a$, $A \to \lambda$ alakok egyike, akkor őt változtatás nélkül másoljuk át H- ból, H'- be. Ha az aktuális szabályra nem teljesül a fenti alakok egyike sem, akkor a következő két eset lehetséges:

- a szabály alakja $A \to a_1 \ldots a_k$ ($k>1$, $a_1, \ldots, a_k \in T$) . Ekkor vegyük fel az N' halmazba, az eddig még nem szereplő $X_1, \ldots, X_{k-1}$ nemterminálisokat, és az aktuális szabály helyett a H'- be vegyük fel a $A \to a_1X_1$, $X_1 \to a_2X_2$, …, $X_{k-2} \to a_{k-1}X_{k-1}$, $X_{k-1} \to a_k$ szabályokat.

- a szabály alakja $A \to a_1 \ldots a_kB$ ($k>1$, $a_1, \ldots, a_k \in T$) . Ekkor vegyük fel az N' halmazba, az eddig még nem szereplő $X_1, \ldots, X_{k-1}$ nemterminálisokat, és az aktuális szabály helyett a H'- be vegyük fel a $A \to a_1X_1$, $X_1 \to a_2X_2$, …, $X_{k-1} \to a_kB$ szabályokat.

Majd tekintsük ugyanilyen eljárás során a kövtekező szabályt H- ból.

Az eljárásunk végén G' gyenge normálformájú reguláris nyelvtan lesz, továbbá benne minden újonnan bevezetett nemterminális pontosan két szabályban szerepel, egyszer a jobb oldalon, egyszer a baloldalon. Ha egy olyan szabályt alkalmazunk, aminek a jobb oldalán új (N- ben nem szereplő) nemterminális van, akkor a levezetés csak úgy folytatódhat, ha azt a szabályt alkalmazzuk, ahol ez a nemterminális van a bal oldalon. Ily módon végig kell alkalmaznunk az eredetileg alkalmazott szabályt helyettesítő lánc szabályait. Ennek alapján indukcióval belátható, hogy a G és a G' által generált nyelv megegyezik. ▮



A fenti ábrán látható, hogy a normálformára hozás tekinthető a levezetési fák olyan transzformációjának, aminek ereményeként bináris (minden csúcsban maximum ketté ágazó) fa áll elő; miközben a levélelemek száma és sorrendje nem változik.

Az előző normálforma tovább alakítható. Az $A \to \lambda$ alakú szabályok kiküszöbölhetőek az Üresszó-lemmánál bizonyított módon (Üresszó-lemma). Így elérhető, hogy vagy (λ- mentes nyelv esetén) egyáltalán nem fordul elő olyan szabály, aminek jobboldala az üresszó, vagy (ha a nyelvben szerepel az üresszó, akkor) csak az $S \to \lambda$ szabályban fordul elő, miközben S nem szerepel egyik szabály jobb oldalán sem (ahol S a mondatszimbólum).

Tekintsük most a nyelv üresszómentes részét, ennek megfelelően nem használunk $A \to \lambda$ alakú szabályt egyáltalán. Megmutatjuk, hogy az $A \to B$ alakú, úgynevezett láncszabályoktól (vagy nemterminális átnevezésektől) is meg tudunk szabadulni.

9. Definíció. Egy reguláris nyelvtanról azt mondjuk, hogy erős normálformában van, ha minden szabályára teljesül az alábbi alakok egyike: $A \to aB$, $A \to a$ (ahol $A, B \in N$, $a \in T$). ★

21. Tétel. Minden reguláris nyelvhez van vele ekvivalens erős normálformájú reguláris nyelvtan.

Bizonyítás. Az előzőek alapján feltehetjük, hogy $G=(N, T, S, H)$ nyelvtanunkban csak $A \to aB$, $A \to a$ és $A \to B$ alakú szabályok vannak ($A, B \in N$, $a \in T$) . Definiáljuk minden $A \in N$ változóhoz a következő halmazokat:

- $U_1(A)=\{A\}$.

- $U_{i+1}(A)=U_i(A) \cup \{B \in N \mid \exists C \in U_i(A) \mid C \to B \in H\}$, ha $i>1$.

Ekkor N véges volta miatt létezik olyan minimális k index, hogy $U_k(A)=U_{k+j}(A)$, ha $j=1, 2, \ldots$. Jelöljük minden $A \in N$ nemterminális jelhez tartozó $U_k(A)$- t $U(A)$- val. Ekkor $U(A)$ éppen azokat a

nemterminálisokat tartalmazza, amelyek levezethetőek az A-ból csupán láncszabályokat felhasználva, vagyis tetszőleges A, $B \in N$ változókra $A \Rightarrow^* B$ pontosan akkor, ha $B \in U(A)$. Ezek után definiáljuk a H' szabályhalmazt a következőképpen:

$H' = \{A \to r \mid$ ha van olyan $B \in N$ hogy $B \to r \in H$, $r \notin N$, $B \in U(A)\}$.

Ekkor $G' = (N, T, S, H')$ nyelvtan erős normálformájú, és ekvivalens az eredetivel. ∎

Itt jegyezzük meg, hogy a szakirodalomban előfordul, hogy a reguláris nyelvtanokat eleve csak az imént ismertetett normálformájú nyelvtanokkal definiálják; az általunk reguláris nyelvtanként definiált nyelvtanokat pedig jobblineáris nyelvtannak hívják. Ahogy láttuk ezek a nyelvtantípusok ekvivalensek egymással, így mi továbbra sem fogunk köztük különbséget tenni.

A ballineáris nyelvtanok és a jobblineáris nyelvtanok ekvivalenciáját a lineáris nyelvtanoknál tárgyaljuk (lásd Bal- és jobb-lineáris nyelvtanok ekvivalenciája [129]).

5.2. Reguláris kifejezések

Legyen $T = \{a_1, \ldots, a_n\}$ tetszőleges nem üres és véges halmaz. Ha a T ábécét kibővítjük a benne nem szereplő $\emptyset$, λ, $+$, $\cdot$, $*$, $(,)$ jelekkel: $V = T \cup \{\emptyset, \lambda, +, \cdot, *, (,)\}$, akkor a V ábécé felett értelmezhetjük a reguláris kifejezések halmazát a következő induktív definícióval:

Elemi kifejezés:

- $\emptyset$ reguláris kifejezés,

- λ reguláris kifejezés,

- minden $a \in T$ reguláris kifejezés.

Indukciós lépések:

- ha p és r reguláris kifejezések, akkor $(p+r)$ is az,

- ha p és r reguláris kifejezések, akkor $(p \cdot r)$ is az,

- ha r reguláris kifejezés, akkor r^* is az.

Továbbá, minden reguláris kifejezés előáll az elemi kifejezésekből az indukciós lépések véges sokszori alkalmazásával.

A konkatenáció $\cdot$ műveleti jelét sokszor, ahogy eddig is, elhagyjuk.

A reguláris kifejezések segítségével nyelveket írhatunk le:

az elemi kifejezésekkel elemi nyelveket:

- $\emptyset$ (üres halmaz) jelöli az $\{\}$ *üres nyelv*et,

- λ jelöli a $\{\lambda\}$ *kezdő nyelv*et,

- a jelöli az $\{a\}$ egyszavas *alapnyelv*et.

Az indukciós lépéssel pedig:

- $(p+r)$ jelöli az $L_p \cup L_r$ nyelvet,

- $(p \cdot r)$ jelöli az $L_p L_r$ nyelvet,

- r^* jelöli az L_r^* nyelvet,

ahol L_p és L_r a p és az r reguláris kifejezések által jelölt nyelvek.

Két reguláris kifejezést ekvivalensnek nevezünk, ha ugyanazt a nyelvet írják le.

A T feletti nyelvek közt, tehát, a következő műveleteket (*reguláris műveleteknek*) hívjuk:

(i) összedás: nem más mint a két nyelv halmazelméleti uniója;

(ii) szorzás: a két nyelv konkatenációja;

(iii) iteráció: L *iteráltján* vagy más néven *lezártján* értjük és L^*- al jelöljük az $L^*=\{p_1p_2\cdots p_k|p_1, \ldots, p_k\in L, k\geq 1\}\cup\{\lambda\}$ nyelvet. Vagy ahogy már korábban is definiáltuk, $L^*=\bigcup_{i=0}^{\infty} L^i$, ahol $L^0=\{\lambda\}$, $L^i=LL^{i-1}$.

Az $L_T=\{L|L\subseteq T^*\}$ halmazt a rajta értelmezett ezen három művelettel T *feletti nyelvalgebrának* hívjuk. A nyelvalgebrában érvényesek az absztrakt algebrából ismert, vagy azokhoz hasonló műveleti tulajdonságok:

$L_1\cup L_2=L_2\cup L_1$ (összeadás kommutatívitása);

$(L_1\cup L_2)\cup L_3=L_1\cup(L_2\cup L_3)$ (összeadás asszociativitása);

$L\cup\varnothing=L$ (additív egységelem létezése);

$(L_1L_2)L_3=L_1(L_2L_3)$ (szorzás asszociativitása);

$L\{\lambda\}=\{\lambda\}L=L$ (multiplikatív egységelem létezése);

$(L_1\cup L_2)L_3=L_1L_3\cup L_2L_3$ (baloldali disztributivitás);

$L_1(L_2\cup L_3)=L_1L_2\cup L_1L_3$ (jobboldali diszributivitás);

$\{\lambda\}^*=\{\lambda\}$;

$\varnothing^*=\{\lambda\}$;

$$L^*=\left(L^k\right)^*\left(\bigcup_{i=1}^{k-1}L^i\right)\quad k\geq 1$$

$LL^*=L^*L$;

$L^*=\{\lambda\}\cup LL^*$;

$(L_1\cup L_2)^*=(L_1^*L_2^*)^*$ (unió kiváltása).

Két kivétellel minden felsorolt azonosság egyszerű számítással adódik a definíciók alapján. Ezért csak ezt a két összefüggést fogjuk bizonyítani.

Mutassuk meg először, hogy $L^*=\left(L^k\right)^*\left(\bigcup_{i=1}^{k-1}L^i\right)\quad k\geq 1$. Ehhez azt kell belátnunk, hogy a baloldali halmaznak pontosan azok az elemei mint a jobboldalinak.

A baloldali halmaznak és a jobboldali halmaznak is eleme az üresszó, így csak az üresszótól különböző elemek vizsgálatára kell szorítkoznunk. A baloldal az üres szón kívül tartalmazza azokat a p szavakat, melyek előállnak valamely $n\geq 1$- re $p=p_1\cdots p_n$ alakban, ahol $p_1, \ldots, p_n\in L$. Ha minden tényező az üresszó, akkor $p=\lambda$, s ezzel az esettel már nem kell tovább foglalkoznunk. Tehát feltehető, hogy van olyan $1\leq i\leq n$, hogy $p_i\neq\lambda$, azaz $p\neq\lambda$. Legyen valamely $u\geq 0$ nemnegatív egészre $n=uk+v$, ahol

$0 \le v<k$. Ekkor $p=q_1q_2$, ahol vagy $q_1=\lambda$ (ekkor $u=0$) vagy pedig $q_1=p_1\cdots p_{uk}$, ahol p_1, …, $p_{uk}\in L$ (ha $u\neq 0$), s ugyanekkor vagy $q_2=\lambda$ (ekkor $v=0$), vagy $q_2=p_{uk+1}\cdots p_{uk+v}$, ahol p_{uk+1}, …, $p_{uk+v}\in L$ (ha $v\neq 0$). Mindenképp fennáll, hogy $q_1\in(L^k)^*$, illetve $q_2\in L^v$, $0 \le v<k$, azaz $p=q_1q_2\in(L^k)^*L^v$. Tehát p eleme mindkét oldalnak, amivel a két oldal egyenlőségét kimutattuk.

Igazoljuk most az $(L_1\cup L_2)^*=(L_1^*L_2^*)^*$ egyenlőséget. Itt is igaz, hogy a baloldali halmaznak és a jobboldali halmaznak is eleme az üresszó, így csak az üresszótól különböző elemek vizsgálatára kell szorítkoznunk. Hasonló a helyzet mint az előző esetben. Mindkét oldal az üresszón kívül pontosan azokat a p szavakat tartalmazza, melyek előállnak valamely $n \ge 1$- re $p=p_1\cdots p_n$ alakban, ahol p_1, …, $p_n\in L_1\cup L_2$. A két halmaz tehát egyenlő.

A reguláris kifejezésekben a fent említett nyelvazonosságok miatt (pl. asszociativitás) zárójeleket hagyhatunk el a leírt nyelv egyértelműségét megtartva. További zárójeleket hagyhatunk el, ha megállapodunk abban a szokásos precedenciában, hogy a * művelet (ahogy a hatványozás szokott lenni) a legerősebb, majd következik a szorzás (konkatenáció) és az összeadás (unió) a leggyengébb.

A reguláris kifejezésekkel a reguláris nyelvek egy alternatív definícióját adhatjuk meg.

Reguláris nyelvnek hívjuk az üres nyelvet, továbbá mindazon nyelveket, amelyek elemi nyelvekből a reguláris műveletek véges számú alkalmazásával előállíthatók. (Így például reguláris nyelvek: $\emptyset$, $\{\lambda\}$, $\{a_1\}$, T^*, $(\{a_1\}\{a_2\})^*\cup\{a_3\}$.) Később igazolni fogjuk, hogy a reguláris nyelv fogalmának korábbi nyelvtannal történő definíciója ezzel a fogalommal ekvivalens. A reguláris kifejezések tehát megmutatják, hogy a reguláris nyelv hogyan áll elő az elemi nyelvek és az alapműveletek segítségével. Egy reguláris kifejezés egyértelműen definiál egy nyelvet, fordítva viszont egy reguláris nyelvet leíró reguláris kifejezés általában nem egyértelműen meghatározott. (Például $L\{\lambda\}=\emptyset^*L=(\{\lambda\}+\{\lambda\})L=\cdots$.) Ezért külön érdekes, hogy hogyan lehet eldönteni az F_T nyelvalgebrában, hogy két különböző reguláris kifejezés ugyanazt a reguláris nyelvet állítja-e elő. Később látni fogjuk, hogy ez a kérdés eldönthető. Sőt, ha pontosan megmondjuk, hogy mit is értünk egy reguláris nyelvet megadó minimális reguláris kifejezésen (például lehet ez alatt érteni (a karakterszámra) a legrövidebbet), akkor annak meghatározására is megadható algoritmus.

Itt jegyezzük meg, hogy az unió és konkatenáció műveletek tartják a végességet, vagyis ha * művelet nem szerepel egy reguláris kifejezésben, akkor az általa leírt nyelv véges. Ily módon éppen a véges nyelvek egy jellemzését adhatjuk meg, hiszen minden véges nyelv megadható a szavai között + jelekkel, mint reguláris kifejezéssel megadott nyelv. A következőkben néha nem teszünk különbséget egy reguláris kifejezés és az általa leírt nyelv között, vagyis magára a nyelvre is hivatkozunk a reguláris kifejezéssel. Amennyiben nem okoz félreértést, az unió jelet ($\cup$), illetve az összeadás jelet (+), is használhatjuk ugyanabban a szerepben. A következő alfejezetben ugyancsak speciális reguláris kifejezéseket fogunk megvizsgálni.

5.2.1. Reguláris kifejezések ekvivalenciája

A reguláris kifejezések között vannak ekvivalensek, vagyis ugyanazt a halmazt (reguláris nyelvet) általában több, egymástól különböző reguláris kifejezés is megadja. Ez alapján az ekvivalencia reláció alapján az ekvivalens formulákat felhasználhatjuk formulák átalakítására.

Néhány ilyen ekvivalencia például:

- $(p\cup q)$ kifejezés jelentése ugyanaz, mint $(q\cup p)$ kifejezésé;
- $(p\cup q)\cup r$ kifejezés jelentése ugyanaz, mint $p\cup(q\cup r)$ kifejezésé;
- $(p\cup q)(r\cup t)$ kifejezés ugyanazt jelenti, mint a $(pr\cup pt\cup qr\cup qt)$;
- rr^* pedig ekvivalens r^*r kifejezéssel.

Az unió művelet kommutativitása és asszociativitása miatt zárójeleket hagyhatunk el, és tekinthetjük az unió műveletet akár kettőnél több argumentumúnak is.

5.2. példa - Nyelv megadása reguláris kifejezéssel 1. feladat

Adjuk meg reguláris kifejezéssel azt a nyelvet a {0,1} ábécé felett, amely azon szavakból áll, amelyek tartalmazzák részszóként a 010 szót!

Megoldás: $L=(0+1)^*010(0+1)^*$ ★

5.3. példa - Nyelv megadása reguláris kifejezéssel 2. feladat

Adjuk meg reguláris kifejezéssel azt a nyelvet a {0,1} ábécé felett, amely azon szavakból áll, amelyek tartalmazzák részszóként a 000 vagy az 111 szót!

Megoldás: $L=(0+1)^*(000+111)(0+1)^*$ ★

5.4. példa - Nyelv megadása reguláris kifejezéssel 3. feladat

Adjuk meg reguláris kifejezéssel azt a nyelvet a {0,1} ábécé felett, amely azon 1-esre végződő szavakból áll, amelyek nem tartalmazzák részszóként a 00 szót!

Megoldás: $L=(1+01)^*$ ★

5.5. példa - Nyelv megadása reguláris kifejezéssel 4. feladat

Adjuk meg reguláris kifejezéssel azt a nyelvet a {0,1} ábécé felett, amely azon szavakból áll, melynek 3. betűje 0!

Megoldás: $L=(00+01+10+11)0(0+1)^*$ ★

5.6. példa - Nyelv megadása reguláris kifejezéssel 5. feladat

Adjuk meg reguláris kifejezéssel azt a nyelvet a {0,1} ábécé felett, amely azon szavakból áll, melyek tartalmaznak legalább három 1-est!

Megoldás: $L=(0+1)^*1(0+1)^*1(0+1)^*1(0+1)^*$ ★

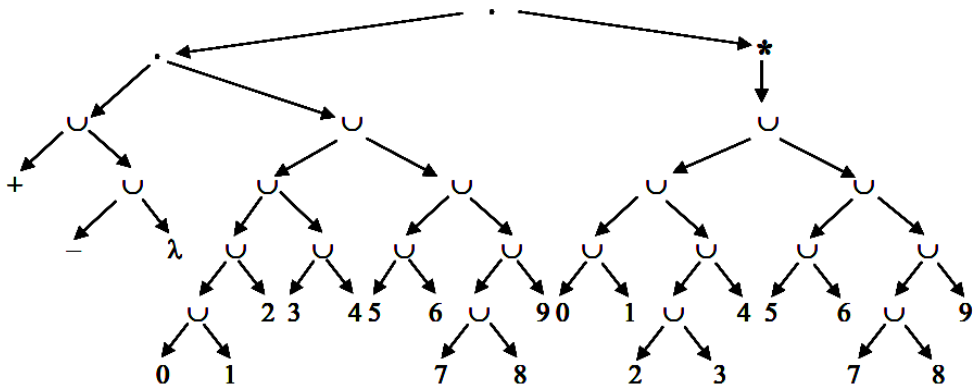
5.7. példa - Nyelv megadása reguláris kifejezéssel 6. feladat

Adjuk meg reguláris kifejezéssel azt a nyelvet a {0,1} ábécé felett, amely azon szavakból áll, melyek 5-tel osztható 1-est tartalmaznak!

Megoldás: $L=(0^*10^*10^*10^*10^*)^*$ ★

5.2.2. A kifejezésfa

A kifejezéseket, illetve a formulákat, amik atomokból (tovább nem bontható egységek), illetve műveletekből állnak szokás fa formájában is ábrázolni. A fa levélelemeiben az atomok (itt most a terminálisok és az üresszó) a többi csomópontban pedig a műveletek szerepelnek. Minden műveletnek megfelelő csúcsból pontosan annyi darab él indul ki a további részkifejezésekhez, ahány argumentumú az operátor. Példaként az egész számokat leíró reguláris kifejezés fa alakban:



5.2.3. Unió-normálforma reguláris kifejezésekhez

A normálformák fontos szerepet játszanak a számítástudomány sok területén, pl. a logikában a konjunktív-, diszjunktív- normálformákról, illetve prenex alakú formulákról beszélhetünk. Ezekben a formulákban a műveletek sorrendjére van valamilyen megszorításunk. Lényeges viszont az, hogy minden formulához létezik vele ekvivalens amely normálformában van. Normálformát értelmezhetünk a reguláris kifejezésekre is.

Egy reguláris kifejezésről akkor mondjuk, hogy *unió-normálformában van*, ha a kifejezésfájában unió művelet csak a fa gyökerében szerepelhet (megengedve bármekkora aritású unió műveletet).

Ekkor igaz a következő eredmény:

A következő ekvivalenciák véges sokszori alkalmazásával bármely (reguláris) kifejezés normálformára hozható:

(1) $(p \cup r)^* \rightarrow (p^* r^*)^*$,

(2) $p(q \cup r) \rightarrow pq \cup pr$,

(3) $(p \cup q)r \rightarrow pr \cup qr$,

(4) $(p \cup q)(r \cup t) \rightarrow pr \cup pt \cup qr \cup qt$.

Tehát egy unió-normálformájú kifejezés véges sok uniómentes kifejezés uniója (a normálformát 2004-ben vezette be Nagy Benedek).

Lássuk végül, hogyan néz ki a tizes számrendszerben felírt egész számokat leíró reguláris kifejezés normálformája. Mivel a kifejezés meglehetősen hosszú, bevezetünk egy rövidítést:

$S \equiv (0^*1^*2^*3^*4^*5^*6^*7^*8^*9^*)^*$.

Így a harminc tagú unió (hogy a + előjelet ne keverjük a reguláris unió művelettel, ez utóbbit itt ∪-val jelöljük):

$0S \cup 1S \cup 2S \cup 3S \cup 4S \cup 5S \cup 6S \cup 7S \cup 8S \cup 9S \cup$

$+0S \cup +1S \cup +2S \cup +3S \cup +4S \cup +5S \cup +6S \cup +7S \cup +8S \cup +9S \cup$

-0S∪-1S∪-2S∪-3S∪-4S∪-5S∪-6S∪-7S∪-8S∪-9S

5.8. példa - Reguláris kifejezés unió-normálformára alakítása - 1. feladat

$(a+b)(c+d+e) \equiv ac+ad+ae+bc+bd+be$ ★

5.9. példa - Reguláris kifejezés unió-normálformára alakítása - 2. feladat

$((ab+c^*)d^*)^* \equiv (abd^*+c^*d^*)^* \equiv ((abd^*)^*(c^*d^*)^*)^*$

ez már uniómentes, de egyszerűsíthető: $((ab)^*c^*d^*)^*$ ★

5.10. példa - Reguláris kifejezés unió-normálformára alakítása - 3. feladat

$(a^*+b)^*(c+d)^* \equiv ((a^*)^*b^*)^*(c^*d^*)^*) \equiv (a^*b^*)^*(c^*d^*)^*$ ★

5.11. példa - Reguláris kifejezés unió-normálformára alakítása - 4. feladat

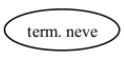
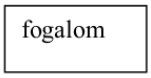
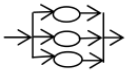
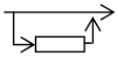
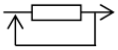
$(a+bab)(bb+ababa)^* \equiv (a+bab)((bb)^*(ababa)^*)^* \equiv a((bb)^*(ababa)^*)^*+bab((bb)^*(ababa)^*)^*$ ★

5.12. példa - Reguláris kifejezés unió-normálformára alakítása - 5. feladat

$(p+m+\lambda)(0+1)(0+1)^* \equiv (p0+m0+0+p1+m1+1)(0+1)^* \equiv (p0+m0+0+p1+m1+1)(0^*1^*)^* \equiv p0(0^*1^*)^*+m0(0^*1^*)^*+0(0^*1^*)^*+p1(0^*1^*)^*+m1(0^*1^*)^*+1(0^*1^*)^*$ ★

5.3. Egyszerű szintaxis gráfok

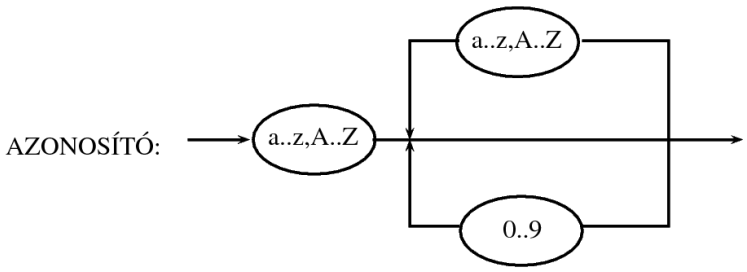
A szintaxis gráfokat pl. programnyelvek egységeinek szintaktikai leírására használhatjuk. Ez a megadási mód a Pascal nyelvvel terjedt el igazán, a grafikus kép miatt iskolások és felnőttek is könnyedén sajátították el segítségével a programozás alapjait. Több elnevezése is ismert, pl. a vasúthálózat diagram név a diagramok alakja alapján találó. A terminálisokat körökkel jelöljük, beírva őket a körbe.

Terminális	Nemterminális	Konkatenáció	Alternatíva	Opció	Iteráció
					

Használható műveletek:

- konkatenáció (összefűzés: több szövegelem egymás mellé/után írása),
- alternatíva (választás: különböző lehetőségek közül egy kiválasztása),
- opció (a szövegelem vagy megjelenik vagy nem),
- iteráció (a szövegelem akárhányszor megjelenhet (általában a nullaszori megjelenést is beleértjük)).

5.13. példa - Azonosítók nyelvének megadása szintaxis gráffal

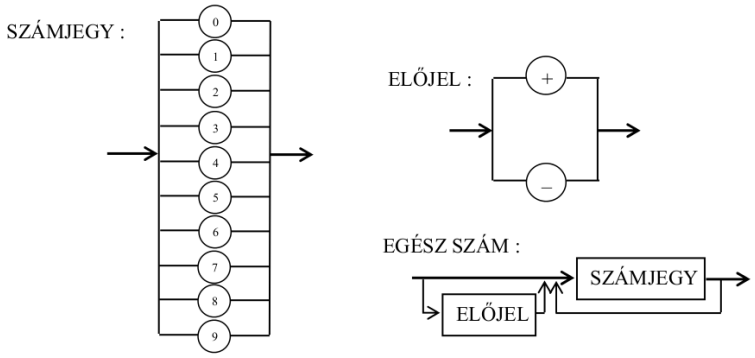


★

Egy szintaxis gráfban mindig van pont egy indulóél és egy érkezőél, ahonnan indulva és ahova érkezve kell egy utat bejárnunk a gráfban. Ha az út során összeolvassuk a terminálisokat, akkor egy szót kapunk. Az összes olyan szó, amelyet megkaphatunk az indulóéltől az érkezőélig valamely úton végighaladva adja a gráf által leírt nyelvet.

A rövidebb és áttekinthetőbb leírás kedvvért bevezethetjük a nemterminálisokat, amik egyszerű szintaxisgráf részeket rövidítenek. Azt is megengedhetjük, hogy ezekben a nemterminális rövidítésekben szerepljenek már korábban definiált nemterminálsiok.

5.14. példa - Tizes számrendszerbeli egész számok nyelvének megadása szintaxis gráffal



★

Az előző példa alapján a -000 szó is egész szám (eleme a nyelvnek), hiszen egy számítógépes programnyelv általában ezt is ugyanúgy érti, mint a 0 szót. Az iskolában tanultaknak megfelelően viszont egy ember számára a -000 szám hibásnak tűnik.

5.15. példa - Egész számok köznapi értelemben vett leírása - Gyakorló feladat

Adjuk meg azt a szintaxis gráfot, amely az emberek számára hétköznapi értelemben vett egész számok leírását adja! ★

Az itt ismertetett szintaxis gráfokkal pontosan a reguláris nyelveket tudjuk leírni, ha a nemterminálisokat nem használjuk. A kifejező erő ugyanennyi, ha a nemterminálisok definíciójában csak a már korábban ugyanígy megadott nemterminálisokat használhatjuk fel. Ekkor ugyanis a szereplő nemterminálisok helyettesíthetőek a definíciójukban megadott leírással, ami ha tartalmaz nem terminálist, akkor az ott szereplő nemterminálisok is helyettesíthetőek stb., amíg végül visszakaphatjuk az eredeti nemterminális mentes definíciót.

5.4. Véges elfogadó automaták (Rabin-Scott automaták)

E fejezetben megmutatjuk, hogy a 3-as típusú, azaz reguláris nyelvtanokkal generálható nyelvek osztálya megegyezik a véges automaták által felismerhető nyelvek osztályával. Más szóval, a 3-as típusú, azaz reguláris nyelvtan, mint generatív eszköz, azonos értékű a véges automatával, mint felismerő eszközzel.

Legyen $FA=(Q, T, q_0, d, F)$ véges felismerő automata (FA, az angol finite automaton alapján), ahol Q az állapotok véges nemüres halmaza, T a bemenő- (vagy szalag-) ábécé, $q_0 \in Q$ a kezdőállapot, d az állapot átmenet függvény, $F \subseteq Q$ pedig a végállapotok vagy elfogadóállapotok halmaza.

A d leképezés alakja alapján beszélhetünk

- nemdeterminisztikus üresszó átmenetet is megengedő automatáról: $d:Q\times(T\cup\{\lambda\}) \rightarrow 2^Q$,

- nemdeterminisztikus üresszó átmenet nélküli automatáról: d:$Q\times T \rightarrow 2^Q$,

- (parciális) determinisztikus automatáról: d:$Q\times T \rightarrow Q$ (parciális függvény),

- teljesen definiált determinisztikus automatáról: d:$Q\times T \rightarrow Q$ (teljesen definiált függvény).

Amint azt a 4. fejezetben (Nemdeterminisztikus és determinisztikus automata és Rabin-Scott automata alfejezetekben) már láttuk, a *nemdeterminisztikus véges automata* olyan $A=(Q, T, q_0, d)$ rendszer, amely hasonlóan működik, mint az iniciális kimenő jel nélküli (determinisztikus) automata, azzal a különbséggel, hogy a d átmeneti függvény a $Q\times T$ szorzathalmazt a Q állapothalmaz részhalmazainak halmazába, vagyis a 2^Q hatványhalmazba képezi le: d:$Q\times T \rightarrow 2^Q$. Minden $q\in Q$, $a\in T$ párra $d(q, a)\subseteq Q$ az átmeneti lehetőségek halmaza. Feltételezzük viszont, hogy az A egy $q\in Q$ állapotból az $a\in T$ jel hatására mindig egy állapotba megy át (kivéve ha $d(q, a)=\emptyset$, amikor az átmenet nincs értelmezve), az átmenet azonban A által nincs egyértelműen meghatározva. Továbbá, az üresszó hatására is történhet átmenet, ilyenkor a szalag olvasása nélkül is állapotot válthat az automata. Tehát az automata a diszkrét időskála mentén minden pillanatban egy jól meghatározott állapotban van, egy lépés során olvas (vagy λ- átmenet esetén nem olvas) az input szalagról egy betűt, és ennek hatására állapotot vált (vagy nem vált).

5.16. példa - Egy nemdeterminisztikus véges automata

A	p	q	r
a	{ q, p }	$\emptyset$	{ p }
b	{ r }	{ q, r }	{ r }

A fenti táblázattal definiált A nemdeterminisztikus véges automata például az a bemenő jel hatására a p állapotból átmehet q- be, de maradhat a p állapotban is. A q állapotban az A az a bemenő jelet nem képes feldolgozni. ★

Az automata egy futásának nevezzük azt az állapotsorozatot, amin egy adott input elolvasása közben végigmehet.

Üresszó átmenetet is megengedő automata esetén az automatát megadó táblázatban a terminálisok mellett az üresszó is egy külön sorban szerepel, ahol a $d(q, \lambda)$ átmenetek szerepelnek megfelelő q állapotok esetén. Gráfokkal hasonlóan adhatjuk meg ezeket az automatákat is azzal a különbséggel, hogy az állapotátmeneteket jelző nyilakon a terminálisok mellett a λ is szerepelhet.

A d függvény értelmezését kiterjesztjük, így definiáljuk a d^* kiterjesztett átmenetfüggvényt. Ehhez először minden állapothoz megadjuk annak λ- lezártját, vagyis azt az állapothalmazt, amit az adott állapotból bemenőjel nélkül is elérhetünk. Formálisan: $l(q_j)$- t definiáljuk rekurzívan tetszőleges q_j állapotra:

$l_0(q_j)=\{q_j\}$;

$l_{i+1}(q_j)=l_i\cup\{q_k \mid q_k\in d(q_m, \lambda)$, valamely $q_m\in l_i(q_j)\}$, ha $i \geq 0$

ekkor a Q végessége miatt lesz olyan i, hogy $l_i(q_j)=l_{i+k}(q_j)$ minden k természetes számra; ekkor $l(q_j)=l_i(q_j)$ halmazt a q_j állapot λ- lezártjának nevezzük.

A λ- lezárt fogalmát értelmezhetjük nemcsak állapotokra, de állapot halmazokra is: $l(Q')=\{q_k \mid q_k\in l(q)$ valamely $q\in Q'$ esetén $\}$.

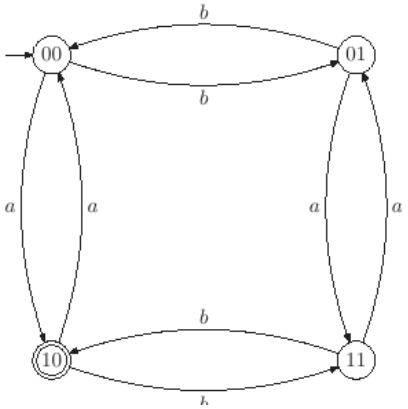
Ekkor a kiterjesztett átmenetfüggvényt a következőképpen definiáljuk: d^*:$Q\times T^* \rightarrow 2^Q$, ahol minden $q\in Q$- ra $d^*(q, \lambda)=l(q)$ és minden $a\in T$, $w\in T^*$ párra $d^*(q, wa)=l\left(\bigcup_{p\in d^*(q,w)} d(p, a)\right)$. (Természetesen előfordulhat, hogy $d(q, wa)=\emptyset$.) Vegyük észre, hogy ez a kiterjesztés lényegesen különbözik az automataelméleti vizsgálatoknál korábban alkalmazott kiterjesztéstől, hisz ott képelemként $2^{(Q^+)}$- beli elemek léptek fel, míg itt csak 2^Q- beliek.

Egy véges automata akkor fogad el egy $w \in T^*$ input szót, ha van olyan futása amely a w elolvasása után végállapotba ér, vagyis $d^*(q_0, w) \cap F \neq \emptyset$. Egy automata által elfogadott szavak halmaza jelenti az automata által elfogadott nyelvet.

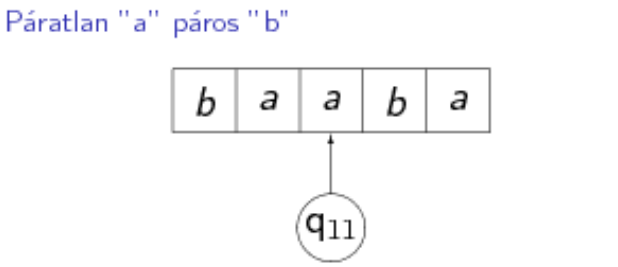
Ez alapján azt is mondhatjuk, hogy az L nyelv előállítható vagy felismerhető az A automatában az $F \subseteq Q$ állapot halmazzal, jelekben: $L(A) = L_A^F$, ha $w \in L \Leftrightarrow d^*(q_0, w) \cap F \neq \emptyset$.

5.17. példa - A páratlan *a*-ból és páros *b*-ből álló szavak nyelve

Az alábbi ábrán olyan automatát láthatunk, mely azokat a szavakat fogadja el, melyek páratlan számú a- t és páros számú b- t tartalmaznak.



Az automata működés közben:



A véges automata azokat a szavakat fogadja el, melyekben páratlan számú "a" és páros számú "b" szerepel.

$$A = (\{q_{00}, q_{01}, q_{10}, q_{11}\}, \{a, b\}, q_{00}, \delta, \{q_{10}\}),$$

$$\delta(q_{00}, a) = q_{10}, \quad \delta(q_{00}, b) = q_{01},$$
$$\delta(q_{01}, a) = q_{11}, \quad \delta(q_{01}, b) = q_{00},$$
$$\delta(q_{10}, a) = q_{00}, \quad \delta(q_{10}, b) = q_{11},$$
$$\delta(q_{11}, a) = q_{01}, \quad \delta(q_{11}, b) = q_{10}.$$

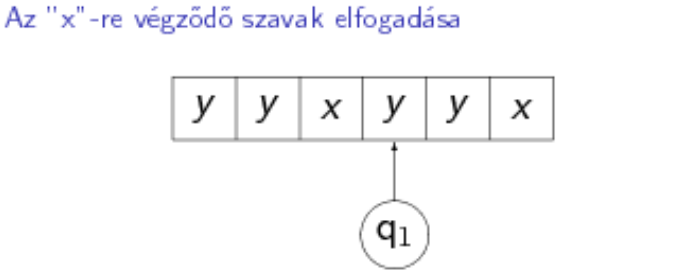
★

5.18. példa - Az *x*-re végződő szavak nyelve

Az alábbi ábrákon látható egy nemdeterminisztikus automata működése, mely az x- re végződő szavakat fogadja el.

Most lássunk néhány példát nemdeterminisztikus automata működésére.

Az automata nemdeterminisztikus működéssel olyan állapotba jut és olyan szimbólumot olvas, melyekre az állapotfüggvény üres:

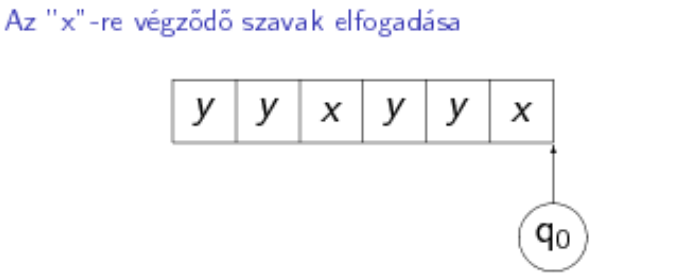


A nemdeterminisztikus véges automata azokat a szavakat fogadja el, melyek "x"-re végződnek.

$$A = (\{q_0, q_1\}, \{x, y\}, q_0, \delta, \{q_1\}),$$

$\delta(q_0, x) = q_0$,
$\delta(q_0, x) = q_1$,
$\delta(q_0, y) = q_0$.

Az automata nemdeterminisztikus működés során végigolvassa az inputot, de a működést nem elfogadó végállapotban fejezi be:

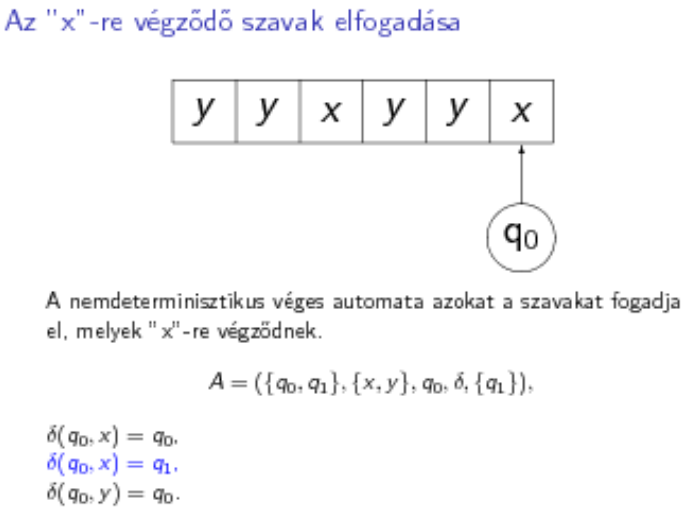


A nemdeterminisztikus véges automata azokat a szavakat fogadja el, melyek "x"-re végződnek.

$$A = (\{q_0, q_1\}, \{x, y\}, q_0, \delta, \{q_1\}),$$

$\delta(q_0, x) = q_0$,
$\delta(q_0, x) = q_1$,
$\delta(q_0, y) = q_0$.

Az automata működése során végigolvassa az inputot és végállapottal elfogadja azt:



★

5.4.1. Véges determinisztikus és nemdeterminisztikus automaták ekvivalenciája

Két automatát ekvivalensnek hívunk, ha az általuk elfogadott nyelvek megegyeznek. Habár az imént bemutatott négy automatafogalom elég különbözőnek tűnhet most megmutatjuk, hogy az általuk elfogadott nyelvek osztálya megegyezik. Egyrészt világos, hogy a bemutatott sorrendben egyre speciálisabbak az automaták, vagyis definíció szerint a nemdeterminisztikus üresszó átmenet nélküli automaták, tulajdonképpen a nemdeterminisztikus üresszó átmenetet is megengedő automaták speciális esetei, amikben üresszó átmenet nem fordul elő. Továbbá a (parciális) determinisztikus automaták a nemdeterminisztikus üresszó átmenet nélküli automaták olyan speciális esetei ahol a d függvény képhalmaza maximum egyelemű halmazokat tartalmaz. A teljesen definiált determinisztikus automaták viszont olyan speciális parciális determinisztikus automaták, amelyekben a d értéke mindig pontosan egyelemű halmaz.

Ezek alapján világos, hogy a teljesen definiált determinisztikus automaták által elfogadott nyelvek osztálya részhalmaza a parciális determinisztikus automaták által elfogadott nyelvek halmazának, amely részhalmaza a nemdeterminisztikus üresszó átmenet nélküli automatákkal elfogadott nyelvek osztályának, az viszont részhalmaza a nemdeterminisztikus üresszó átmenetet is megengedő automaták által elfogadott nyelvek halmazának. Azt, hogy itt nem valódi részhalmaz relációkról van szó a következő tétel, illetve annak bizonyításában szereplő konstrukcióval látjuk be.

22. Tétel. Minden nemdeterminisztikus üresszó átmenetet is megengedő véges automatához van vele ekvivalens teljesen definiált determinisztikus automata.

Bizonyítás. Legyen $NFA=(Q, T, q_0, d, F)$ egy nemdeterminisztikus üresszó átmenetet is megengedő véges automata. Konstruáljuk meg a $DFA=(2^Q, T, l(q_0), d', F')$ véges automatát, ahol az állapotok 2^Q halmaza az eredeti automata állapotainak a lehetséges részhalmazaiból áll. A d' állapotátmenet-függvény definíciója pedig $d'(p, a) = l\left(\bigcup_{q \in p} d(l(q), a)\right)$, ahol $p \in 2^Q$, vagyis $p \subseteq Q$ és $a \in T$. Az új automata végállapotai pedig $F' = \{p \mid$ van olyan $q \in p$, hogy $q \in F\}$. Világos, hogy DFA teljesen definiált determinisztikus véges automata.

Másrészt belátható, hogy $L(DFA)=L(NFA)$, hiszen az eredeti NFA automata minden elfogadó futásához tartozik az új DFA automatának egy elfogadó futása, és viszont. ∎

Az előző tételben megkonstruált DFA tartalmazhat olyan állapotokat, amelyek a kezdőállapotból nem érhetőek el. Például csak olyan $p \in 2^Q$ állapotok érhetőek el melyekre $p=l(p)$ teljesül.

Ez alapján, egy determinisztikus automata megkonstruálásakor akkor járunk el észszerűen, ha az $l(q_0)$-nak megfelelő állapotból indulunk el, és csak azokat az állapotokat (vagyis Q azon részhalmazait) vesszük fel az állapotok közé, amely előáll valamilyen eddigi állapotból valamely bemenőjel hatására.

5.19. példa - Automata determinizálása 1. feladat

Adjunk meg az $A=(\{ a_0, a_1, a_2 \}, \{ x, y \}, a_0, \delta, \{ a_1 \})$ nemdeterminisztikus, parciálisan definiált, kimenő jel nélküli, iniciális, végállapotokkal bővített véges automatával ekvivalens A_d determinisztikus, teljesen definiált automatát!

δ	a_0	a_1	a_2
x	$\{ a_2 \}$	$\{ a_0 \}$	$\{ a_1, a_2 \}$
y	$\{ a_1 \}$	$\{ a_2 \}$	-

Megoldás

I. Első lépésben meg kell határozni az új A_d automata belső állapotainak halmazát.
Az A automata kezdőállapotához, valamint azon állapothalmazaihoz,
melyek elérhetőek a kezdőállapotból, rendeljünk új betűket!
Ezek az új betűk alkotják az új állapothalmazt.
Jelen esetben:
Az A automata kezdőállapotához és a kezdőállapotból egy lépésben elérhető
állapothalmazokhoz rendeljünk új betűket:
Jelöljük b_i-vel ezeket az elemeket!

$b_0=\{ a_0 \}, b_1=\{ a_2\}, b_2=\{ a_1 \}.$

Vegyük fel azokat az állapothalmazokat is, melyek egy lépésben elérhetőek
a már meglévő állapothalmazokból és még nem szerepeltek:

$b_3=\{ a_1, a_2 \}, b_4 = \emptyset.$

Mindezt addig kell folytatni, amíg van újabb elérhető állapothalmaz.
Amennyiben több állapot is szerepel egy állapothalmazban, - mint az megfigyelhető a b_3 állapothalmaz esetén, - akkor az ezen állapotokból elérhető állapothalmazok unióját kell venni:

$b_5=\{ a_0, a_1, a_2 \}.$

Mivel újabb bővítés nem lehetséges, készen vagyunk az első lépéssel.

II. Második lépésben meghatározzuk az A_d automata δ' átmenetfüggvényét.
Ehhez a következő lépésekre van szükség:

1. Ha az automata $b_i = \emptyset$ állapotban van, akkor bármely
bemenő jel hatására ugyanebben az állapotban marad.

2. Ha az automata egy $b_i \neq \emptyset$ állapotban van, akkor meg kell
vizsgálni, hogy a b_i halmazban lévő $a_{l_1}, ..., a_{l_n}$
állapotok az adott bemenő jel hatására mely $a_{k_1}, ... ,a_{k_m}$ állapotokba mennek át.
Az A_d automata a b_i állapotból az adott bemenő jel hatására az
$a_{k_1}, ... ,a_{k_m}$ halmazhoz tartozó b_j állapotba megy át.

δ'	b_0	b_1	b_2	b_3	b_4	b_5
x	b_1	b_3	b_0	b_5	b_4	b_5

δ'	b_0	b_1	b_2	b_3	b_4	b_5
y	b_2	b_4	b_1	b_1	b_4	b_3

III. Harmadik lépésben meg kell határozni az A_d automata
bemenő jeleinek halmazát, kezdőállapotát, valamint végállapotainak halmazát.

- Az A_d automata bemenő jeleinek halmaza megegyezik az A automata bemenő jeleinek a halmazával.
- Az A_d automata kezdőállapota az A automata kezdőállapotához rendelt b_i lesz.
- Az A_d automata végállapotainak a halmaza pedig tartalmazni fog minden olyan b_j állapotot, melynek mint halmaznak eleme az A automata bármely végállapota.

Jelen esetben:
A_d=({ b_0, b_1, b_2, b_3, b_4, b_5 }, { x, y }, b_0, δ', { b_1, b_2, b_3, b_5 }).

★

5.20. példa - Automata determinizálása 2. feladat

Adjunk meg az A=({ a_0, a_1 }, { x, y }, a_0, δ, { a_1 })
nemdeterminisztikus, parciálisan definiált, kimenő jel nélküli, iniciális, végállapotokkal bővített véges automatával ekvivalens A_d determinisztikus, teljesen definiált automatát!

δ	a_0	a_1
x	{ a_1 }	{ a_0, a_1 }
y	{ a_0 }	-

Megoldás:

I. b_0={ a_0 }, b_1={ a_1 }, b_2={ a_0, a_1 }, $b_3 = \emptyset$.

II.

δ'	b_0	b_1	b_2	b_3
x	b_1	b_2	b_2	b_3
y	b_0	b_3	b_0	b_3

III.A_d=({ b_0, b_1, b_2, b_3 }, { x, y }, b_0, δ', { b_1, b_2 }).

★

5.21. példa - Automata determinizálása 3. feladat

Adjunk meg az A=({ a_0, a_1 }, { x, y, z }, a_0, δ, { a_0 })
nemdeterminisztikus, parciálisan definiált, kimenő jel nélküli, iniciális, végállapotokkal bővített véges automatával ekvivalens A_d determinisztikus, teljesen definiált automatát!

δ	a_0	a_1
x	{ a_1 }	-
y	-	{ a_0, a_1 }
z	{ a_0, a_1 }	{ a_0 }

Megoldás:

I. b_0={ a_0}, b_1={ a_1 }, $b_2 = \emptyset$, b_3={ a_0, a_1 }.

II.

δ'	b_0	b_1	b_2	b_3
x	b_1	b_2	b_2	b_1

δ'	b_0	b_1	b_2	b_3
y	b_2	b_3	b_2	b_3
z	b_3	b_0	b_2	b_3

III.A_d=({ b_0, b_1, b_2, b_3 }, { x, y, z }, b_0, δ', { b_0, b_3 }).

★

5.22. példa - Automata determinizálása 4. feladat

Adjunk meg az A=({ a_0, a_1 }, { x, y }, a_0, δ, { a_1 })
nemdeterminisztikus, parciálisan definiált, kimenő jel nélküli, iniciális, végállapotokkal bővített véges automatával ekvivalens A_d determinisztikus, teljesen definiált automatát!

δ	a_0	a_1
x	{ a_0, a_1 }	{ a_0, a_1 }
y	{ a_0, a_1 }	-

Megoldás:

I. b_0={ a_0 }, b_1={ a_0, a_1 }.

II.

δ'	b_0	b_1
x	b_0	b_1
y	b_1	b_1

III.A_d=({ b_0, b_1 }, { x, y }, b_0, δ', { b_1 }).

★

5.23. példa - Automata determinizálása 5. feladat

Adjunk meg az A=({ a_0, a_1, a_2 }, { x, y }, a_0, δ, { a_0, a_1 })
nemdeterminisztikus, parciálisan definiált, kimenő jel nélküli, iniciális, végállapotokkal bővített véges automatával ekvivalens A_d determinisztikus, teljesen definiált automatát!

δ	a_0	a_1	a_2
x	{ a_2 }	{ a_1 }	{ a_1, a_2 }
y	{ a_0 }	{ a_2 }	{ a_0 }

Megoldás:

I. b_0={ a_0 }, b_1={ a_2 }, b_2={ a_1, a_2 }, b_3={ a_0, a_2 }.

II.

δ'	b_0	b_1	b_2	b_3
x	b_1	b_2	b_2	b_2
y	b_0	b_0	b_3	b_0

III.A_d=({ b_0, b_1, b_2, b_3 }, { x, y }, b_0, δ', { b_0, b_2, b_3 }).

★

5.24. példa - Automata determinizálása 6. feladat

Adjunk meg az A=({ a_0, a_1, a_2 }, { x, y }, a_0, δ, { a_1 })
nemdeterminisztikus, parciálisan definiált, kimenő jel nélküli, iniciális, végállapotokkal bővített véges automatával ekvivalens A_d determinisztikus, teljesen definiált automatát!

δ	a_0	a_1	a_2
x	$\{ a_1 \}$	$\{ a_1, a_2 \}$	$\{ a_2 \}$
y	-	-	$\{a_1\}$

Megoldás:

I. $b_0=\{ a_0 \}, b_1=\{ a_1 \}, b_2 = \emptyset, b_3=\{ a_1, a_2 \}$.

II.

δ'	b_0	b_1	b_2	b_3
x	b_1	b_3	b_2	b_3
y	b_2	b_2	b_2	b_1

III.$A_d=(\{ b_0, b_1, b_2, b_3 \},\{ x, y \}, b_0, \delta', \{ b_1, b_3 \})$.

★

5.25. példa - Automata determinizálása 7. feladat

Adjunk meg az $A=(\{ a_0, a_1, a_2 \}, \{ x, y, z \}, a_0, \delta, \{ a_1, a_2 \})$
nemdeterminisztikus, parciálisan definiált, kimenő jel nélküli, iniciális, végállapotokkal bővített véges automatával ekvivalens A_d determinisztikus, teljesen definiált automatát!

δ	a_0	a_1	a_2
x	$\{ a_0 \}$	-	$\{ a_0, a_2 \}$
y	$\{ a_1, a_2 \}$	$\{ a_0 \}$	-
z	-	$\{ a_0, a_1 \}$	-

Megoldás:

I. $b_0=\{ a_0 \}, b_1=\{ a_1, a_2 \}, b_2 = \emptyset, b_3=\{ a_0, a_2 \}, b_4=\{ a_0, a_1 \}, b_5=\{ a_0, a_1, a_2 \}$.

II.

δ'	b_0	b_1	b_2	b_3	b_4	b_5
x	b_0	b_3	b_2	b_3	b_0	b_3
y	b_1	b_0	b_2	b_1	b_5	b_5
z	b_2	b_4	b_2	b_2	b_4	b_4

III.$A_d=(\{ b_0, b_1, b_2, b_3, b_4, b_5 \},\{ x, y ,z \}, b_0, \delta', \{ b_1, b_3, b_4, b_5 \})$.

★

5.26. példa - Automata determinizálása 8. feladat

Adjunk meg az $A=(\{ a_0, a_1, a_2, a_3 \}, \{ x, y, z \}, a_0, \delta, \{ a_1, a_3 \})$
nemdeterminisztikus, parciálisan definiált, kimenő jel nélküli, iniciális, végállapotokkal bővített véges automatával ekvivalens A_d determinisztikus, teljesen definiált automatát!

δ	a_0	a_1	a_2	a_3
x	$\{ a_0 \}$	-	$\{ a_2, a_3 \}$	$\{ a_1, a_2 \}$
y	$\{ a_1, a_2, a_3 \}$	$\{ a_0 \}$	-	$\{ a_0 \}$
z	-	$\{ a_1, a_2 \}$	$\{ a_1, a_3 \}$	$\{ a_1 \}$

Megoldás:

I. $b_0=\{ a_0 \}, b_1=\{ a_1, a_2 , a_3\}, b_2 = \emptyset$.

II.

δ'	b_0	b_1	b_2
x	b_0	b_1	b_2
y	b_1	b_0	b_2
z	b_2	b_1	b_2

III.A_d=({ b_0, b_1, b_2 },{ x, y ,z }, b_0, δ', { b_1 }).

★

5.4.2. Véges determinisztikus elfogadó automaták minimalizálása

Itt mutatjuk be az Aufenkamp-Hohn-féle Minimalizációs Algoritmus determinisztikus elfogadó automatákra működő változatát.

Legyen adott DFA=(Q, T, q_0, d, F). A valódi minimalizásiós algoritmus végrehajtása előtt az iniciális összefüggőséget kell ellenőriznünk, illetve a kezdőállapotból nem elérhető állapotokat törölni: azaz a Q'={$d^*(q_0, w)|w\in T^*$} állapothalmazzal és F'=$F\cap${$d^*(q_0, w)|w\in T^*$} végállapothalmazzal rendelkező DFA'=(Q', T, q_0, d', F') iniciálisan összefüggő (kimenő jel nélküli) állapot-részautomatát minimalizáljuk.

A továbbiakban legyen DFA=(Q, T, q_0, d, F) iniciálisan összefüggő. A C_{DFA} osztályozást osztályozások egy C_1, C_2, … sorozatán keresztül szerkesztjük meg, melyeket a következőképp definiálunk:

Kezdetként a C_1 osztályozással bontsuk a Q állapothalmazt két részre: F és $Q\setminus F$.

Ezután, ha $i \geq 1$, úgy a C_{i+1} osztályozás szerint p és q akkor és csak akkor esnek egy osztályba, ha egyrészt már C_i szerint is egy osztályba esnek, másrészt pedig minden bemenő jel hatására egy és ugyanazon C_i szerinti osztályba mennek át.

Képletben: ha $i \geq 1$, akkor $C_{i+1}[p]=C_{i+1}[q]\Leftrightarrow C_i[p]=C_i[q]$ és $\forall a\in T: C_i[d(p, a)]=C_i[d(q, a)]$.

A Q végessége miatt lesz olyan m, hogy C_m osztályozás megegyezik C_{i+m} osztályozással (vagyis megkaptuk a C_{DFA} osztályozást), ekkor a teljesen definiált determinisztikus minimális automatát a következőképpen adjuk meg:

A/C_m=(C_m, T, $C_m[q_0]$, d_{C_m}, F_{C_m}), ahol minden $C_m[q]\in C_m$, $a\in T$- re $d_{C_m}(C_m[q], a)=C_m[d(q, a)]$, illetve F_{C_m}={$C_m[q]$ | $q\in F$}.

Az így kapott automata ekvivalens az eredetivel, ugyanazt az L nyelvet fogadja el; teljesen definiált determinisztikus és minimális állapotszámú.

Ha egy DFA=(Q, T, q_0, d, F) minimális automata esetén vannak olyan T^*- beli szavak amelyek nem prefixei (kezdőszeletei) egyetlen L=$L(DFA)$ nyelvbeli szónak sem, azaz van olyan $u\in T^*$, hogy nincs olyan $v\in T^*$, hogy $uv\in L$, akkor a minimális determinisztikus teljesen definiált automatának van nyelőállapota, vagyis olyan q állapot, amire bármilyen $a\in T$ bemenőjelre $d(q, a)=q$ és $q\notin F$.

Például ez a nyelő állapot felel meg egy nemdeterminisztikus automata nem definiált átmeneteinek, vagyis a determinizáló algoritmusban az $\emptyset\subset Q$ halmaznak.

Természetesen parciális véges automata esetén ez a nyelő állapot elhagyható, így ekkor az állapotszám eggyel csökkenthető.

Amint később látni fogjuk a minimális determinisztikus automata létezésének fontos elméleti jelentősége is van.

5.27. példa - Véges elfogadó automata minimalizálása 1. feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus
segítségével az
$A=(\{a_0,a_1,a_2,a_3,a_4,a_5,a_6\},\{x,y\},a_0,\delta,\{a_2,a_4,a_5,a_6\})$
kimenő jel nélküli, iniciális, végállapotokkal bővített
véges automatával ekvivalens A_0 minimális állapotszámú automatát!

δ	a_0	a_1	a_2	a_3	a_4	a_5	a_6
x	a_2	a_5	a_1	a_1	a_2	a_1	a_0
y	a_1	a_0	a_3	a_4	a_5	a_3	a_2

Megoldás:

(O.) Mielőtt az érdemi munkához hozzáfognánk, meg kell vizsgálni, hogy
mely állapotok érhetőek el az A automata kezdőállapotából.
Azokat az állapotokat, melyek nem érhetőek el, egyszerűen töröljük, mivel
nem fognak előfordulni semelyik számítás során sem.

Jelen esetben az a_0 állapotból elérhető állapotok:
$\{a_0,a_2,a_1,a_3,a_5,a_4\}$.
Látható, hogy semmilyen input szó esetén sem kerülhet az A
automata a_6 állapotba, ezért ezt az állapotot töröljük. Az így kapott
A' automatát kell a továbbiakban minimalizálnunk az
Aufenkamp-Hohn-féle minimalizációs algoritmus segítségével:
$A'=(\{a_0,a_1,a_2,a_3,a_4,a_5\},\{x,y\},a_0,\delta',\{a_2,a_4,a_5\})$.

δ'	a_0	a_1	a_2	a_3	a_4	a_5
x	a_2	a_5	a_1	a_1	a_2	a_1
y	a_1	a_0	a_3	a_4	a_5	a_3

(I.) A feladat megoldásának első lépéseként különböző osztályokba fogjuk
sorolni az A' automata belső állapotait. A kezdeti osztályozáskor
mindig két osztályba kerülnek az állapotok, attól függően, hogy végállapotok,
vagy pedig nem végállapotok.

Jelen esetben:
$C_1=\{a_0,a_1,a_3\},\{a_2,a_4,a_5\}$.
Ezek után a C_{i+1}-edik osztályozás esetén két állapot akkor esik egy
osztályba, ha egyrészt a C_i-edik osztályozás esetén is azonos
osztályba tartoztak, másrészt pedig minden bemenő jel hatására azonos
C_i-beli osztályban található állapotokba mennek át. Az osztályozás
véget ér, amennyiben $C_i=C_{i+1}$ valamely $i \geq 1$ esetén.

Jelen esetben:
$C_2=\{a_0,a_1\},\{a_3\},\{a_2,a_5\},\{a_4\}$.
$C_3=\{a_0,a_1\},\{a_3\},\{a_2,a_5\},\{a_4\}$.
Mivel $C_2=C_3$, ezért az osztályozás véget ért. A C_2 osztályait
jelöljük valamely új betűvel, például b-vel:
$b_0=\{a_0,a_1\},b_1=\{a_3\},b_2=\{a_2,a_5\},b_3=\{a_4\}$.
(II.) Készítsük el az A' automatával ekvivalens, minimális állapotszámú
A_0 automatát, mely állapothalmazát az osztályozás és az új jelölés
bevezetése után kapott b_i betűk alkotják, a bemenő jeleinek halmaza megegyezik

az A automata bemenő jeleinek halmazával, az átmenetfüggvényét megkapjuk
úgy, hogy megnézzük, hogy az adott b_i osztálybeli állapotok
az adott bemenő jel hatására mely b_j osztálybeli állapotokba mentek át az A '
automata esetén, az új kezdőállapot az a b_i lesz, melynek eleme a_0, és
végül az A_0 automata végállapotait azon $b_{k_1},...,b_{k_m}$
osztályok alkotják, mely osztályok elemei az
A ' automata végállapotaiból állnak.

Jelen esetben:
$A_0=(\{b_0,b_1,b_2,b_3\},\{x,y\},b_0,\ \delta_0,\{b_2,b_3\})$.

δ_0	b_0	b_1	b_2	b_3
x	b_2	b_0	b_0	b_2
y	b_0	b_3	b_1	b_2

★

5.28. példa - Véges elfogadó automata minimalizálása 2. feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus
segítségével az
$A=(\{a_0,a_1,a_2,a_3,a_4,a_5,a_6\},\{x,y\},a_0,\ \delta,\{a_0,a_1,a_2,a_3\})$
kimenő jel nélküli, iniciális, végállapotokkal bővített
véges automatával ekvivalens A_0 minimális állapotszámú automatát!

δ	a_0	a_1	a_2	a_3	a_4	a_5	a_6
x	a_6	a_6	a_6	a_6	a_5	a_0	a_2
y	a_6	a_3	a_0	a_3	a_1	a_4	a_1

Megoldás:

(O.) Az a_0 kezdőállapotból elérhető belső állapotok:
$\{a_0,a_6,a_2,a_1,a_3\}$.

$A'=(\{a_0,a_1,a_2,a_3,a_6\},\{x,y\},a_0,\ \delta',\{a_0,a_1,a_2,a_3\})$.

δ'	a_0	a_1	a_2	a_3	a_6
x	a_6	a_6	a_6	a_6	a_2
y	a_6	a_3	a_0	a_3	a_1

(I.)
$C_1=\{a_0,a_1,a_2,a_3\},\{a_6\}$.
$C_2=\{a_0\},\{a_1,a_2,a_3\},\{a_6\}$.
$C_3=\{a_0\},\{a_1,a_3\},\{a_2\},\{a_6\}$.
$C_4=\{a_0\},\{a_1,a_3\},\{a_2\},\{a_6\}$.
$b_0=\{a_0\},b_1=\{a_1,a_3\},b_2=\{a_2\},b_3=\{a_6\}$.

(II.)
$A_0=(\{b_0,b_1,b_2,b_3\},\{x,y\},b_0,\ \delta_0,\{b_0,b_1,b_2\})$.

δ_0	b_0	b_1	b_2	b_3
x	b_3	b_3	b_3	b_2

δ_0	b_0	b_1	b_2	b_3
y	b_3	b_1	b_0	b_1

★

5.29. példa - Véges elfogadó automata minimalizálása 3. feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus
segítségével az
$A=(\{a_0,a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8\},\{x,y\},a_0,\delta,\{a_2,a_7\})$
kimenő jel nélküli, iniciális, végállapotokkal bővített
véges automatával ekvivalens A_0 minimális állapotszámú automatát!

δ	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
x	a_3	a_5	a_7	a_0	a_2	a_0	a_4	a_2	a_5
y	a_8	a_1	a_3	a_2	a_6	a_7	a_3	a_5	a_2

Megoldás:

(O.) Az a_0 kezdőállapotból elérhető belső állapotok:
$\{a_0,a_3,a_8,a_2,a_5,a_7\}$.

$A'=(\{a_0,a_2,a_3,a_5,a_7,a_8\},\{x,y\},a_0,\delta',\{a_2,a_7,a_8\})$.

δ'	a_0	a_2	a_3	a_5	a_7	a_8
x	a_3	a_7	a_0	a_0	a_2	a_5
y	a_8	a_3	a_2	a_7	a_5	a_2

(I.)
$C_1=\{a_0,a_3,a_5,a_8\},\{a_2,a_7\}$.
$C_2=\{a_0\},\{a_3,a_5,a_8\},\{a_2,a_7\}$.
$C_3=\{a_0\},\{a_3,a_5\},\{a_8\},\{a_2,a_7\}$.
$C_4=\{a_0\},\{a_3,a_5\},\{a_8\},\{a_2,a_7\}$.
$b_0=\{a_0\},b_1=\{a_3,a_5\},b_2=\{a_8\},b_3=\{a_2,a_7\}$.
(II.)
$A_0=(\{b_0,b_1,b_2,b_3\},\{x,y\},b_0,\delta_0,\{b_3\})$.

δ_0	b_0	b_1	b_2	b_3
x	b_1	b_0	b_1	b_3
y	b_2	b_3	b_3	b_1

★

5.30. példa - Véges elfogadó automata minimalizálása 4. feladat

Készítsük el az Aufenkamp-Hohn-féle minimalizációs algoritmus
segítségével az
$A=(\{a_0,a_1,a_2,a_3,a_4,a_5,a_6,a_7\},\{x,y,z\},a_0,\delta,\{a_2,a_4,a_5,a_7\})$
kimenő jel nélküli, iniciális, végállapotokkal bővített
véges automatával ekvivalens A_0 minimális állapotszámú automatát!

δ	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
x	a_6	a_3	a_1	a_7	a_1	a_0	a_4	a_1
y	a_4	a_7	a_0	a_2	a_3	a_1	a_5	a_6
z	a_5	a_2	a_6	a_1	a_3	a_6	a_0	a_6

Megoldás:

(O.) Az a_0 kezdőállapotból elérhető belső állapotok:
$\{a_0,a_6,a_4,a_5,a_1,a_3,a_7,a_2\}$.
Mivel a kezdőállapotból minden állapot elérhető, ezért nem törlünk
egyetlen állapotot sem, tehát $A'=A$.
(I.)
$C_1=\{a_0,a_1,a_3,a_6\},\{a_2,a_4,a_5,a_7\}$.
$C_2=\{a_0,a_1\},\{a_3,a_6\},\{a_2,a_4,a_5,a_7\}$.
$C_3=\{a_0,a_1\},\{a_3,a_6\},\{a_2,a_5\},\{a_4,a_7\}$.
$C_4=\{a_0,a_1\},\{a_3,a_6\},\{a_2,a_5\},\{a_4,a_7\}$.
$b_0=\{a_0,a_1\},b_1=\{a_3,a_6\},b_2=\{a_2,a_5\},b_3=\{a_4,a_7\}$.
(II.)
$A_0=(\{b_0,b_1,b_2,b_3\},\{x,y,z\},b_0,\ \delta_0,\{b_2,b_3\})$.

δ_0	b_0	b_1	b_2	b_3
x	b_1	b_3	b_0	b_0
y	b_3	b_2	b_0	b_1
z	b_2	b_0	b_1	b_1

★

5.4.3. Véges automaták és reguláris nyelvtanok ekvivalenciája

23. Tétel. A 3- as típusú nyelvek osztálya egybeesik a véges automatákkal felismerhető nyelvek osztályával.

Bizonyítás. Legyen adott $G=(N, T, S, H)$ reguláris nyelvtan gyenge normálformában. Ekkor megadunk egy nemdeterminisztikus üresszó átmenetet is megengedő $NFA=(Q, T, q_0, d, F)$ automatát, amely $L(G)$ nyelvet fogadja el. Legyen $Q=N\cup\{q_f\}$, ahol $q_f\notin N$. Legyen $q_0=S$, $F=\{q_f\}$, továbbá a d állapotátmenet-függvényt adjuk meg a következőképpen:

minden $A \to aB\in H$ szabály ($A, B\in N, a\in T\cup\{\lambda\}$) esetén legyen $B\in d(A, a)$ és

minden $A \to a\in H$ szabály ($A\in N, a\in T\cup\{\lambda\}$) esetén legyen $q_f\in d(A, a)$.

A konstrukció alapján látható, hogy G minden termináló levezetéséhez pontosan egy elfogadó futása lesz az NFA automatának és fordítva.

Most nézzük a fordított konstrukciót: az egyszerűség kedvéért nemdeterminisztikus üresszó átmenet nélküli automatából kiindulva.

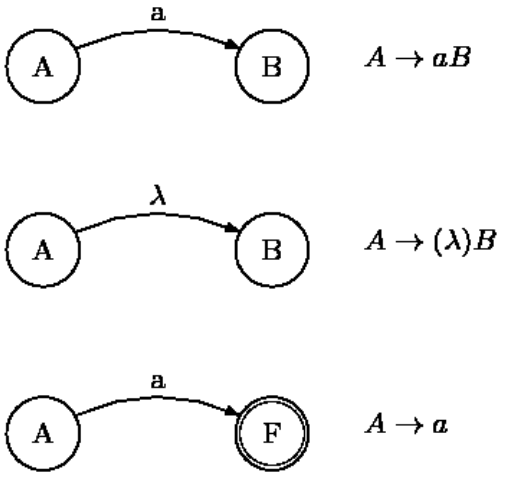
Legyen, tehát, adott $NFA=(Q, T, q_0, d, F)$ úgy, hogy $Q\cap T=\emptyset$, definiáljuk a $G=(N, T, S, H)$ nyelvtant a következőképpen: legyen $N=Q$, $S=q_0$. A szabályok pedig:

minden $p\in d(q, a)$ esetén ($p, q\in Q, a\in T$) legyen $q \to ap\in H$.

továbbá, minden $q\in F$ esetén legyen $q \to \lambda\in H$.

Könnyen ellenőrizhető, hogy *NFA* minden elfogadó futásához lesz a *G* nyelvtanban termináló levezetés, és fordítva. ∎

Képletesen szólva, az előző konstrukciók esetén a következő átalakításokat végezhetjük el:



Ebben a fejezetben tehát beláttuk, hogy a pontosan a reguláris nyelveket fogadják el a véges automaták, akár nemdeterminisztikus üresszó átmenetet is megengedő automatáról, nemdeterminisztikus üresszó átmenet nélküli automatáról, (parciális) determinisztikus automatáról, teljesen definiált determinisztikus automatáról, vagy akár minimális determinisztikus automatákról van szó.

5.31. példa - Ekvivalens reguláris nyelvtan megadása véges automatához 1. feladat

Feladat: Adjunk meg az $A=(\{q_0,q_1,q_2\},\{a,b\},q_0,\delta,\{q_0,q_2\})$

$\delta(q_0,a)=\{q_1,q_2\}$, $\delta(q_0,b)=\{q_2\}$,
$\delta(q_1,a)=\{q_0\}$, $\delta(q_2,b)=\{q_1\}$

véges automatával ekvivalens *G* reguláris nyelvtant!

Megoldás:

A *G* nyelvtan nemterminálisainak halmaza az automata állapothalmazával egyezik meg, a terminálisok halmazát az automata bemenő jeleinek halmaza alkotja, a startszimbólum az automata kezdőállapota lesz.

A *G* nyelvtan szabályainak halmazát két csoportba oszthatjuk.
Az első csoportba tartoznak azok a szabályok, melyeket az átmenetfüggvényből kaphatunk meg.
Ha az *A* automata a q_i állapotból az *x* bemenőjel hatására a q_j állapotba megy át,
akkor a szabályok közé bekerül a $q_i \to xq_j$ szabály.
A második csoportba azok a szabályok tartoznak, melyeket a végállapotok alapján adhatunk meg.
Amennyiben a q_k állapot szerepel az *A* automata végállapotainak a halmazában,
úgy fel kell vennünk *G* nyelvtan szabályai közé a $q_k \to \lambda$ szabályt.

Jelen esetben:
$G=(\{q_0,q_1,q_2\},\{a,b\},q_0,H)$,
$H=\{\ q_0 \to aq_1,\ q_0 \to aq_2,\ q_0 \to bq_2,\ q_1 \to aq_0,\ q_2 \to bq_1,\ q_0 \to \lambda,\ q_2 \to \lambda\ \}$. ★

5.32. példa - Ekvivalens reguláris nyelvtan megadása véges automatához 2. feladat

Adjunk meg az $A=(\{q_0,q_1\},\{x,y\},q_0,\delta,\{q_1\})$,

$\delta(q_0,x)=\{q_0,q_1\}$, $\delta(q_1,y)=\{q_0\}$

véges automatával ekvivalens G reguláris nyelvtant!

Megoldás:
$G=(\{q_0,q_1\},\{x,y\},q_0,H)$,
$H=\{\ q_0 \to xq_0,\ q_0 \to xq_1,\ q_1 \to yq_0,\ q_1 \to \lambda\ \}$. ★

5.33. példa - Ekvivalens reguláris nyelvtan megadása véges automatához 3. feladat

Adjunk meg az $A=(\{q_0,q_1,q_2\},\{x,y\},q_0,\delta,\{q_2\})$,

$\delta(q_0,x)=\{q_1\}$, $\delta(q_0,y)=\{q_2\}$, $\delta(q_1,x)=\{q_0,q_2\}$,
$\delta(q_1,y)=\{q_1,q_2\}$, $\delta(q_2,x)=\{q_0\}$, $\delta(q_2,y)=\{q_1\}$

véges automatával ekvivalens G reguláris nyelvtant!

Megoldás:
$G=(\{q_0,q_1,q_2\},\{x,y\},q_0,H)$,
$H=\{\ q_0 \to xq_1,\ q_0 \to yq_2,\ q_1 \to xq_0,\ q_1 \to xq_2,\ q_1 \to yq_1,$
$q_1 \to yq_2,\ q_2 \to xq_0,\ q_2 \to yq_1,\ q_2 \to \lambda\ \}$. ★

5.34. példa - Ekvivalens reguláris nyelvtan megadása véges automatához 4. feladat

Adjunk meg az $A=(\{q_0,q_1,q_2,q_3\},\{x,y,z\},q_0,\delta,\{q_0,q_2,q_3\})$,

$\delta(q_0,x)=\{q_1,q_3\}$, $\delta(q_0,y)=\{q_2\}$, $\delta(q_1,z)=\{q_0,q_2\}$,
$\delta(q_2,x)=\{q_0\}$, $\delta(q_3,y)=\{q_1\}$.

véges automatával ekvivalens G reguláris nyevtant!

Megoldás:
$G=(\{q_0,q_1,q_2,q_3\},\{x,y,z\},q_0,H)$,
$H=\{\ q_0 \to xq_1,\ q_0 \to xq_3,\ q_0 \to yq_2,\ q_1 \to zq_0,\ q_1 \to zq_2,$
$q_2 \to xq_0,\ q_3 \to yq_1,\ q_0 \to \lambda,\ q_2 \to \lambda,\ q_3 \to \lambda\ \}$. ★

5.35. példa - Ekvivalens véges automata megadása reguláris nyelvtanhoz 1. feladat

Adjunk meg a $G=(\{S,A,B\},\{a,b\},S,H)$ $H=\{\ S \rightarrow abaB,\ A \rightarrow B,\ A \rightarrow cacb,\ B \rightarrow bA,\ B \rightarrow S,\ B \rightarrow \lambda\ \}$ nyelvtannal ekvivalens véges üresszóátmenet nélküli automatát!

Megoldás:

(I.) Első lépésben megadunk egy G_1 nyelvtant, ami ekvivalens a G nyelvtannal és nem szerepelnek benne $Y \rightarrow y_1y_2 \ldots y_n$ és $Y \rightarrow y_1y_2 \ldots Y_n$, $n \geq 3$ alakú szabályok. A H szabályhalmaz ilyen alakú szabályait helyettesítjük új szabályokkal, a többi szabályt pedig változtatás nélkül átvesszük a H_1 szabályhalmazba.

Minden $Y \rightarrow y_1y_2 \ldots y_n$, $n \geq 3$ alakú szabályhoz vezessünk be $Z_1,Z_2, \ldots ,Z_{n-1}$ új nemterminálisokat, Z_i-ből fogjuk levezetni az $y_{i+1} \ldots y_n$. Ehhez az összes $Y \rightarrow y_1y_2 \ldots y_n$, $n \geq 3$ alakú szabályt helyettesítsük a következő szabályokkal:

$Y \rightarrow y_1Z_1$,
$Z_1 \rightarrow y_2Z_2$,
.
.
.
$Z_{n-2} \rightarrow y_{n-1}Z_{n-1}$,
$Z_{n-1} \rightarrow y_n$.

Minden $Y \rightarrow y_1y_2 \ldots Y_n$, $n \geq 3$ alakú szabályhoz vezessünk be $Z_1,Z_2, \ldots,Z_{n-2}$ új nemterminálisokat. Z_i-ből az $y_{i+1} \ldots y_n$ szót fogjuk levezetni:
az összes $Y \rightarrow y_1y_2 \ldots Y_n$, $n \geq 3$ alakú szabályt helyettesítsük a következő szabályokkal:

$Y \rightarrow y_1Z_1$,
$Z_1 \rightarrow y_2Z_2$,
.
.
.
$Z_{n-3} \rightarrow y_{n-2}Z_{n-2}$,
$Z_{n-2} \rightarrow y_{n-1}Y_n$.

Jelen esetben:

$\{G_1=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5\},\{a,b\},S,H_1)$. $H_1=\{\ S \rightarrow aZ_1,\ Z_1 \rightarrow bZ_2,\ Z_2 \rightarrow aB,\ A \rightarrow B,\ A \rightarrow cZ_3,\ Z_3 \rightarrow aZ_4,\ Z_4 \rightarrow cZ_5,\ Z_5 \rightarrow b,\ B \rightarrow bA,\ B \rightarrow S,\ B \rightarrow \lambda\ \}$

(II.) Második lépésben megadunk egy G' nyelvtant, ami ekvivalens a G nyelvtannal és nem szerepelnek benne $X \rightarrow Z$ alakú szabályok. Ehhez két lépésre van szükség.

- Először meghatározunk egy $U(Z)$ halmazt minden olyan Z nemterminálishoz, mely levezethető legalább egy másik nemterminálisból a G_1 nyelvtanban és szerepel olyan H_1 halmazban lévő szabály bal oldalán, amelynek jobb oldalán egy terminális, vagy egy terminális és egy nemterminális betű, vagy pedig az üresszó áll. Az $U(Z)$ halmaz tartalmazni fogja az összes olyan nemterminálist, melyből egy vagy több lépésben levezethető a Z betű.

Jelen esetben:
$U(B)=\{A\},U(S)=\{B,A\}$.

- Majd a H' szabályhalmazba átvesszük a H_1 szabályhalmaz
mindazon szabályait, melyek nem $X \to Z$ alakúak, majd hozzávesszük mindazon szabályokat, melyeket úgy kapunk, hogy a már átvett szabályok bal oldalán szereplő betűt a hozzá tartozó U halmaz elemeivel helyettesítjük.

Formálisan:

$H_2=(H_1 \cup \{W \to p | Z \to p \in H_1, W \in U(Z)\}) \setminus \{X \to Y | X,Y \in N_1\}$.

Jelen esetben:

$G'=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5\},\{a,b\},S,H')$. $H'=\{ S \to aZ_1,\ B \to aZ_1,\ A \to aZ_1,\ Z_1 \to bZ_2,\ Z_2 \to aB,\ A \to cZ_3,\ Z_3 \to aZ_4,\ Z_4 \to cZ_5,\ Z_5 \to b,\ B \to bA,\ A \to bA,\ B \to \lambda,\ A \to \lambda \}$

(III.) Harmadik lépésben megadjuk a G nyelvtannal ekvivalens A véges automatát. Az A automata állapothalmazát úgy kapjuk, hogy a G' nyelvtan nemterminálisainak a halmazához hozzáadunk egy új q_v állapotot. Az automata bemenő jeleinek a halmaza megegyezik a G nyelvtan terminálisainak a halmazával, a kezdőállapota a startszimbólum lesz, a végállapotok halmaza pedig tartalmaz minden olyan X állapotot, mely szerepelt $X \to \lambda$ alakú szabályban, valamint tartalmazza az újonnan bevezett q_v állapotot is.

Az A automata δ átmenetfüggvényét úgy kapjuk, hogy minden H' halmazban szereplő $X \to yZ$ szabály esetén felvesszük a $\delta(X,y)=Z$ átmenetet, valamint az $X \to y$ alakú szabályok esetén a $\delta(X,y)=q_v$ átmenetet.

Jelen esetben:
$A=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5,q_v\},\{a,b\},S,\delta,\{B,A,q_v\})$.

$\delta(S,a)=\{Z_1\}$, $\delta(B,a)=\{Z_1\}$, $\delta(A,a)=\{Z_1\}$, $\delta(Z_1,b)=\{Z_2\}$, $\delta(Z_2,a)=\{B\}$, $\delta(A,c)=\{Z_3\}$, $\delta(Z_3,a)=\{Z_4\}$, $\delta(Z_4,c)=\{Z_5\}$, $\delta(Z_5,b)=\{q_v\}$, $\delta(B,b)=\{A\}$, $\delta(A,b)=\{A\}$ ★

5.36. példa - Ekvivalens véges automata megadása reguláris nyelvtanhoz 2. feladat

Adjunk meg a $G=(\{S,A,B\},\{x,y\},S,H)$
$H=\{\ S \to xA,\ S \to yyB,\ A \to B,\ B \to yS,\ B \to xyx,\ B \to \lambda\ \}$

nyelvtannal ekvivalens véges automatát!

Megoldás:
(I.)
$G_1=(\{S,A,B,Z_1,Z_2,Z_3\},\{x,y\},S,H_1)$.
$H_1=\{\ S \to xA,\ S \to yZ_1,\ Z_1 \to yB,\ A \to B,$
$B \to yS,\ B \to xZ_2,\ Z_2 \to yZ_3,\ Z_3 \to x,$
$B \to \lambda\ \}$

(II.)
$U(B)=\{A\}$.
$G'=(\{S,A,B,Z_1,Z_2,Z_3\},\{a,b\},S,H')$.
$H'=\{\ S \to xA,\ S \to yZ_1,\ Z_1 \to yB,\ B \to yS,A \to yS,$
$B \to xZ_2,A \to xZ_2,\ Z_2 \to yZ_3,\ Z_3 \to x,\ B \to \lambda,A \to \lambda\ \}$

(III.)
$A=(\{S,A,B,Z_1,Z_2,Z_3,q_v\},\{x,y\},S,\delta,\{B,A,q_v\})$.

$\delta(S,x)=\{A\},\ \delta(S,y)=\{Z_1\},\ \delta(Z_1,y)=\{B\},\ \delta(B,y)=\{S\},\ \delta(A,y)=\{S\},$
$\delta(B,x)=\{Z_2\},\ \delta(A,x)=\{Z_2\},\ \delta(Z_2,y)=\{Z_3\},\ \delta(Z_3,x)=\{q_v\}$ ★

5.37. példa - Ekvivalens véges automata megadása reguláris nyelvtanhoz 3. feladat

Adjunk meg a $G=(\{S,A,B\},\{x,y\},S,H)$
$H=\{\ S \to xA,\ S \to yB,\ S \to B,\ B \to A,\ A \to xS,\ A \to y\ \}$
nyelvtannal ekvivalens véges automatát!

Megoldás:
(I.)
Mivel a G nyelvtanban nincs $Y \to y_1y_2 \ldots y_n$ és $Y \to y_1y_2 \ldots Y_n$, $n \geq 3$ alakú szabály, ezért
$G_1=G$.

(II.)
$U(A)=\{B,S\}$.
$G'=(\{S,A,B\},\{x,y\},S,H')$.
$H'=\{\ S \to xA,\ S \to yB,\ A \to xS,\ B \to xS,\ S \to xS,\ A \to y,\ B \to y,\ S \to y\ \}$

(III.)
$A=(\{S,A,B,q_v\},\{x,y\},S,\delta,\{q_v\})$.

$\delta(S,x)=\{A\},\ \delta(S,y)=\{B\},\ \delta(A,x)=\{S\},\ \delta(B,x)=\{S\},$
$\delta(S,x)=\{S\},\ \delta(A,y)=\{q_v\},\ \delta(B,y)=\{q_v\},\ \delta(S,y)=\{q_v\}$ ★

5.38. példa - Ekvivalens véges automata megadása reguláris nyelvtanhoz 4. feladat

Adjunk meg a $G=(\{S,A,B\},\{x,y\},S,H)$
$H=\{\ S \to xxxA,\ S \to yyyB,\ A \to yS,\ B \to xS,\ A \to xx,\ B \to yy\ \}$

nyelvtannal ekvivalens véges automatát!

Megoldás:
(I.)
$G'=(\{S,A,B,Z_v,Z_2,Z_3,Z_4,Z_5,Z_6\},\{x,y\},S,H')$.
$H'=\{\ S \to xZ_1,\ Z_1 \to xZ_2,\ Z_2 \to xA,\ S \to yZ_3,\ Z_3 \to yZ_4,\ Z_4 \to yB,$
$A \to yS,\ B \to xS,\ A \to xZ_5,\ Z_5 \to x,\ B \to yZ_6,\ Z_6 \to y\ \}$

(II.)
Mivel a G_1 nyelvtanban nincs $X \to Z$ alakú szabály, ezért $G_2=G_1$.

(III.)
$A=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5,Z_6,q_v\},\{x,y\},S,\delta,\{q_v\})$.

$\delta(S,x)=\{Z_1\},\ \delta(Z_1,x)=\{Z_2\},\ \delta(Z_2,x)=\{A\},\ \delta(S,y)=\{Z_3\},\ \delta(Z_3,y)=\{Z_4\},\ \delta(Z_4,y)=\{B\},$
$\delta(A,y)=\{S\},\ \delta(B,x)=\{S\},\ \delta(A,x)=\{Z_5\},\ \delta(Z_5,x)=\{q_v\},\ \delta(B,y)=\{Z_6\},\ \delta(Z_6,y)=\{q_1\}$ ★

5.39. példa - Ekvivalens véges automata megadása reguláris nyelvtanhoz 5. feladat

Adjunk meg a $G=(\{S,X,Y\},\{x,y,z\},S,H)$

$H=\{\ S \to xX,\ S \to z,\ S \to \lambda,\ X \to yY,\ X \to zS,\ X \to x,\ Y \to xX,\ Y \to y,\ \}$

nyelvtannal ekvivalens véges automatát!

Megoldás:

(I.)
Mivel a G nyelvtanban nincs $Y \to y_1y_2 \ldots y_n$ és
$Y \to y_1y_2 \ldots Y_n$, $n \geq 3$ alakú szabály, ezért $G_1=G$.

(II.)
Mivel a G_1 nyelvtanban nincs $X \to Z$ alakú szabály, ezért $G_2=G_1$.

(III.)
$A=(\{S,X,Y,q_v\},\{x,y,z\},S,\delta,\{S,q_v\})$.

$\delta(S,x)=\{X\},\ \delta(S,z)=\{q_v\},\ \delta(X,y)=\{Y\},\ \delta(X,x)=\{q_v\},\ \delta(X,z)=\{S\},\ \delta(Y,y)=\{q_v\},\ \delta(Y,x)=\{X\}$ ★

5.40. példa - Ekvivalens véges automata megadása reguláris nyelvtanhoz 6. feladat

Adjunk meg a $G=(\{S,A\},\{0,1\},S,H)$
$H=\{\ S \to 0,\ S \to 1A,\ A \to 0A,\ A \to 1A,\ A \to \lambda\ \}$

nyelvtannal ekvivalens véges automatát!

Megoldás:
(I.)
Mivel a G nyelvtanban nincs $Y \to y_1y_2 \ldots y_n$ és $Y \to y_1y_2 \ldots Y_n$, $n \geq 3$
alakú szabály, ezért $G_1=G$.

(II.)
Mivel a G_1 nyelvtanban nincs $X \to Z$ alakú szabály, ezért $G_2=G_1$.

(III.)
$A=(\{S,A,q_v\},\{0,1\},S,\delta,\{A,q_v\})$.

$\delta(S,0)=\{q_v\},\ \delta(S,1)=\{A\},\ \delta(A,0)=\{A\},\ \delta(A,1)=\{A\}$ ★

5.41. példa - Ekvivalens véges automata megadása reguláris nyelvtanhoz 7. feladat

Adjunk meg a $G=(\{S,A\},\{0,1,+,-\},S,H)$
$H=\{\ S \to 0,\ S \to 1A,\ S \to -1A,\ A \to 0A,$
$A \to 1A,\ A \to \lambda,\ A \to +1A,\ A \to -1A\ \}$

nyelvtannal ekvivalens véges automatát!

Megoldás:

(I.)
$G_1=(\{S,A,Z_1,Z_2,Z_3\},\{0,1,+,-\},S,H)$
$H=\{\ S \to 0,\ S \to 1A,\ S \to -Z_1,\ Z_1 \to 1A,\ A \to 0A,\ A \to 1A,$
$A \to \lambda,\ A \to +Z_2,\ Z_2 \to 1A,\ A \to -Z_3,\ Z_3 \to 1A\ \}$

(II.)
Mivel a G_1 nyelvtanban nincs $X \to Z$ alakú szabály, ezért $G_2=G_1$.

(III.)
$A=(\{S,A,Z_1,Z_2,Z_3,q_v\},\{0,1\},S,\delta,\{A,q_v\})$.

$\delta(S,0)=\{q_v\},\ \delta(S,1)=\{A\},\ \delta(S,-)=\{Z_1\},\ \delta(Z_1,1)=\{A\},\ \delta(A,0)=\{A\},$
$\delta(A,1)=\{A\},\ \delta(A,+)=\{Z_2\},\ \delta(Z_2,1)=\{A\},\ \delta(A,-)=\{Z_3\},\ \delta(Z_3,1)=\{A\}$ ★

5.5. Nyelvek előállítása automatákban

Az automaták és az automata leképezések kapcsolatát illetően, mint már utaltunk rá, két ellentétes kérdés fogalmazható meg:

I. Ha adva van egy iniciális automata, meghatározandó az általa indukált leképezés (analízis);

II. Ha adva van egy iniciális automata leképezés, meghatározandó legalább egy olyan iniciális automata, amely ezt a leképezést indukálja (szintézis).

A szintézis, ahogy utaltunk rá egyértelművé tehető a minimalizálás feladatával. Az analízis és szintézis akkor fogalmazható meg egzakt módon, ha megállapodunk abban, hogy mit értünk automata és automata leképezés megadásán. Ezzel kapcsolatban, amint a Automaták Analízise és Szintézise fejezetben utaltunk rá, a következő problémát kell megoldani:

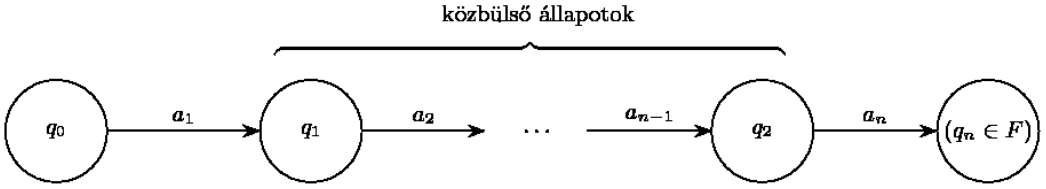
0. Meg kell adnunk a véges automaták által indukálható automata leképezéseknek az automatáktól független leírását. Más szóval, keresnünk kell olyan írásmódot, amelyben el tudjuk dönteni, hogy valamely, ebben az írásmódban megadott automata leképezés indukálható-e véges automatával vagy sem. Ezek után véges automatákra az analízis és szintézis problémája így fogalmazható meg:

I'. Bármely adott véges automata esetén meg kell tudnunk adni a szóban forgó írásmódban azt a leképezést, amelyet az automata indukál;

II'. Ha ebben az írásmódban meg van adva egy véges automata által indukálható leképezés, akkor meg kell tudnunk adni legalább egy olyan automatát, mely ezt a leképezést indukálja és állapothalmaza véges.

Egy olyan leképezés megadási módszer, mely eleget tesz a 0.,I.',II.' feltételeknek, először S. C. Kleene talált 1956-ban, bevezetve a reguláris kifejezés fogalmát. A fejezet további részében az elfogadó automatákra vizsgáljuk az analízis és szintézis fogalmát, vagyis az analízis és szintézis foglamát módosítjuk és e kettős feladat megoldását kimenő jel nélküli automaták vizsgálatára vezetjük vissza. Amint láttuk, a generatív rendszerek egyik fontos típusát képezik a generatív nyelvtanok, melyek a formális nyelvek előállítására alkalmas eszközök. A formális nyelvek felismerésére viszont nem az analitikus nyelvtanokat, hanem a felismerő (Rabin-Scott) automatákat szokás alkalmazni.

Tetszőleges $DFA=(Q, T, q_0, d, F)$ determinisztikus automata esetén akkor mondjuk, hogy $w \in L(DFA)$ ha az automata a $q_0 \in Q$ kezdőállapotból a $w \in T^*$ inputszó hatására a $d^*(q_0, w) \in F$ állapotba megy át.



24. Tétel. (Kleene Tétele) Bármely adott *DFA* véges determinisztikus automatához és állapotai *F* részhalmazához meg tudjuk adni annak a nyelvnek egy reguláris kifejezését, amely *DFA*- ban az *F* halmazzal előállítható. Megfordítva, minden reguláris kifejezéshez megadható egy *FA* véges (felismerő) automata, amely állapotai *F* részhalmazával éppen a tekintett reguláris kifejezéssel leírt nyelvet fogadja el.

Kleene tételének konstruktív bizonyítását fogjuk adni a következő fejezetekben.

Azt mondjuk, hogy a T^* monoidon (azaz egységelemes félcsoporton) definiált valamely ϱ ekvivalencia reláció *jobbkongruencia,* ha minden $u, v, w \in T^*$ hármasra igaz, hogy $u\varrho v \Rightarrow uw\varrho vw$. A T^* monoid egy C_ϱ osztályozását *jobbról stabilnak* (vagy *jobbról kompatibilisnek*) hívjuk, ha a ϱ reláció jobbkongruencia. Végül, a ϱ relációt, illetve a C_ϱ osztályozást *véges indexűnek* mondjuk, ha véges sok osztályból áll. Analóg módon definiálható a balkongruncia és a balról kompatibilis osztályozás. Egy, a T^* monoidon megadott relációt *kongruenciának* hívunk, ha egyidejűleg bal- és jobbkongruencia. Másként kifejezve, a T^* monoidon értelmezett ϱ reláció kongruencia, ha bármely $u, v, w, s \in T^*$- ra $u\varrho v \Rightarrow wus\varrho wvs$. (Megjegyezzük, hogy természetesen u, v, w, s bármelyike lehet az üresszó.) Egy ϱ kongruenciához tartozó C_ϱ osztályozást *kompatibilisnek* mondunk. Másként kifejezve, kompatibilisnek hívjuk az egyidejűleg balról és jobbról stabil osztályozást.

A véges automatákban előállítható nyelvek egy relációelméleti jellemzését adja Myhill (ejtsd: májhil) és Nerode (ejtsd: neród) tétele.

25. Tétel. (Myhill és Nerode tétele) Egy véges T ábécé feletti L nyelv akkor és csak akkor állítható elő egy véges automatában az állapotok valamely részhalmazával, ha megadható T^*- nak olyan véges indexű kompatibilis C osztályozása, hogy L előáll bizonyos C- beli osztályok egyesítési halmazaként.

Bizonyítás. Mindenekelőtt megjegyezzük, hogy T^*- nak az a triviális osztályozása, amelynél minden elem egymagában alkot egy osztályt, nyilván kompatibilis és egy L nyelv mindig előáll ennél az osztályozásnál bizonyos osztályok egyesítési halmazaként. Ennél fogva egy tetszőleges L nyelv esetén mindig van olyan kompatibilis osztályozása T^*- nak, hogy az L előáll bizonyos osztályok egyesítési halmazaként.

A tétel szükségességének bizonyításához tegyük fel, hogy egy L nyelv előállítható a $DFA=(Q, T, q_0, d, F)$ véges automata állapotainak F részhalmazában, vagyis DFA Rabin-Scott automata éppen az L nyelvet ismeri fel. Definiáljunk T^*- on egy kongruenciát a következőképp:

$\forall u, v \in T^*: (u \varrho_{DFA} v \Leftrightarrow \forall q \in Q: d^*(q, u) = d^*(q, v))$.

(Vagyis minden $u, v \in T^*$ esetén $u \varrho_{DFA} v$ akkor és csak akkor teljesüljön, ha tetszőleges Q- beli állapotra az u szó a q állapotot ugyanabba az állapotba viszi át mint a v szó.) Ekkor a ϱ- hoz tartozó C_ϱ kompatibilis osztályozásnál egy-egy osztályba pontosan azok a T^*- beli elemek tartoznak, melyek a DFA automata kezdőállapotát egy meghatározott állapotba viszik. Nyilvánvaló, hogy a DFA végessége miatt a Q halmaz véges, azaz a C_ϱ kompatibilis osztályozás véges sok osztályt tartalmaz. Tehát C_ϱ valóban véges indexű.

Azt kell még kimutatnunk, hogy L előáll bizonyos C_ϱ- beli osztályok egyesítési halmazaként. Jelölje most tetszőleges $q \in Q$ mellett $C^{(q)}$ ezen C_ϱ kompatibilis osztályozás azon osztályát, melynek elemei a q_0 kezdőállapotot a q állapotba viszik át. Mivel feltettük, hogy a DFA az L nyelvet állapotainak F halmazával ismeri fel, ekkor $L = \cup \{C^{(q)} | q \in F\}$. (Természetesen $L = \emptyset$ előfordulhat. Ilyen esetben $F = \emptyset$.) Ezzel a tétel szükségességét kimutattuk.

Most igazoljuk az elegendőséget. Tegyük fel, hogy megadható T^*- nak olyan véges indexű kompatibilis C osztályozása, hogy L előáll bizonyos C- beli osztályok egyesítési halmazaként. Korábbi konvenciónknak megfelelően tetszőleges $u \in T^*$ mellett $C[u]$- val fogjuk jelölni azt az osztályt, melynek egy adott $u \in T^*$ eleme. Ha $L = \emptyset$, akkor L bármely véges automatában előállítható a végállapotok üres halmazával. Így a továbbiakban feltehetjük, hogy L nem üres. Mivel C kompatibilis, ez azt jelenti, hogy tetszőleges $u, v \in T^*$ mellett $C[u] = C[v] \Rightarrow \forall a \in T: C[ua] = C[va]$ (vagyis tetszőleges $u, v \in T^*$ esetén a u és a v egy osztályba esése azt eredményezi, hogy minden $a \in T$ esetén az ua és a va is egy osztályba fognak esni.) Ekkor viszont jól definiált (azaz az értelmezési tartomány minden elemének valóban pontosan egy elem felel meg az értékkészletben) az a $d: C \times T \to C$ leképezés, melyre minden $u \in T^*$, $a \in T$ pár mellett $d(C[u], a) = C[ua]$. Tekintsük most az ezen d átmeneti függvénnyel rendelkező $DFA=(C, T, C[\lambda], d, F)$ véges automatát (ahol $C[\lambda]$ jelöli a C azon osztályát, mely az üresszót tartalmazza); és F jelöli a C azon részhalmazát, melyre teljesül, hogy az L előáll a F- beli osztályok egyesítéseként. A DFA definíciója értelmében minden $u \in T^*$- ra $d(C[\lambda], u) = C[u]$. Világos, hogy ekkor minden $u \in T^*$ esetén $C[u] \in C'$ pontosan akkor áll fenn, ha $C[\lambda]$- t az u szó a DFA automatában valamely C'- beli állapotába viszi át. Vagyis DFA állapotainak F részhalmazával épp az L nyelvet ismeri fel. Ezzel a tétel bizonyítását befejeztük. ∎

Valamely $A=(Q, T, q_0, d)$ kimenő jel nélküli (nem feltétlenül véges) automata esetén a $\forall u, v \in T^*: (u \varrho_A v \Leftrightarrow \forall q \in Q: d^*(q, u) = d^*(q, v)$- val definiált ϱ kongruenciát az A *Myhill-Nerode féle kongruenciájának* hívjuk, a hozzá tartozó T^*/ϱ fakorfélcsoportot pedig az A automata *karakterisztikus félcsoportjának* nevezzük.

Az A karakterisztikus félcsoportja nyilván úgy is megadható, mint a Q állapothalmaz feletti teljes transzformáció félcsoport azon részfélcsoportja, amelyet az összes, a $d_a(q) = d(q, a)$, $q \in Q$ összefüggésnek eleget tevő $d_a: Q \to Q$ leképezések $\{d_a | a \in T\}$ halmaza generál. Ilyen interpretációban az $S(A)$ elemei azok az $\eta_u: Q \to Q$, $u \in T^*$ alakú leképezések lesznek, melyekre $\eta_u(q) = d^*(q, u)$, $q \in Q$.

5.42. példa - Egy véges automatával nem felismerhető nyelv

Az $L=\{a^m b^n | m>n \geq 0\}$ nyelv véges (kimenő jel nélküli iniciális) automatában nem állítható elő. (a^n egy n hosszúságú, csupa a betűből álló szót jelöl.) Valóban, ha L véges kimenő jel nélküli iniciális automatában előállítható lenne, akkor a Myhill-Nerode tétel szerint lenne a T^*- nak véges indexű olyan C kompatibilis osztályozása, hogy L előáll véges sok osztály egyesítéseként. Mivel L végtelen elemű, s véges sok osztály egyesítéseként előáll, van olyan osztály, mely valamely $i-j>k-l>0$ feltételnek eleget tevő i, j, k, l természetes számokra $a^i b^j$- t és $a^k b^l$- t tartalmazza. Viszont $a^i b^{j+k-l} (=a^i b^j b^{k-l}) \in L$, s ugyanekkor $a^k b^k (=a^k b^l b^{k-l}) \notin L$. Vagyis van olyan $u \in X^*$, hogy $a^i b^j u \in L$ mellett $a^k b^l u \notin L$. Mivel feltételezésünk szerint L előáll (véges sok) C- beli osztály egyesítéseként, L- nek azok és csak azok az osztályok lehetnek a részhalmazai, melyek egyesítéseként L előáll. Ez viszont azt is jelenti, hogy $a^i b^j u$ és $a^k b^l p$ nem tartozhatnak a C osztályozás egy és ugyanazon osztályába. Tehát, feltételezésünknek ellentmondva a C nem kongruencia osztályozás. Ekkor viszont nem lehet az L nyelvet véges kimenő jel nélküli iniciális automatában előállítani. ★

Itt jegyezzük meg, hogy a Myhill-Nerode tétel és a nyelvet elfogadó minimális (véges determinisztikus teljesen definiált) automata szoros kapcsolatban állnak egymással. Ugyanis a bizonyításban előállított *DFA* éppen az L nyelvhez tartozó minimális (állapotszámú) teljesen definiált automata. Itt hívjuk fel a figyelmet arra a nagyon fontos tényre, hogy a teljesen definiált minimális determinisztikus automata (az állapothalmaz izometriájától eltekintve) egyértelműen definiálható minden reguláris nyelvhez. Ez alapján lehet eldönteni, hogy két reguláris nyelv megegyezik-e egymással.

5.6. Véges automaták analízise

Az automaták analízisének problémáját véges automatákra így módosítjuk:

A véges automaták analízise jelentse olyan univerzális algoritmus megadását, amelynek alkalmazásával bármely adott determinisztikus $FA=(Q, T, q_0, d, F)$ véges automatához meg tudjuk adni annak a nyelvnek egy reguláris kifejezését, amely *FA*- ban az F halmazzal előállítható. Ilyen algoritmus létezését S. C. Kleene bizonyította be 1956-ban. Mi a továbbiakban McNaughton és Yamada algoritmusát ismertetjük, amely egyben konstruktív bizonyítását is szolgáltatja a következő tételnek.

26. Tétel. Ha a véges T halmaz feletti L nyelv előállítható véges kimenő jel nélküli iniciális automata állapotai valamely F részhalmazával, akkor az L nyelv megadható reguláris kifejezéssel.

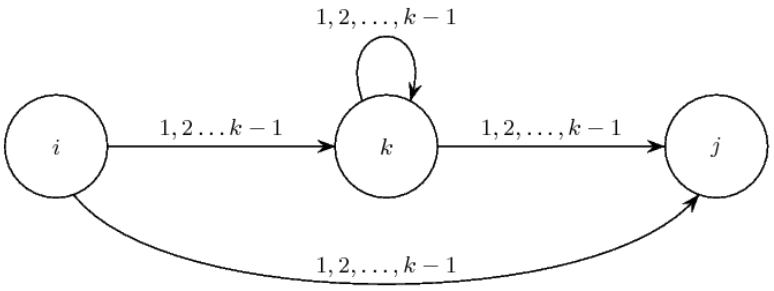
Bizonyítás. A kívánt algoritmus megadásához nyilván elég olyan eljárást megkonstruálni, amely lehetővé teszi az olyan nyelv egy reguláris kifejezésének felírását, amely *FA*- ban tetszőleges, egyetlen állapottal előállítható, hisz ha $F=\{q_1, q_2, \ldots, q_n\}$, akkor $L_{FA}{}^F = L_{FA}{}^{q_1} \cup L_{FA}{}^{q_2} \cup \cdots \cup L_{FA}{}^{q_n}$. Így a továbbiakban csupán az olyan reguláris nyelvek reguláris kifejezéssel történő megadását vizsgáljuk, melyek mindegyike *FA*- ban egy-egy állapottal (azaz az állapotok egy-egy egyelemű részhalmazával) előállítható. Legyen $FA=(\{1, \ldots, n\}, T, 1, d, F)$ egy véges iniciális kimenő jel nélküli automata, melyben az állapothalmaz 1- től alkalmas n- ig terjedő természetes számok halmaza, s az 1 természetes szám a kezdőállapot. Az állapothalmaz és a kezdőállapot ilyen megválasztása nem jelent igazi megszorítást, hisz tetszőleges véges iniciális kimenő jel nélküli automata esetén jelölhetjük 1- el a kezdőállapotot, s amennyiben az állapotok száma n, az $\{1, 2, \ldots, n\}$- el az állapothalmazt (aszerint hogy az állapotok egy tetszőleges, kezdőállapottal kezdődő felsorolásánál beszélünk első, …, n- edik állapotról). Jelölje $L_{i,j}{}^k$ tetszőleges $i, j \in \{1, \ldots, n\}$- re és $k \in \{0, 1, \ldots, n\}$- re azt a nyelvet, mely azokból és csak azokból a bemenő szavakból áll, amelyek hatására az *FA* az $i \in \{1, \ldots, n\}$ állapotból átmegy a $j \in \{1, \ldots, n\}$ állapotba úgy, hogy $k=0$ esetén nincs közbülső állapot, $k>0$ esetén pedig legfeljebb az 1, …, k állapotok léphetnek fel közbülső állapotként. Speciálisan, $L_{i,j}{}^0$ jelöli azt a nyelvet, amelynek elemei hatására *FA* az $i \in \{1, \ldots, n\}$ állapotból közvetlenül, közbülső állapotok nélkül megy át a $j \in \{1, \ldots, n\}$ állapotba. (Itt tehát k nem kitevőt hanem egyszerűen indexet jelöl!)

Elegendő megmutatni, hogy bármely $m \in \{1, \ldots, n\}$ esetén az $L_{1,m}{}^n$ reguláris, hiszen $L_{FA}{}^m = L_{1,m}{}^n$, s elegendő ezen nyelvek egy reguláris kifejezését megadni. Ennél valamivel többet bizonyítunk: Igazolni fogjuk, hogy minden $L_{i,j}{}^k$ nyelv reguláris és rekurzív formulát adunk az ilyen nyelvek egy

reguláris kifejezésének felírásához. Ez egyben azt is jelenti, hogy tételünknek egy teljes indukciós bizonyítását adjuk.

Legyen k=0. Mivel az $L_{i,j}^{k}$ nyelvek mindegyike vagy az üres nyelv, vagy pedig T bizonyos elemeinek egyesítési halmaza, ezért minden ilyen nyelv reguláris és az automata átmeneti táblázata alapján felírható. Nevezetesen, a táblázat i állapothoz tartozó oszlopában kigyűjtjük azokat az állapotokat, melyek j- vel megegyeznek, s tekintjük az összes olyan $a_{i_1}, \ldots a_{i_s} \in T$ bemenő jeleket, melyek az FA átmeneti táblázatában ezen j- vel megegyező elemekkel egy sorba esnek. Ezek a bemenő jelek lesznek azok, melyek hatására az FA átmegy az i állapotából a j állapotába. Ha ilyen bemenő jel nincs és $i \neq j$, akkor $L_{i,j}^{0}=\emptyset$ lesz a tekintett reguláris kifejezés. Amennyiben van olyan bemenő jel, mely az i állapotot a j állapotba viszi át és $i \neq j$, akkor $L_{i,j}^{0}=a_{i_1}+a_{i_2}+\ldots+a_{i_s}$ lesz a megfelelő reguláris kifejezés. Ha nincs az i- t a j- be átvivő bemenő jel, de $i=j$, akkor $L_{i,j}^{0}=\lambda$, hisz az üresszó hatására minden állapot át tud menni önmagába közbülső állapot nélkül, definíció szerint. Végül, ha van olyan bemenő jel, mely az i állapotot a j állapotba viszi át és $i=j$, akkor nyilvánvalóan igaz, hogy $L_{i,j}^{0}=\lambda+a_{i_1}+a_{i_2}+\ldots+a_{i_s}$ megfelelő reguláris kifejezés.

Legyen most k>0, s tegyük fel, hogy az $L_{i,j}^{0}$, …, $L_{i,j}^{k-1}$, $i, j \in \{1, \ldots, n\}$ nyelvek regularitását már bebizonyítottuk és fel is írtuk ezen nyelvek egy-egy reguláris kifejezését. Mutassuk meg, hogy érvényes az $L_{i,j}^{k}=L_{i,j}^{k-1}+L_{i,k}^{k-1}(L_{k,k}^{k-1})^{*}L_{k,j}^{k-1}(i, j, k \in \{1, \ldots, n\})$ formula. Az egyenlőség igazolásához először azt gondoljuk végig, hogy milyen módon tudunk egy $w \in T^{*}$ szóval eljutni az i állapotból a j állapotba úgy, hogy legfeljebb 1, …, k léphetnek fel közbülső állapotként.



Ha valamely $w \in T^{*}$ szóval az i állapotból a j állapotba el tudunk jutni úgy, hogy legfeljebb az 1, 2, …k léphetnek fel közbülső állapotokként, akkor vagy el tudunk jutni ezen w szóval az i állapotból a j állapotba úgy, hogy legfeljebb 1, …, k-1 léphet fel közbülső állapotként (azaz ha k=1 akkor ezesetben nem lép fel közbülső állapot) és ekkor $w \in L_{i,j}^{k-1}$, vagy nem. Utóbbi esetben fel kell lépnie a k állapotnak legalább egyszer, de persze véges sokszor közbülső állapotként. Vagyis ekkor először eljutunk a k állapotba úgy, hogy legfeljebb 1, …, k-1 léphet fel közbülső állapotként, s ha k csak egyszer lépett fel közbülső állapotként, akkor k- ból máris tovább juthatunk j- be úgy, hogy ismét legfeljebb 1, …, k-1 léphet fel közbülső állapotként. Ezesetben tehá $w=w_1w_2$, ahol $w_1 \in L_{i,k}^{k-1}$, $w_2 \in L_{k,j}^{k-1}$. Az az eset van még hátra, mikor k egynél többször (de persze véges sokszor) lép fel közbülső állapotként. Ekkor is először eljutunk a k állapotba úgy, hogy legfeljebb 1, …, k-1 léphet fel közbülső állapotként, majd véges sokszor k- ból elindulva visszatérünk k- ba úgy, hogy ismét legfeljebb 1, …, k-1 léphessenek fel közbülső állapotként, s végül mint az előző esetben, k- ból tovább juthatunk j- be úgy, hogy ismét legfeljebb 1, …, k-1 léphet fel közbülső állapotként. Ilyenkor w előáll valamely t>2 természetes számra $w=w_1w_2\ldots w_{t-1}w_t$ alakban úgy, hogy $w_1 \in L_{i,k}^{k-1}$, w_2, …, $w_{t-1} \in L_{k,k}^{k-1}$, illetve $w_t \in L_{k,j}^{k-1}$. Minden lehetséges esetben azt kaptuk, hogy ha w benne van az egyenlőségünk baloldalán szereplő nyelvben, akkor benne van az egyenlőség jobboldalán álló nyelvben. Az egyenlőség fennállásához be kell még látni az is, hogy ha $w \in T^{*}$ benne van az egyenlőség jobboldalán lévő nyelvben akkor benne van a baloldalán lévőben is. $L_{i,j}^{k-1} \subseteq L_{i,j}^{k}$ definíció szerint fennáll, így csak azzal az esettel kell foglalkoznunk, mikor $w \in L_{i,k}^{k-1}(L_{k,k}^{k-1})^{*}L_{k,j}^{k-1}$. Ekkor $w=w_1\cdots w_s$, $s \geq 1$, ahol w_1 az i állapotot a j állapotba, w_s a k állapotot a j állapotba, s ha s>2, akkor minden egyes $w_l, l \in \{2, \ldots, s\text{-}1\}$ tényező a k állapotból önmagába viszi át az FA automatát úgy, hogy legfeljebb 1, …, k-1 léphetnek fel közbülső állapotként. Világos,

hogy ekkor w hatására úgy juthatunk el az i állapotból a j állapotba, hogy legfeljebb 1, …, k léphetnek fel közbülső állapotként. Ezzel a tétel teljes bizonyítást nyert. ∎

Az előző bizonyításban szereplő konstrukció segítségével tetszőleges véges automatához elkészíthetjük a megfelelő reguláris kifejezést.

5.43. példa - Ekvivalens reguláris kifejezés megadása véges automatához 1. feladat

Adjunk meg az $A=(\{a_1,a_2\},\{x,y\},a_1,\delta,\{a_2\})$ determinisztikus véges automatával ekvivalens reguláris kifejezést! Ahol az átmenetfüggvény a következő:

δ	a_1	a_2
x	a_2	a_2
y	a_1	$\varnothing$

Megoldás:
(I.) Jelöljük n-el az automata állapotainak számát. (Jelen eseben n=2.)
A feladat megoldásához előállítjuk az r_{ij}^{k} reguláris kifejezéseket, a következő rekurzív definíció segítségével:
$r_{ij}^{0}=\{x|\delta(a_i,x)=a_j\}$, ha $i \neq j$,
$r_{ij}^{0}=\{x|\delta(a_i,x)=a_j\}\cup\{\lambda\}$, ha $i=j$,
$r_{ij}^{k}=r_{ik}^{k-1}(r_{kk}^{k-1})^{*}r_{kj}^{k-1}\cup r_{ij}^{k-1}$, ha $1 \leq k \leq n$.

Jelen esetben r_{ij}^{k} reguláris kifejezések:

δ	k**=0**	k**=1**	k**=2**
r_{11}^{k}	$y\cup\lambda$	y^{*}	y^{*}
r_{12}^{k}	x	$y^{*}x$	$y^{*}xx^{*}$
r_{21}^{k}	$\varnothing$	$\varnothing$	$\varnothing$
r_{22}^{k}	$x\cup\lambda$	$x\cup\lambda$	x^{*}

Például az $r_{11}^{1}=r_{11}^{0}(r_{11}^{0})^{*}r_{11}^{0}\cup r_{11}^{0}=(y\cup\lambda)(y\cup\lambda)^{*}(y\cup\lambda)\cup(y\cup\lambda)=y^{*}$ és
az $r_{12}^{2}=r_{12}^{1}(r_{22}^{1})^{*}r_{22}^{1}\cup r_{12}^{1}=y^{*}x(x\cup\lambda)^{*}(x\cup\lambda)\cup y^{*}x=y^{*}xx^{*}$.

(II.)

Ezek után $L(A)=\bigcup_{r_m\in F} r_m^{n}$, ahol r_1 a kezdőállapot, az r_m pedig befutja a végállapotok halmazát. Jelen esetben egyetlen végállapotunk van, így $L(A)=r_{12}^{2}=y^{*}xx^{*}$. Megfigyelhető, hogy k=n esetén elegendő azokat az r_{ij}^{k} elemeket kiszámolni, ahol i kezdőállapot és j végállapot. A továbbiakban ezek szerint fogunk eljárni. ★

5.44. példa - Ekvivalens reguláris kifejezés megadása véges automatához 2. feladat

Adjunk meg az $A=(\{a_1,a_2\},\{x,y,z\},a_1,\delta,\{a_1,a_2\})$ determinisztikus véges automatával ekvivalens reguláris kifejezést, ahol a δ átmenetfüggvény a következő táblázattal adott:

δ	a_1	a_2
x	a_2	$\varnothing$
y	a_1	a_2
z	$\varnothing$	a_1

Megoldás:

(I.)

Az r_{ij}^{k} reguláris kifejezések:

δ	$k=0$	$k=1$
r_{11}^{k}	$y\cup\lambda$	y^*
r_{12}^{k}	x	y^*x
r_{21}^{k}	z	zy^*
r_{22}^{k}	$y\cup\lambda$	$zy^*x\cup y\cup\lambda$

(II.)
$r_{11}^{2}=r_{12}^{1}(r_{22}^{1})^*r_{21}^{1}\cup r_{11}^{1}=y^*x(zy^*x\cup y\cup\lambda)^*zy^*\cup y^*=$
$=y^*x(zy^*x\cup y)^*zy^*\cup y^*,$

$r_{12}^{2}=r_{12}^{1}(r_{22}^{1})^*r_{22}^{1}\cup r_{12}^{1}=y^*x(zy^*x\cup y\cup\lambda)^*(zy^*x\cup y\cup\lambda)\cup y^*x=$
$=y^*x(zy^*x\cup y)^*.$

Végül $L(A)=r_{11}^{2}\cup r_{12}^{2}=(y^*x(zy^*x\cup y)^*zy^*\cup y^*)\cup(y^*x(zy^*x\cup y)^*)=$
$=y^*x(zy^*x\cup y)^*(zy^*\cup\lambda)\cup y^*$. ★

5.45. példa - Ekvivalens reguláris kifejezés megadása véges automatához 3. feladat

Adjunk meg az $A=(\{a_1,a_2,a_3\},\{x,y\},a_1,\delta,\{a_2,a_3\})$ determinisztikus véges automatával ekvivalens reguláris kifejezést, ahol a δ átmenetfüggvény:

δ	a_1	a_2	a_3
x	a_2	a_3	a_1
y	a_3	a_2	$\varnothing$

Megoldás:

(I.)

Az r_{ij}^{k} reguláris kifejezések:

δ	k**=0**	k**=1**	k**=2**
r_{11}^{k}	$\varnothing$	$\varnothing$	$\varnothing$
r_{12}^{k}	x	x	xy^*
r_{13}^{k}	y	y	$y \cup xy^*x$
r_{21}^{k}	$\varnothing$	$\varnothing$	$\varnothing$
r_{22}^{k}	$y \cup \lambda$	$y \cup \lambda$	y^*
r_{23}^{k}	x	x	y^*x
r_{31}^{k}	x	x	x
r_{32}^{k}	$\varnothing$	xx	xxy^*
r_{33}^{k}	λ	$\lambda \cup xy$	$\lambda \cup xy \cup xxy^*x$

(II.)

$r_{12}^{3}=(y \cup xy^*x)\ (\lambda \cup xy \cup xxy^*x^*xxy^* \cup xy^*=$
$=(y \cup xy^*x)(xy \cup xxy^*x)^*xxy^* \cup xy^*,$

$r_{13}^{3}=(y \cup xy^*x)(\lambda \cup xy \cup xxy^*x)^*\ (\lambda \cup xy \cup xxy^*x) \cup (y \cup xy^*x)=$
$=(y \cup xy^*x)(xy \cup xxy^*x)^*,$

$L(A)=r_{12}^{3} \cup r_{13}^{3}=((y \cup xy^*x)\ (xy \cup xxy^*x)^*xxy^* \cup xy^*) \cup ((y \cup xy^*x)(xy \cup xxy^*x)^*)=$
$(y \cup xy^*x)\ (xy \cup xxy^*x)^*(xxy^* \cup \lambda) \cup xy^*$ ★

5.7. Véges automaták szintézise

Az automaták szintézisének problémáját véges automatákra a következőképp módosítjuk: A véges automaták szintézise jelentse olyan univerzális algoritmus megadását, amelynek alkalmazásával bármely véges T halmaz feletti, reguláris kifejezéssel megadott L reguláris nyelvhez meg tudunk

konstruálni olyan véges $FA=(Q, T, q_0, d, F)$ automatát, amelyben az adott L nyelv az állapothalmaz F részhalmazával előállítható, (vagyis legyen FA az L reguláris kifejezéssel megadott nyelv felismerő automatája). Egy ilyen algoritmus megadása egyben a következő tétel konstruktív bizonyítását szolgáltatja:

27. Tétel. Egy véges T halmaz feletti tetszőleges r reguláris kifejezéshez megadható olyan FA véges felismerő automata amely az r által leírt nyelvet fogadja el.

Bizonyítás. A tétel első bizonyítása ugyancsak S. C. Kleene eredménye 1956-ból. Mi egy V. M. Gluskovtól származó egyszerű eljárást ismertetünk, amely eléggé gazdaságos is abban az értelemben, hogy az esetek többségében a minimálishoz közeli automatát eredményez. Másrészt lehetővé teszi, hogy az adott véges sok, reguláris kifejezéssel megadott reguláris nyelv esetén egyszerre szerkesszünk meg egy olyan véges automatát, amelyben a nyelvek mindegyike előállítható (általában az állapothalmaz más-más részhalmazaival).

A módszer lényege, hogy számba vesszük azoknak a szavaknak a struktúráját, amelyek az adott nyelvek közül legalább egyikben előfordulnak. Az eljárás ismertetése során a könnyebb megértés kedvéért egy példát is kidolgozunk. Legyenek $L_1, L_2, \ldots, L_k$ olyan nyelvek a $T=\{a^{(1)}, a^{(2)}, \ldots, a^{(n)}\}$ ábécé felett (az $1, 2, \ldots, n$ itt nem kitevőt, hanem indexet jelentenek), amelyek reguláris kifejezésekkel vannak megadva. A módszer alkalmazása előtt a nyelvalgebra alapműveleteire megismert műveleti azonosságok alkalmazásával hozzuk a reguláris kifejezéseket minél rövidebb alakra. Ezt követően indexeljük a reguláris kifejezésekben előforduló betűket balról jobbra haladva úgy, hogy az azonos betűk különböző előfordulási helyükön más-más indexet kapjanak.

5.46. példa - Automata megadása reguláris kifejezéshez

Legyenek az előállítandó nyelvek $L_1=aa^*$, $L_2=bb^*$, $L_3=(aa^*b+bb^*a)(a+b)^*$.

Ekkor egy lehetséges indexelés eredménye:

$L_1=a_1a_2^*$, $L_2=b_3b_4^*$, $L_3=(a_5a_6^*b_7+b_8b_9^*a_{10})(a_{11}+b_{12})^*$.

Legyenek u és v indexelt betűk. Azt mondjuk, hogy az u indexelt betű *megelőzi* a v indexelt betűt, jelekben: $u \ll v$, ha az u index nélküli u' és v index nélküli v' formájához létezik olyan p szó, mely benne van legalább egy adott $L_i (1 \le i \le k)$ nyelvben és előáll $p=wu'v'z$ alakban valamely $w, z \in T^*$- ra. Az u indexelt betűt *kezdőbetűnek* nevezzük, ha index nélkül formája első betűje legalább egy olyan szónak, amely legalább egy $L_i (1 \le i \le k)$- ben benne van. Végül, az u indexelt bezűt *záróbetűnek* hívunk, ha index nélküli formája utolsó betűje olyan szónak, mely legalább egy $L_i (1 \le i \le k)$- ben benne van.

Az 5.46. példa - Automata megadása reguláris kifejezéshez megoldásának folytatása: betűk osztályozása

Példánkban a kezdőbetűk: a_1, b_3, a_5, b_8.

Záróbetűk: $a_1, a_2, b_3, b_4, b_7, a_{10}, a_{11}, b_{12}$.

Emellett:

$a_1 \ll a_2$;	$b_7 \ll a_{11}, b_{12}$;
$a_2 \ll a_2$;	$b_8 \ll b_9, a_{10}$;
$b_3 \ll b_4$;	$b_9 \ll b_9, a_{10}$;
$b_4 \ll b_4$;	$a_{10} \ll a_{11}, b_{12}$;
$a_5 \ll a_6, b_7$;	$a_{11} \ll a_{11}, b_{12}$;
$a_6 \ll a_6, b_7$;	$b_{12} \ll a_{11}, b_{12}$;

Ezek után a keresett véges felismerő $FA=(Q, T, q_0, d, F)$ automatát a következő módon definiáljuk:

Legyen q_0 tetszőleges szimbólum. Legyen ezután $d(q_0, a^{(i)})$ minden i=1, …, n- re az $a^{(i)}$ betű azon indexelt változatainak azon $q_1^{(i)}$ halmaza, melyek kezdőbetűk. Ily módon definiáljuk először a $q_1^{(1)}$, $q_1^{(2)}$, …, $q_1^{(n)}$ halmazokat mint *FA*- beli állapotokat. Megjegyezzük, hogy ha valamelyik *T*- beli $a^{(i)}$ betű egyik indexelt változata sem kezdőbetű, akkor $q^{(i)}$ az üres halmaz.

A többi állapotot rekurzióval definiáljuk a következő módon: Ha már egy q állapot definiálva van, akkor minden i=1, …, n- re a $d(q, a^{(i)})$ állapot legyen az $a^{(i)}$ betű azon indexelt változatainak halmaza, melyeket megelőz legalább egy q- beli betű. (Ez a q- beli indexelt betű természetesen nem szükségképp az $a^{(i)}$ betű indexelt változata.) Természetesen előfordulhat az is, hogy az $a^{(i)}$ betűnek nincs egyetlen ilyen indexelt változata sem, s ekkor $d(q, a^{(i)})$ az üres halmaz. Az is előfordulhat, hogy q maga is az üres halmaz. Ezesetben $d(q, a^{(i)})=q$ fog fennállni a konstrukciónk értelmében.

Világos, hogy az így definiált *FA* automata iniciálisan összefüggő, azaz a kezdőállapotából alkalmas bemenő szavakkal bármely állapota elérhető. Az is világos, hogy véges, hisz állapotainak száma legfeljebb annyi mint a bemenő ábécé összes részhalmazainak halmaza, azaz 2^n+1. Végül, konstrukciónk alapján minden $p \in L_i$, i=1, …, k nem üresszó esetén a p szó az így megkonstruált automatát a kezdőállapotból egy olyan állapotba viszi át, melynek legalább egy indexelt betűje az L_i nyelv záróbetűje. Ugyancsak konstrukciónk alapján látható, hogy ha $q \notin L_i$ akkor ez nem áll fenn. Tehát minden i=1, …, k- ra az $L_i \setminus \{\lambda\}$ nyelv előállítható az *FA* automatában azzal az F_i halmazzal, amely azokból az állapotokból áll, melyek legalább egy L_i- beli záróbetűt tartalmaznak. Amennyiben az üresszó eleme L_i- nek, akkor F_i- hez még hozzá kell venni a q_0 kezdőállapotot is.

Az 5.46. példa - Automata megadása reguláris kifejezéshez folytatása: a megkonstruált automata:

Példánkban az *FA* automata megszerkesztése így alakul:

$d(q_0, a)=\{a_1, a_5\}=q_1$,	$d(q_4, b)=\{b_{12}\}=q_8$,
$d(q_0, b)=\{b_3, b_8\}=q_2$,	$d(q_5, a)=\{a_{11}\}=q_7$,
$d(q_1, a)=\{a_2, a_6\}=q_3$,	$d(q_5, b)=\{b_{12}\}=q_8$,
$d(q_1, b)=\{b_7\}=q_4$,	$d(q_6, a)=\{a_{11}\}=q_7$,
$d(q_2, a)=\{a_{10}\}=q_5$,	$d(q_6, b)=\{b_{12}\}=q_8$,
$d(q_2, b)=\{b_4, b_9\}=q_6$,	$d(q_7, a)=\{a_{11}\}=q_7$,
$d(q_3, a)=\{a_2, a_6\}=q_3$,	$d(q_7, b)=\{b_{12}\}=q_8$,
$d(q_3, b)=\{b_7\}=q_4$,	$d(q_8, a)=\{a_{11}\}=q_7$,
$d(q_4, a)=\{a_{11}\}=q_7$,	$d(q_4, b)=\{b_{12}\}=q_8$.

A kapott q_0 kezdőállapotú automata átmenet táblázata:

FA	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
a	q_1	q_3	q_5	q_3	q_7	q_7	q_5	q_7	q_7
b	q_2	q_4	q_6	q_4	q_8	q_8	q_6	q_8	q_8

Ebben az automatában előállítható

az L_1 nyelv az $F_1=\{q_1, q_3\}$ végállapothalmazzal,

az L_2 nyelv az $F_2=\{q_2, q_6\}$ végállapothalmazzal,

az L_3 nyelv az $F_3=\{q_4, q_5, q_7, q_8\}$ végállapothalmazzal. ▮ ★

Ezzel tehát beláttuk, hogy a reguláris kifejezések által leírt nyelvosztály éppen a véges automatákkal felismerhető nyelvek osztálya, vagyis a Chomsky-féle 3. típusú nyelvek osztálya.

Ezek szerint egy reguláris nyelvet megadhatunk reguláris nyelvtannal, véges (determinisztikus vagy nemdeterminisztikus) felismerő automatával vagy reguláris kifejezéssel (illetve szintaxis gráffal a Szintaxis gráf alfejezetben ismertetett módon).

5.8. Véges automaták által indukálható leképezések

Ebben a részben a véges automaták által indukálható leképezéseket (lásd Az automaták által indukált leképezések fejezet) a reguláris nyelvekkel kapcsolatosan vizsgáljuk.

Érvényes a következő állítás.

3. Segédtétel. Tetszőleges $\varphi:T^* \to V^*$ automata leképezés esetén egy $b\in V$ akkor és csak akkor lesz valamely $w\in T^*$- ra a $\varphi(w)$ képszó valamely betűje, ha van olyan $w'\in T^*$, hogy b a $\varphi(w')$ képszó utolsó betűje.

Bizonyítás. Legyen $\varphi:T^* \to V^*$ tetszőleges automata leképezés, s tételezzük fel, hogy valamely $w\in T^*$ esetén a $b\in Y$ betű előfordul a $\varphi(w)$ képszóban. Ekkor $\varphi(w)$ előáll $\varphi(w)=pbr$ alakban, ahol p, $r\in V^*$. Mivel φ hossztartó, $|w|=|\varphi(w)|$. Így w előáll $w=uav$ alakban, ahol $|u|=|p|$, $a\in T$ és $|v|=|r|$. Ráadásul a hossztartó tulajdonság miatt $\varphi(ua)$ és pb hossza is szükségképp megegyezik. Másrészt φ kezdőszelet tartó, azaz $\varphi(ua)=pb$. Ha tehát $b\in V$ előfordul egy képszóban, akkor előfordul egy (esetleg másik) képszó utolsó betűjeként is. ∎

A továbbiakban olyan $\varphi:T^* \to V^*$ automata leképezésekre szorítkozunk, ahol minden $b\in V$ előfordul legalább egy $w\in T^*$ szó képszavában. Fenti segédtételünk értelmében ekkor minden $b\in V$ elő fog fordulni legalább egy $w\in T^*$ szó képszavának utolsó betűjeként.

Legyenek T és V véges halmazok és legyen $\varphi:T^* \to V^*$ tetszőleges (nem feltétlen véges állapotú) automata leképezés. Megmutatjuk, hogy φ jellemezhető nyelvek egy alkalmas rendszerével. Tetszőleges $b\in V$- re legyen $L_b=\{w\in T^* | \gg\varphi(w)=b\}$. Más szóval, az L_b legyen azoknak a T^*- beli w szavaknak a halmaza, amelyek a φ leképezésnél a b- re végződő szóba esnek át. Tekintsük az ilyen L_b nyelvek halmazát: $H_\varphi=\{L_b | b\in V\}$. H_φ eleget tesz a következő feltételeknek:

(a) H_φ véges sok elemből áll.

(b) $\forall b, b'\in V: b\neq b' \Rightarrow L_b\cap L_{b'}=\emptyset$.

(c) Minden nem üres $w\in T^*$- hoz van olyan $b\in V$, hogy $w\in L_b$.

(d) A λ üresszó nincs benne egyik L_b- ben sem.

Az (a)-(d) feltételek jelentése az, hogy az $L_b (b\in V)$ nyelvek a $T^*\setminus\{\lambda\}$ halmaz egy véges indexű osztályozását alkotják. Az (a)-(d) feltételeknek eleget tevő nyelvrendszereket *automata nyelvrendszereknek* nevezzük.

28. Tétel. Ha a T és V véges halmazok, akkor minden $\varphi:T^* \to V^*$ automata leképezéshez hozzárendelhető a T feletti nyelvalgebra egy H_φ automata nyelvrendszere. Megfordítva, a véges T ábécé feletti nyelvalgebra bármely H automata nyelvrendszere a véges V halmaz elemeinek jelölésétől eltekintve egyértelműen meghatározza azt a $\varphi:T^* \to V^*$ automata leképezést, melyre $H_\varphi=H$.

Bizonyítás. A tétel első felét már beláttuk. Második felének bizonyításához legyen H az LA_Y nyelvalgebra egy tetszőleges automata nyelvrendszere és legyen $H=\{H_1, \ldots, H_n\}$. Legyen továbbá $V=\{1, \ldots, n\}$ és határozzuk meg azt a $\varphi:T^* \to V^*$ automata leképezést, melyre tetszőleges $w=a_1\cdots a_k\in T^*$, $a_1, \ldots, a_k\in T$ szó esetén $\varphi(w)=i_1\cdots i_k$, ahol $a_1\in H_{i_1}$, $a_1a_2\in H_{i_2}$, $\ldots$, $a_1\cdots a_k\in H_{i_k}$. Ez a φ leképezés nyilván automata leképezés és rá $H_\varphi=H$ teljesül. ∎

A tétel azt fejezi ki, hogy a véges T és V halmazokhoz tartozó $\varphi:T^* \to V^*$ alakú automata leképezések a V elemeinek jelölésétől eltekintve egy-egy értelmű módon jellemezhetők nyelvek

bizonyos rendszerével. Így valóban eljutottunk az automata leképezések olyan megadási módjához, mely a korábban kirótt feltételeinket teljesíti.

Egy A automatához tartozó K_A kanonikus nyelvrendszeren az A automata által indukált leképezéshez tartozó automata nyelvrendszert értjük. Képletben: $K_A=H_{\varphi_A}$. (Itt feltesszük, hogy A iniciális automata, melynek bemenő és kimenő jelhalmaza véges.) A definícióból közvetlenül adódik:

29. Tétel. Egy véges T és V halmazokkal rendelkező $A=(Q, T, V, q_0, d, f)$ Mealy automata akkor és csak akkor indukálja a $\varphi:T^* \rightarrow V^*$ automata leképezést, ha a φ- hez tartozó automata nyelvrendszer egybeesik az A- hoz tartozó kanonikus nyelvrendszerrel. ∎

Ebben a fejezetben eddig végig feltételeztük, hogy az T és V halmazok végesek. Most tovább szűkítjük vizsgálataink körét és az állapothalmaz végességét is kikötjük. Más szóval, a továbbiakban csupán véges automatákkal foglalkozunk. Arra vonatkozóan, hogy milyen leképezések indukálhatóak véges automatákkal, érvényes a következő tétel.

30. Tétel. (Véges Automaták Alaptétele) A véges T és V halmazok esetén egy $\varphi:T^* \rightarrow V^*$ automata leképezés akkor és csak akkor indukálható véges automatában, ha a φ- hez tartozó H_φ automata nyelvrendszer minden eleme reguláris nyelv.

Bizonyítás. A szükségesség igazolásához tegyük fel, hogy T és V véges halmazok, s a $\varphi:T^* \rightarrow V^*$ automata leképezés indukálható a véges $A=(Q, T, V, q_0, d, f)$ automatával. Akkor Gill tétele szerint van olyan véges $A'=(Q', T, V, q_0, d', g)$ Moore-automata is, mely ugyancsak indukálja a φ leképezést. Minden $b\in V$- re legyen $Q'_b=\{q'\in Q' | g(q')=b\}$, azaz legyen Q'_b az A' automata b állapotjelű állapotainak halmaza és jelölje L'_b azt a nyelvet, mely az $A''=(Q', T, q_0, d')$ kimenő jel nélküli automatában a Q'_b halmazzal előállítható (vagyis a (Q', T, q_0, d', Q'_b) által elfogadott nyelv), azaz legyen $L'_b=L_{A''}{}^{Q'_b}$. Mivel az A véges, az analízisre vonatkozó tétel alkalmazásával kapjuk, hogy L'_b minden $b\in V$- re reguláris nyelv. Ezért a szükségességhez azt kell csupán megmutatni, hogy ha $H_\varphi=\{L_b | b\in V\}$, akkor minden $b\in V$- ra $L_b=L'_b$. Ez pedig a következő számolással adódik: $w\in L_b \Leftrightarrow \gg \varphi(w)=b \Leftrightarrow \gg \varphi_A(w)=b \Leftrightarrow \varphi_{A'}(w)=b \Leftrightarrow \gg g(d(q_0, w))=b \Leftrightarrow \gg d(q_0, w)\in Q'_b \Leftrightarrow w\in L'_b$. Ezzel a szükségesség igazolását befejeztük.

Az elegendőség kimutatásához tegyük fel, hogy a véges T és V halmazokra megadott $\varphi:T^* \rightarrow V^*$ automata leképezés $H_\varphi=\{L_{b_1}, \ldots, L_{b_m}\}$ automata nyelvrendszeréhez létezik olyan $A''=(Q, T, q_0, d)$ iniciális kimenő jel nélküli automata, amelyben az $L_{b_1}, \ldots, L_{b_m}$ nyelvek rendre előállíthatók a Q állapothalmaz $Q_1, \ldots, Q_m$ részhalmazaival. Bővítsük ki az $A''=(Q, T, q_0, d)$ automatát az $A=(Q, T, V, q_0, d, g)$ kimenő jellel bíró Moore-automatává a g jelfüggvény következő definíciójával: Legyen $g(q_0)$ egy tetszőlegesen rögzített V- beli elem, továbbá legyen minden $q\in Q_i$, $i\in\{1, \ldots, m\}$ mellett $g(q)=b_i$. Minthogy H_φ automata nyelvrendszer, azért a $Q_1, \ldots, Q_m$ páronként diszjunkt halmazok. Így g definíciója egyértelmű. Megmutatjuk, hogy az A Moore-automata indukálja a φ leképezést. Ehhez az előzőek alapján elegendő kimutatni, hogy a $K_A=\{R_{b_i} | b_i\in V\}$ kanonikus nyelvrendszer egybeesik a H_φ automata nyelvrendszerrel. Ez pedig a következő számolással adódik: $w\in R_{b_i} \Leftrightarrow \gg \varphi_A(w)=b_i \Leftrightarrow \gg g(d(q_0, w))=b_i \Leftrightarrow \gg d(q_0, w)\in Q_i \Leftrightarrow w\in L_{b_i}$.

Ezzel az elegendőség igazolását is befejeztük. ∎

Az analízisre és a szintézisre vonatkozó tétel a most bebizonyított tétellel együtt - konstruktív bizonyításukat konkrét esetre alkalmazva - algoritmust szolgáltat az analízis és a szintézis eredeti értelemben vett megoldására.

5.47. példa - Automata nyelvrendszer

Legyen $T=\{a, b\}$ és az $T^*\setminus\{\lambda\}$ félcsoportot bontsuk fel a következő módon:

L_1 legyen az összes, csupa a jelből álló szavak halmaza,

L_2 legyen az összes, csupa b jelből álló szavak halmaza,

L_3 legyen az összes többi szavak halmaza.

Ez a nyelvrendszer automata nyelvrendszer, mégpedig minden eleme reguláris nyelv, hiszen nyilván $L_1=aa^*$, $L_2=bb^*$, $L_3=(aa^*b+bb^*a)(a+b)^*$. Vegyük észre, hogy a véges automaták szintézisénél tárgyalt Gluskov módszert épp erre a három nyelvre alkalmaztuk, így az ott kapott 9 állapotú A automata játszhatja az alaptétel elegendősége bizonyítása során A"- vel jelölt kimenő jel nélküli automata szerepét. Ha most ezt az automatát kimenő jellel bíró automatává egészítjük ki úgy hogy az L_1, L_2, L_3 nyelvekhez rendre hozzárendeljük az u, v, w szimbólumokat, s a jelfüggvényt az alaptétel elegendősége bizonyításánál látott módon definiáljuk, akkor az alábbi Moore-automatához jutunk, mely az L_1, L_2, L_3 nyelvrendszerrel megadott leképezést indukálja. Az Aufenkamp-Hohn algoritmussal meghatározható az ezen leképezést indukáló minimális (négy állapotú) automata. Ennek ismertetését mellőzzük.

A	tetsz.	u	v	u	w	w	v	w	w
	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
a	q_1	q_3	q_5	q_3	q_7	q_7	q_5	q_7	q_7
b	q_2	q_4	q_6	q_4	q_8	q_8	q_6	q_8	q_8

★

Eredményeink alapján algoritmust tudunk konstruálni annak eldöntésére is, hogy két reguláris kifejezés ekvivalens-e, azaz egy és ugyanazon reguláris nyelv reguláris kifejezéseiről van-e szó. Gluskov módszerével ugyanis mindkettőhöz meg tudunk szerkeszteni egy-egy iniciális kimenő jel nélküli automatát, melyek állapotaik alkalmas részhalmazával ezeket a nyelveket felismerik. Vehetjük a véges determinisztikus felismerő automatákat és minimalizálhatjuk őket. Ezután már csak azt kell eldöntenünk, hogy a kapott két automata állapot-izomorf-e egymással úgy, hogy egy alkalmas izomorfizmus a kezdőállapotokat egymáshoz rendeli. Véges automatákra ez nyilván eldönthető. Ha igen a válasz, akkor a két reguláris kifejezés ekvivalens, különben nem.

A fentiek alapján algoritmus adható egy reguláris kifejezéssel megadott reguláris nyelvhez a legrövidebb vele ekvivalens reguláris kifejezés meghatározásához is: Végig kell vizsgálni az összes, a mi reguláris kifejezésünknél nem hosszabb olyan reguláris kifejezéseket, melyek összes változói legalább egyszer előfordulnak az adott reguláris kifejezésben. A vizsgálat során ki kell választani azokat a reguláris kifejezéseket, melyek ekvivalensek az adott reguláris kifejezéssel, s a legrövidebbet (vagy ha több ilyen van akkor az egyik legrövidebbet) meg kell tartanunk.

Cél lehet egy adott reguláris kifejezéshez megtalálni egy vele ekvivalens olyan reguláris kifejezést, melyben terminálisok a lehető legkevesebbszer fordulnak elő. Nyilván csak azok a reguláris kifejezések jöhetnek számításba, melyekben a terminálisok előfordulási száma nem nagyobb, mint az adott reguláris kifejezésben. Ha ez a szám és a zárójel-párok száma ismert, akkor felső korlátot tudunk adni a vizsgálandó reguláris kifejezések hosszára. Nevezetesen, legfeljebb annyi iterációs jelet tartalmazhat, mint a zárójel-párok száma plusz a terminálisok előfordulási száma. Az összeadásjelek és a szorzások számának összege ennél az értéknél kisebb, mert a legelső előfordulás elé (ami vagy egy kezdő zárójel, vagy egy terminális) nem teszünk műveleti jelet. Belátható, hogy a nem elhagyható zárójelpárok száma kisebb mint a terminálisok előfordulási száma. Vagyis ha csupán nem elhagyható zárójeleket tételezünk fel, a minimális terminális előfordulási számú reguláris kifejezés (vagy ha több van akkor ezek egyike) meghatározható.

5.9. Szóprobléma

A reguláris nyelvek esetén a szóprobléma nyagyon hatékonyan megoldható. Ha megszerkesztünk egy az adott nyelvet elfogadó determinisztikus véges automatát, akkor annak segítségével a szót betűnként elolvasva végig követve az automata futását (legkésőbb) a szó végére érve megkapjuk a választ a kérdésre: ha végállapotba jutottunk a szó végén, akkor a szó benne van az adott reguláris nyelvben; ha nem végállapotba jutottunk, vagy (parciális automata esetén) időközben elakadtunk a feldolgozással, akkor a keresett szó nincs a nyelvben.

Tehát a probléma valós időben megoldható, ahány betűből áll az input szó, annyi lépés után tudjuk a választ.

Hasonlóan hatékony algoritmus készíthető, ha $G=(N, T, S, H)$ erős normálformájú reguláris nyelvtannal adott a nyelv. Ekkor egy lineáris felismerési mátrixot készíthetünk (a mátrix mezőibe (celláiba) a nemterminálisok részhalmazai fognak kerülni).

Legyen a w elemzendő szó hossza n, ekkor egy 1 soros $n+1$ mezőből álló sormátrixot készítünk, a mezőket 0-tól n- ig számozzuk. Az i- edik mező fölé odaírjuk az input szó i- edik betűjét, a_i- t, a 0. mezőbe pedig a nyelvtan S mondatszimbólumát.

Ezután a cellákat balról jobbra haladva töltjük ki: az i- edik ($0<i<n$) cellába beírunk egy A nemterminálist, pontosan akkor ha az (i-1)-edik cellában van olyan B nemterminális, amelyre $B \rightarrow a_iA \in H$.

Az n- edik cella kitöltése: akkor írunk egy + jelet a cellába, ha van olyan nemterminális B az (n-1)-edik cellában, melyre $B \rightarrow a_n \in H$. A w szó pontosan akor van benne az $L(G)$ generált nyelvben, ha az n- edik cellába + került.

Az ismertetett algoritmus éppen a nemdeterminisztikus üresszóátmenet nélküli automata lehetséges futásait szimulálja az adott bemenő szóra.

5.10. Iterációs lemma reguláris nyelvekre

A következő tétel egy hatékony eszköz lehet annak megmutatására, hogy valamely nyelv nem reguláris. A tételt iterációs- vagy pumpáló lemma néven szokás emlegetni.

31. Tétel. Legyen L reguláris nyelv. Ekkor létezik az L- hez olyan n konstans, hogy ha z egy n- nél hosszabb L- beli szó, akkor alkalmas u, v, w szavakra teljesülnek a következők: $z=uvw$, $|uv| \leq n$, $|v|>0$, továbbá tetszőleges nemnegatív i- re $uv^iw \in L$.

Bizonyítás. Legyen L reguláris, ekkor legyen $FA=(Q, T, q_0, d, F)$ minimális determinisztikus automata úgy, hogy $L(FA)=L$. Legyen $n=|Q|+1$. Ekkor bármely $z \in L$, $|z|>n$ szót is tekintjük az automata bemenetének, biztosan van olyan állapot, amelyet FA a szó elfogadása során legalább kétszer érint, sőt az első n lépés alatt is lennie kell ilyen q állapotnak. Bontsuk fel z szót ennek megfelelően: legyen u a z olyan prefixe amely a kezdőállapotból (először) q- ba viszi az automatát, v pedig olyan, amely az automatát q- ból indulva ugyancsak q- ba viszi (másodszor). Ekkor $|uv| \leq n$ fennáll. Mivel a q két előfordulása közt az automatának legalább egy lépése történt, azaz legalább egy input betűt beolvasott $|v|>0$ is teljesül.

Amennyiben az automata a q állapotból indulva a w szót olvassa, elfogadó állapotba kerül. Ennek megfelelően az uw inputot FA elfogadja. Hasonlóan minden uv^iw ($i>0$) alakú szót is elfogad, hiszen u a q állapotba viszi, ahonnan indulva a v^i szavak elolvasása után ugyancsak q állapotba jutunk, végül innen w egy végállapotba vezet.

A tétel igazolását ezzel befejeztük. ∎

5.48. példa - Iterációs lemma alkalmazása

Legyen $L=\{a^mb^m | m>0\}$. Felhasználva a reguláris pumpáló lemmát, kimutatjuk, hogy ez a nyelv nem reguláris. Tegyük fel hogy az, s jelöljön n egy konstanst, mely mellett L kielégíti a reguláris pumpáló lemma tulajdonságait. Ilyen n- t viszont nem fogunk találni, hiszen az a^nb^n szó minden olyan uvw felbontására, melyre $|uv| \leq n$ és $|v|>0$, azt kapjuk, hogy $uw=a^{n-|v|}b^n$, azaz $uw \notin L$. (Hasonlóan, minden $i>1$- re $uv^iw=a^{n+(i-1)|v|}b^n \notin L$.) Ezt az ellentmondást csak az indirekt feltevésünk hamissága okozhatja, tehát L nem reguláris. ★

Egyébként a példabeli nyelv a $G=(\{S\}, \{a, b\}, S, \{S \rightarrow aSb, S \rightarrow ab\})$ lineáris nyelvtannal generálható.

5.11. Zártsági tulajdonságok

A reguláris nyelvek zártak a reguláris műveletekre nézve, ami a reguláris kifejezésekkel való megadásukra emlékezve magától értetődő.

Ezzel szemben, ha csak a reguláris nyelvtanokkal, vagy véges automatákkal tudnánk leírni ezeket a nyelveket a probléma nem tűnne ilyen egyszerűnek.

A következő tétel bizonyításában konstrukciót adunk a nyelvműveletekkel előállított nyelvek nyelvtanokkal történő előállítására.

32. Tétel. A reguláris nyelvek osztálya zárt a reguláris műveletekre nézve, azaz tetszőleges $G_1=(N_1, T, S_1, H_1)$ és $G_2=(N_2, T, S_2, H_2)$ reguláris nyelvtanokhoz megkonstruálhatók olyan nyelvtanok amelyek $L(G_1)\cup L(G_2)$, $L(G_1)L(G_2)$, illetve $(L(G_1))^*$ nyelveket generálnak.

Bizonyítás. A bizonyítás során az általánosság csorbítása nélkül feltehetjük, hogy $N_1\cap N_2=\emptyset$.

Unió (*egyesítés*). Legyen S olyan új nemterminális, amely eddig nem szerepelt sem N_1- ben, sem pedig N_2- ben. A

$$G_+=(N_1\cup N_2\cup\{S\}, T, S_0, H_1\cup H_2\cup\{S \rightarrow S_1, S \rightarrow S_2\})$$

reguláris nyelvtan ekkor az $L_1\cup L_2$ nyelvet generálja. A G_+ nyelvtanban az S mondatszimbólumra két szabály alkalmazható, vagy a G_1 vagy a G_2 mondatszimbólumát vezetjük be, ezután viszont N_1 és N_2 diszjunktsága miatt csak a választott nyelvtan szabályai lesznek alkalmazhatóak.

Konkatenáció. Legyen

$$G_.=(N_1\cup N_2, T, S_1, H_2\cup\{A \rightarrow vB \mid A \rightarrow vB\in H_1, B\in N_1\}\cup\{A \rightarrow vS_2 \mid A \rightarrow v\in H_1, v\in T^*\}).$$

Világos hogy ez a nyelvtan reguláris, másrészt pontosan akkor amikor az első nyelvtanban befejeződne egy levezetés, az új nyelvtanban az első nyelvbeli levezetett szó után megjelenik a második nyelvtan mondatszimbóluma a mondatformában, így $G_.$ az $L(G_1)L(G_2)$- t generálja.

Kleene iteráció (*a konkatenáció lezárása*). Legyen $S\notin N_1$, továbbá legyen H' az a szabályrendszer, amiben helyettesítjük a H_1 minden $A \rightarrow v$ ($A\in N_1$, $v\in T^*$) szabályát az $A \rightarrow vS$ szabállyal. Az így kapott H' szabályhalmazzal és S- sel képezett

$$G_*=(N_1\cup\{S\}, T, S, H\cup H'\cup\{S \rightarrow \lambda, S \rightarrow S_1\})$$

nyelvtan az $(L(G_1))^*$ nyelvet generálja és reguláris. ∎

A bizonyításban használt konstrukciók, illetve az alapnyelveket generáló egyszerű reguláris nyelvtanok (pl. $(\{S\}, \{a, b, c\}, S, \{S \rightarrow a\})$) segítségével bármily reguláris kifejezéshez legyárthatjuk azt a reguláris nyelvtant, amely a kifejezéssel leírt nyelvet generálja.

A továbbiakban halmazműveletekre való zártságot fogunk bizonyítani determinisztikus elfogadó automatát felhasználva.

33. Tétel. A reguláris nyelvek halmaza zárt a metszet műveletre.

Bizonyítás. Legyen adott két reguláris nyelv az őket elfogadó determinisztikus automatákkal: $FA=(Q, T, q_0, d, F)$ és $FA'=(Q', T, q'_0, d', F')$ ekkor megkonstruáljuk azt az automatát, amely az $L(FA)\cap L(FA')$ nyelvet fogadja el: legyen

$$FA_\cap=(Q\times Q', T, (q_0, q'_0), d_\cap, F\times F'),$$

ahol $d_\cap((q, q'), a)=(d(q, a), d'(q', a))$ minden $q\in Q$, $q'\in Q'$, $a\in T$ esetén.

Az automata állapotpárokkal dolgozik, a pár első tagja FA- t, a második tagja FA'- t szimulálja, elfogadni pontosan akkor fog, ha mindkét szimulált automata egyszerre fogadna el. ∎

34. Tétel. A reguláris nyelvek halmaza zárt a komplementerképzés műveletre.

Bizonyítás. Legyen adott a reguláris nyelv a teljesen definiált determinisztikus elfogadó automatával, $FA=(Q, T, q_0, d, F)$- el. Ekkor egy $w \in T^*$ szót az automata pontosan akkor fogad el, ha $d^*(q_0, w) \in F$. Legyen tehát $FA_c=(Q, T, q_0, d, Q \setminus F)$, ekkor minden szóra pontosan ugyanaz a futás lesz, ami az eredeti *FA* automatában, azzal a különbséggel, hogy éppen akkor fog a futás végén az új automata elfogadni, amikor a régi nem fogadott el, és fordítva. ∎

5.49. példa - Reguláris nyelvtan megadása reguláris kifejezéshez 1. feladat

Adjuk meg az $L=a^*b \cup b^*a$ reguláris kifejezéssel
megadott nyelvet generáló reguláris nyelvtant!
Megoldás:

(I.) A feladat megoldásához első lépésben vezessünk be minden terminális betűhöz
egy reguláris nyelvtant, mely kizárólag az adott terminálist generálja.

Jelen esetben a kiinduló nyelvtanok a következők:
$G_a=(\{A\},\{a,b\},A,\{A \rightarrow a\})$,
$G_b=(\{B\},\{a,b\},B,\{B \rightarrow b\})$.

(II.) Ezek után a kifejezésben belülről kifelé haladva építjük fel a nyelvtant a következőképpen:
1. Adott $G=(N,T,S,H)$ nyelvtan esetén az $L(G)^*$ nyelvet generáló nyelvtant megkaphatjuk,
ha az összes $A \rightarrow p$ alakú szabály mellé - ahol $p \in T^*$, - bevezetjük az $A \rightarrow pS$ alakú szabályt is,
valamint bevezetjük az S_2 új mondatszimbólumot és az $S_2 \rightarrow \lambda$, $S_2 \rightarrow S$ szabályokat.

Jelen esetben:
$G_{a^*}=(\{A,S\},\{a,b\},S,\{A \rightarrow a, A \rightarrow aA, S \rightarrow \lambda, S \rightarrow A\})$,
$G_{b^*}=(\{B,S\},\{a,b\},S,\{B \rightarrow b, B \rightarrow bB, S \rightarrow \lambda, S \rightarrow B\})$.

2. Adott $G_1=(N,T,S,H)$ és $G_2=(N',T,S_2,H')$ nyelvtanok esetén - ahol $N \cap N'=\emptyset$ az
$L(G)=L(G_1)L(G_2)$ konkatenált nyelvet generáló nyelvtan előállításához
első lépésben az összes H-beli $A \rightarrow p$ szabályt - ahol $p \in T^*$ - az $A \rightarrow pS_2$ szabályra cseréljük.
Az így kapott szabályhalmazt jelöljük H''-vel. Ezek után $G=(N \cup N',T,S,H'' \cup H')$.

Jelen esetben:

$G_{a^*b}=(\{A,S,B\},\{a,b\},S,\{A \rightarrow aB, A \rightarrow aA, S \rightarrow B, S \rightarrow A, B \rightarrow b\})$,

$G_{b^*a}=(\{B,S,A\},\{b,a\},S,\{B \rightarrow bA, B \rightarrow bB, S \rightarrow A, S \rightarrow B, A \rightarrow a\})$.

3. Adott $G_1=(N,T,S,H)$ és $G_2=(N',T,S_2,H')$ nyelvtanok esetén - ahol
$N \cap N'=\emptyset$
- az $L(G)=L(G_1) \cup L(G_2)$ nyelvet generáló nyelvtan előállításához
a $G=(N \cup N' \cup \{S_3\}, T, S_3, H \cup H' \cup \{S_3 \rightarrow S, S_3 \rightarrow S_2\})$ nyelvtan megfelelő lesz,
ahol S_3 nem eleme az N, N', T halmazok egyikének sem.
Jelen esetben a 2. pontban megadott G_{a^*b} *és* G_{b^*a} nyelvtanok esetén nem teljesül az a feltétel,
miszerint a nemterminálisok halmazai diszjunktak, ezért az egyik nyelvtanban először át kell
jelölni a nemterminálisokat.
Legyen például
$G'_{a^*b}=(\{C,S_2,D\},\{a,b\},S_2,\{C \rightarrow aD, C \rightarrow aC, S_2 \rightarrow D, S_2 \rightarrow C, D \rightarrow b\})$!

Természetesen a nemterminálisok átjelölése nem befolyásolja a generált nyelvet, azaz
$L(G'_{a^*b})=L(G_{a^*b})$.
Ezek után már meg tudjuk adni a
$G_{a^*b \cup b^*a}=(\{C,S_2,D,B,S,A,S_3\},\{a,b\},S_3, \{C \rightarrow aD, C \rightarrow aC, S_2 \rightarrow D, S_2 \rightarrow C, D \rightarrow b, B \rightarrow bA,$
$B \rightarrow bB, S \rightarrow A, S \rightarrow B, A \rightarrow a, S_3 \rightarrow S_2, S_3 \rightarrow S\})$
reguláris nyelvtant, mely pontosan az L nyelvet generálja. ★

5.50. példa - Reguláris nyelvtan megadása reguláris kifejezéshez 2. feladat

Adjuk meg az $L=(a \cup b)^*$ reguláris kifejezéssel megadott nyelvet generáló reguláris nyelvtant!

Megoldás:
(I.)
$G_a=(\{A\},\{a,b\},A,\{A \to a\})$,
$G_b=(\{B\},\{a,b\},B,\{B \to b\})$,

(II.)
1. $G_{a \cup b}=(\{A,B,S\},\{a,b\},S,\{A \to a,\ B \to b, S \to A, S \to B\})$,
2. $G_{(a \cup b)^*}=(\{A,B,S,S_2\},\{a,b\},S_2,\{A \to a,\ A \to aS,\ B \to b,\ B \to bS,\ S \to A,\ S \to B,\ S_2 \to \lambda,\ S_2 \to S\})$. ★

5.51. példa - Reguláris nyelvtan megadása reguláris kifejezéshez 3. feladat

Adjuk meg az $L=ab^*c$ reguláris kifejezéssel megadott nyelvet generáló reguláris nyelvtant!

Megoldás:
(I.)
$G_a=(\{A\},\{a,b,c\},A,\{A \to a\})$,
$G_b=(\{B\},\{a,b,c\},B,\{B \to b\})$,
$G_c=(\{C\},\{a,b,c\},C,\{C \to c\})$.
(II.)
1. $G_{b^*}=(\{B,S\},\{a,b,c\},S,\{B \to b,\ B \to bB,\ S \to \lambda,\ S \to B\})$
2. $G_{ab^*}=(\{A,B,S\},\{a,b,c\},A,\{A \to aS,\ B \to b,\ B \to bB,\ S \to \lambda,\ S \to B\})$
3. $G_{ab^*c}=(\{A,B,S,C\},\{a,b,c\},A,\{A \to aS,\ B \to bC,\ B \to bB,\ S \to C,\ S \to B,\ C \to c\})$. ★

5.52. példa - Reguláris nyelvtan megadása reguláris kifejezéshez 4. feladat

Adjuk meg az $L=(ab^*)^*$ reguláris kifejezéssel megadott nyelvet generáló reguláris nyelvtant!

Megoldás:
(I.)
$G_a=(\{A\},\{a,b\},A,\{A \to a\})$,
$G_b=(\{B\},\{a,b\},B,\{B \to b\})$.

(II.)
1. $G_{b^*}=(\{B,S\},\{a,b\},S,\{B \to b,\ B \to bB,\ S \to \lambda,\ S \to B\})$,
2. $G_{ab^*}=(\{A,B,S\},\{a,b\},A,\{A \to aS,\ B \to b,\ B \to bB,\ S \to \lambda,\ S \to B\})$,
3. $G_{(ab^*)^*}=(\{A,B,S,S_2\},\{a,b\},S_2,\{A \to aS,\ B \to b,\ B \to bA,\ B \to bB,\ S \to \lambda,$
$S \to A,\ S \to B,\ S_2 \to \lambda,\ S_2 \to A\})$. ★

5.53. példa - Reguláris nyelvtan megadása reguláris kifejezéshez 5. feladat

Adjuk meg az $L=a^*b\cup b^*$ reguláris kifejezéssel megadott nyelvet generáló reguláris nyelvtant!

Megoldás:

(I.)
$G_a=(\{A\},\{a,b\},A,\{A \to a\})$,
$G_b=(\{B\},\{a,b\},B,\{B \to b\})$.

(II.)
1. $G_{a^*}=(\{A,S\},\{a,b\},S,\{A \to a,\ A \to aA,\ S \to \lambda,\ S \to A\})$,
2. $G_{a^*b}=(\{A,S,B\},\{a,b\},S,\{A \to aB,\ A \to aA,\ S \to B,\ S \to A,\ B \to b\})$,
3. $G_{b^*}=(\{B,S\},\{a,b\},S,\{B \to b,\ B \to bB,\ S \to \lambda,\ S \to B\})$,
4. $G'_{b^*}=(\{D,Z\},\{a,b\},Z,\{D \to b,\ D \to bD,\ Z \to \lambda,\ Z \to D\})$,
5. $G_{a^*b\cup b^*}=(\{A,S,B,D,Z,S_2\},\{a,b\},S_2,\{A \to aB,\ A \to aA,\ S \to B,\ S \to A,\ B \to b,\ D \to b,\ D \to bD,\ Z \to \lambda,\ Z \to D,\ S_2 \to S,\ S_2 \to Z\})$. ★

5.54. példa - Reguláris nyelvtan megadása reguláris kifejezéshez 6. feladat

Adjuk meg az $L=a^*\cup b^*\cup c^*$ reguláris kifejezéssel megadott nyelvet generáló reguláris nyelvtant!

Megoldás:
(I.)
$G_a=(\{A\},\{a,b,c\},A,\{A \to a\})$,
$G_b=(\{B\},\{a,b,c\},B,\{B \to b\})$,
$G_c=(\{C\},\{a,b,c\},C,\{C \to c\})$.
(II.)
1. $G_{a^*}=(\{A,S\},\{a,b,c\},S,\{A \to a,\ A \to aA,\ S \to \lambda,S \to A\})$,
2. $G_{b^*}=(\{B,S\},\{a,b,c\},S,\{B \to b,\ B \to bB,\ S \to \lambda,\ S \to B\})$,
3. $G'_{a^*}=(\{A,S_2\},\{a,b,c\},S_2,\{A \to a,\ A \to aA,\ S_2 \to \lambda,\ S_2 \to A\})$,
4. $G_{a^*\cup b^*}=(\{A,S_2,B,S,S_3\},\{a,b,c\},S_3,\{A \to a,\ A \to aA,\ S_2 \to \lambda,\ S_2 \to A,\ B \to b,\ B \to bB,\ S \to \lambda,\ S \to B,\ S_3 \to S_2,\ S_3 \to S\})$,
5. $G_{c^*}=(\{C,S\},\{a,b,c\},S,\{C \to c,\ C \to cC,\ S \to \lambda,\ S \to C\})$,
6. $G'_{c^*}=(\{C,S_4\},\{a,b,c\},S_4,\{C \to c,\ C \to cC,\ S_4 \to \lambda,\ S_4 \to C\})$,
7. $G_{a^*\cup b^*\cup c^*}=(\{A,S_2,B,S,S_3,C,S_4,S_5\},\{a,b,c\},S_5,\{A \to a,\ A \to aA,\ S_2 \to \lambda,\ S_2 \to A,\ B \to b,\ B \to bB,\ S \to \lambda,\ S \to B,\ S_3 \to S_2,\ S_3 \to S,\ C \to c,\ C \to cC,\ S_4 \to \lambda,\ S_4 \to C,S_5 \to S_3,\ S_5 \to S_4\})$. ★

5.55. példa - Reguláris nyelvtan megadása reguláris kifejezéshez 7. feladat

Adjuk meg az $L=(ab \cup c)^*(ba)^*$ reguláris kifejezéssel megadott nyelvet generáló reguláris nyelvtant!

Megoldás:
(I.)
$G_a=(\{A\},\{a,b,c\},A,\{A \to a\})$,
$G_b=(\{B\},\{a,b,c\},B,\{B \to b\})$,
$G_c=(\{C\},\{a,b,c\},C,\{C \to c\})$.
(II.)
1. $G_{ab}=(\{A,B\},\{a,b,c\},A,\{A \to aB,\ B \to b\})$,
2. $G_{ba}=(\{B,A\},\{a,b,c\},B,\{B \to bA,\ A \to a\})$,
3. $G_{ab \cup c}=(\{A,B,C,S\},\{a,b,c\},S,\{A \to aB,\ B \to b,\ C \to c,\ S \to A,\ S \to C\})$,
4. $G_{(ab \cup c)^*}=(\{A,B,C,S,S_2\},\{a,b,c\},S_2,\{A \to aB,\ B \to b,\ B \to bS,\ C \to c,\ C \to cS,\ S \to A,\ S \to C,\ S_2 \to \lambda,\ S_2 \to S\})$,
5. $G_{(ba)^*}=(\{B,A,S\},\{a,b,c\},S,\{B \to bA,\ A \to a,\ A \to aB,\ S \to \lambda,\ S \to B\})$,
6. $G'_{(ba)^*}=(\{D,C,Z\},\{a,b,c\},Z,\{D \to bC,\ C \to a,\ C \to aD,\ Z \to \lambda,\ Z \to D\})$,
7. $G_{(ab \cup c)^*(ba)^*}=(\{A,B,C,S,S_2,D,C,Z\},\{a,b,c\},S_2,\ \{A \to aB,\ B \to bZ,\ B \to bS,\ C \to cZ,\ C \to cS,\ S \to A,\ S \to C,\ S_2 \to Z,S_2 \to S,\ D \to bC,\ C \to a,\ C \to aD,\ Z \to \lambda,\ Z \to D\})$. ★

5.56. példa - Reguláris nyelvtan megadása reguláris kifejezéshez 8. feladat

Adjuk meg az $L=(a^*ba^* \cup aa)^*$ reguláris kifejezéssel megadott nyelvet generáló reguláris nyelvtant!

Megoldás:
(I.)
$G_a=(\{A\},\{a,b\},A,\{A \to a\})$,
$G_b=(\{B\},\{a,b\},B,\{B \to b\})$.

(II.)
1. $G_{a^*}=(\{A,S\},\{a,b\},S,\{A \to a,A \to aA,\ S \to \lambda,S \to A\})$,
2. $G_{a^*b}=(\{A,S,B\},\{a,b\},S,\{A \to aB,A \to aA,\ S \to B,S \to A,B \to b\})$,
3. $G'_{a^*}=(\{C,Z\},\{a,b\},Z,\{C \to a,C \to aC,\ Z \to \lambda,Z \to C\})$,
4. $G_{a^*ba^*}=(\{A,S,B,C,Z\},\{a,b\},S,\{A \to aB,\ r;aA,S \to B,S \to A,B \to bZ,\ C \to a,\ C \to aC,\ Z \to \lambda,\ Z \to C\})$,
5. $G'_a=(\{E\},\{a,b\},E,\{E \to a\})$,
6. $G_{aa}=(\{A,E\},\{a,b\},A,\{A \to aE,E \to a\})$,
7. $G'_{aa}=(\{F,E\},\{a,b\},F,\{F \to aE,E \to a\})$,
8. $G_{a^*ba^* \cup aa}=(\{A,S,B,C,Z,F,E,S_2\},\{a,b\},S_2,\{A \to aB,\ A \to aA,S \to B,S \to A,\ B \to bZ,\ C \to a,C \to aC,\ Z \to \lambda,\ Z \to C,\ F \to aE,E \to a,S_2 \to S,\ S_2 \to F\})$,
9. $G_{(a^*ba^* \cup aa)^*}=(\{A,\ S,\ B,\ C,\ Z,\ F,\ E,\ S_2,\ S_3\},\{a,b\},\ S_3,\ \{A \to aB,\ A \to aA,\ S \to B,\ S \to A,\ B \to bZ,\ C \to aS_2,\ C \to aC,\ Z \to S_2,\ Z \to C,\ F \to aE,\ E \to aS_2,S_2 \to S,\ S_2 \to F,\ S_3 \to \lambda,\ S_3 \to S_2\})$. ★

5.12. Irodalmi megjegyzések

A reguláris kifejezéseket [Kleene 1956]-ban vezette be és bizonyította ekvivalenciájukat a véges elfogadó automatákkal. A reguláris műveletekre való zártsági tulajdonságok bizonyítása ugyancsak itt jelent meg. A pumpáló lemma a [Bar-Hillel et al. 1961]-ben közölt eredmény speciális esete. A szóprobléma megoldása megtalálható [Moore 1956]-ban. A minimális automata előállítására az első algoritmus [Huffman 1954]-ben jelent meg. Habár a reguláris nyelvek elmélete és alkalmazása (pl. Unix operációs rendszereknél) is nagy múltra tekint vissza, a témakör ma is aktív kutatási

terület. A reguláris kifejezések unió-normálformája [Nagy 2004] munkában került bevezetésre, néhány ezzel kapcsolatos eredmény [Nagy 2010d]-ben is található; [Afonin, Golomazov 2009]-ben bizonyították, hogy a reguláris nyelvek unióbonyolultságának meghatározása, mint feladat megoldható. Az uniómentes nyelvek és speciális véges automaták kapcsolatát a [Nagy 2006a] cikk tárgyalja.

6. fejezet - Lineáris nyelvek

A Chomsky-féle nyelvcsaládok közt a 3. (reguláris) és a 2. (környezetfüggetlen) között elhelyezkedő nyelvcsalád. Definíció szerint egy nyelvtan akkor lineáris, ha környezetfüggetlen és minden szabály jobb oldalán maximum egy nemterminális áll.

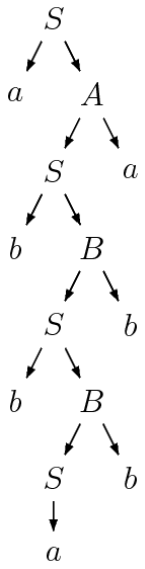
6.1. példa - A kétbetűs ábécé feletti palindrómák (vagy tükörszavak) nyelvét generáló lineáris nyelvtan

Palindromák (olyan szavak amelyek visszafele olvasva megegyeznek saját magukkal) nyelve a következő lineáris nyelvtannal generáló: $(\{S\}, \{a, b\}, S, \{S \to \lambda, S \to a, S \to b, S \to aSa, S \to bSb\})$. ★

6.2. példa - 0^n1^n nyelv - Gyakorló feladat

Generáljuk az $\{0^n1^n\}$ nyelvet lineáris nyelvtannal. ★

A levezetéseket fagráfokkal reprezentálhatjuk itt is, hasonlóan a reguláris nyelveknél látott fákhoz, azzal a különbséggel, hogy a főág (a nemterminálisok ága) mellett mindkét oldalon vezethetünk be terminális szimbólumokat egy általános lineáris nyelvtan esetén. A következő ábrán egy ilyen levezetési fa található.



6.1. Normálforma

10. Definíció. Egy lineáris nyelvtant gyenge normálformájúnak mondunk, ha minden szabályára a következő alakok egyike teljesül $A \to aB, A \to Ba, A \to a, A \to B, A \to \lambda$ $(a \in T; A, B \in N)$. ★

35. Tétel. Minden lineáris nyelv generálható gyenge normálformájú nyelvtannal.

Bizonyítás. Hasonlóan a reguláris nyelvtanok esetéhez, itt is új nemterminálisok bevezetése segítségével helyettesítjük a hosszabb szabályokat rövidebbekkel. $A \to \lambda$ alakúak kiküszöbölhetőek az üresszó lemma alapján, $A \to B$ alakúak is kiküszöbölhetőek a reguláris normálalaknál ismertetett módon. ∎

Az így kapott, csak $A \to aB, A \to Ba, A \to a$ alakú szabályokat tartalmazó lineáris nyelvtanokat, erős normálformájúnak nevezzük (itt minden lépésben pontosan egy terminális kerül levezetésre).

36. Tétel. Minden lineáris nyelvtanhoz van vele ekvivalens (a generált nyelv legfeljebb az üres-szóban különbözik), amely erős normálformájú.

A fenti, levezetési fát ábrázoló ábrán a palindromák nyelvét generáló erős normálformában levő nyelvtanban példa levezetésként az *abbabba* szót vezettük le.

6.3. példa - Erős lineáris normálalak 1. példa

Adjunk meg a $G=(\{S,A,B\},\{a,b\},S,H)$
$H=\{ S \to ababA, A \to Bbba, B \to aaSbab, B \to b, A \to abba, A \to B, B \to S \}$

nyelvtannal ekvivalens erős lineáris normálalakú nyelvtant!

Megoldás:

(I.)
Első lépésben megadunk egy G_1 nyelvtant, ami ekvivalens a G nyelvtannal és nem szerepelnek benne
$Y \to y_1y_2 ...y_n$,
$Y \to y_1y_2 ...Y_n$,
$Y \to Y_1y_2 ...y_n$, $n \geq 3$ alakú szabályok.
A H szabályhalmaz ilyen alakú szabályait helyettesítjük új szabályokkal, a többi szabályt pedig változtatás nélkül átvesszük a H_1 szabályhalmazba.

Minden $Y \to y_1y_2 ...y_n$, $n \geq 3$ alakú szabályhoz
vezessünk be $Z_1,Z_2, ...,Z_{n-1}$ új
nemterminálisokat, ahol Z_i szerepe az, hogy belőle vezetjük le az $y_{i+1} ...y_n$ szót! Ezek után az összes $Y \to y_1y_2 ...y_n$, $n \geq 3$ alakú
szabályt helyettesítsük a következő szabályokkal:

$Y \to y_1Z_1$,
$Z_1 \to y_2Z_2$,

.
.
.

$Z_{n-2} \to y_{n-1}Z_{n-1}$,
$Z_{n-1} \to y_n$.

Minden $Y \to y_1y_2 ...Y_n$, $n \geq 3$ alakú szabályhoz
vezessünk be $Z_1,Z_2, ...,Z_{n-2}$ új
nemterminálisokat, ahol Z_i szerepe az, hogy belőle vezetjük le az $y_{i+1} ...Y_n$ szót! Ezek után az összes $Y \to y_1y_2 ...Y_n$, $n \geq 3$ alakú
szabályt helyettesítsük a következő szabályokkal:

$Y \to y_1Z_1$,
$Z_1 \to y_2Z_2$,

.
.
.

$Z_{n-3} \to y_{n-2}Z_{n-2}$,
$Z_{n-2} \to y_{n-1}Y_n$.

Minden $Y \to Y_1y_2 ...y_n$, $n \geq 3$ alakú szabályhoz
vezessünk be $Z_1,Z_2, ...,Z_{n-2}$ új

nemterminálisokat, ahol Z_i szerepe az, hogy belőle vezetjük le az $Y_1y_2 ...y_{n-i}$ szót!
Tehát az összes $Y \to Y_1y_2 ...y_n$, $n \geq 3$ alakú
szabályt helyettesítsük a következő szabályokkal:

$Y \to Z_1y_n$,
$Z_1 \to Z_2y_{n-1}$,

.
.
.

$Z_{n-3} \to Z_{n-2}y_3$,
$Z_{n-2} \to Y_1y_2$.

Jelen esetben:
$G_1=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5,Z_6,Z_7,Z_8\},\{a,b\},S,H_1)$.
$H_1=\{ S \to aZ_4,\ Z_4 \to bZ_5,\ Z_5 \to aZ_6,\ Z_6 \to bA,\ A \to Z_7a,\ Z_7 \to Z_8b,\ Z_8 \to Bb,\ B \to aaSbab,$
$B \to b,\ A \to aZ_1,\ Z_1 \to bZ_2,\ Z_2 \to bZ_3,\ Z_3 \to a,\ A \to B,\ B \to S\ \}$

(II.)Második lépésben megadunk egy G_2 nyelvtant, ami
ekvivalens a G_1 nyelvtannal és lineáris normálalakú.
A G_1 nyelvtanból indulunk ki. A H_1 szabályhalmaz
$Y \to y_1y_2 ...y_{k-1}Y_ky_{k+1} ...y_n$, $n \geq 3$
alakú szabályait helyettesítjük új szabályokkal, a többi szabályt pedig
változtatás nélkül átvesszük a H_2 szabályhalmazba.
Minden $Y \to y_1y_2 ...y_{k-1}Y_ky_{k+1} ...y_n$, $n \geq 3$
alakú szabályhoz vezessünk be $Z_1,Z_2, ...,Z_{n-1}$ új nemterminálisokat!
Ezek után az összes $Y \to y_1y_2 ...y_{k-1}Y_ky_{k+1} ...y_n$, $n \geq 3$
alakú szabályt helyettesítsük a következő szabályokkal:

$Y \to y_1Z_1$,

$Z_1 \to y_2Z_2$,

.
.
.

$Z_{k-2} \to y_{k-1}Z_{k-1}$,
$Z_{k-1} \to Z_ky_n$,
$Z_k \to Z_{k+1}y_{n-1}$,

.
.
.

$Z_{n-2} \to Z_{n-1}y_{k+2}$,
$Z_{n-1} \to Y_ky_{k+1}$.

Jelen esetben:
$G_2=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5,Z_6,Z_7,Z_8,Z_9,Z_{10},Z_{11},Z_{12}\},\{a,b\},S,H_2)$.
$H_2=\{ S \to aZ_4,\ Z_4 \to bZ_5,\ Z_5 \to aZ_6,\ Z_6 \to bA,\ A \to Z_7a,\ Z_7 \to Z_8b,\ Z_8 \to Bb,\ B \to aZ_9,\ Z_9 \to aZ_{10},$
$Z_{10} \to Z_{11}b,\ Z_{11} \to Z_{12}a,\ Z_{12} \to Sb,\ B \to b,\ A \to aZ_1,\ Z_1 \to bZ_2,\ Z_2 \to bZ_3,\ Z_3 \to a,\ A \to B,\ B \to S\ \}$

(III.) Harmadik lépésben megadjuk az eredeti nyelvtannal ekvivalens G' erős lineáris normálalakú nyelvtant.
Ehhez két lépésre van szükség. Első lépésben meghatározunk egy $U(Z)$ halmazt minden

olyan Z nemterminálishoz, mely levezethető legalább egy másik nemterminálisból a G_2 nyelvtanban és szerepel olyan H_2 halmazban lévő szabály bal oldalán,
amelynek jobb oldalán egy terminális vagy pedig egy terminális és egy nemterminális betű áll.
Az $U(Z)$ halmaz tartalmazni fogja az összes olyan nemterminálist,
melyből egy vagy több lépésben levezethető a Z betű.

Jelen esetben:
$U(B)=\{A\}, U(S)=\{B,A\}$.
Második lépésben a H' szabályhalmazba átvesszük a H_2 szabályhalmaz
mindazon szabályait, melyek jobb oldalán egy terminális betű vagy pedig egy
terminális és egy nemterminális található, majd hozzávesszük mindazon szabályokat,
melyeket úgy kapunk, hogy a már átvett szabályok bal oldalán szereplő betűt
a hozzá tartozó U halmaz elemeivel helyettesítjük.

Formálisan:
$H'=(H_2 \cup \{W \to p | Z \to p \in H_2, W \in U(Z)\}) \setminus \{X \to Y | X,Y \in N\}$.
Jelen esetben:
$G'=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5,Z_6,Z_7,Z_8,Z_9,Z_{10},Z_{11},Z_{12}\},\{a,b\},S,H')$.
$H'=\{ S \to aZ_4,\ B \to aZ_4,\ A \to aZ_4,\ Z_4 \to bZ_5,\ Z_5 \to aZ_6,\ Z_6 \to bA,\ A \to Z_7a,\ Z_7 \to Z_8b,\ Z_8 \to Bb,\ B \to aZ_9,$
$A \to aZ_9,\ Z_9 \to aZ_{10},\ Z_{10} \to Z_{11}b,\ Z_{11} \to Z_{12}a,\ Z_{12} \to Sb,\ B \to b,\ A \to b,\ A \to aZ_1,\ Z_1 \to bZ_2,\ Z_2 \to bZ_3,$
$Z_3 \to a\ \}$ ★

6.4. példa - Erős lineáris normálalak 2. feladat

Adjunk meg a $G=(\{S,A,B\},\{x,y,z\},S,H)$
$H=\{ S \to Bxxx,\ B \to xyAyx,\ A \to B,\ A \to S,\ A \to z\ \}$

nyelvtannal ekvivalens erős lineáris normálalakú nyelvtant!

Megoldás:
(I.)
$G_1=(\{S,A,B,Z_1,Z_2\},\{x,y,z\},S,H_1)$.
$H_1=\{ S \to Z_1x,\ Z_1 \to Z_2x,\ Z_2 \to Bx,\ B \to xyAyx,\ A \to B,\ A \to S,\ A \to z\ \}$
(II.)
$G_2=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5\},\{x,y,z\},S,H_2)$.
$H_2=\{ S \to Z_1x,\ Z_1 \to Z_2x,\ Z_2 \to Bx,\ B \to xZ_3,\ Z_3 \to yZ_4,\ Z_4 \to Z_5x,\ Z_5 \to Ay,\ A \to B,\ A \to S,\ A \to z\ \}$
(III.)
$U(B)=\{A\}, U(S)=\{A\}$.
$G'=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5\},\{x,y,z\},S,H')$.
$H'=\{ S \to Z_1x, A \to Z_1x,\ Z_1 \to Z_2x,\ Z_2 \to Bx,\ B \to xZ_3,\ A \to xZ_3,\ Z_3 \to yZ_4,\ Z_4 \to Z_5x,\ Z_5 \to Ay,\ A \to z\ \}$ ★

6.5. példa - Erős lineáris normálalak 3. feladat

Adjunk meg a $G=(\{S,A\},\{x,y,z\},S,H)$
$H=\{ S \to xSx,\ S \to A,\ A \to yAy,\ A \to z\ \}$

nyelvtannal ekvivalens erős lineáris normálalakú nyelvtant!

Megoldás:

(I.)
Mivel a G nyelvtanban nincsenek $Y \to y_1y_2 \dots y_n$, $Y \to y_1y_2 \dots Y_n$, $Y \to Y_1y_2 \dots y_n$, $n \geq 3$ alakú szabályok, ezért $G_1=G$.

(II.)
$G_2=(\{S,A,Z_1,Z_2\},\{x,y,z\},S,H_2)$.
$H_2=\{\ S \to xZ_1,\ Z_1 \to Sx,\ S \to A,\ A \to yZ_2,\ Z_2 \to Ay,\ A \to z\ \}$
(III.)
$U(A)=\{S\}$.
$G'=(\{S,A,B,Z_1,Z_2\},\{x,y,z\},S,H')$.
$H'=\{\ S \to xZ_1,\ Z_1 \to Sx,\ A \to yZ_2,\ S \to yZ_2,\ Z_2 \to Ay,\ A \to z,\ S \to z\ \}$ ★

6.6. példa - Erős lineáris normálalak 4. feladat

Adjunk meg a $G=(\{S,X,Y\},\{x,y,z\},S,H)$
$H=\{\ S \to Xxx,\ S \to yyY,\ S \to zzz,\ X \to xS,\ Y \to Sy,\ X \to S,\ Y \to S\ \}$

nyelvtannal ekvivalens erős lineáris normálalakú nyelvtant!

Megoldás:

(I.)
$G_1=(\{S,X,Y,Z_1,Z_2,Z_3,Z_4\},\{x,y,z\},S,H_1)$.
$H_1=\{\ S \to Z_1x,\ Z_1 \to Xx,\ S \to yZ_2,\ Z_2 \to yY,\ S \to zZ_3,\ Z_3 \to zZ_4,\ Z_4 \to z,\ X \to xS,\ Y \to Sy,\ X \to S,\ Y \to S\ \}$

(II.) Mivel a G_1 nyelvtanban nincs $Y \to y_1y_2 \dots y_{k-1}Y_ky_{k+1} \dots y_n$, $n \geq 3$ alakú szabály, ezért $G_2=G_1$.

(III.)
$U(S)=\{X,Y\}$.
$G'=(\{S,A,B,Z_1,Z_2,Z_3,Z_4\},\{x,y,z\},S,H')$.
$H'=\{\ S \to Z_1x, X \to Z_1x, Y \to Z_1x,\ Z_1 \to Xx,\ S \to yZ_2, X \to yZ_2,\ Y \to yZ_2,\ Z_2 \to yY,\ S \to zZ_3,$
$X \to zZ_3,\ Y \to zZ_3,\ Z_3 \to zZ_4,\ Z_4 \to z,\ X \to xS,\ Y \to Sy\ \}$ ★

6.7. példa - Erős lineáris normálalak 5. feladat

Adjunk meg a $G=(\{S,A,B\},\{x,y\},S,H)$
$H=\{\ S \to yyAxx,\ A \to xxB,\ B \to Syy,\ S \to xxyy\ \}$

nyelvtannal ekvivalens erős lineáris normálalakú nyelvtant!

Megoldás:
(I.)
$G_1=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5\},\{x,y\},S,H_1)$.
$H_1=\{\ S \to yyAxx,\ A \to xZ_1,\ Z_1 \to xB,\ B \to Z_2y,\ Z_2 \to Sy,\ S \to xZ_3\ S \to xZ_4\ S \to yZ_5\ Z_5 \to y\ \}$

(II.)
$G_2=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5,Z_6,Z_7,Z_8\},\{x,y,z\},S,H_2)$.
$H_2=\{\ S \to yZ_6,\ Z_6 \to y,Z_7\ Z_7 \to Z_8x,\ Z_8 \to Ax,\ A \to xZ_1,\ Z_1 \to xB,\ B \to Z_2y,\ Z_2 \to Sy,\ S \to xZ_3,\ S \to xZ_4,$
$S \to yZ_5,\ Z_5 \to y\ \}$

(III.)
Mivel a G_2 nyelvtan már erős lineáris normálalakú, ezért $G'=G_2$. ★

6.8. példa - Erős lineáris normálalak 6. feladat

Adjunk meg a $G=(\{S,A,B\},\{x,y\},S,H)$
$H=\{\ S \to Bx,\ B \to yA,\ S \to A,\ A \to S,\ S \to z\ \}$
nyelvtannal ekvivalens erős lineáris normálalakú nyelvtant!

Megoldás:

(I.)
Mivel a G nyelvtanban nincsenek $Y \to y_1y_2 \ldots y_n$, $Y \to y_1y_2 \ldots Y_n$, $Y \to Y_1y_2 \ldots y_n$, $n \geq 3$
alakú szabályok, ezért $G_1=G$.

(II.)
Mivel a G_1 nyelvtanban nincs $Y \to y_1y_2 \ldots y_{k-1}Y_ky_{k+1} \ldots y_n$, $n \geq 3$ alakú szabály, ezért $G_2=G_1$.

(III.)
$U(S)=\{A\}$.
$G'=(\{S,A,B\},\{x,y,z\},S,H')$.
$H'=\{\ S \to Bx,\ A \to Bx,\ B \to yA,\ S \to z,\ A \to z\ \}$ ★

6.9. példa - Erős lineáris normálalak 7. feladat

Adjunk meg a $G=(\{S,A,B\},\{0,1,+\},S,H)$
$H=\{\ S \to 1S0,\ S \to A,\ S \to B,\ A \to 0A1,\ A \to B,\ B \to +,\ B \to S\ \}$

nyelvtannal ekvivalens erős lineáris normálalakú nyelvtant!

Megoldás:

(I.)
Mivel a G nyelvtanban nincsenek $Y \to y_1y_2 \ldots y_n$, $Y \to y_1y_2 \ldots Y_n$, $Y \to Y_1y_2 \ldots y_n$, $n \geq 3$
alakú szabályok, ezért $G_1=G$.

(II.)
$G_1=(\{S,A,B,Z_1,Z_2\},\{0,1,+\},S,H_1)$
$H_1=\{\ S \to 1Z_1,\ Z_1 \to S0,\ S \to A,\ S \to B,\ A \to 0Z_2,\ Z_2 \to A1,\ A \to B,\ B \to +,\ B \to S\ \}$

(III.)
$U(A)=\{S,B\}, U(B)=\{S,A\}, U(S)=\{B,A\}$.
$G'=(\{S,A,B,Z_1,Z_2\},\{0,1,+\},S,H')$.
$H'=\{\ S \to 1Z_1, B \to 1Z_1, A \to 1Z_1,\ Z_1 \to S0,\ A \to 0Z_2, S \to 0Z_2, B \to 0Z_2,\ Z_2 \to A1,\ B \to +, S \to +, A \to +\ \}$ ★

Most egy pillanatra lépjünk vissza: a reguláris nyelvek speciális lineáris nyelvek, amelyek generálhatók olyan lineáris nyelvtannal, hogy minden $A \to uBv$ alakú szabályában $v=\lambda$ (azaz a nyelvtan jobb-lineáris). Itt jegyezzük meg, hogy a szakirodalomban előforduló jobb-lineáris nyelvtan mellett előfordul a bal-lineáris kifejezés is (ez megfelel a mi reguláris nyelvtan definíciónk szimmetrikus alakjának): ha a szabályok alakja csak $A \to v$ és $A \to Bv$ lehet, ahol $A, B \in N$, $v \in T^*$. Érdekes kérdés hogy milyen a bal-lineáris nyelvek és a reguláris (vagyis jobb-lineáris) nyelvek viszonya.

37. Tétel. A bal-lineáris nyelvtanokkal generált nyelvek osztálya egybeesik a reguláris nyelvek osztályával.

Bizonyítás. Legyen $G=(N, T, S, H)$ bal-lineáris nyelvtan, legyen $N=\{S, A_1, A_2, \ldots, A_n\}$. Az általánosság csorbítása nélkül feltehetjük, hogy S nem szerepel egyetlen szabály jobb oldalán sem. Szerkesszük meg a $G'=(N', T, S', H')$ nyelvtant úgy, hogy $N'=\{S', B_1, \ldots, B_n\}$ és a H'- beli szabályok a következők:

- $S' \to v \in H'$, ha $S \to v \in H$ és $v \in T^*$,

- $S' \to vB_k \in H'$, ha $A_k \to v \in H$ és $v \in T^*$,

- $B_j \to vB_k \in H'$, ha $A_k \to A_j v \in H$ és $v \in T^*$,

- $B_j \to v \in H'$, ha $S \to A_j v \in H$ és $v \in T^*$.

Az így kapott G' nyelvtan nyilván reguláris, és belátható, hogy $L(G)=L(G')$: Legyen $w \in L(G)$. Ha $S \to w \in H$, akkor $S' \to w \in H'$. Ha pedig létezik egy $S \Rightarrow A_{i,1}v_1 \Rightarrow A_{i,2}v_2v_1 \Rightarrow \ldots \Rightarrow A_{i,m-1}v_{m-1}\ldots v_1 \Rightarrow v_m v_{m-1}\ldots v_1$ alakú levezetés G- ben, ahol $v_1, v_2, \ldots, v_m \in T^*$, akkor G'- ben ezzel analóg módon megadható egy $S \Rightarrow v_m B_{i,\,m-1} \Rightarrow v_m v_{m-1} B_{i,\,m-2} \Rightarrow \ldots \Rightarrow v_m \ldots v_2 B_{i,\,1} \Rightarrow v_m v_{m-1}\ldots v_2 v_1$ alakú levezetés, tehát $w \in L(G')$. Megfordítva ugyanígy belátható, hogy $L(G') \subseteq L(G)$. ∎

Fordított konstrukcióval pedig éppen az bizonyítható, hogy minden reguláris nyelv generálható bal-lineáris módon is. (Ez egyébként automatanyelven megfelel annak, mintha egy végállapotból visszafelé haladva fogadnánk el a szót olyan módon, hogy végül a kezdőállapotba érkezzünk.)

Itt jegyezzük meg az előző tétel és konstrukciók egy azonnali következményét:

A reguláris nyelvek halmaza zárt a tükrözésre nézve, ahol egy $w=a_1a_2\ldots a_n$ szó tükörképén a $w^{-1}=a_n\ldots a_2a_1$ szót értjük ($a_1, a_2, \ldots, a_n \in T$). Egy L nyelv tükörképén pedig az $L^{-1}=\{w \mid \exists v \in L,\ w=v^{-1}\}$ nyelvet.

6.2. 2-fejű (véges állapotú) automata

11. Definíció. A rendezett (Q, T, q_0, d, F) ötöst kétfejű véges automatának hívjuk, ha (a hagyományos véges automatákhoz hasonlóan)

Q: nemüres véges halmaz: állapotok halmaza,

T: szalagábécé,

$q_0 \in Q$ kezdőállapot,

$F \subseteq Q$ vég- (vagy elfogadó)állapotok halmaza,

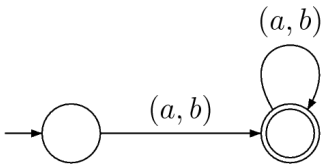
$d: Q \times (T \cup \{\lambda\}) \times (T \cup \{\lambda\}) \to 2^Q$ leképezés az állapotátmenet-függvény. ★

A konfigurációk halmaza: a még feldolgozandó input és az állapot (u, q) ahol $u \in T^*$, $q \in Q$,

kezdeti konfiguráció: (w, q_0) ahol w az inputszó (feldolgozandó input).

A konfigurációk az állapotátmenet-függvénynek megfelelően változhatnak az automata számítási lépései alapján: $(aub, q) \vdash (u, q')$ ha $q' \in d(q, a, b)$, ahol $a, b \in T \cup \{\lambda\}$.

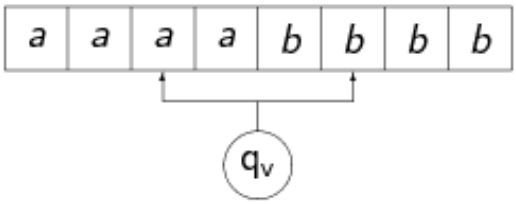
$\vdash^*$ jelölje a $\vdash$ reláció tranzitív és reflexív lezártját. Akkor mondjuk, hogy egy 2-fejű automata elfogad egy w input szót, ha $(w, q_0) \vdash^* (\lambda, q)$ valamely $q \in F$ állapotra. Egy automata által elfogadott szavak halmaza jelenti az automata által elfogadott/felismert nyelvet.



Az ábrán látható automata az $L=\{a^n b^n \mid n>0\}$ nyelvet fogadja el.

Az automata működés közben:

$a^n b^n$ elfogadása



A kétfejű automata az $L = \{a^n b^n \mid n > 0\}$ nyelv szavait fogadja el.

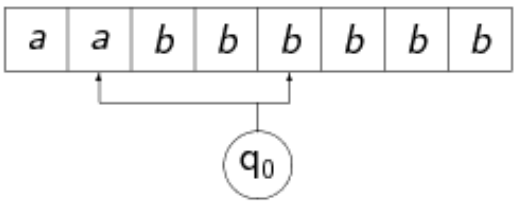
$$A = (\{q_0, q_v\}, \{a, b\}, q_0, \delta, \{q_v\}),$$

$\delta(q_0, a, b) = q_v$,
$\delta(q_v, a, b) = q_v$.

Ha az állapotátmenet-függvény alakja $d:((Q_1 \times T \times \{\lambda\}) \cup (Q_2 \times \{\lambda\} \times T)) \rightarrow Q$, ahol $Q=Q_1 \cup Q_2$, Q_1 és Q_2 diszjunktak, akkor determinisztikus 2-fejű automatáról beszélünk.

További példák kétfejű automaták működésére:

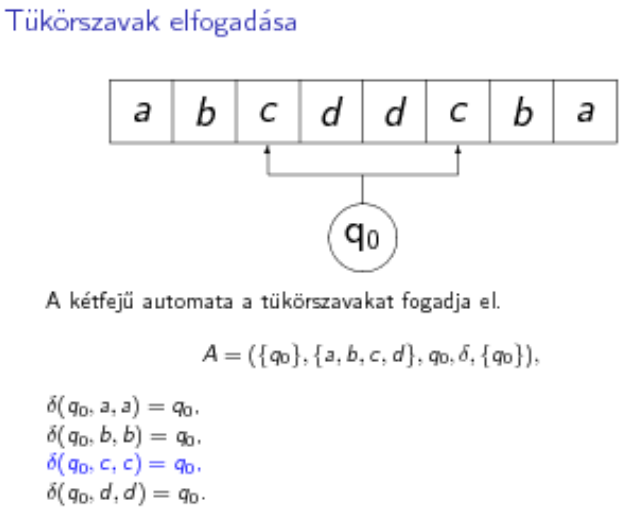
$a^n b^{3n}$ elfogadása



A kétfejű automata az $L = \{a^n b^{3n} \mid n \geq 0\}$ nyelv szavait fogadja el.

$$A = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \delta, \{q_0\}),$$

$\delta(q_0, \lambda, b) = q_1$,
$\delta(q_1, \lambda, b) = q_2$,
$\delta(q_2, a, b) = q_0$.



38. Tétel. A nemdeterminisztikus 2-fejű automaták által elfogadott nyelvek osztálya megegyezik a lineáris nyelvek osztályával.

Bizonyítás. Konstruktív mindkét irányban. Legyen adva egy (N, T, S, H) lineáris nyelvtan normálformában. Ekkor megkonstruáljuk azt a 2-fejű automatát, amely a megadott nyelvtan által generált nyelvet fogadja el:

a (Q, T, q_0, d, F) elemeit adjuk meg a következő módon:

Q=N∪{q_f}, ahol q_f∉N ;

T a terminális ábécé megegyezik az automata inputábécéjével;

q_0=S;

F={q_f} ;

d pedig a következőképpen van definiálva a H szabályhalmaz alapján:

A∈d(B, a, b), ha $B \rightarrow aAb$∈H(A, B∈N;a, b∈T∪{λ}),

q_f∈d(A, a, λ), ha $A \rightarrow a$∈H(A∈N;a∈T∪{λ}).

Könnyen belátható lépésenkénti indukcióval, hogy a nyelvtan minden egyes termináló levezetésének pontosan egy elfogadó út fog megfelelni az automatában, és más elfogadó út nem lesz. Legyen most adva egy (Q, T, q_0, d, F) 2-fejű automata, amihez megkonstruáljuk azt a nyelvtant, ami az automata által elfogadott nyelvet fogja generálni. Az általánosság csorbítása nélkül feltehetjük, hogy Q és T halmazok diszjunktak (ha ez nem teljesül, a Q halmaz elemeinek átnevezésével ez elérhető). Legyen ez alapján az (N, T, S, H) nyelvtan definiálva a következőképpen:

N=Q;

T a terminális ábécé megegyezik az automata inputábécéjével;

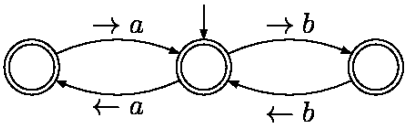
S=q_0;

a H halmaz elemeit pedig a következőképpen definiáljuk:

legyen $A \rightarrow aBb$∈H, ha B∈d(A, a, b) (A, B∈Q;a, b∈T∪{λ}), és

legyen $A \to ab \in H$, ha $B \in d(A, a, b)$ $(A, B \in Q; B \in F; a, b \in T \cup \{\lambda\})$.

Könnyen belátható, hogy az automata minden elfogadó útjához pontosan egy termináló levezetés fog tartozni, és csak olyan termináló levezetések lesznek amik ilyenek. ∎



Az ábrán a palindromák nyelvét elfogadó determinisztikus automata látható, az átmenteknél a nyíl iránya mutatja hogy melyik fej (melyik irányba lépő fej) lép az adott átmenetben. (A $\to a$ átmenet megfelel a korábban használt (a,λ); míg a $\leftarrow a$ a korábbi (λ,a)-val jelölt átmenetnek.)

6.10. példa - Kétfejű automata készítése lineáris nyelvtanhoz

Adott a következő nyelvtan:($\{S, A, B, C\}$, $\{a, b, c\}$, S, $\{S \to aaaAbb, S \to aBaa, A \to aaaAb, A \to c, B \to aBaa, B \to Ca, C \to cC, C \to c\}$). Az ezzel ekvivalens erős normálformájú nyelvtan: ($\{S, A, B, C, D, E, F, G, I, J, K, L, M, O, P\}$, $\{a, b, c\}$, S, $\{S \to aD, D \to aE, E \to aF, F \to Gb, G \to Ab, S \to aI, I \to Ja, J \to Ba, A \to aK, K \to aL, L \to aM, M \to Ab, A \to c, B \to aO, O \to Pa, P \to Ba, B \to Ca, C \to cC, C \to c\}$).

Ennek megfelelően, az ez alapján konstruált 2-fejű elfogadó automata:

	(a, λ)	(b, λ)	(c, λ)	(λ, a)	(λ, b)	(λ, c)
S	D,I					
A	K		q_f			
B	O			C		
C			C,q_f			
D	E					
E	F					
F					G	
G					A	
I				J		
J				B		
K	L					
L	M					
M						
O				P	A	
P				B		
q_f						

Az első sor a lehetséges átmeneteket, az első oszlop pedig az állapotokat tartalmazza, S a kezdőállapot, q_f pedig az egyetlen végállapot. ★

7. *Megjegyzés.* A determinisztikus 2-fejű automaták által elfogadott nyelvek osztálya valódi része a lineáris nyelvek osztályának, jele: 2detLin.

Itt jegyezzük meg, hogy a szakirodalomban egy másik automatatípus (egyszerforduló veremautomaták [164]) is ismert, amely pontosan a lineáris nyelveket ismeri fel, ezekről is lesz szó a következő fejezetben.

6.3. A szóprobléma megoldása

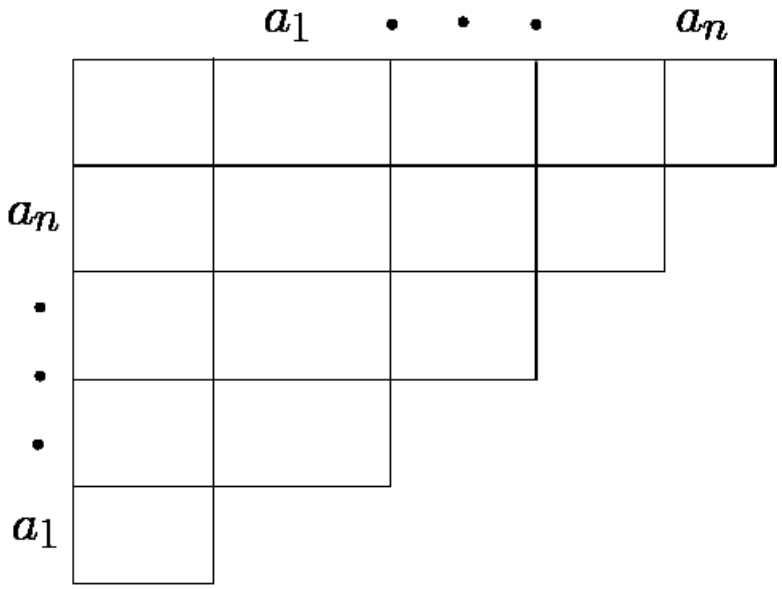
A szóprobléma megoldása (annak eldöntése, hogy egy adott szó szerepel-e az adott lineáris nyelvben) egyben a szó egy lehetséges levezetését is magában foglalja, így szóelemző algoritmusnak is hívhatjuk. Az algoritmus alapja egy felismerési mátrix.

Algoritmus (szóelemzés lineáris nyelvtan esetén)

Input:

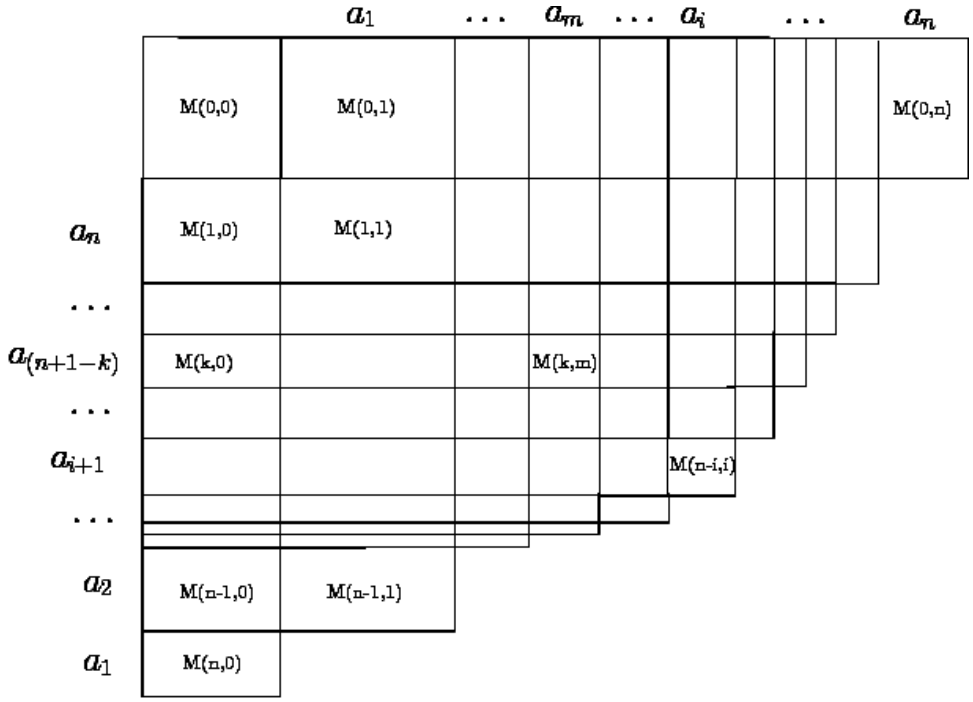
Legyen adott egy lineáris nyelvtan (N, T, S, H) erős normálformában és egy inputszó $w \in T^*$, (jelöljük a szó betűit: a_1, a_2, $\ldots a_n$ –nel ahol $n=|w|$).

1. a mátrix megrajzolása: legyen a háromszögmátrixban a sorok és oszlopok száma n+1, vagyis az input szó hosszánál eggyel több (az oszlopok fölé (eredeti sorrendben) és a sorok elé (visszafelé, vagyis fordított sorrendben) pedig írjuk be az input betűit a következőképpen:



Mint látni fogjuk, valójában az utolsó sorra nem lesz szükség.

2. a mátrix kitöltése (sorfolytonosan):



A kitöltést az M(0,0) cellával kezdjük, az első sort az M(0,*n*) cellával fejezzük be ezután kezdjük a második sort, végül az utolsó előtti sorban az M(*n*-1,1) mezővel fejezzük be. A cellák tartalmai a nemterminálisok részhalmazai lesznek, kivéve az átlóbeli cellák, ahol + vagy – fog szerepelni.

2a. az M(0,0) cella kitöltése: írjuk be az *S* mondatszimbólumot ebbe a cellába.

2b. egy belső (nem átlóbeli) M(*k,m*) cella kitöltése, ($k + m < n$ esetén) a cella kitöltése a cella baloldali szomszédja és a cella feletti cella alapján (ha vannak ilyenek) történik:

- ha $k > 0$, (azaz nem a 0. sorban vagyunk) akkor legyen $A \in M(k, m)$, ha van olyan $B \in M(k\text{-}1, m)$, amelyre teljesül, hogy $B \rightarrow Aa_{(n+1-k)} \in H$.

- ha $m > 0$, (vagyis nem a 0. oszlopban vagyunk) akkor legyen $A \in M(k, m)$, ha van olyan $B \in M(k, m\text{-}1)$, amelyre teljesül, hogy $B \rightarrow a_m A \in H$.

2c. Átlóbeli elemek kitöltése: M(*n*-*i*,*i*) cella kitöltése: az M(*n*-*i*,*i*-1) cellában (balszomszéd cella) szereplő nemterminálisok és az $A \rightarrow a$ alakú szabályok alapján: pontosan akkor írunk + jelet a cellába, ha az M(*n*-*i*,*i*-1) cellában van olyan nemterminális szimbólum *B*, amire van $B \rightarrow a_i$ szabály a nyelvtanban. Ellenkező esetben a – jelet írjuk a cellába.

3. az eredmény leolvasása: Ha az átlóban van + elem, akkor a szó benne van a nyelvtan által generált nyelvben, különben nincs.

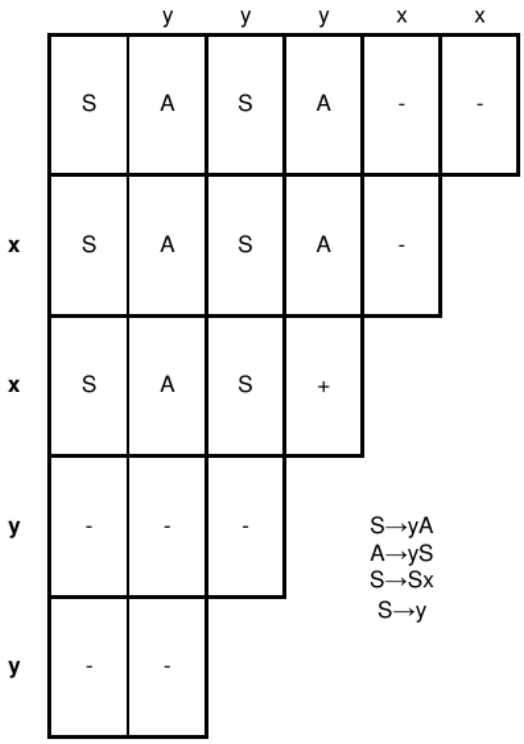
8. *Megjegyzés.* ha a főátló valamely már kitöltött eleme (+) alapján el tudjuk dönteni a választ, vagyis sikeres levezetés van a táblában, akkor a többi mező kitöltése nem szükséges.

Az algoritmus az erős normálforma alapján készített 2-fejű automata működését szimulálja, vízszintes lépésnél az első fej, függőleges lépésnél pedig a második fej lépésével, a főátlóban levő mezők jelzik a két fej lehetséges találkozási pontjait. Amennyiben a szó benne van a generált nyelvben, a kezdő *S* szimbólumtól a tábla valamely + szimbólumáig vezető út (ahogy a kitöltés során a szomszédos cellákat figyelembe vettük) megadja a szó egy lehetséges levezetését is. Az algoritmus a mátrix kitöltéséből, és a megoldás leolvasásából áll, időben és térben is (determinisztikusan) négyzetes bonyolultságú (a táblázat mérete miatt).

6.11. példa - A szóprobléma megoldása lineáris nyelvekre 1.feladat

Adott a $G=(\{S,A\}, \{x,y\}, S, H)$, ahol H szabályai:
$\{ S \to yA, A \to yS, S \to Sx, S \to y \}$.
Benne van-e a nyelvtan által generált nyelvben az *yyyxx* szó?

Megoldás:

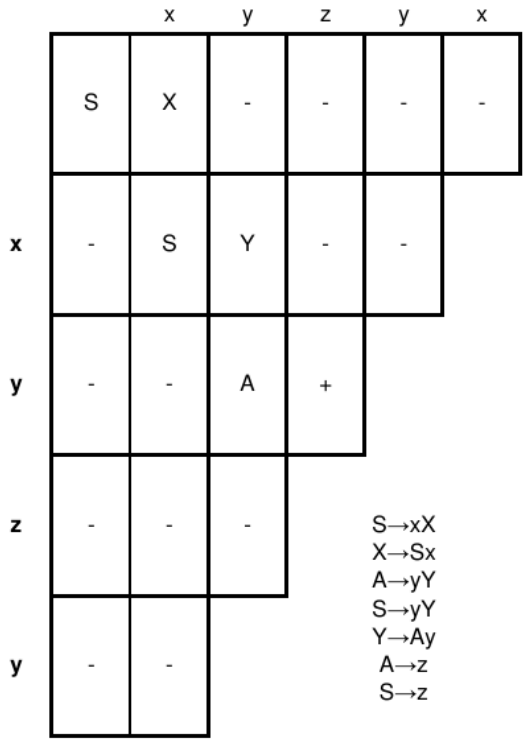


Mivel az átlóban szerepel +-jel, ezért a szó előállítható az adott nyelvtan segítségével. ★

6.12. példa - A szóprobléma megoldása lineáris nyelvekre 2.feladat

Adott a $G=(\{S,A\}, \{x,y\}, S, H)$, ahol H szabályai:
$\{S \to xX, X \to Sx, A \to yY, S \to yY, Y \to Ay, A \to z, S \to z\}$.
Benne van-e a nyelvtan által generált nyelvben az *xyzyx* szó?

Megoldás:



Mivel az átlóban szerepel +-jel, ezért a szó előállítható az adott nyelvtan segítségével. ★

6.4. A lineáris nyelvek tulajdonságai

6.4.1. Iterációs lemma

Lineáris nyelvekre a következő pumpálási (iterációs) tulajdonságot fogjuk bizonyítani:

39. Tétel. Legyen L egy lineáris nyelv. Ekkor létezik olyan (a nyelvtől függő) n természetes szám, hogy minden $z \in L$ szóra, melyre $|z|>n$, van a szónak olyan $z=uvwxy$ felbontása amelyre teljesülnek a következő feltételek

1. $uv^iwx^iy \in L$ minden i természetes számra ($i \geq 0$)

2. $|vx|>0$

3. $|uvxy| \leq n$.

Bizonyítás. Tegyük fel, hogy L nyelv lineáris, ekkor legyen $G=(N, T, S, H)$ olyan lineáris nyelvtan, amely L-et generálja és erős normálformában van. Legyen $n=|N|+2$. Ekkor legyen $w \in L$ tetszőleges

olyan szó, amelyre $|w|>n$. Tekintsük w levezetési fáját. Mivel miden lépésben pontosan egy terminális kerül bevezetésre a levezetés hossza (lépéseinek száma) éppen megegyezik a szó hosszával. A levezetési fa főága (amely tartalmazza a levezetés során szereplő összes nemterminálist és egy terminális levélelemet) legalább $|N|+1$ nemterminális szimbólumot tartalmaz, tehát legalább egy nemterminálist kétszer tartalmaz az első $|N|+1$ eleme közt. Legyen ez a nemterminális A. A első előfordulásakor a levezetés uAy mondatformánál tart (definiáljuk ezzel az u és y részszavakat), míg az A második előfordulásakor legyen az $uvAxy$ az aktuális mondatforma (definiáljuk így a v és x részszavakat), Végül legyen $A\Rightarrow^* w$ az A második előfordulásából generált részszó. Ennek megfelelően világos, hogy $A\Rightarrow^* vwx$ az A első előfordulásából generált részszó, valamint $A\Rightarrow^* vAx$ mondatforma. Amennyiben az A első előfordulásakor a levezetés folytatása $uAy\Rightarrow^* uwy$, akkor éppen az i=0 esetbeli szót vezethetjük le. Ha az A második előfordulásakor nem a w hanem vwx szót vezetjük le belőle, akkor kapjuk az i=2 esetnek megfelelő szót, viszont ebben a levezetésben $uAy\Rightarrow^* uvAxy\Rightarrow^* uvvAxxy$ mondatforma is szerepel, ahol az A-ból a w helyett az uwy szót levezetve az i=3 esetbeli iterált szót kapjuk és így tovább. Az i értéke bármely természetes számra növelhető. Az eredeti felosztásunk és az erős normálforma miatt a $|vx|>0$ és az $|uvxy| \leq n$ feltételek automatikusan teljesülnek. Ezzel a bizonyítást befejeztük. ∎

Sajnos az iterációs lemma nem adja szükséges és elégséges feltételét annak, hogy egy nyelv lineáris legyen, vagyis ha a nyelv lineáris akkor a lemma feltételeinek szükségszerűen teljesülnie kell, viszont ha teljesül a lemma valamely nyelvre az még nem elégséges bizonyíték arra, hogy a nyelv lineáris legyen. Ezek alapján a lemmát arra használhatjuk, hogy amennyiben sikerül belátnunk egy adott nyelvről, hogy nem teljesülnek rá a lemma feltételei, akkor a nyelv biztosan nem lineáris.

6.13. példa - Lineáris iterációs lemma alkalmazása

Legyen $L=\{ a^k b^k a^m b^m \mid k, m>0\}$. Az iterációs lemma segítségével belátjuk, hogy L nyelv nem lineáris. Tegyük fel, indirekt, hogy L lineáris. Ekkor L-re teljesülnie kell a lemmának. Legyen n az a konstans ami a lemma alapján ehhez a nyelvhez tartozik. Vegyük az $a^{2n}b^{2n}a^{2n}b^{2n}$ alakú szót, ami eleme L-nek, másrészt a hossza $8n$, így nagyobb, mint n. Tehát a szót fel kell tudnunk bontani $uvwxy$ részszavakra, hogy v és x nem lehet egyszerre az üresszó, valamint $|uvxy| \leq n$. Viszont ekkor $|uv| \leq n$ és $|xy| \leq n$ is fennáll, vagyis v csak az első a^{2n} beli résznek lehet részszava, így ha nem nulla a hossza, akkor a^i $(0<i \leq n)$ alakú. Viszont az első b^{2n} blokk egésze csak a w-hez tartozhat. Így az iteráció során az első a-kból álló blokkban az a-k száma megváltozna, míg a hozzátratozó első b-ket tartalmaz blokk maradna b^{2n}. Így nem L-beli szót kapunk, tehát v-nek az üresszónak kell lennie. Ekkor viszont x nem lehet üres és csak az utolsó b^{2n} blokkból tartalmazhat betűket. Az előző esettel analóg módon belátható, hogy a pumpálás kivezet az L nyelvből, ha x nem az üresszó. De v és x nem lehet egyszerre üres a lemma állítása miatt. Az ellentmondást csak az indirekt feltételünk okozhatja, tehát a nyelv nem lineáris. ★

6.4.2. Zártsági tulajdonságok

40. Tétel. A lineáris nyelvek osztálya zárt az unió műveletre.

Bizonyítás. Legyenek L_1 és L_2 lineáris nyelvek $G_1=(N_1, T, S_1, H_1)$ és $G_2=(N_2, T, S_2, H_2)$ lineáris nyelvtanokkal, ahol N_1 és N_2 diszjunkt halmazok. Konstruáljuk meg az $(N_1 \cup N_2 \cup \{S\}, T, S, H)$ nyelvtant, ahol S új szimbólum, nem szerepel sem N_1, sem N_2 elemei közt, H pedig a következőképpen definiált: $H=H_1 \cup H_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}$. Könnyen belátható, hogy az új nyelvtan lineáris és éppen L_1 és L_2 unióját generálja. ∎

41. Tétel. A lineáris nyelvek osztálya nem zárt a konkatenációra, a Kleene-iterációra.

Bizonyítás. Korábbi példaként láttuk, hogy $L=\{a^n b^n \mid n>0\}$ lineáris nyelv, saját magával konkatenálva az $LL=\{a^k b^k a^m b^m \mid k, m>0\}$ nyelvet kapjuk, amiről az imént mutattuk meg, hogy nem lineáris. Mivel $LL \subseteq L^*$, ennek esetünkben egyenes következménye, hogy L^* nyelv sem lineáris. ∎

42. Tétel. A lineáris nyelvek osztálya nem zárt a metszet és a komplementer műveletekre.

Lásd környzetfüggetlen nyelvekre vonatkozó 56. és 57. tételben.

A továbbiakban a lineáris nyelvtanoknak (és nyelveknek) speciális alosztályát, illetve kiterjesztését vizsgáljuk. Amar és Putzolu az 1960-as években definiálta a következő, a reguláris és a lineáris nyelvcsaládok közti nyelvosztályokat.

12. Definíció. Legyen k egy rögzített nemnegatív racionális szám. Ha egy G lineáris nyelvtan minden $A \to uBv$ alakú szabályára igaz, hogy $k = \frac{|v|}{|u|}$, , akkor G-t k-arányú (vagy k-fokú) lineáris nyelvtannak, az általa generált nyelvet pedig k-arányú lineáris nyelvnek hívjuk. Egy nyelvtant (illetve nyelvet) fix-arányú lineárisnak nevezünk, ha k-arányú lineáris valamely nemnegatív racionális k értékre. ★

Speciális esetben k=0 érték esetén éppen a reguláris nyelveket kapjuk vissza. Nevezetes még a k=1 eset, ezeket a nyelveket és nyelvtanokat páros lineárisnak hívjuk. Erre példa a palindromák nyelve.

43. Tétel. Minden k nemnegatív racionális számra, a k-fokú lineáris nyelvek osztálya tartalmazza a reguláris nyelvek osztályát.

44. Tétel. Minden fix-arányú lineáris L nyelvhez létezik olyan determinisztkus 2-fejű automata, ami éppen L-et fogadja el.

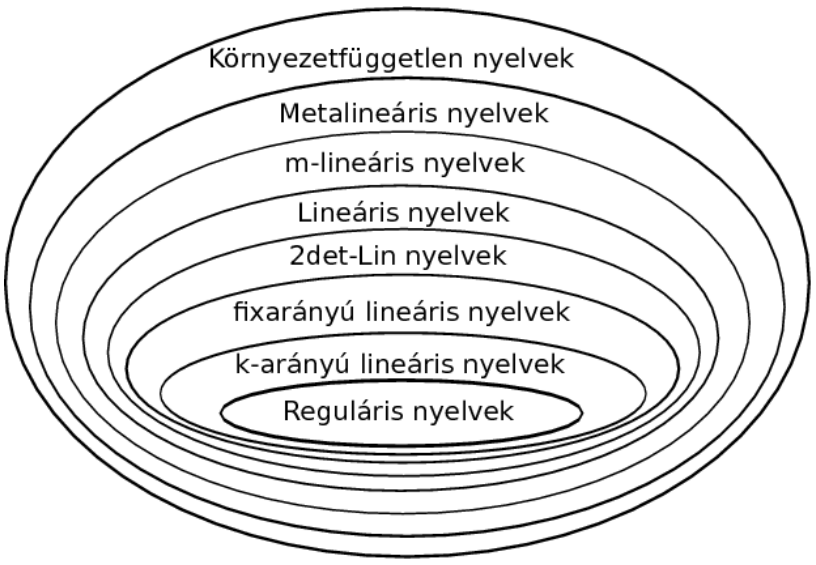
13. Definíció. Egy G=(N, T, S, H) nyelvtant m-lineárisnak nevezünk (ahol m>0 egész szám), ha szabályainak mindegyike eleget tesz a lineáris nyelvtanoknál megadott definíciónak, kivéve az egyetlen $S \to S_1S_2\ldots S_m$ szabályt, és S nem szerepel egyik szabály jobb oldalán sem. Az m-lineáris nyelvtanok által generált nyelvek osztályát m-lineáris nyelvek osztályának nevezzük. Egy nyelvtan, illetve nyelv metalineáris ha m-lineáris valamilyen m-re. ★

Egy metalineáris nyelvtanban a mondatformában már több nemterminális is előfordulhat egyszerre, de számuk maximálva van, ha a nyelvtan m-lineáris, akkor maximum m nemterminális lehet egyszerre. A lineáris nyelvek konkatenációjával éppen a metalineáris nyelveket állíthatjuk elő.

6.14. példa - Az $a^kb^ka^mb^m$ metalineáris nyelv

G=({S, A, B}, {a, b}, S, {$S \to AB$, $A \to aAb$, $A \to ab$, $B \to aBb$, $B \to ab$}) nyelvtan éppen a korábban már bemutatott {$a^kb^ka^mb^m$ | k, m>0} nyelvet generálja. ★

A következő ábrán a lineáris nyelvekhez kapcsolódó hierarchiát mutatjuk be, a hierarchia éles, vagyis minden tartalmazás szigorú (kivéve a k=0 és m=1 eseteket, amikor a 0-fokú lineáris nyelvek éppen a reguláris az 1-lineáris nyelvek, pedig éppen a lineáris nyelvekkel esnek egybe).



6.5. Irodalmi megjegyzések

A páros lineáris és a fix-arányú lineáris nyelvtanokat és nyelveket az [Amar, Putzolu 1964, 1965] vezették be. A lineáris nyelvek, illetve azok speciális változatainak, kiterjesztéseinek vizsgálata

ma is aktív terület, pl. tanulási problémákban is alkalmazzák őket [Sempere, García 1994]. Ezen nyelvosztályokhoz tartozó iterációs lemmák találhatók a [Horváth, Nagy 2010] cikkben. A kétfejű automatákkal és ezek speciális változataival elfogadott nyelvosztályokról pl. a [Nagy 2008, 2011b, 2011c] és [Leupold, Nagy 2010] cikkekben lehet olvasni.

7. fejezet - Környezetfüggetlen nyelvek

Ebben a fejezetben a Chomsky-féle 2-típusú nyelvek sajátosságait tárgyaljuk. E nyelvosztály ugyancsak sok területen jól alkalmazható és vannak rá hatékony algoritmusok, melyek közül néhányat részletesen is bemutatunk.

7.1. Programnyelvek szintaktikájának leírása

Különböző programnyelvek elemeinek (mint pl. számjegy, szám, változó név, utasítás, eljárás stb.) megadásához sokféle módszer ismert. Ilyenek pl. a BNF (Backus-Naur Form), a COBOL-szerű megadási mód, a szintaxis gráf és a hibrid módszer. A leírás terminális egységeket és nemterminális egységeket tartalmaz BNF-ben. A nemterminálisokat más egységekből a konkatenáció, az alternatíva, az iteráció és ezek egy speciális esetének, az opciónak a segítségével adhatjuk meg. Ugyanezek a lépések grafikusan a szintaxis-gráfban is megtalálhatóak. Ezeket a leírási módokat ismertetjük röviden a következőkben, kiterjesztve az egyszerű szintaxis gráfok korábban bemutatott (lásd Szintaxis gráf) alakját is.

A szintaktika leírásához, a különböző programnyelvek megadásánál, használt műveletek:

- konkatenáció (összefűzés: több szövegelem egymás mellé/után írása),
- alternatíva (választás: különböző lehetőségek közül egy kiválasztása),
- opció (a szövegelem vagy megjelenik vagy nem),
- iteráció (a szövegelem akárhányszor megjelenhet (általában a nullaszori megjelenést is beleértjük)).

Lássuk, akkor most hogyan is néznek ki a már említett leírási módok.

7.1.1. A BNF megadási mód

Ezt a megadási módot abban az időben találták ki, amikor az ALGOL60 nyelvet fejlesztették.

Terminális	Nemterminális	Alternatíva	Opció	Iteráció
íráskép	< > : : = magyarázat	|	[ ]	{ }

A konkatenációnak nincs külön jele, egyszerűen egymás után írjuk a megfelelő szövegelemeket. A nemterminális jeleket < > zárójelek közé tesszük, és ::= definiáló egyenlőségjel segítségével definiáljuk.

7.1. példa - A számok jelölésére használható karaktersorozatok

Az egész számok megadásához szükség van az opcióra (van előjel vagy nincs), ha van: alternatíva (plusz vagy mínusz), a számjegyek sorozatát pedig iterációval adjuk meg:

< számjegy > ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

< előjel > ::= + | –

< egész szám > ::= [< előjel >] < számjegy > { < számjegy > } ★

7.1.2. A COBOL-szerű megadási mód

Ez a megadási mód a PL/1 nyelv idején volt leginkább használatban.

Terminális	Nemterminális	Alternatíva	Opció	Iteráció
kisbetű v. íráskép	nagybetű : magyarázat	{ }	[ ]	... a megelőző szövegelem ismétlése

7.2. példa - A számok megadása COBOL-szerű módszerrel

$$\text{SZÁMJEGY} : \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \right\} \qquad \text{ELŐJEL} : \left\{ \begin{array}{c} + \\ - \end{array} \right\} \qquad \text{EGÉSZ SZÁM} : [\ \text{ELŐJEL}\ ]\ \text{SZÁMJEGY} \ldots$$

★

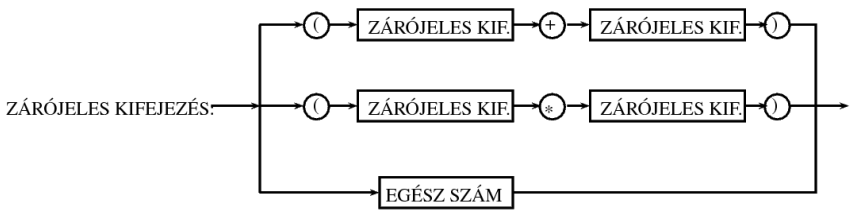
7.1.3. A szintaxis gráf

Itt a korábban már a reguláris nyelvek esetén ismertetett Szintaxis gráf módszer teljes kifejezőerejű változatát mutatjuk be.

A korábbiakhoz képest a változás az, hogy egy nemterminális definíciójában felhasználhatunk még nem definiált nemterminálisokat is, akár a definiálandó nemterminálist saját magát is. Így megengedjük a rekurzió kialakulását, akár közvetlenül egy saját magát "meghívó" definícióval, vagy akár egymást hívó több nemterminális definíciójával.

Természetesen, így minden műveletnél, a konkatenációnál, alternatívánál nemcsak terminálisok, hanem nemterminálisok, illetve bármilyen, a módszerrel már összetett gráfok is előfordulhatnak. Az iteráció pedig akár úgy is előfordulhat, mint az opció, csak fordított nyíliránnyal az adott szövegelemnél (megengedve a nullaszoros ismétlést).

Egy szintaxis gráfban mindig van pont egy indulóél és egy érkezőél, ahonnan indulva és ahova érkezve kell egy utat bejárnunk a gráfban. Az út során összeolvassuk a terminálisokat, illetve ha egy nemterminálishoz érünk akkor az adott nemterminális definíciója alapján a neki megfelelő szintaxis gráfban megyünk végig egy úton, ha annak végére értünk akkor folytatjuk az utunkat az eredeti gráfban az adott nemterminális után. Egy nemterminális tehát egy rekurzív hívást jelent, az adott gráfban lefutott út után folytathatjuk csak utunkat. Az hogy megengedjük nem csak a már korábban definiált nemterminálisok használatát egy nemterminális definíciójában, azt jelenti, hogy a rekurziót nem korlátozzuk, annak mélysége bármennyi lehet. A reguláris nyelveknél (5.14. példa - Tizes számrendszerbeli egész számok nyelvének megadása szintaxis gráffal) megadott egész szám fogalmát felhasználva adhatjuk meg például a zárójeles kifejezés fogalmát:



7.1.4. A Hibrid megadási mód

A 90-es évek gyakori leíró nyelve. Tulajdonképpen az előző módszerek keveréke, illetve egyvelege. Az egész számok például a következő módon adhatók meg:

számjegy : { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }

egész szám : [{ + | – }] számjegy ...

Az ismertetett leírási módoknak a kifejezőereje azonos. Mindegyikkel pontosan a reguláris halmazokat tudjuk definiálni, ha a nemterminálisokat nem használjuk. A kifejező erő ugyanennyi, ha a nemterminálisok definíciójában csak a már korábban ugyanígy megadott nemterminálisokat használhatjuk fel. Ezzel szemben, ha megengedjük a rekurziót, vagyis, hogy egy adott nemterminális definíciójában saját magát is felhasználjuk, vagy előtte még nem definiált nemterminálist, akkor a kifejezőerő megnő; így mindegyik ismertetett módszer segítségével pontosan a környezetfüggetlen nyelvek írhatók le. Ezeknek a megadási módoknak a kifejezőereje tehát azonos, velük a környezetfüggetlen nyelveket tudjuk definiálni.

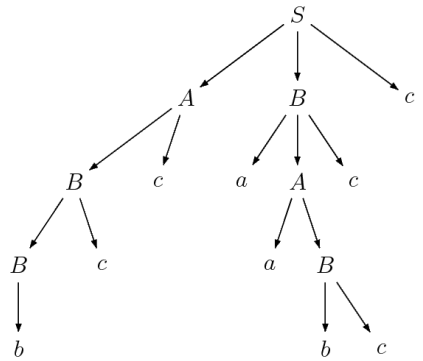
7.2. Levezetési fa

A környezetfüggetlen nyelvek (nyelvtanok) egy tulajdonsága (mely népszerűségükben is központi szerepet játszik), hogy a levezetések fa alakban ábrázolhatók.

Környezetfüggetlen nyelvtanban minden levezetés nagyon egyszerűen ábrázolható egy irányított gráf (fa) segítségével. Tekintsünk egy $S \Rightarrow^* p$ levezetést, ahol $p \in (N \cup T)^*$. A gráf csúcsainak nemterminális, illetve terminális szimbólumokat feleltethetünk meg: a fa gyökeréhez S- t, leveleihez rendre terminális és nemterminális szimbólumokat, a közbülső csúcsaihoz pedig nemterminális jeleket rendelhetünk. Az egy csúcsból kiinduló élek annak a helyettesítési szabálynak az alkalmazását jelölik, amelynek bal oldalán az élek közös kiindulási pontjában található nemterminális jel áll, a jobb oldalán pedig az élek végpontjaiban található nemterminális, illetve terminális jelek sorozata (az egyes éleket balról jobbra véve sorra). Azt, hogy egyes szabályokat milyen sorrendben alkalmazunk, nem mindig tudjuk egyértelműen rekonstruálni. Egy levezetési gráf mélységén a benne az S gyökértől induló és levélelemhez tartó irányított utak hosszának maximumát értjük. Ha a fa minden levéleleme terminális címkéjű, akkor befejezett levezetésről és befejezett levezetési fáról beszélhetünk.

7.3. példa - Levezetési fa

Legyen $G=(\{S, A, B\}, \{a, b, c\}, S, H)$, ahol $H=\{S \to ABc,\ A \to aB,\ A \to Bc,\ B \to aAc,\ B \to bc\}$. Ebben a nyelvtanban megadható a következő levezetés: $S \Rightarrow ABc \Rightarrow AaAcc \Rightarrow BcaAcc \Rightarrow BcaaBcc \Rightarrow bccaaBcc \Rightarrow bccaabccc$.



★

Egy levezetési fa általában nem egy levezetést ábrázol (vagyis több olyan levezetés is lehetséges melynek ábrázolásával ugyanazt a fát kapjuk). Az egy fa által reprezentált levezetések viszont egy

ekvivalencia osztályt alkotnak, hiszen mindegyikükben ugyanaz a mondatforma (vagy szó) van levezetve, illetve az ugyanott megjelenő ugynarra a nemterminálisra ugyanazt a szabályt alkalmazzuk. Egyedül a szabályalkalmazások sorrendje lehet különböző.

Legbaloldalibb levezetésnek hívunk egy levezetést, ha a levezetés során minden lépésben az aktuális mondatforma legelső (legbaloldalibb) nemterminálisára végzünk el egy helyettesítést valamely alkalmas levezetési szabály segítségével.

45. Tétel. Minden levezetési fához egyértelműen hozzárendelhető egy legbaloldalibb levezetés.

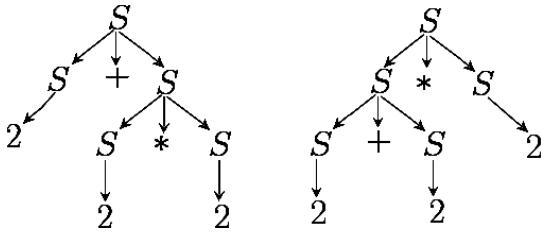
Az adott levezetési fához tartozó levezetések közül tehát egyértelműen kiválaszthatunk egyet, amivel az osztályt reprezentálhatjuk: a legbaloldalibbat.

A legbaloldalibb levezetés arra hasonlít, mintha a levezetési fát a mesterséges intelligenciában mélységi keresőként ismert algoritmus segítségével építenénk fel.

7.2.1. Többalakúság

Fontos szerepet játszik az egyértelműség kérdése a következő tekintetben. Adott nyelvtan esetén előfordulhat, hogy egy adott szóhoz több különböző levezetési fa létezik. A természetes nyelvekre nagyon jellemző ez a többalakúság. A "láttam Pétert egy távcsővel." mondat értelmezése lehet kétféle: vagy a tárgy rendelkezik egy részeshatározóval: Péter a kezében vitt egy távcsövet, amikor láttam, hogy megy kirándulni. A másik elemzés szerint az alany (én) rendelkezem egy távcsővel (eszközhatározós szerkezet) és ennek segítségével láttam Pétert, ahogy jön föl a hegyre. A természetes nyelvek esetén általában a szövegkörnyezet, vagy az adott szituáció segít kiválasztani a helyes elemzést...

Ezzel szemben, pl. a programnyelvek leírásakor törekednünk kell az egyértelműségre. Például a $G=(\{S\}, \{2, +, *\}, S, \{S \rightarrow S+S, S \rightarrow S*S, S \rightarrow 2\})$ nyelvtanban $S \Rightarrow S+S \Rightarrow S+S*S \Rightarrow^* 2+2*2$ valamint $S \Rightarrow S*S \Rightarrow S+S*S \Rightarrow^* 2+2*2$ ugyanannak a szónak két különböző levezetése, a levezetési fákat a következő ábra mutatja.



Viszont a levezetés elemzése alapján az első értelmezés szerint 2+(2*2)=6 az eredmény, míg a második értelmezés (2+2)*2=8- at jelent. Ez pl. gondot okozhat egy számítógépes fordító részére. Ezért fontos, hogy lehetőleg kerüljük a többalakúságot egy programnyelv tervezése során, nem célszerű ilyen döntéseket a fordítóra hagyni.

7.4. példa - Csellengő "else" feladat

Ismert a csellengő "else" problémája, amikoris nem egyértelmű, hogy egy else-ág melyik feltételhez tartozik. Példaként tekintsük a következő kódot:

if a then if b then do else print

Adjunk példát olyan nyelvtanra, ahol ez a probléma fellép. ★

Ismert tény, hogy vannak olyan környezetfüggetlen nyelvek, amelyeket nem lehet olyan nyelvtannal generálni, hogy ne legyen olyan szó amelynél a többalakúság fellép. Ilyen nyelvre példa: $L=\{a^n b^m c^m d^n\} \cup \{a^n b^n c^m d^m\}$.

7.3. Normálformák a környezetfüggetlen nyelvtanokhoz

Gyakorlati szempontból fontos lehet, hogy a nyelvtant minimalizáljuk olyan értelemben, hogy megszabadulunk az olyan felesleges nemterminálisoktól, amik egyetlen terminálló levezetésben sem jelennek meg. Egy A nemterminális két okból lehet felesleges:

- A nem vezethető be mondatformában, vagyis nincs olyan szabálysorozat, hogy az S mondatszimbólumból indulva valamilyen uAv mondatformát kapjunk (valamilyen u, $v \in (N \cup T)^*$ szavakra).

- az A- ból nem lehet terminálni, vagyis nincs olyan $w \in T^*$, hogy $A \Rightarrow^* w$ lenne.

Az első típusba tartozó felesleges nemterminálisokat a következőképpen határozhatjuk meg egy $G=(N, T, S, H)$ nyelvtan esetén:

Legyen $U_0=\{S\}$.

Legyen $U_{i+1}=U_i \cup \{A \mid \text{létezik } B \rightarrow uAv \in H, A \in N, B \in U_i, u, v \in (N \cup T)^*\}$ ($i \geq 0$)

Ekkor az N végessége miatt van olyan i hogy $U_i=U_{i+1}$, és ekkor az $N \setminus U_i$ nemterminálisok feleslegesek, mert nem érhetőek el S- ből kezdődő levezetésekkel.

A második típusba tartozó felelsleges nemterminálisok meghatározása egy $G=(N, T, S, H)$ nyelvtan esetén a következő rekurzív módon történhet:

Legyen $U_0=\{A \mid \text{létezik } A \rightarrow w \in H, A \in N, w \in T^*\}$.

Legyen $U_{i+1}=U_i \cup \{A \mid \text{létezik } A \rightarrow u \in H, A \in N, u \in (U_i \cup T)^*\}$ ($i \geq 0$)

Ekkor az N végessége miatt van olyan i hogy $U_i=U_{i+1}$, és ekkor az $N \setminus U_i$ nemterminálisok feleslegesek, mert belőlük nem vezethető le terminális (vagy üres) szórész. Ha az S nincs benne az U_i halmazban, akkor a generált nyelv üres.

Ha G nemüres nyelvet generál, akkor világos, hogy az így meghatározott felesleges nemterminálisokat, és az összes olyan szabályt, amely tartalmaz ilyen nemterminálist (akár a bal, akár a jobboldalán) elhagyva a kapott G' nyelvtan ekvivalens az eredetivel: a terminálló levezetések megmaradnak, így a generált nyelv nem változik.

Ha speciálisan reguláris nyelvtanból (vagy ennek megfelelő véges automatából) indulunk ki, akkor az első rész éppen a startszóból (vagyis a kezdőállapotból) való elérhetőséget jelenti. A második rész, vagyis azon nemterminálisok (állapotok) összegyűjtése, amiből nem tudunk terminálni a teljesen definiált determinisztikus véges automata esetén az egyetlen nyelő állapotnak felel meg.

A fejezet további részében két fontos normálformát mutatunk be, az elsőnél algoritmust is adunk arra, hogy egy tetszőleges környezetfüggetlen nyelvtanhoz vele ekvivalenset állítsunk elő. Természetesen, ha nem akarunk feleslegesen sok szabállyal dolgozni, akkor célszerű a normálformákat az imént bemutatott felesleges nemterminálisokat már nem tartalmazó nyelvtanokra meghatározni.

7.3.1. Chomsky-féle normálalak

Egy nyelvtant λ- *mentesnek* nevezünk, ha a szabályok jobb oldalán egyáltalán nem fordul elő a λ.

Itt jegyezzük meg, hogy minden 2-típusú $\lambda \notin L$ nyelv esetén megadható olyan 2-típusú λ- mentes grammatika ami L- et generálja (lásd Üresszó-lemma). Ha viszont $\lambda \in L$, akkor igaz az, hogy megadható olyan 2-típusú λ- mentes G grammatika ami a $L \setminus \{\lambda\}$ nyelvet generálja. (Ilyenkor G nemterminális halmazát kiegészítve S' új mondatszimbólummal, és a H szabályrendszert kiegészítve az $S' \rightarrow \lambda$, $S' \rightarrow S$ szabályokkal kapunk egy L- et generáló nyelvtant.)

14. Definíció. Egy környezetfüggetlen nyelvtanról azt mondjuk, hogy *Chomsky-féle normálalak*ban van, ha minden szabálya $A \to a$ vagy $A \to BC$ alakú, ahol A, B, $C \in N$ és $a \in T$. ★

Minden λ- mentes környezetfüggetlen nyelv generálható olyan nyelvtannal, amely Chomsky-féle normálalakú. Avagy, kicsit másképp fogalmazva:

46. Tétel. Minden G környezetfüggetlen nyelvtanhoz van olyan vele ekvivalens környezetfüggetlen G_{Ch} nyelvtan, amely Chomsky-féle normálalakú.

Bizonyítás. Ha a $G=(N, T, S, H)$ nyelvtan nem λ- mentes, vagyis van benne olyan szabály, aminek jobboldala az üresszó, akkor először alkalmazzuk rá az Üresszó-lemma -t, és legyen $G'=(N, T, S, H')$ az a nyelvtan, amit az üresszó lemmánál ismertetett algoritmussal létrehozunk a G által generált nyelv λ- mentes részének generálására, ekkor G és G' ekvivalensek (az általuk generált nyelvek különbsége legfeljebb az üresszót tartalmazza) és G' λ- mentes.

Hozzuk a nyelvtanunkat olyan alakra, hogy terminális jel csak $A \to a$ alakú szabályokban ($A \in N$, $a \in T$) forduljon elő: Vezessük be az X_a új nemterminálisokat a következő módón: legyen $N_t=\{X_a | a \in T\}$ és $N''=N \cup N_t$, ahol $N_t \cap N=\emptyset$. Továbbá a H' minden nem $A \to a (A \in N, a \in T)$ alakú szabályára: cseréljük ki a szabály jobboldalán található a terminálisokat a nekik megfelelő imént bevezetett X_a nemterminálisra, valamint vegyük fel az $X_a \to a$ új szabályokat. Formálisan,

$H''=\{A \to a | a \in T, A \to a \in H\} \cup \{A \to r | A \to r' \in H, r' \notin T$, és $h(r')=r$, ahol h az az izomorfizmus, amely N minden eleméhez önmagát, a T elemeihez pedig N_t megfelelő elemeit rendeli$\} \cup \{X_a \to a | a \in T\}$.

Az így létrejött $G''=(N'', T, S, H'')$ nyelvtan ekvivalens a G'- vel és szabályaiban a terminális csak $A \to a$ alakú szabályban fordul elő.

Ekkor $G''=(N'', T, S, H'')$ nyelvtanban az összes nem $A \to a$ alakú szabályunk $A \to r$ alakú ($A \in N$, $r \in N''^* \setminus \{\lambda\}$). Tekintsük ezek közül azokat a szabályokat, amelyekben $|r|>2$. Egy ilyen $A \to B_1B_2 \ldots B_k$ ($k>2$) alakú szabályt helyettesíthetünk az

$A \to B_1Z_1$

$Z_1 \to B_2Z_2$

…

$Z_{k-2} \to B_{k-1}B_k$

szabályok halmazával, ahol Z_1, Z_2, …, Z_{k-2} újonnan bevezetett nemterminálisok. Ezt a helyettesítést minden kettőnél hosszabb jobb oldalú szabályra külön-külön végrehajtva a $G'''=(N''', T, S, H''')$ nyelvtanunk ekvivalens az előzővel (így az eredetivel is) és H''' csak az alábbi háromféle szabályt tartalmazhatja:

(1) $A \to a$,

(2) $A \to B$,

(3) $A \to BC$.

Tehát a G''' nyelvtanra ki kell küszöbölni a H'''- ből az $A \to B$ alakú (úgynevezett lánc-)szabályokat, hasonlóan ahogy a reguláris és lineáris nyelvtanoknál megtettük.

(Lásd pl. erős normálformájú reguláris nyelvtanok.)

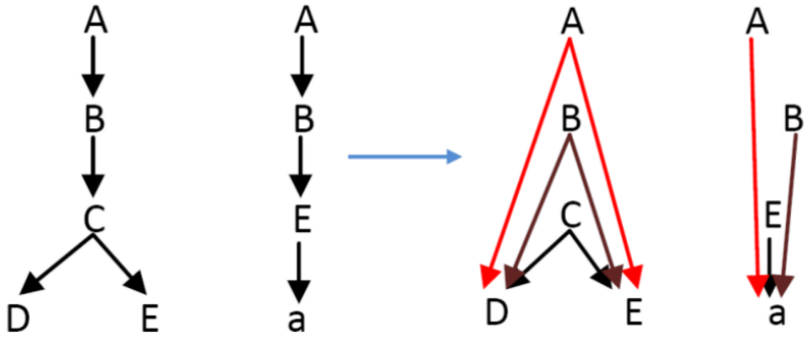
Tehát definiáljuk minden $A \in N'''$ változóhoz a következő halmazokat:

- $U_1(A)=\{A\}$

- $U_{i+1}(A)=U_i(A) \cup \{B \in N''' | \exists C \in U_i(A) | C \to B \in H'''\}$, ha $i>1$

Ekkor N''' véges volta miatt létezik olyan minimális k index, hogy $U_k(A)=U_{k+j}(A)$, ha j=1, 2, …. Jelöljük minden $A\in N'''$ nemterminális jelhez tartozó $U_k(A)$- t $U(A)$- val. Ekkor $U(A)$ éppen azokat a nemterminálisokat tartalmazza, amelyek levezethetőek az A- ból csupán láncszabályokat felhasználva, vagyis tetszőleges A, $B\in N'''$ változókra $A\Rightarrow^{*}B$ pontosan akkor, ha $B\in U(A)$. Ezek után definiáljuk a H'''' szabályhalmazt a következőképpen:

$H''''=\{A \to r \mid$ van olyan $B\in N'''$, hogy $B \to r\in H'''$, $r\notin N$, $B\in U(A)\}$.



Az így kapott $G''''=(N'''', T, S, H'''')$ nyelvtanra $L(G)\setminus\{\lambda\}=L(G'''')$, mivel az $A \to B$ szabályok alkalmazását az előbbi két csoportba tartozó szabályok alkalmazásával biztosítjuk és fordítva. ∎

A Chomsky normálforma használata esetén a leveztési fa egy bináris fa lesz, minden lépésben vagy a mondatforma hossza nő ($A \to BC$ alakú szabály alkalmazása) vagy egy terminális bevezetésével egy ágon befejeződik a levezetés ($A \to a$ alakú szabály esetén).

7.5. példa - Chomsky normálforma 1.feladat

Adjunk meg a $G=(\{S,A,B\},\{a,b,c\},S,H)$ nyelvtannal ekvivalens Chomsky-féle normálalakú nyelvtant, ahol $H = \{ S \to ABaba, A \to B, A \to c, B \to AbA, B \to S \}$.

Megoldás:
(I.) Első lépésben megadunk egy G_1 nyelvtant, ami ekvivalens a G nyelvtannal és (terminális) normálalakú.
Ehhez először a G nyelvtan minden olyan x_i terminális betűjéhez, amely szerepel olyan szabályban,
ami nem normálalakú, új X_i nemterminálist vezetünk be.
Ezután a G_1 nyelvtan H_1 szabályhalmazát úgy kapjuk, hogy felvesszük az $X_i \to x_i$ szabályokat, valamint
a H szabályhalmaz elemeit átvesszük úgy, hogy a szabályokban az x_i betűk azon előfordulásait,
melyek nem (terminális) normálalakú szabályban szerepelnek, X_i-re cseréljük.

Jelen esetben legyenek az új nemterminálisok az X_a és az X_b, ekkor:
$G_1=(\{S,A,B,X_a,X_b\},\{a,b,c\},S,H_1)$, ahol
$H_1=\{ X_a \to a, X_b \to b, S \to ABX_aX_bX_a, A \to B, A \to c, B \to AX_bA, B \to S \}$

(II.) Második lépésben megadunk egy G_2 nyelvtant, ami ekvivalens
az eredeti nyelvtannal, normálformájú és nem szerepel benne $Y \to Y_1Y_2...Y_n$, n>2 alakú szabály.
Ehhez a G_1 nyelvtanból indulunk ki.
A H_1 szabályhalmaz $Y \to Y_1Y_2...Y_n$, n>2 alakú szabályait helyettesítjük új szabályokkal,
a többi szabályt pedig változtatás nélkül átvesszük a H_2 szabályhalmazba.
Minden $Y \to Y_1Y_2...Y_n$, n>2 alakú szabályhoz
vezessünk be $Z_1,Z_2, ... ,Z_{n-2}$ új nemterminálisokat,
úgy, hogy Z_i-ből az $Y_{i+1}...Y_n$ szót tudjuk levezetni:
az összes $Y \to Y_1Y_2...Y_n$, n>2 alakú szabályt helyettesítsük
a következő szabályokkal:
$Y \to Y_1Z_1$,

$Z_1 \rightarrow Y_2Z_2$,
.
.
.
$Z_{n-3} \rightarrow Y_{n-2}Z_{n-2}$,
$Z_{n-2} \rightarrow Y_{n-1}Y_n$.

Jelen esetben:
$G_2=(\{S,A,B,X_a,X_b,Z_1,Z_2,Z_3,Z_4\},\{a,b,c\},S,H_2)$, ahol
$H_2=\{ X_a \rightarrow a, X_b \rightarrow b, S \rightarrow AZ_1, Z_1 \rightarrow BZ_2, Z_2 \rightarrow X_aZ_3, Z_3 \rightarrow X_bX_a, A \rightarrow B, A \rightarrow c, B \rightarrow AZ_4, Z_4 \rightarrow X_bA, B \rightarrow S \}$

(III.) Harmadik lépésben megadjuk az eredeti nyelvtannal ekvivalens
G' Chomsky-féle normálalakú nyelvtant. Ehhez két lépésre van szükség.
Első lépésben meghatározunk egy $U(Z)$ halmazt minden olyan Z nemterminálishoz, mely levezethető legalább egy másik nemterminálisból a G_2 nyelvtanban és
szerepel olyan H_2 halmazban lévő szabály bal oldalán, amelynek jobb oldalán egy terminális vagy pedig két nemterminális betű áll.
Az $U(Z)$ halmaz tartalmazni fogja az összes olyan nemterminálist,
melyből egy vagy több lépésben levezethető a Z betű.

Jelen esetben:
$U(B)=\{A\}$,
$U(S)=\{B,A\}$.
Második lépésben a H' szabályhalmazba átvesszük a H_2 szabályhalmaz mindazon szabályait,
melyek jobb oldalán egy terminális betű vagy pedig kettő nemterminális található,
majd hozzávesszük mindazon szabályokat, melyeket úgy kapunk, hogy a már
átvett szabályok bal oldalán szereplő betűt a hozzá tartozó U halmaz elemeivel helyettesítjük.

Formálisan:
$H'=(H_2 \cup \{Z \rightarrow p \mid X \rightarrow pZ \in H_2, Z \in U(X)\}) \setminus \{X \rightarrow Y \mid X,Y \in N\}$.Jelen esetben:
$G'=(\{S,A,B,X_a,X_b,Z_1,Z_2,Z_3,Z_4\},\{a,b,c\},S,H')$, ahol
$H'=\{ X_a \rightarrow a, X_b \rightarrow b, S \rightarrow AZ_1, B \rightarrow AZ_1, A \rightarrow AZ_1, Z_1 \rightarrow BZ_2, Z_2 \rightarrow X_aZ_3, Z_3 \rightarrow X_bX_a, A \rightarrow c, B \rightarrow AZ_4, A \rightarrow AZ_4, Z_4 \rightarrow X_bA \}$ ★

7.6. példa - Chomsky normálforma 2.feladat

Adjunk meg a $G=(\{S,A,B\},\{x,y,z\},S,H)$ nyelvtannal ekvivalens Chomsky-féle normálalakú nyelvtant, ahol
$H=\{ S \rightarrow BB, A \rightarrow S, A \rightarrow xxzz, A \rightarrow y, B \rightarrow AxzxA, B \rightarrow A \}$.

Megoldás:
(I.)
Legyenek az új nemterminálisok az X és a Z, ekkor:
$G_1=(\{S,A,B,X,Z\},\{x,y,z\},S,H_1)$, ahol
$H_1=\{ X \rightarrow x, Z \rightarrow z, S \rightarrow BB, A \rightarrow S, A \rightarrow XXZZ, A \rightarrow y, B \rightarrow AXZXA, B \rightarrow A \}$
(II.)
$G_2=(\{S,A,B,X,Z,Z_1,Z_2,Z_3,Z_4,Z_5\},\{x,y,z\},S,H_2)$, ahol
$H_2=\{ X \rightarrow x, Z \rightarrow z, S \rightarrow BB, A \rightarrow S, A \rightarrow XZ_1, Z_1 \rightarrow XZ_2, Z_2 \rightarrow ZZ, A \rightarrow y, B \rightarrow AZ_3, Z_3 \rightarrow XZ_4, Z_4 \rightarrow ZZ_5, Z_5 \rightarrow XA, B \rightarrow A \}$
(III.)
$U(S)=\{A,B\}, U(A)=\{B\}$.
$G'=(\{S,A,B,X,Z,Z_1,Z_2,Z_3,Z_4,Z_5\},\{x,y,z\},S,H')$, ahol
$H'=\{ X \rightarrow x, Z \rightarrow z, S \rightarrow BB, A \rightarrow BB, B \rightarrow BB, A \rightarrow XZ_1, B \rightarrow XZ_1, Z_1 \rightarrow XZ_2, Z_2 \rightarrow ZZ, A \rightarrow y, B \rightarrow y, B \rightarrow AZ_3, Z_3 \rightarrow XZ_4, Z_4 \rightarrow ZZ_5, Z_5 \rightarrow XA \}$ ★

7.7. példa - Chomsky normálforma 3.feladat

Adjunk meg a $G=(\{S,A,B\},\{x,y\},S,H)$ nyelvtannal ekvivalens Chomsky-féle normálalakú nyelvtant, ahol $H=\{\ S \to ABBAB,\ S \to x,\ A \to BB,\ A \to S,\ A \to B,\ B \to ASA,\ B \to y\ \}$.

Megoldás:
(I.) Mivel a G nyelvtan már normálalakban van, ezért $G_1=G$.
(II.)
$G_2=(\{S,A,B,Z_1,Z_2,Z_3,Z_4\},\{x,y\},S,H_2)$, ahol
$H_2=\{\ S \to AZ_1,\ Z_1 \to BZ_2,\ Z_2 \to BZ_3,\ Z_3 \to AB,\ S \to x,\ A \to BB,\ A \to S,\ A \to B,$
$B \to AZ_4,\ Z_4 \to SA,\ B \to y\ \}$
(III.)
$U(S)=\{A\}, U(B)=\{A\}$.
$G'=(\{S,A,B,Z_1,Z_2,Z_3,Z_4\},\{x,y\},S,H')$, ahol
$H'=\{\ S \to AZ_1,\ A \to AZ_1,\ Z_1 \to BZ_2,\ Z_2 \to BZ_3,\ Z_3 \to AB,\ S \to x,\ A \to x,\ A \to BB,\ B \to AZ_4,$
$A \to AZ_4,\ Z_4 \to SA,\ B \to y,\ A \to y\ \}$ ★

7.3.2. Greibach normálforma

A Chomsky-féle normálforma segítségével kiküszöböltük a levezetésekből az $A \Rightarrow^* A$ alakú formákat, amelyek a levezetés hosszát a végtelenségig megnövelhették kellő odafigyelés hiányában. Egy másik féle probléma merülhet fel $A \to AB$ alakú szabály használata során: ha legbaloldalibb módon szeretnénk a keresett szót levezetni, és pont egy ilyen alakú szabály alkalmazható, akkor ugyanígy alkalmazható lesz a következő mondatformára, majd az azt követőre, és így tovább. Ha a levezetés során mást nem veszünk figyelembe, akkor itt a levezetési fa felépítésével egy végtelen ágba kerültünk így a visszalépéses kereső algoritmusa nem alkalmazható a szóprobléma megoldására.

Balrekurziónak nevezzük, ha egy $G=(N, T, S, H)$ nyelvtanban van olyan $A \in N$, hogy $A \Rightarrow^* Ar$ (valamely $r \in (N \cup T)^+$ esetén). Közvetlen balrekurzióról beszélünk, ha $A \Rightarrow Ar$ vagyis van olyan szabály, hogy $A \to Ar$.

A közvetlen balrekurzió megszüntetésére szolgáló algoritmus lépéseivel analóg módon továbbalakítva a nyelvtant jutunk el a Greibach-féle normálalakig, ami a balrekurzió teljes kiküszöbölésével adódik:

A következő normálforma az amerikai Sheila Greibach nevéhez fűződik.

15. Definíció. Egy környezetfüggetlen nyelvtanról akkor mondjuk, hogy Greibach-féle normálalakú, ha minden szabálya $A \to ar$ alakú, ahol $A \in N$, $a \in T$, $r \in N^*$. ★

Igaz a következő tétel, aminek bizonyítását most nem közöljük, de a konstruktív algoritmust egy példán bemutatjuk.

47. Tétel. Minden környezetfüggetlen nyelvtanhoz van vele ekvivalens Greibach normálformájú nyelvtan.

Az eredmény jelentősége az, hogy a környezetfüggetlen nyelvek is generálhatóak (a reguláris és a lineáris nyelvekhez hasonlóan) oly módon, hogy a levezetés minden lépésében történik terminális bevezetése. Ennek következtében a levezetés lépésszáma pontosan annyi kell hogy legyen, mint a levezetendő szó hossza.

7.8. példa - Greibach normálforma 1.feladat

Adjunk meg a $G(\{A,B,C\},\{0,1\},S,H)$ nyelvtannal ekvivalens Greibach normálformájú nyelvtant, ahol
$H=\{$
$S \to BC,$
$B \to CS,$
$B \to 1,$
$C \to SB,$
$C \to 0\}$

Megoldás:
(I.) A balrekurzió kiiktatása

Állítsunk fel egy sorrendet a nemterminálisok közt. Legyen ez most $S < B < C$!
Alakítsuk át a szabályokat úgy, hogy a sorrendben később következő szabályból ne legyen levezethető olyan szó, melynek első betűje egy sorrendben előrébb lévő nemterminális (és haladjunk a választott sorrendben: tekintsük először az $A \to u$ alakú szabályokat... ($u \in (N \cup T)^+$):
$S \to BC$ maradhat ($S < B$),
$B \to CS$ maradhat ($B < C$),
$B \to 1$ maradhat.

$C \to SB$ helyett $C \to BCB$, mert ($C > S$ miatt, mintha a $C \Rightarrow SB \Rightarrow BCB$ levezetésben már az S -re is alkalmaztunk volna szabályt.)
majd $C \to BCB$ helyett $C \to CSCB$ és $C \to 1CB$ ($C > B$ miatt, mintha a BCB első B -jére is alkalmaztunk volna már levezetési szabályt.)

$C \to 0$ maradhat.

(II.) A közvetlen balrekurzió kiiktatása

$S \to BC,$
$B \to CS,$
$B \to 1,$
$C \to CSCB,$
$C \to 1CB,$
$C \to 0.$

Csak a $C \to CSCB$ szabály tartalmaz közvetlen balrekurziót.
Csoportosítjuk a $C \to u$ ($u \in (N \cup T)^+$) alakú szabályokat az alapján, hogy balrekurzívak, vagy nem. Megjegyezzük, hogy egyik csoport sem lehet üres, ha rekurzió előfordul és a C nem felesleges szimbóluma a nyelvtannak.

Vezessük be a C' új nemterminálist, mely a sorrendben előzze meg a már létezőeket!
A sorrend tehát $C' < S < B < C$ legyen!

A balrekurzív $C \to CSCB$ szabály helyett vegyük fel a következőket:

$C \to 0C',$
$C \to 1CBC',$
(a nem balrekurzív szabályokat vegyük fel úgy is, hogy az új C' szimbólum szerepel a jobboldal végén)
$C' \to SCB,$
$C' \to SCBC'.$
(Az új nemterminálissal a baloldalon vegyük fel a balrekurzív szabály(ok) jobb oldalát a balrekurziót okozó kezdő C nélkül. Vegyük fel ezeket a szabály(oka)t úgy is, hogy a jobboldal végén a C' is szerepel.)

Így a szabályok:

$S \to BC$,
$B \to CS$,
$B \to 1$,
$C \to 1CB$,
$C \to 0$,
$C \to 0C'$,
$C \to 1CBC'$,
$C' \to SCB$,
$C' \to SCBC'$.

(III.) Ha a jobboldal első betűje nemterminális, akkor helyette a belőle levezethető jobboldalakat kell behelyettesíteni:

$C \to 1CB$;
$C \to 0$;
$C \to 0C'$;
$C \to 1CBC'$; megfelelő szabályok

$B \to 1$;
$B \to CS$ helyett $B \to 1CBS$, $B \to 0S$, $B \to 0C'S$, $B \to 1CBC'S$; (vagyis a C helyett az előző négy szabály jobboldalait)

$S \to BC$ helyett $S \to 1C$, $S \to 1CBSC$, $S \to 0SC$, $S \to 0C'SC$, $S \to 1CBC'SC$; (vagyis a B helyett az előző öt szabály jobboldalait)

$C' \to SCB$ helyett $C' \to 1CCB$, $C' \to 0SCCB$, $C' \to 1CBSCCB$, $C' \to 1CBC'SCCB$, $C' \to 0C'SCCB$;
$C' \to SCBC'$ helyett $C' \to 1CCBC'$, $C' \to 0SCCBC'$, $C' \to 1CBSCCBC'$, $C' \to 1CBC'SCCBC'$,
$C' \to 0C'SCCBC'$

Ezzel nyelvtanunk megfelelő formájú lett.
G=({S,B,C,C'},{0,1},S,H'), ahol H' szabályai:{
$C \to 1CB$,
$C \to 0$,
$C \to 0C'$,
$C \to 1CBC'$,
$B \to 1$,
$B \to 1CBS$,
$B \to 0S$,
$B \to 0C'S$,
$B \to 1CBC'S$,
$S \to 1C$,
$S \to 0SC$,
$S \to 1CBSC$,
$S \to 1CBC'SC$,
$S \to 0C'SC$,
$C' \to 1CCB$,
$C' \to 0SCCB$,
$C' \to 1CBSCCB$,
$C' \to 1CBC'SCCB$,
$C' \to 0C'SCCB$,
$C' \to 1CCBC'$,
$C' \to 0SCCBC'$,
$C' \to 1CBSCCBC'$,
$C' \to 1bCBC'SCCBC'$,
$C' \to 0C'SCCBC'$
}
Ha ezek a szabályok a jobboldali első terminális után még tartalmaznának terminálist,
akkor azt már csak a normális alakra hozásnál tanult módon, új nemterminálisok bevezetésével

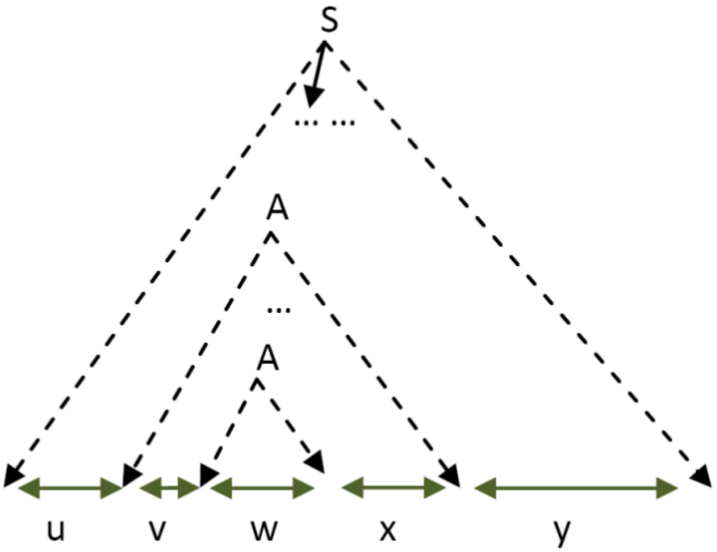
kellene kiiktatni. ★

7.4. A Bar-Hillel lemma

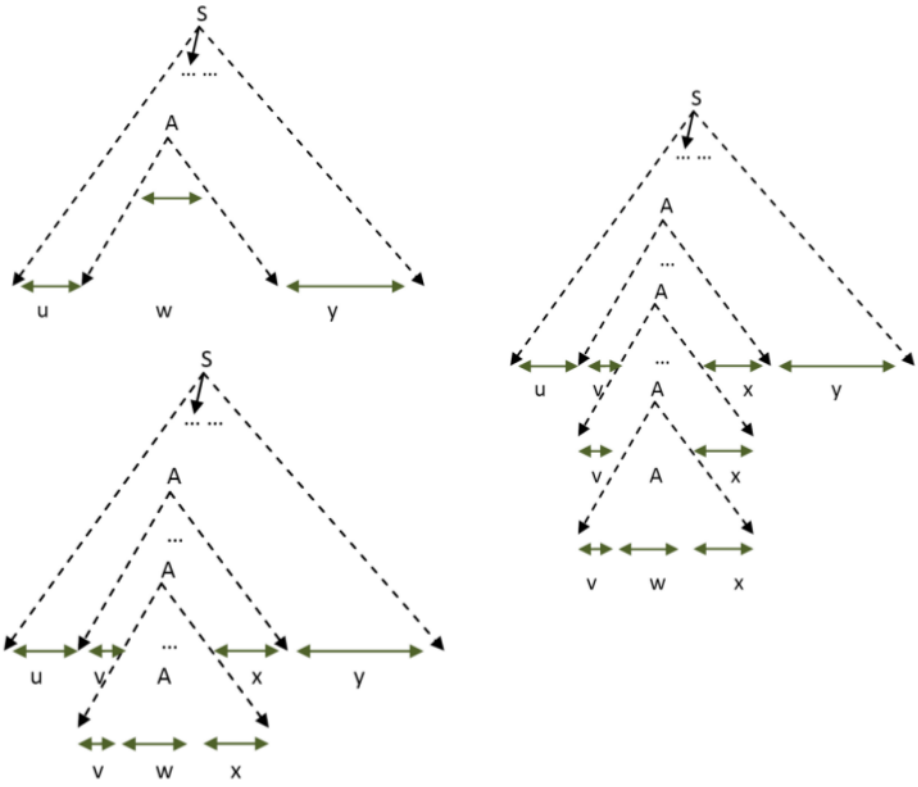
A környezetfüggetlen nyelvek esetén a levezetések fa ábrázolása (a levezetési fa) segítségével fogjuk belátni a következő *iterációs lemmát*:

48. Tétel. (Bar-Hillel lemma) Bármely L környezetfüggetlen nyelvhez létezik olyan n természetes szám, hogy $\forall z \in L$ esetén $|z|>n$ szóra $z=uvwxy$ alakban írható, ahol $|vwx| \leq n$, $vx \neq \lambda$ és $uv^i wx^i y \in L$ minden $i \geq 0$ egész számra.

Bizonyítás. Röviden ez a lemma azt mondja ki, hogy a nyelvben minden elég hosszú szóhoz végtelen sok rokon szerkezetű további szó található. A bizonyításnál elég λ- mentes nyelvtanra szorítkoznunk. Feltételezzük továbbá, hogy a nyelvtanunk Chomsky-féle normálalakban adott. Ha egy $z \in L(G)$ szónak a levezetése olyan fával ábrázolható, amelyben a leghosszabb út k hosszúságú, akkor a Chomsky-féle normálalak miatt $|z| \leq 2^k$. Tegyük fel, hogy az N elemeinek száma j és legyen $n=2^{j+1}$. Ha most $z \in L$ és $|z|>n$, akkor az $S \Rightarrow^* z$ levezetési fájában a leghosszabb útnak j- nél hosszabbnak kell lennie. Vegyük ennek az útnak az utolsó $j+1$ hosszúságú szakaszát. Lesz olyan $A \in N$ változó, amely ezen a szakaszon legalább kétszer előfordul.



Vegyük ennek a változónak két ilyen előfordulását. Ezek közül az elsőhöz (az S- hez közelebb fekvőhöz) tartozó részfa végpontjainak megfelelő szó legyen r, a másik részfa végpontjainak megfelelő szó legyen w. Ezekre nyilván $A \Rightarrow^* r$ és $A \Rightarrow^* w$, továbbá a w részszava r- nek, tehát $r=vwx$ valamely v, $x \in T^*$ szavakra. Emellett nyilván $z=ury$ is teljesül valamilyen u, $y \in T^*$ szavakra. Az A változó szóban forgó előfordulásainak a megválasztása miatt $|r| \leq 2^{j+1}$. Másrészt $S \Rightarrow^* uAy$ és $A \Rightarrow^* vAx$ is fennáll, tehát tetszőleges $i \geq 0$ egész számra $S \Rightarrow^* uv^i wx^i y$. Itt nem lehet v és x mindkettő λ, mert az $A \Rightarrow^* vAx$ levezetés legalább egy lépést tartalmaz, tekintve, hogy az A- nak két különböző előfordulásáról van szó. Akkor pedig ebben a levezetésben az első lépés csak egy $A \to BC$ alakú szabály alkalmazása lehet, s ezért $|vx| \geq 1$, miután a nyelvtanunk λ- mentes. Ezzel befejeztük a lemma bizonyítását. ▮



7.9. példa - A Bar-Hillel lemma alkalmazása

Az $\{a^j b^j c^j | j \geq 1\}$ nyelv nem környezetfüggetlen.

Ha volna olyan környezetfüggetlen nyelvtan, amely generálja ezt a nyelvet, akkor a lemma szerint volna olyan $z=uvwxy$ szó, hogy $vx \neq \lambda$, és minden $i \geq 0$- ra $uv^i wx^i y$ ehhez a nyelvhez tartozik. Az $uv^i wx^i y = a^j b^j c^j$ összefüggést azonban az u, v, w, x, y szavak semmilyen konkrét választása mellett sem lehet végtelen sok i- re és j- re kielégíteni. Ugyanis v és x közül egyik sem tartalmazhat többféle betűt (hiszen ekkor a pumpálás során létrejövő szó alakja nem $a^* b^* c^*$ lenne). Ha viszont csak egyfélét tartalmaznak, akkor az a, b, c közül legalább az egyiknek a kitevője i- től független lesz. ★

A lemma alapján azt is kijelenthetjük, hogy ha egy környezetfüggetlen nyelv végtelen, akkor igaz rá a konstansnövekmény elve, vagyis van olyan konstans $n \in \mathbb{N}$, hogy minden u szavához van a nyelvnek olyan szava, melynek hossza nem több, mint $|u|+n$.

További példákat mutatunk a lemma alkalmazására:

7.10. példa - Bar-Hillel lemma 1. feladat

Bizonyítsa be, hogy az $L = \{a^i b^j c^i d^j \mid i,j \geq 1\}$ nyelv nem környezetfüggetlen.

Megoldás:
Tegyük fel indirekt, hogy megadható olyan n természetes szám, amely a Bar-Hillel lemma szerint minden környezetfüggetlen nyelv esetében létezik!
Tekintsük az $a^{2n}b^{2n}c^{2n}d^{2n}$ szót, ami nyilvánvalóan eleme a nyelvnek és hosszabb n -nél.

Legyen $a^{2n}b^{2n}c^{2n}d^{2n} = uvwxy$, ahol $vx \neq \lambda$, ekkor $uv^2wx^2y \in L$.

Ha v és x közül valamelyik tartalmaz a-t, akkor $|vwx| \leq n$ miatt egyikük sem tartalmazhat c-t.
Így a pumpálás kivezet a nyelvből.

Ha viszont v és x közül valamelyik tartalmaz b-t, akkor $|vwx| \leq n$ miatt egyikük sem tartalmazhat d-t; a pumpálás így is kivezet a nyelvből.

Így tehát sem v, sem x nem tartalmazhat sem a-t, sem b-t, akkor viszont a szó hossza nem változhat, ami ellentmond a lemma állításának. ★

7.11. példa - Bar-Hillel lemma 2. feladat

Bizonyítsa be, hogy a $L=\{wcw \mid w \in \{a,b\}^*\}$ nyelv nem környezetfüggetlen.

Megoldás:
Tegyük fel indirekt, hogy L környezetfüggetlen, ekkor legyen n a lemma által adott konstans.
Tekintsük az $a^{2n}b^{2n}ca^{2n}b^{2n}$ szót. Ekkor $vx \neq \lambda$ miatt a v az első
$a^{2n}b^{2n}$ blokk részszava kell, hogy legyen, amíg az x a c utáni blokk részszava.

Ellenkező esetben vagy a c-k száma változna a pumpálás során, vagy a c előtti, illetve utáni blokkok hossza változna különbözőre.

$|vwx| \leq n$ miatt viszont így a v csak b-ket, az x pedig csak a-kat tartalmazhat, így viszont az iteráció kivezet a nyelvből. ★

7.12. példa - Bar-Hillel lemma 3. feladat

Bizonyítsa be, hogy az $L=\{a^{k^2} \mid k \geq 1\}$ nyelv nem környezetfüggetlen!
Megoldás:
Tegyük fel indirekt, hogy L környezetfüggetlen!
Ekkor igaz rá a Bar-Hillel lemma, tehát létezik olyan n szám,
hogy minden n-től hosszabb L-beli szó felírható a lemma szerinti $uvwxy$ alakban,
hogy $uvvwxxy$, $uvvvwxxxy$ ... is L-beli.
Legyen $m^2 > n$ és $a^{m^2}=uvwxy$ és $|vx| = c > 0$!
Ekkor m^2+c, m^2+2c, m^2+3c... is négyzetszám kell legyen.
Kérdés tehát, hogy van-e olyan szigorúan monoton növekvő számtani sorozat,
melynek minden eleme négyzetszám?
Mivel két szomszédos négyzetszám különbsége minden határon túl nő,
ezért egyszer meghaladja c-t is, tehát nincs.
Vagyis ellentmondásra jutottunk. Az ellentmondás oka csak az indirekt feltétel hamis volta lehet. ★

7.13. példa - Bar-Hillel lemma 4. feladat

Környezetfüggetlen-e az $L=\{0^m1^{2m}| \ m \geq 1\}$ nyelv?
Ha igen, adjon meg hozzá környezetfüggetlen grammatikát,ha nem, bizonyítsa be!

Megoldás:
Ha a v csak 1-eseket, az x csak kétszer annyi 0-t tartalmaz, akkor azok hatványozásával továbbra is L-beli szavakat kapunk, ezért a Bar- Hillel lemmával nem jutunk ellentmondásra.
pl:
v=0,
w=λ,
x=11.
Ekkor a lemma szerinti állandó n = 3.
A nyelv tehát lehet környezetfüggetlen, és meg is tudunk adni egy, a nyelvet generáló nyelvtant:
$G=(\{S\},\{0, 1\}, S, \{S \rightarrow 0S11, S \rightarrow 011\})$. ★

7.14. példa - Bar-Hillel lemma 5. feladat

Környezetfüggetlen-e az $L=\{0^i1^{2i}0^j| \ i,j \geq 1\}$ nyelv?
Ha igen, adjon meg hozzá környezetfüggetlen grammatikát, ha nem, bizonyítsa be!

Megoldás:
Két alapjában eltérő módon is kielégíthető a Bar-Hillel lemma:
- v-t 0^i-ből, x-et 1^{2i}-ből választjuk, úgy, hogy x kétszer hosszabb.
Pl:
v=0,
w=λ,
x=11.
Ekkor n=3.
- v-t és x-et is 0^j-ből válaszjuk.
Pl:
v=0,
w=λ,
x=λ.
Ekkor n=3.
$G=(\{S, A, B\},\{0, 1\}, S,\{S \rightarrow AB, A \rightarrow 0A11, A \rightarrow 011, B \rightarrow 0, B \rightarrow 0B\})$ pedig egy grammatika, amely a keresett nyelvet állítja elő. ★

7.15. példa - Bar-Hillel lemma 6. feladat

Környezetfüggetlen-e az $L=\{0^i1^j0^{2i}|i,j \geq 1\}$ nyelv?

Megoldás:
Ha a két szélén választjuk ki a "pumpálandó" részeket, akkor ezek távolsága bármekkorára nőhet, így nem tudjuk n alatt tartani, ezért a két "pumpálandó" részt válasszuk az 1-esek közül, illetve az egyik legyen λ:
v=1,
w=λ,
x=λ,
n=3.
Vagyis a Bar-Hillel lemmával nem kerültünk ellentmondásba, és ez a nyelv egyébként tényleg környezetfüggetlen:
A $G=(\{S, A\}, \{0, 1\}, S, \{S \rightarrow 0S00, S \rightarrow 0A00, A \rightarrow 1A, A \rightarrow 1\})$
2-es típusú grammatika pontosan ezt a nyelvet generálja.
Először a 0-kat állítjuk elő $S \rightarrow 0S00$ szabály i-1 szeri alkalmazásával,
majd a szó közepét töltjük fel 1-esekkel. ★

7.16. példa - Bar-Hillel lemma 7. feladat

$L=\{a^{p_i}|p_i$ az i-edik prím $i \geq 1\}$
Bizonyítsa be, hogy az L nyelv nem környezetfüggetlen!

Megoldás:
Tegyük fel, hogy L környezetfüggetlen! Ekkor alkalmazható rá a Bar-Hillel lemma, vagyis, ha c olyan prím, hogy nagyobb a lemma által létező n-nél, és az vx hossza m, akkor $c+m$, $c+2m$, ..., $c+cm$ is prím.
Márpedig $c+cm=c(1+m)$ nem lehet prím. Vagyis ellentmondásra jutottunk. ★

7.17. példa - Bar-Hillel lemma 8. feladat

Legyen $L=\{pp^{-1}|p^{-1}$ a p tükörképe, $p \in T\}$, ahol T tetszőleges ábécé.
Környezetfüggetlen-e az L nyelv?

Megoldás:
Tetszőleges L-beli szó esetén, ha a Bar-Hillel lemma szerinti v és x részszavakat szimmetrikusan választjuk, akkor ezek "pumpálásával" továbbra is L-beli szavakat kapunk, tehát az L nyelv "kiállja" a Bar-Hillel lemmát.
Hogy valóban környezetfüggetlen, ahhoz csak azt kell látni, hogy ha a grammatikának csak $S \rightarrow aSa$ ($a \in T$) és $S \rightarrow$ λ típusú szabályai vannak, akkor az pontosan az L nyelvet állítja elő. ★

7.18. példa - Bar-Hillel lemma 9. feladat

$L=\{a^x | x=m^2+3m+1,\ m \geq 1\}$
Bizonyítsa be, hogy az L nyelv nem környezetfüggetlen!

Megoldás:
A bizonyítás azon múlik, hogy két szomszédos ilyen alakú szám távolsága
$[(m+1)^2+3(m+1)+1]-[m^2+3m+1]=2m+4$
minden határon túl nőhet, az *uvwxy, uvvwxxy, uvvvwxxxy* ... hosszai pedig számtani sorozatot alkotnak. ★

7.5. Veremautomaták

Ha a véges automata definíciójában az állapothalmaz végességére vonatkozó követelményt elhagyjuk, akkor (a véges ábécéjű) végtelen automata fogalmához jutunk. Ez a fogalom e formájában túl általánosnak bizonyult, ezért bevezették a végtelen automaták speciális fajtáit.

7.19. példa - Verem működése

Az egyetemi menzán az ételeket tálcán szokás a kiszolgáló pultoktól az asztalokhoz vinni. A tálcák a kiszolgáló pultoknál egymás tetején vannak elhelyezve, s mindig csak a legfelső tálcát lehet elvenni. Ha a legfelső tálca valamiért nem szimpatikus (színe vagy más miatt), ahhoz hogy az alatta levő valamelyik tálcához hozzáférjünk, le kell venni a legfelsőt, s félretenni. Ezt a mozdulatot mindaddig folytatnunk kell, míg a kívánt (valamiért szimpatikus) tálcához nem jutunk. Ezután (ha rendeseknek akarunk látszani) akkor a félretett tálcákat visszapakoljuk (többnyire fordított sorrendben mint ahogy elvettük) úgy, hogy minden visszapakolt tálca fölé helyezzük a következő visszapakolandót. ★

A veremautomatát eredetileg aritmetikai kifejezések számítógéppel történő kiértékelésére vezették be, s történeti érdekessége, hogy egy veremautomatát realizáló szoftver volt az első olyan számítógépes szoftver termék, mely szabadalmi oltalmat kapott az USA-ban. Egyben ez volt a világon az első szoftver szabadalom. (A szabadalmaztatás a hatvanas években történt.)

A verem olyan alapvető adatszerkezet, amely fontos szerepet játszik a programozás során is. A számítógép, anélkül, hogy erre konkrét veremkezelő utasításokat használnánk alkalmazza a verem adatszerkezetet a rekurzív programozási módszerek esetén. Tekintsük a következő C-szerű pszeudo-kódot, amely a jól ismert Hanoi-tornyai probléma megoldására írodott.

7.20. példa - Verem, mint rekurziós eszköz

Adott n páronként különböző méretű korong és három rúd. Kezdetben mind az n korong az első (source) rúdon van. Bármelyik rúdon bármelyik pillanatban a korongok csak méretüknek megfelelő sorrendben lehetnek: adott korong alatt nála kisebb sohasem lehet. A cél hogy az összes n korongot átrakjuk az első rúdról a második (target) rúdra a harmadik (help) rúd alkalmas segítségével, oly módon, hogy minden lépésben egy korongot (a legfelsőt) tudjuk áthelyezni egy rúdról egy másik tetejére az előbb ismertetett feltételt betartva. A megoldás során egyes rudakhoz, mint verem adatszerkezethez tudunk hozzáférni.

```
function Hanoi (n,s,t,h){
 if (n>0){
 Hanoi (n-1,s,h,t);
 movedisk (s,t);
 Hanoi (n-1,h,t,s);}
}
```

Magyarázat: a Hanoi(*n,s,t,h*) hívásával *n* korongot helyezünk át az *s* rúdról a *t* rúdra úgy, hogy közben a *h* rudat segédrúdként használhatjuk.

Másrészt a megoldás során a rekurzív hívások ugynacsak egy verem segítségével hajtódnak végre, mindig a megfelelő helyre visszadva a vezérlést. Ahogy az előző kód kompaktságán is látszik a rekurzió segítségével nagyon tömör forráskóddal tudunk feladatokat megoldani. ★

A veremautomata esetén egy potenciálisan végtelen befogadóképességű veremmemória összes lehetséges tartalma eredményez végtelen sok állapotot. A veremmemóriát úgy képzelhetjük el, mint egy pozíciókra felosztott, egyirányban végtelen szalagot.

Minden pozícióba egy-egy jel írható. A jelek kiolvasása, illetve törlése a bevitelhez képest fordított sorrendben történik (LIFO: Last In First Out adatszerkezet). Más szóval, a verem belsejének tartalmához közvetlenül nem férünk hozzá, hanem mindig csak a verem tetején lévő jelet tudjuk kiolvasni, illetve módosítani (felülírni, vagy törölni) és csak a verem tetejére (a benn lévők fölé) helyezhetünk el újabb jelet.

A verem alján kezdetben csak egy speciális szimbólum van, a kezdő veremszimbólum.

Általában az input szalaggal ellentétben, amit vízszintesen elhelyezkedőnek gondolunk, a vermet legtöbbször függőleges elrendezésűnek képzeljük/rajzoljuk. A formális leírásunkban a hellyel való takarékosság miatt, vízszintesen úgy fogjuk elképzelni, hogy a legelső betű a legfelső; vagyis mintha ez a szalag balra volna végtelen.

A veremautomata a véges bemenő szót egy input szalagon kapja meg, melyről egy olvasófejjel, balról jobbra haladva betűnként tudja leolvasni a bemenő szót. Az is megengedett, hogy az input szalag üres legyen, azaz az üresszót tartalmazza.

A veremautomatához a potenciálisan végtelen veremmemórián és az input szalagon kívül tartozik még egy véges iniciális nemdeterminisztikus kimenő jel nélküli automata, mely az input szalagról beolvasott betű, a verem tetején levő betű, valamint a belső állapota alapján fogja a verem tetejét és belső állapotát megváltoztatni. Az is megengedett, hogy egy-egy ilyen elemi lépés során az input szalagon a soron következő betűt ne vegye figyelembe, és/vagy az átmenet lényegében ne függjön a verem tetején lévő betűtől. Azt az elemi lépést, mikor a veremautomata az input szalagon soron következő betűt nem olvassa be (az olvasó fej helyben marad), λ- lépésnek is szokás hívni. Ezt a véges iniciális nemdeterminisztikus kimenő jel nélküli automatát a fejezet további részében a rövidség kedvéért a vereamutomata véges automatájának (vagy véges vezérlőjének) hívjuk.

A veremautomata működésének kezdetekor a verem szinte üres (csak a kezdő veremszimbólum van a verem alján), az input szalag olvasófeje a szalag első betűjére mutat (vagy ha az input szalag tartalma az üresszó, ezt érzékeli), s a veremautomatához tartozó véges automata pedig a kezdő állapotában van.

A működés veremautomata esetén is diszkrét időskála mentén haladva történik. Ha a veremautomatához tartozó véges automata a működés során a teljes input elolvasásával eljut egy végállapotba, akkor veremautomata megáll. A veremautomata megáll akkor is, ha a hozzá tartozó véges automata egy olyan állapotban van, melyhez nem tartozik egyetlen alkalmas átmenet sem. Az alkalmas átmenet létezése azt jelenti, hogy a verem tetején lévő jel figyelembe vételével vagy anélkül és az input szalagon lévő következő betű (ha van olyan) figyelembe vételével vagy anélkül az adott állapotból van lehetséges átmenet egy másik állapotba a veremautomata véges automatájában. A veremautomatához tartozó véges automata belső állapotát a továbbiakban röviden a veremautomata belső állapotának, vagy a veremautomata állapotának mondjuk. (Szigorúan véve a veremautomata belső állapotát egy pár határozza meg, melynek első eleme a verem tartalma, a második pedig a veremautomatához tartozó véges automata belső állapota, s így tekintve a veremautomata végtelen.)

Ha az input szalagon egy nem üresszó van, s a veremautomata úgy áll meg végállapotban, hogy előzőleg az input szalag utolsó betűjét is beolvasta, akkor azt mondjuk, hogy a veremautomata az input szalagon levő szót elfogadta. Ha a veremautomata úgy áll meg végállapotban, hogy az input szalag tartalma az üresszó, akkor azt mondjuk, hogy a veremautomata az üresszót elfogadta. Akkor mondjuk, hogy a (nemdeterminisztikus működésű) veremautomata egy bemenő szót elfogad, ha

van olyan futása (működésmódja), hogy a bemenő szót elfogadja. A veremautomata által elfogadott bemenő szavak összeségét a veremautomata által elfogadott nyelvnek hívjuk. A veremautomatáknak többféle definíciója is ismert, a következőkben többféle (általánosabb és egyszerűbb modellt is megvizsgálunk). Látni fogjuk, hogy a veremautomaták által elfogadott nyelvek osztálya épp a környezetfüggetlen nyelvek osztálya.

Formálisan, egy *veremautomata* (PushDown Automaton) a következő hetes: $PDA=(Q, T, Z, q_0, Z_0, d, F)$ ahol

Q a belső *állapotok* nem üres és véges *halmaza,* vagy más néven állapot ábécé,

T az *inputábécé, vagy szalagábécé*,

Z a *veremábécé,*

$q_0 \in Q$ a *veremautomata kezdő állapota,*

$Z_0 \in Z$ a *verem kezdőjele* (*"alja"*),

d leképezés a $Q\times(T\cup\{\lambda\})\times Z$- ból a $Q\times Z^*$ véges részhalmazaiba a (nemdeterminisztikus) *átmeneti függvény,*

$F \subseteq Q$ a *veremautomata végállapotainak halmaza.*

Egy veremautomata pillanatnyi konfigurációján értjük azt a (u, q, z) hármast, ahol $u \in T^*$ az input még fel nem dolgozott része, $z \in Z^*$ a veremmemória pillanatnyi tartalma (z első betűje a verem tetején lévő betű), $q \in Q$ pedig a veremautomata pillanatnyi belső állapota. Egy veremautomata minden lépésben a pillanatnyi konfigurációból a d átmeneti függvény definíciója értelmében egy vagy több pillanatnyi konfigurációba mehet át (vagy épp nem mehet át egybe sem). Mint már említettük, az átmenet történhet oly módon, hogy közben a veremautomata nem kéri a bemenő jelsorozat következő jelét (λ- lépés).

A *PDA* veremautomata egy $(au, q, Xz) \in (T^*\times Q\times Z^+)$ konfigurációt *egy lépésben átalakít* a $(u, p, tz) \in (T^*\times Q\times Z^*)$ konfigurációba, ahol $a \in (T\cup\{\lambda\})$, $X \in Z$ pedig a verem tetején levő szimbólum, (jelekben: $(au, q, Xz) \vdash_{PDA} (u, p, tz)$, vagy csak egyszerűen: $(au, q, Xz) \vdash (u, p, tz)$), ha van olyan $a \in T\cup\{\lambda\}$, $p, q \in Q$, valamint $t \in Z^*$, hogy a következő összefüggés fennáll: $(p, t) \in d(q, a, X)$.

Egy veremautomata a $(u, q, z) \in (T^*\times Q\times Z^+)$ konfigurációt (véges lépésben) *átalakít* (átalakíthat) a (v, p, t) konfigurációba - jelekben: $(u, q, z) \vdash^* (v, p, t)$ - ha van olyan véges konfigurációsorozat $P_0, P_1, \ldots, P_n$, melyenk első tagja $P_0=(u, q, z)$, utolsó tagja $P_n=(v, p, t)$, és bármely két egymást követő tagjára $P_i \vdash_{PDA} P_{i+1}$ $(i=0,\ldots,n-1)$ fennáll.

A veremautomata által (végállapottal) elfogadott nyelv:

$L(PDA_f)=\{w \in T^* | (w, q_0, Z_0) \vdash^* (\lambda, p, z)$ valamely $z \in Z^*$ és $p \in F$ esetén $\}$.

Látható, hogy a veremautomata egy lépésben egy jelpáron tud operációt végrehajtani, aholis a jelpár egyik tagja a veremábécé egy jele, a második tagja pedig az input ábécé egy betűje, vagy az üresszó.

Az eddigiek során a veremautomata végállapottal fogadta el a nyelvet, a szakirodalomban ugyanilyen jól ismert az üres veremmel elfogadó automata. Sok esetben, mint pl. a rekurzív programok esetén is, akkor tekintjük a rendszer működését helyesen befejezettnek, ha végül kiürül a verem.

Egy $PDA=(Q, T, Z, q_0, Z_0, d, F)$ veremautomata üres veremmel fogadja el az

$L(PDA_e)=\{w \in T^* | (w, q_0, Z_0) \vdash^* (\lambda, p, \lambda)$ valamely $p \in Q$ esetén $\}$ nyelvet.

Egy üres veremmel elfogadó veremautomata esetén (vagyis ha az $L(PDA_e)$ nyelvre vagyunk kíváncsiak), tulajdonképpen az F végállapotok halmazára nincs is szükség.

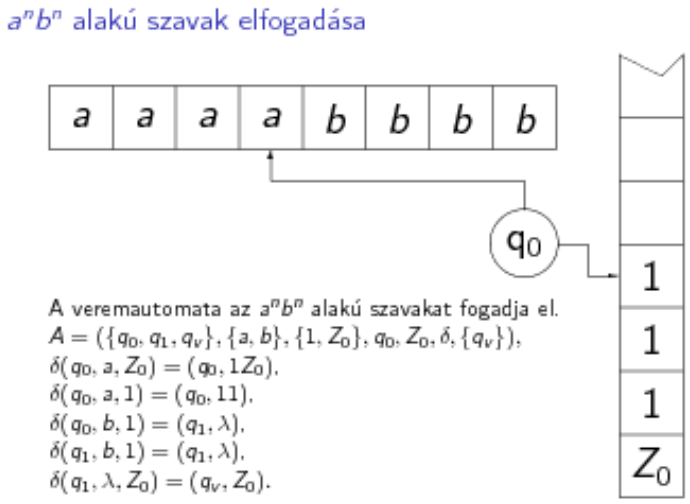
A veremautomatákat is megadhatjuk grafikusan a véges automatákhoz hasonló módon: körökkel jelöljük az állapotokat, külön (egy bemenő nyíllal) megjelölve a kezdőállapotot. Végállapottal

elfogadó automata esetén duplakarikákkal jelöljük a végállapotokat, üres veremmel elfogadó automata esetén pedig nincs végállapot (vagyis a végállapotok hiányával jelezzük, hogy üresveremmel elfogadásról van szó). A d átmenetfüggvény alapján, ha $(p, t) \in d(q, a, X)$, akkor egy irányított él vezet a q állapotból a p állapotba és az élen az $(a, X/t)$ címke ($a \in T \cup \{\lambda\}$, $X \in Z$, $t \in Z^*$) mutatja, hogy az átmenet akkor lehetséges ha a- t olvasunk az input szalagról miközben X van a verem tetején, amit az átmenet során a t- vel helyettesítünk.

Egy tetszőleges veremautomata esetén a végállapottal, illetve az üres veremmel elfogadott nyelvek nagyon különbözhetnek egymástól. Ezért nagyon fontos, hogy, ha adva van egy veremautomata, akkor legyen adott az is, hogy mely módon akarjuk azt nyelv elfogadásra használni.

7.21. példa - Végállapottal elfogadó veremautomata működése

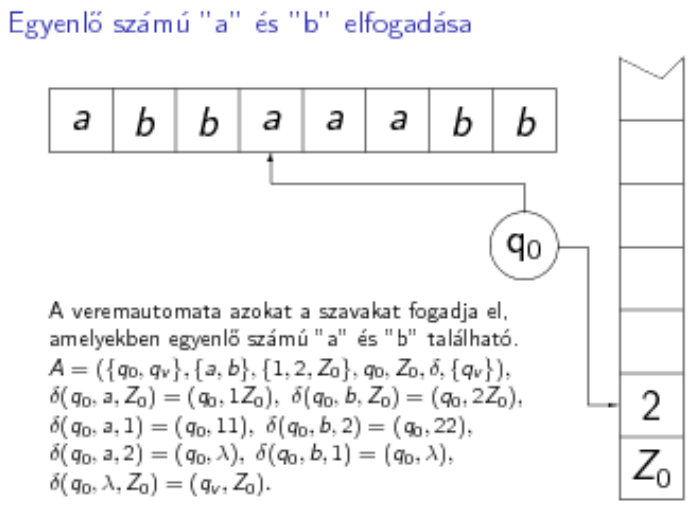
Olyan veremautomata működés közben, mely az $a^n b^n$ alakú szavakat fogadja el:



★

7.22. példa - Az azonos számú *a* és *b* betűből álló szavakat végállapottal elfogadó veremautomata

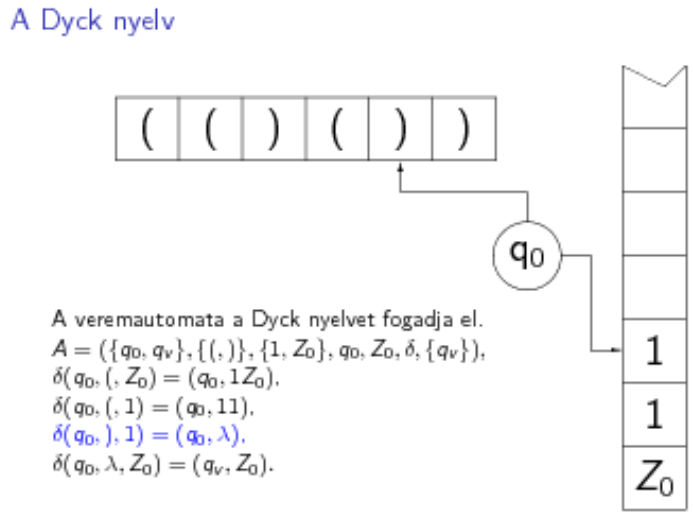
A következő veremautomata olyan szavakat fogad el, melyekben az a- k és b- k száma megegyezik.



★

7.23. példa - A Dyck nyelvet végállapottal elfogadó automata

Az alábbi veremautomata a helyes zárójelek nyelvét (Dyck nyelvet) fogadja el.



★

A következőkben megmutatjuk, hogy a veremautomákkal végállapottal elfogadott nyelvek osztálya megegyezik a veremautomákkal üres veremmel elfogadott nyelvek osztályával. Az állítást két részben, konstruktívan bizonyítjuk.

49. Tétel. Legyen L nyelv olyan amit a PDA_f veremautomata végállapottal elfogad. Ekkor van olyan PDA_e veremautomata amely ugyanezt a nyelvet üres veremmel fogadja el.

Bizonyítás. Legyen $PDA_f=(Q, T, Z, q_0, Z_0, d, F)$. Ekkor megkonstruálunk egy $PDA_e=(Q', T, Z', q'_0, Z'_0, d', \emptyset)$ automatát a következőképpen:

$Q'=Q\cup\{q'_0, q_f\}$, ahol $q'_0, q_f\notin Q$;

$Z'=Z\cup\{Z'_0\}$, ahol $Z'_0\notin Z$;

d'- t pedig a következőképpen származtatjuk d- ből.

Legyen $d'(q'_0, Z'_0)=\{(q_0, Z_0Z'_0)\}$. Továbbá, minden $p, q\in Q$, $a\in T\cup\{\lambda\}$, $X\in Z$, $t\in Z^*$ esetén, legyen $(p, t)\in d'(q, a, X)$ pontosan akkor, ha $(p, t)\in d(q, a, X)$; valamint legyen $(q_f, \lambda)\in d'(q, \lambda, X)$ minden $q\in(F\cup\{q_f\})$ és $X\in Z'$ esetén.

Az így konstruált PDA_e automata első lépése csak az lehet, hogy átmegy a szimulálandó PDA_f kezdőállapotának megfelelő állapotba, miközben a verembe a PDA_f kezdő veremszimbóluma bekerül a PDA_e kezdő veremszimbóluma fölé.

Ezután az automata a PDA_f automata egy futását szimulálja, amíg az egy végállapotba nem jut vagy el nem akad.

Ha a PDA_f egy végállapotba jutna, akkor, és csak ekkor, a PDA_e automata átmehet a q_f állapotba input olvasása nélkül, és kiürítheti a verem tartalmát.

PDA_e pontosan akkor fogad el egy szót, ha a szimulálandó PDA_f végigolvasta a szót és ekkor végállapotba került, vagyis pontosan a kívánt nyelvet fogadja el. ∎

50. Tétel. Legyen L nyelv olyan amit a PDA_e veremautomata üres veremmel elfogad. Ekkor van olyan PDA_f veremautomata amely ugyanezt a nyelvet végállapottal fogadja el.

Bizonyítás. Legyen $PDA_e=(Q, T, Z, q_0, Z_0, d, F)$. Ekkor megkonstruálunk egy $PDA_f=(Q', T, Z', q'_0, Z'_0, d', \{q_f\})$ automatát a következőképpen:

$Q'=Q\cup\{q'_0, q_f\}$, ahol $q'_0, q_f\notin Q$;

$Z'=Z\cup Z'_0$, ahol $Z'_0\notin Z$;

d'- t pedig a következőképpen származtatjuk d- ből.

Legyen $d'(q'_0, Z'_0)=\{(q_0, Z_0Z'_0)\}$. Továbbá, minden $p, q\in Q$, $a\in T\cup\{\lambda\}$, $X\in Z$, $t\in Z^*$ esetén, legyen $(p, t)\in d'(q, a, X)$ pontosan akkor ha $(p, t)\in d(q, a, X)$. Ezen kívül legyen $(q_f, \lambda)\in d'(q, \lambda, Z'_0)$ minden $q\in Q$ esetén.

Az így konstruált PDA_f automata első lépése csak az lehet, hogy átmegy a szimulálandó PDA_e kezdőállapotának megfelelő állapotba, miközben a verembe a PDA_e kezdő veremszimbóluma bekerül a PDA_f kezdő veremszimbóluma fölé.

Ezután az automata a PDA_e automata egy futását szimulálja, amíg annak verme ki nem ürül (ekkor most Z'_0 a veremtartalom) vagy el nem akad.

Ha a szimulálandó verem kiürült, és csak ekkor, a PDA_f átmehet végállapotba.

PDA_f pontosan akkor fogad el egy szót, ha a szimulálandó PDA_e végigolvasta a szót és ekkor kiürült a verme, vagyis pontosan a kívánt nyelvet fogadja el. ∎

Az előző konstrukcióval kicsit többet is beláttunk a tervezettnél: minden a veremautomaták által elfogadodtt L nyelvhez van olyan PDA veremautomata, amely L-et fogadja el végállapottal és üres veremmel is, ráadásul egyszerre végállapottal és üresveremmel fogadja el L minden szavát (és csak ezeket).

3. Következmény. A veremautomatákkal végállapottal elfogadott nyelvek osztálya megegyezik a veremautomákkal üres veremmel elfogadott nyelvek osztályával.

A következőkben a veremautomaták és a környezetfüggetlen nyelvosztály kapcsolatára derítünk fényt.

51. Tétel. Minden környezetfüggetlen L nyelvhez megadható egy PDA veremautomata, amely L-et fogadja el üres veremmel.

Bizonyítás. Legyen adott $G=(N, T, S, H)$ környezetfüggetlen nyelvtan, amely L-et generálja. Ekkor definiáljuk $PDA=(Q, T, Z, q, Z_0, d, \emptyset)$- t a követekezőképpen:

legyen $Q=\{q\}$;

T ábécé megegyezik a G nyelvtannál megadott terminális ábécével, hiszen ugyanazt az L nyelvet szeretnénk elfogadtatni, amit G generál;

$Z=N\cup T$; $Z_0=S$; d-t pedig a H alapján a következőképpen definiáljuk:

legyen $(q, t)\in d(q, \lambda, a)$ pontosan akkor, ha $A\to t\in H$ ezen kívül legyen $(q, \lambda)\in d(q, a, a)$ minden $a\in T$- re.

Könnyen belátható, hogy az automata a w szó elfogadása során annak egy legbalodali levezetését szimulálja. ∎

Az előző konstrukcióban egy állapottal dolgoztunk, tulajdonképpen állapotra így nincs is szükség. Állapotnélküli veremautomatáról beszélünk, ha az eredeti definíciónknak megfelelő automatának csak egy állapota van. Világos hogy ilyen automaták esetén az elfogadás üres veremmel történik. Bevezethetjük a következő rövidítést: $PDA_n=(T, Z, Z_0, d)$- vel jelöljük az üres veremmel elfogadó állapotnélküli veremautomatát (egyszerűen elhagyva a hetesből, illetve a d leképezésből az állapotokat). Az előző tétel bizonyításával, a tételnél erősebb állítást bizonyítottunk be: minden környezetfüggetlen nyelv üres veremmel elfogadtatható állapotnélküli veremautomatával.

Ha az előbbi konstrukciót speciális, pl. Chomsky vagy Greibach normálalakú nyelvtanra végezzük el, akkor megspórolhatóak azok a lépések, amelyek a terminálisokat a verembe helyezik (ha a verem

tetején terminális van, akkor csak olyan lépés következhet, ami ezt kiveszi onnan, miközben a szalagról olvasunk).

Chomsky-féle normálalakú $G=(N, T, S, H)$ nyelvtan esetén a $PDA_n=(T, N, S, d)$ fogadja el a nyelvet ahol $\lambda \in d(a, A)$ pontosan akkor, ha $A \to a \in H$ és $BC \in d(\lambda, A)$ pontosan akkor, ha $A \to BC \in H$.

Greibach normálforma használata esetén $t \in d(a, A)$ pontosan akkor, ha $A \to at \in H$ valamely $A \in N$, $a \in T$, $t \in N^*$ esetén.

Ezek alapján gondolhatnánk azt, hogy ha állapotokat is megengedünk, akkor a veremautomaták a környezetfüggetlen nyelveknél bővebb nyelvosztály elfogadására képesek. Ezzel ellentétben belátható, hogy az állapotok szimulálhatóak állapotnélküli automatával is (a veremábécé megfelelő kitejesztésével). Nem az állapotnélküli és állapotokkal rendelkező automaták ekvivalenciáját fogjuk közvetlenül bizonyítani, hanem tetszőleges *PDA* veremautomatához megkonstruáljuk azt a környezetfüggetlen nyelvtant, amit *PDA* elfogad üres veremmel.

52. Tétel. Minden *PDA* veremautomatához létezik olyan *G* környezetfüggetlen nyelvtan, hogy $L(PDA_e)=L(G)$.

Bizonyítás. Legyen tehát $PDA=(Q, T, Z, q_0, Z_0, d, F)$ adott. Készítsük el a $G=(N, T, S, H)$ nyelvtant a következő módon. Legyen $N=\{[p, X, q] \mid p, q \in Q, X \in Z\} \cup \{S\}$. A szabályok pedig legyenek:

- $S \to [q_0, Z_0, q]$ minden $q \in Q$- ra.

- $[p, X, q] \to a[q_1, Y_1, q_2][q_2, Y_2, q_3]\ldots[q_n, Y_n, q]$ minden lehetséges $q_1, q_2, \ldots, q_n$- re ($n \geq 1$), ha $(q_1, Y_1\ldots Y_n) \in d(p, a, X)$

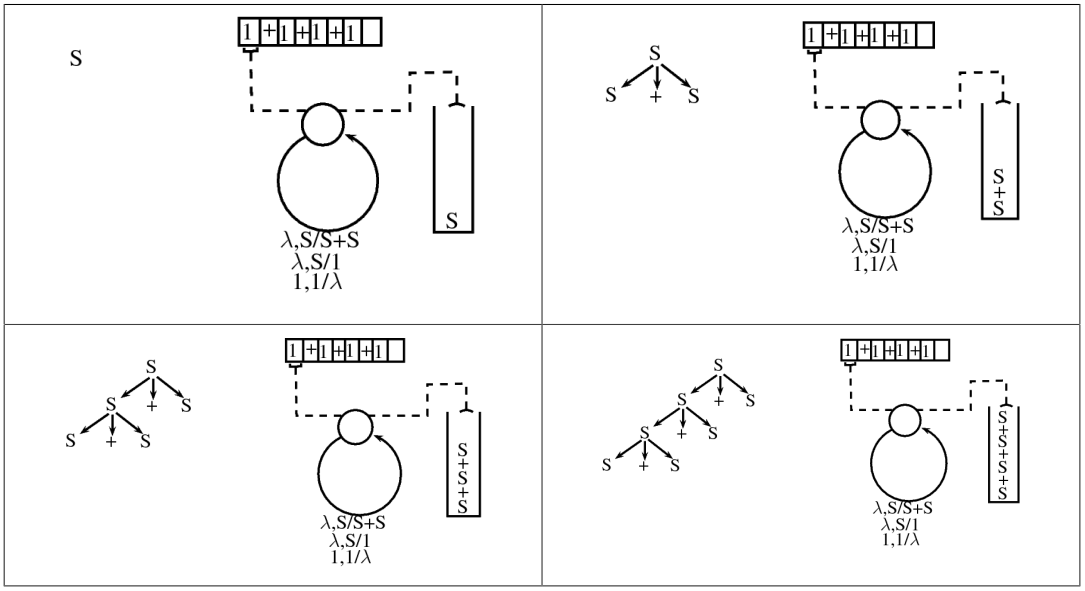
- $[p, X, q] \to a$ pontosan akkor, ha $(q, \lambda) \in d(p, a, X)$.

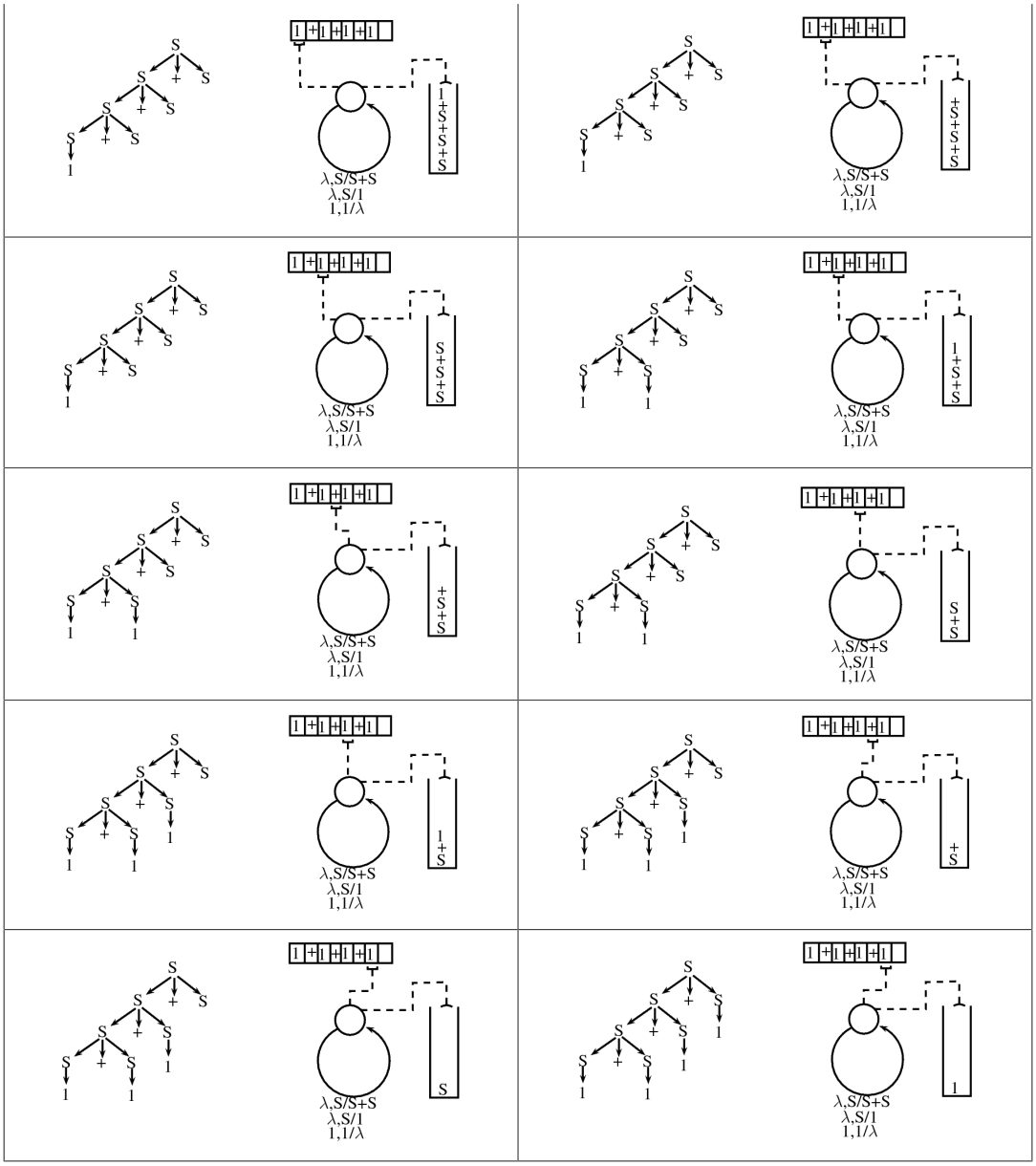
Az első lépésben nemdeterminisztikusan megtippeljük az utolsó állapotot az elfogadáskor. A második lépésbeli szabályokkal nemdeterminisztikusan tippelünk a közbülső $q_2, \ldots, q_n$ állapotokra. Ekkor indukcióval megmutatható, hogy pontosan akkor lesz termináló levezetés *G*-ben egy *w* szóra, ha azt a *PDA* automata elfogadja. ∎

4. Következmény. A környezetfüggetlen nyelvek osztálya egybeesik a veremautomaták által (akár végállapottal, akár üres veremmel, akár egyszerre mindkettővel) elfogadott nyelvek osztályával.

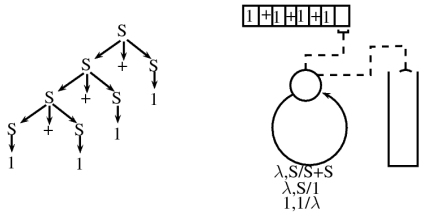
7.24. példa - Állapotnélküli, üres veremmel elfogadó automata működése és a legbaloldalibb levezetés kapcsolata

Legyen $G=(\{S\},\{1,+\},S,\{S \to S+S,\ S \to 1\})$. A következő ábrákon egy, a *G*-vel ekvivalens veremautomata működése, mellette pedig egy *G*-beli legbaloldalibb levezetési fa felépülése látható.





A veremautomata üres veremmel elfogadta az 1+1+1+1 szót:



★

A továbbiakban a veremautomaták néhány speciális változatát vizsgáljuk meg.

Ha a $PDA=(Q, T, Z, q_0, Z_0, d, F)$ olyan, hogy

- minden $p \in Q$, $a \in T$ és $X \in Z$ esetén fennáll, hogy $|d(p, \lambda, X)|+|d(p, a, X)| \geq 1$ (vagyis nem lehet üresszavas átmenet értelmezve olyan állapot és veremszimbólum párra, amire van értelmezve nem üresszavas átmenet is), valamint

- minden $p \in Q$, $a \in T \cup \{\lambda\}$ és $X \in Z$ esetén $|d(p, a, X)| \geq 1$ (vagyis maximum egyféle átmenet van értelmezve),

akkor determinisztikus veremautomatáról beszélünk. Érvényes a következő tétel.

53. Tétel. A determinisztikus veremautomaták által elfogadott nyelvek osztálya valódi részosztálya a nemdeterminisztikus veremautomaták által elfogadott nyelvek osztályának.

A determinisztikus veremautomaták által végállapottal elfogadott nyelvek osztályát determinisztikus környezetfüggetlen nyelveknek hívjuk.

A determinisztikus veremautomatákkal üres veremmel elfogadható nyelvek osztálya valódi része a végállapottal elfogadható nyelvek osztályának. A determinisztikus veremautomatákkal üres veremmel elfogadható nyelvek osztálya egy nagyon szűk nyelvosztály, hiszen az automatának determinisztikus módón ki kell ürítenie a vermet az adott szó elolvasásakor (után) anélkül, hogy tudná van-e még hátra az inputból.

Itt jegyezzük meg, hogy a lineáris (nyelvtanok által generált) nyelvek éppen az *egyszerforduló veremautomaták* által elfogadott nyelvek osztálya. Az egyszerforduló veremautomaták olyan speciális veremautomaták amelyek működésük közben előbb csak "töltik" a vermet, aztán pedig csak "ürítik". Az átmenetfüggvényük olyan, hogy ha volt már törlés a veremből (vagyis olyan átmenet, amikor a legfelső veremszimbólum helyére csak a λ került), akkor már nem tehetnek a verembe újabb szimbólumokat.

A lineáris nyelvtanok levezetési fáit tekintve éppen az történik, hogy a baloldali ágakon levő terminálisok elolvasása közben a főág (a nemterminálisok) bekerülnek a verembe, és ennek megfelelően, őket fordított sorrendben kiszedve olvassuk el a jobb oldali terminálisokat.

A determinisztikus egyfordulós veremautomaták által elfogadott nyelveket tekinthetjük egyféle deterministzikus lineáris nyelveknek (detLin). Ez a nyelvosztály viszont halmazelméletileg nem összemérhető a determinisztikus 2-fejű automaták által definiált 2detLin nyelvosztállyal a tartalmazási relációra nézve.

A palindromák nyelve 2detLin nyelv, de nem detLin (a veremautomata nem tudja mikor értünk a szó közepére, így determinisztikusan nem tud "fordulni"). Viszont az $\{a^nba^n\}\cup\{a^{2n}ca^n\}$ nyelv detLin, de nem 2detLin.

Egy veremautomatát *számláló-automatának* hívunk, ha a verem tartalma csak X^*Z_0 lehet, ekkor a nem törölhető kezdő veremszimbólumon (a veremalja jelen) kívül csupán egyetlen veremszimbólum áll rendelkezésre. Tulajdonképpen a veremnek így csak kétféle szimbólum lehet a tetején: X ha a verem "nemüres", és Z_0, ha a verem "üres". Tulajdonképpen egy számlálónk van amely értéke tetszőleges természetes szám lehet, viszont az automatánk csak azt a két esetet tudja megkülönböztetni, hogy nulla vagy pozitív az érték. Az ilyen számlálóautomatával elfogadott nyelveket számlálónyelveknek hívjuk. Megmutatható, hogy pl. a palindrómák nyelve nem számlálónyelv, így teljesül a következő

54. Tétel. A számlálónyelvek osztálya a környezetfüggetlen nyelvek osztályának valódi része.

7.25. példa - Veremautomaták 1. feladat

Adjunk meg a G 2-es típusú grammatikához egy olyan veremautomatát, amely
a G grammatika által generált nyelvet ismeri fel, majd mutassuk meg, hogy
az 10011 szót felismeri az automata!
$G=(\{S,A,B\},\{0,1\},S,H)$,ahol H szabályai:
$S \to SA,\ S \to AB,$
$A \to BS,\ B \to SA,$
$A \to 1,\ S \to 1,\ B \to 0.$

Megoldás:
Egy olyan megoldást fogunk adni, amely a G nyelvtan által generált nyelv szavait
egyszerre fogadja el végállapottal és üres veremmel.

- A szalagábécé álljon a grammatika terminálisaiból,
- A veremábécé pedig tartalmazza a terminálisokat, a nemterminálisokat és a kezdő veremszimbólumot!
- A belső állapotok halmaza 3 elemből áll, melyből egy kezdő- egy általános- és egy végállapot.
- Az átmenetfüggvény definiálása:
 - Kell egy olyan szabály, hogy a kezdő veremszimbólum fölé írja az S kezdőszimbólumot, és átviszi az automatát általános állapotba!
 - Kellenek olyan szabályok, hogy az automata a szalagról nem olvas, a verem tetejéről olvassa a H-beli szabályok bal oldalát, majd visszaírja fordított sorrendben a jobb oldalát!
 - Kellenek olyan szabályok, hogy a szalagról és a verem tetejéről ugyanazt olvassuk, majd nem írunk semmit a verem tetejére!

$PDA = (\{q_0, q_1, q_2\}, \{0, 1\}, \{S, A, B, 0, 1, z_0\}, q_0, z_0, \delta, \{q_2\})$.
$\delta(q_0, \lambda, z_0) = \{(q_1, Sz_0)\}$,
$\delta(q_1, \lambda, S) = \{(q_1, SA), (q_1, AB), (q_1, 1)\}$,
$\delta(q_1, \lambda, A) = \{(q_1, BS), (q_1, 1)\}$,
$\delta(q_1, \lambda, B) = (q_1, SA), (q_1, 0)\}$,
$\delta(q_1, 1, 1) = \{(q_1, \lambda)\}$,
$\delta(q_1, 0, 0) = \{(q_1, \lambda)\}$,
$\delta(q_1, \lambda, z_0) = \{(q_2, \lambda)\}$.

A Cocke-Younger-Kasami algoritmusos fejezet 7.35. példa - Cocke-Younger-Kasami algoritmus 1. feladat segítségével találhatunk a következő legbaloldalibb levezetésére az 10011 szónak:

$$S \Rightarrow SA \Rightarrow SAA \Rightarrow ABAA \Rightarrow 1BAA \Rightarrow 10AA \Rightarrow 10BSA \Rightarrow 100SA \Rightarrow 1001A \Rightarrow 10011.$$

Ezt levezetést használva mutatjuk meg az automata miyen lépéseken át ismerheti fel az 10011 szót:

Az automata müködése előtt a konfiguráció: (szalagon még hátra lévő inputrész, állapot, verem)

$(10011, q_0, z_0)$

1. lépés: Írjuk a kezdő veremszimbólum felé S-t! Ezt megteheti az automata, mert $(q_1, Sz_0) \in \delta(q_0, \lambda, z_0)$.

$(10011, q_1, Sz_0)$

2. lépés: Az automata a verem tetejét SA-ra cserélheti mivel $(q_1, SA) \in \delta(q_1, \lambda, S)$. (Alkalmaztuk az $S \rightarrow SA$ szabályt.)

$(10011, q_1, SAz_0)$

3. lépés: Az automata a verem tetejét SA-ra cserélheti mivel $(q_1, SA) \in \delta(q_1, \lambda, S)$. (Alkalmazzuk az $S \rightarrow SA$ szabályt.)

$(10011, q_1, SAAz_0)$

4. lépés: A verem tetején az S helyett egy B-t majd egy A-t rakhat, mivel $(q_1, AB) \in \delta(q_1, \lambda, S)$. (Alkalmazzuk az $S \rightarrow AB$ szabályt.)

$(10011, q_1, ABAAz_0)$

5. lépés: A verem tetején lévő A-t 1-esre cserélheti, mert $(q_1, 1) \in \delta(q_1, \lambda, A)$. (Alkalmazzuk az $A \rightarrow 1$ szabályt.)

$(10011, q_1, 1BAAz_0)$

6. lépés: A szalagról olvashat az automata és a veremből törölhet, mert $(q_1, \lambda) \in \delta(q_1, 1, 1)$.

$(0011, q_1, BAAz_0)$

7. lépés: A verem tetején lévő B-t 0-ásra cserélheti az automata, mert $(q_1,\ 0) \in \delta(q_1, \lambda, B)$. (Alkalmazzuk a $B \rightarrow 0$ szabályt.)

$(0011,\ q_1,\ 0AAz_0)$

8. lépés: A szalagról olvashat és a veremből törölhet az automata, mert $(q_1, \lambda) \in \delta(q_1,\ 0, 0)$.

$(011,\ q_1,\ AAz_0)$

9. lépés: A verem tetején az A helyett egy S-t majd egy B-t rakhat, mivel $(q_1,\ BS) \in \delta(q_1,\lambda,A)$. (Alkalmazzuk az $A \rightarrow BS$ szabályt.)

$(011,\ q_1,\ BSAz_0)$

10. lépés: A verem tetején lévő B-t 0-ásra cserélheti az automata, mert $(q_1,\ \ 0) \in \delta(q_1,\ \lambda,\ B)$. (Alkalmazzuk a $B \rightarrow 0$ szabályt.)

$(011,\ q_1,\ 0SAz_0)$

11. lépés: A szalagról olvashat és a veremből törölhet az automata, mert $(q_1, \lambda) \in \delta(q_1,\ 0, 0)$.

$(11,\ q_1,\ SAz_0)$

12. lépés: A verem tetején lévő S-t 1-esre cserélheti az automata, mert $(q_1,\ \ 1) \in \delta(q_1,\ \lambda,\ S)$. (Alkalmazzuk az $S \rightarrow 1$ szabályt.)

$(11,\ q_1,\ 1Az_0)$

13. lépés: A szalagról olvashat és a veremből törölhet az automata, mert $(q_1, \lambda) \in \delta(q_1,\ 1, 1)$.

$(1,\ q_1,\ Az_0)$

14. lépés: A verem tetején lévő A-t 1-esre cserélheti az automata, mert $(q_1,\ \ 1) \in \delta(q_1,\ \lambda,\ A)$. (Alkalmazzuk az $A \rightarrow 1$ szabályt.)

$(1,\ q_1,\ 1z_0)$

15. lépés: A szalagról olvashat és a veremből törölhet az automata, mert $(q_1, \lambda) \in \delta(q_1,\ 1, 1)$.

$(\lambda,\ q_1,\ z_0)$

16. lépés: Az automata átmehet végállapotba és kiürítheti a vermet, mert $(q_2,\lambda) \in \delta(q_1,\ \lambda,\ z_0)$

$(\lambda,\ q_2,\ \lambda)$,

vagyis az automata üres veremmel és végállapottal felismerte az 10011 szót.

II. megoldás:

Mivel a nyelvtanunk (terminális) normálalakú, azaz terminális csak $X \rightarrow x$ szabályokban fordul elő, ahol $X \in N$ és $x \in T$, ezért az átmenetfüggvény a következő lehet:
- Kell egy olyan szabály, hogy a kezdő veremszimbólum fölé írja az S kezdőszimbólumot, és átviszi az automatát általános állapotba!
- Kellenek olyan szabályok, hogy az automata a szalagról nem olvas,
a verem tetejéről olvassa a H-beli szabályok bal oldalát, majd
visszaírja a jobb oldalát, ha a jobboldal nem tartalmaz terminálist!
- Kellenek olyan szabályok, hogy a szalagról x-et és a verem tetejéről X-et olvassuk,
majd nem írunk semmit a verem tetejére, ha $X \rightarrow x \in H$, és $X \in N,\ x \in T$!

- És kell egy olyan szabály, hogy ha általános állapotban olvassa az automata a kezdő veremszimbólumot, akkor menjen át végállapotba!
Ekkor a veremautomata:
$A(\{q_0, q_1, q_2\}, \{0, 1\}, \{S, A, B, 0, 1, z_0\}, q_0, z_0, \delta, \{q_2\})$.
$\delta(q_0, \lambda, z_0)=\{(q_1, Sz_0)\}$,
$\delta(q_1, \lambda, S)=\{(q_1, SA),(q_1, AB)\}$
$\delta(q_1, \lambda, A)=\{(q_1, BS)\}$,
$\delta(q_1, \lambda, B)=\{(q_1, SA)\}$,
$\delta(q_1, 1, A)=\{(q_1, \lambda)\}$,
$\delta(q_1, 1, S)=\{(q_1, \lambda)\}$,
$\delta(q_1, 0, B)=\{(q_1, \lambda)\}$,
$\delta(q_1, \lambda, z_0)=\{(q_2, \lambda)\}$.

A második automata müködése:

$(10011, q_0, z_0)$

1. lépés: Írjuk a kezdő veremszimbólum felé S-t! Ezt megteheti az automata, mert $(q_1, Sz_0) \in \delta(q_0, \lambda, z_0)$.

$(10011, q_1, Sz_0)$

2. lépés: Az automata a verem tetejét kicserélheti SA-ra mivel $(q_1, SA) \in \delta(q_1, \lambda, S)$. (Alkalmaztuk az $S \rightarrow SA$ szabályt.)

$(10011, q_1, SAz_0)$

3. lépés: Az automata a verem tetejét kicserélheti SA-ra mivel $(q_1, SA) \in \delta(q_1, \lambda, S)$. (Alkalmaztuk az $S \rightarrow SA$ szabályt.)

$(10011, q_1, SAAz_0)$

4. lépés: Alkalmazzuk az $S \rightarrow AB$ szabályt! $(q_1, AB) \in \delta(q_1, \lambda, S)$.

$(10011, q_1, ABAAz_0)$

5. lépés: A szalagról olvasunk, a veremből törlünk. $(q_1, \lambda) \in \delta(q_1, 1, A)$.

$(0011, q_1, BAAz_0)$

6. lépés: A szalagról olvasunk, a veremből törlünk. $(q_1, \lambda) \in \delta(q_1, 0, B)$.

$(011, q_1, AAz_0)$

7. lépés: $(q_1, BS) \in \delta(q_1, \lambda, A)$. (Alkalmazzuk az $A \rightarrow BS$ szabályt.)

$(011, q_1, BSAz_0)$

8. lépés: A szalagról olvasunk, a veremből törlünk. $(q_1, \lambda) \in \delta(q_1, 0, B)$.

$(11, q_1, SAz_0)$

9. lépés: A szalagról olvasunk, a veremből törlünk. $(q_1, \lambda) \in \delta(q_1, 1, S)$.

$(1, q_1, Az_0)$

10. lépés: A szalagról olvasunk, a veremből törlünk. $(q_1, \lambda) \in \delta(q_1, 1, A)$.

(λ, q_1, z_0)

11. lépés: $(q_2,\lambda) \in \delta(q_1, \lambda, z_0)$

(λ, q_2, λ),

vagyis az automata üres veremmel és végállapottal felismerte az 10011 szót. ★

7.26. példa - Veremautomaták 2. feladat

Adjunk meg olyan veremautomatát, amely pontosan a következő grammatika által generált nyelv szavait ismeri fel, és mutassuk meg, hogy a *bbcbba* szót is elfogadja!
G=($\{S,A,B,C,D\},\{a,b,c\},S,H$), ahol H szabályai:
$S \to AB$, $A \to CA$, $A \to SS$, $B \to CD$,
$A \to b$, $D \to a$, $C \to c$, $C \to b$.
Megoldás:
$A(\{q_0, q_1, q_2\}, \{a, b, c\}, \{S, A, B, C, D, a, b, c, z_0\}, q_0, z_0, \delta, \{q_2\})$.
$(q_1, Sz_0) \in \delta(q_0, \lambda, z_0)$,
$(q_1, AB) \in \delta(q_1, \lambda, S)$,
$(q_1, CA) \in \delta(q_1, \lambda, A)$,
$(q_1, SS) \in \delta(q_1, \lambda, A)$,
$(q_1, CD) \in \delta(q_1, \lambda, B)$,
$(q_1, b) \in \delta(q_1, \lambda, A)$,
$(q_1, a) \in \delta(q_1, \lambda, D)$,
$(q_1, c) \in \delta(q_1, \lambda, C)$,
$(q_1, b) \in \delta(q_1, \lambda, C)$,
$(q_1, \lambda) \in \delta(q_1, a, a)$,
$(q_1, \lambda) \in \delta(q_1, b, b)$,
$(q_1, \lambda) \in \delta(q_1, c, c)$,
$(q_2, \lambda) \in \delta(q_1, \lambda, z_0)$. Az egyetlen legbaloldalibb levezetés (pl. a Cocke-Younger-Kasami algoritmusos fejezet 7.37. példa - Cocke-Younger-Kasami algoritmus 3. feladat alapján):
$S \Rightarrow AB \Rightarrow CAB \Rightarrow bAB \Rightarrow bCAB \Rightarrow bbAB \Rightarrow bbCAB \Rightarrow bbcAB \Rightarrow bbcbB \Rightarrow bbcbCD \Rightarrow bbcbbD \Rightarrow bbcbba$
A szó felismerésének lépései:
$(bbcbba,q_0,z_0)$

1. lépés $(q_1, Sz_0) \in \delta(q_0, \lambda, z_0)$
$(bbcbba,q_1, Sz_0)$

2. lépés $(q_1, AB) \in \delta(q_1, \lambda, S)$
$(bbcbba,q_1, ABz_0)$

3. lépés $(q_1, CA) \in \delta(q_1, \lambda, A)$
$(bbcbba,q_1, CABz_0)$

4. lépés $(q_1, b) \in \delta(q_1, \lambda, C)$
$(bbcbba,q_1, bABz_0)$

5. lépés $(q_1, \lambda) \in \delta(q_1, b, b)$
$(bcbba,q_1, ABz_0)$

6. lépés $(q_1, CA) \in \delta(q_1, \lambda, A)$
$(bcbba,q_1, CABz_0)$

7. lépés $(q_1, b) \in \delta(q_1, \lambda, C)$
$(bcbba,q_1, bABz_0)$

8. lépés $(q_1, \lambda) \in \delta(q_1, b, b)$
$(cbba,q_1, ABz_0)$

9. lépés $(q_1, CA) \in \delta(q_1, \lambda, A)$
$(cbba,q_1, CABz_0)$

10. lépés $(q_1, c) \in \delta(q_1, \lambda, C)$
$(cbba,q_1, cABz_0)$

11. lépés $(q_1, \lambda) \in \delta(q_1, c, c)$
(bba,q_1, ABz_0)

12. lépés $(q_1, b) \in \delta(q_1, \lambda, A)$
(bba,q_1, bBz_0)

13. lépés $(q_1, \lambda) \in \delta(q_1, b, b)$
(ba,q_1, Bz_0)

14. lépés $(q_1, CD) \in \delta(q_1, \lambda, B)$
(ba,q_1, CDz_0)

15. lépés $(q_1, b) \in \delta(q_1, \lambda, C)$
(ba,q_1, bDz_0)

16. lépés $(q_1, \lambda) \in \delta(q_1, b, b)$
(a,q_1, Dz_0)

17. lépés $(q_1, a) \in \delta(q_1, \lambda, D)$
(a,q_1, az_0)

18. lépés $(q_1, \lambda) \in \delta(q_1, a, a)$
(λ ,q_1, z_0)

19. lépés $(q_2, \lambda) \in \delta(q_1, \lambda, z_0)$
$(\lambda, ,q_2, \lambda)$

Tehát az automata végállapottal és üres veremmel felismerte a szót! ★

7.27. példa - Veremautomaták 3. feladat

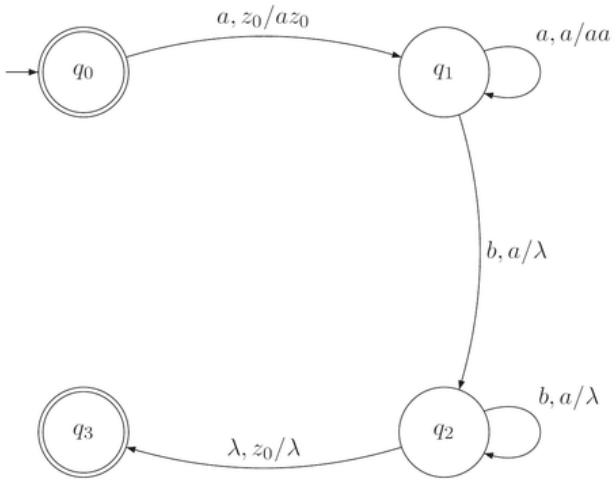
Adjunk meg olyan veremautomatát, mely pontosan a G grammatika által generált nyelv szavait ismeri fel!
G=($\{S, A, B\},\{x, y\}, S, H$), ahol H a következő szabályokból áll:
$S \to AB$, $A \to BSB$, $A \to BB$, $B \to xAy$,
$B \to \lambda$, $B \to x$, $B \to y$.
Megoldás:
$A(\{q_0, q_1, q_2\}, \{x, y\}, \{S, A, B, x, y, z_0\}, q_0, z_0, \delta, \{q_2\})$.
$(q_1, Sz_0) \in \delta(\lambda, q_0, z_0)$,
$(q_1, AB) \in \delta(\lambda, q_1, S)$,
$(q_1, BSB) \in \delta(\lambda, q_1, A)$,
$(q_1, BB) \in \delta(\lambda, q_1, A)$,
$(q_1, xAy) \in \delta(\lambda, q_1, B)$,
$(q_1, \lambda) \in \delta(\lambda, q_1, B)$,
$(q_1, x) \in \delta(\lambda, q_1, B)$,
$(q_1, y) \in \delta(\lambda, q_1, B)$,
$(q_1, \lambda) \in \delta(x, q_1, x)$,
$(q_1, \lambda) \in \delta(y, q_1, y)$,
$(q_2, \lambda) \in \delta(\lambda, q_1, z_0)$. ★

7.28. példa - Veremautomaták 4. feladat

Készítsünk olyan veremautomatát, amely az $L=\{a^n b^n | \ n \geq 0\}$ nyelvet ismeri fel!

Megoldás:
- Mivel az üresszó eleme L-nek, az automata q_0 kezdőállapota egyben egy végállapot is.
- Ha olvasunk egy a betűt, akkor írunk a verembe egy a-t, és átmegyünk a q_1 állapotba.
- Ha olvasunk egy a-t akkor a verem tetejére rakunk még egy a-t, és maradunk ebben az állapotban.
- Ha b-t olvasunk, akkor kiveszünk egy a-t a veremből, majd átmegyünk a q_2 állapotba.
- Ha b-t olvasunk a szalagon és a-t a veremben, akkor maradunk itt, és kiveszünk egy a-t.
- Ha z_0-t olvasunk a veremből és nincs betű a szalagon,

akkor ürítjük a vermet, és átmegyünk q_3 állapotba, mely szintén végállapot.
Az elkészült automatát a következő ábra mutatja:



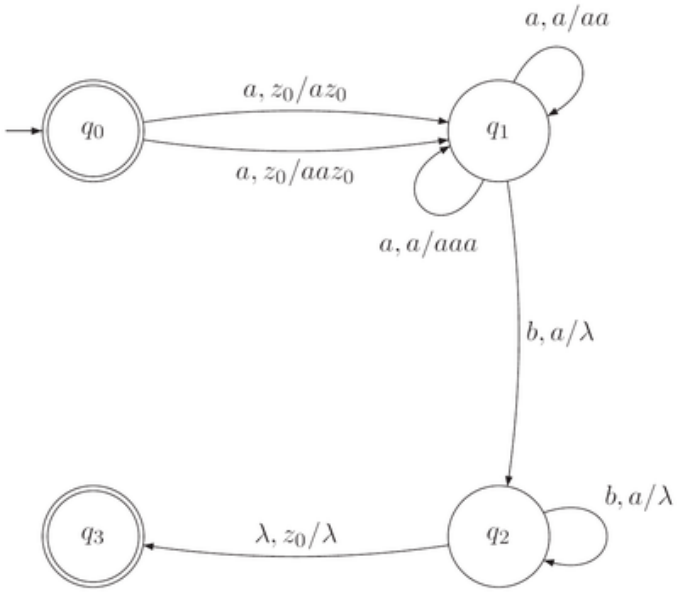
Itt jegyezzük meg, hogy az L nyelv egy számlálónyelv. Ha a fenti megoldásban az utolsó lépésben nem töröljük ki a z_0-t a veremből, úgy megyünk át a q_3 elfogadóállapotba, akkor az elkészített automatánk számláló-automata. ★

7.29. példa - Veremautomaták 5. feladat

Készítsünk olyan veremautomatát, amely az $L=\{a^n b^m | \ 0 \leq n \leq m \leq 2n\}$ nyelvet ismeri fel!

Megoldás:
- Mivel az üresszó eleme L-nek, az automata q_0 kezdőállapota egyben egy végállapot is.
- Ha olvasunk egy a betűt, akkor írunk a verembe egy a-t vagy két a-t, és átmegyünk a q_1 állapotba.
- Ha olvasunk egy a-t akkor a verem tetejére rakunk még egy a-t vagy két a-t, és maradunk ebben az állapotban.
- Ha b-t olvasunk, akkor kiveszünk egy a-t a veremből, majd átmegyünk a q_2 állapotba.
- Ha b-t olvasunk a szalagon és a-t a veremben, akkor maradunk itt, és kiveszünk egy a-t.
- Ha z_0-t olvasunk a veremből és nincs betű a szalagon,

akkor ürítjük a vermet, és átmegyünk q_3 állapotba, mely szintén végállapot. A kész automata:



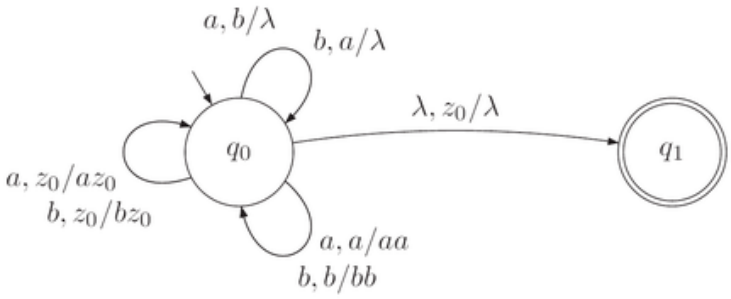
Gyakorló feladatként bizonyítsuk be, hogy az *L* nyelv egy számlálónyelv, vagyis készítsünk egy *L*-et elfogadó számláló-automatát. ★

7.30. példa - Veremautomaták 6. feladat

Készítsünk olyan veremautomatát, amely az L=$\{w \mid |w|_a = |w|_b\}$ nyelvet ismeri fel!
(Azok a szavak az $\{a,b\}$ ábécé felett, melyekben az *a*-k és a *b*-k száma megegyezik.)

Megoldás:
A veremben csak legfeljebb egyféle z_0-tól eltérő szimbólum lehet.
- csak z_0 van: ugyanannyi *a* és *b* betű került elolvasásra eddig.
- *k* db *a* betű van a z_0 felett, akkor eddig k-val több *a*-t olvastunk be.
- *k* db *b* betű van a z_0 felett, akkor eddig k-val több *b*-t olvastunk be.

az automata működése:
- Ha a veremben csak z_0 van, akkor a szalagról olvasott jelet tegyük a verem tetejére.
- Ha a verem tetején lévő és a beolvasott jel egyforma, akkor ezek számát a veremben növeljük 1-gyel.
- Ha a verem tetején lévő és a beolvasott jel eltérő, akkor a veremben lévők számát csökkentjük 1-gyel.
- Ha a veremben csak z_0 van, akkor kiüríthetjük a vermet, és átmehetünk vágállapotba.

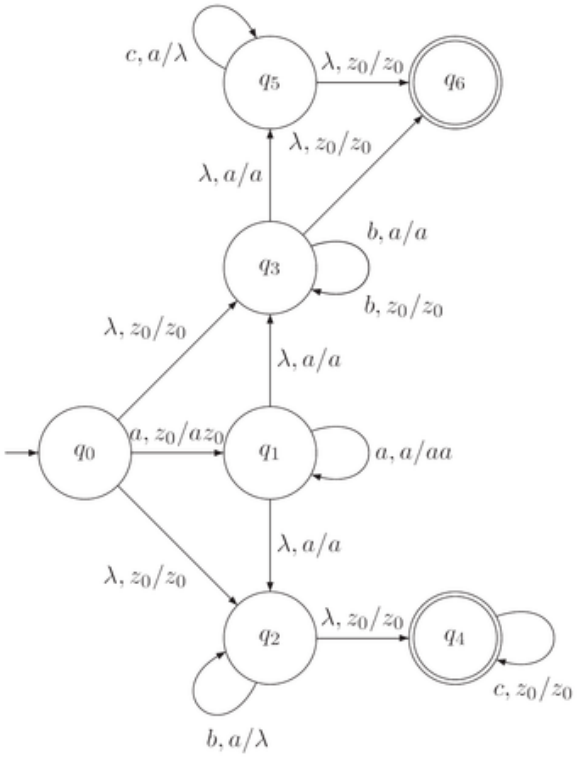


★

7.31. példa - Veremautomaták 7. feladat

Készítsünk olyan veremautomatát, amely az $L=\{a^i b^j c^k \mid i,j,k \geq 0$ és $i=j$ vagy $i=k\}$ nyelvet ismeri fel!

Megoldás:
Fontos, hogy az automatánk nemdeterminisztikus.
- Az egyik "ág" (q_0,q_1,q_2,q_4) azokat a szavakat ismeri fel, mikor az *a*-k és a *b*-k száma egyforma.
- A másik "ág" (q_0,q_1,q_3,q_5,q_6) azokat a szavakat ismeri fel, mikor az *a*-k és a *c*-k száma egyforma.
- Arra kell még figyelni, hogy $a^* b^* c^*$ alakú legyen a szó és lehet $ijk=0$ (*i,j,k* közül bármelyik, vagy több is lehet 0) is.
- Az automata végállapottal ismer fel.

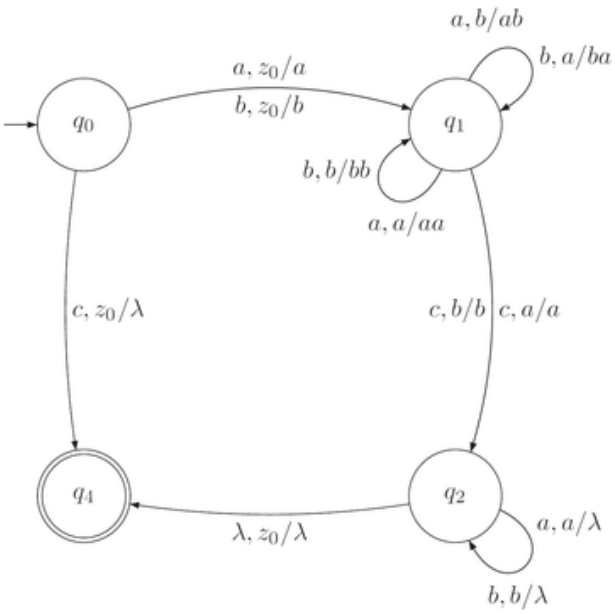


★

7.32. példa - Veremautomaták 8. feladat

Készítsünk olyan veremautomatát, amely $L=\{wcw^{-1} | w \in \{a,b\}^*$, w^{-1} a w fordítottja$\}$ nyelvet ismeri fel,

Megoldás:
- Ha *c*-t olvasunk, akkor rögtön kiürítjük a vermet, és átmegyünk végállapotba.
- Ha *a*-t vagy *b*-t olvasunk, átmegyünk egy másik állapotba (töltő állapotba), és berakjuk a verembe, amit olvasunk.
- Ha töltő állapotban *a*-t vagy *b*-t olvasunk, akkor az megy a verembe.
- Ha *c*-t olvasunk, akkor a vermet változatlanul hagyjuk, és újabb állapotba megyünk (kiürítő állapot).
- Ha kiürítő állapotban a verem tetején lévő betű megegyezik az olvasottal, akkor azt kivesszük a veremből.
- Ha z_0-t elérjük, akkor kiürítjük a vermet, és átmegyünk végállapotba.

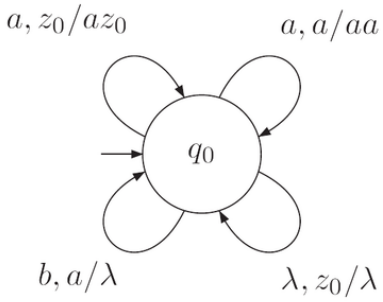


★

7.33. példa - Veremautomaták 9. feladat

Készítsünk olyan üres veremmel elfogadó veremautomatát, amely az $\{a,b\}$ ábécé fölötti helyes zárójelszavak nyelvét, ahol a a nyitó, b pedig a záró zárójel (vagyis a Dyck nyelvet) ismeri fel!

Megoldás:
- Ha a-betűt olvasunk a szalagon, akkor a verembe írjuk.
- Ha b-t olvasunk a szalagon, törlünk egy a-t.
- Ha z_0-t olvasunk, akkor kiüríthetjük a vermet, vagy folytathatjuk a olvasásával is.



★

7.6. Zártsági tulajdonságok

Ebben a fejezetben a környezetfüggetlen nyelvek zártsági tulajdonságait vizsgáljuk meg.

55. Tétel. A környezetfüggetlen nyelvek osztálya zárt a reguláris műveletekre nézve.

Bizonyítás. A bizonyítás során tételezzük fel, hogy L_1=$L(G_1)$, L_2=$L(G_2)$, ahol G_1=(N_1, T, S_1, H_1) és G_2=(N_2, T, S_2, H_2), valamint $N_1 \cap N_2 = \emptyset$. A bizonyítást az egyes műveletekre külön-külön végezzük el.

Egyesítés (*unió*). S legyen egy olyan nemterminális, amely eddig nem szerepelt sem N_1- ben, sem N_2- ben. A

$G_{\cup}=(N_1 \cup N_2 \cup \{S\}, T, S, H_1 \cup H_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\})$

környezetfüggetlen nyelvtan ekkor az $L_1 \cup L_2$ nyelvet generálja. A levezetés során első lépésként alkalmazható $S \rightarrow S_1$, illetve $S \rightarrow S_2$ szabály választásával eldöntjük, hogy melyik nyelvtan alapján szeretnénk generálni a következő szót; ezután pedig, mivel N_1 és N_2 diszjunkt halmazok, csak a megfelelő H_1, illetve H_2- beli szabályok alkalmazhatóak a levezetés során.

Konkatenáció. Legyen ismét $S \notin N_1 \cup N_2$. Ekkor a

$G_{\cdot}=(N_1 \cup N_2 \cup \{S\}, T, S, H_1 \cup H_2 \cup \{S \rightarrow S_1 S_2\})$

környezetfüggetlen nyelvtan az $L_1 L_2$ nyelvet generálja, hiszen az első lépésben generált $S_1 S_2$ mondatformában S_1- ből egy L_1- beli szó, S_2- ből pedig egy L_2- beli szó generálható egymástól függetlenül.

Kleene-csillag (*konkatenáció lezárása*). Legyen $S \notin N_1$. Ekkor a

$G_{*}=(N_1 \cup \{S\}, T, S, H_1 \cup \{S \rightarrow \lambda, S \rightarrow SS_1\})$

környezetfüggetlen nyelvtan az L^* nyelvet generálja. ∎

A fejezet hátralevő részében további halmazműveleteket vizsgálunk.

56. Tétel. A környezetfüggetlen nyelvek osztálya nem zárt metszetképzésre.

Bizonyítás. Legyen $G_1=(\{S_1, A\}, \{a, b, c\}, S_1, \{S_1 \rightarrow aS_1, S_1 \rightarrow A, A \rightarrow bAc, A \rightarrow \lambda\})$. Ekkor könnyen ellenőrizhető, hogy $L(G_1)=\{a^i b^j c^j | i, j \geq 0\}$. Hasonlóan, legyen $G_2=(\{S_2, B\}, \{a, b, c\}, S_2, \{S_2 \rightarrow S_2 c, S_2 \rightarrow B, B \rightarrow aBb, B \rightarrow \lambda\})$, így $L(G_2)=\{a^i b^i c^j | i, j \geq 0\}$. G_1 és G_2 is környezetfüggetlen (sőt lineáris) nyelvtan, viszont $L(G_1) \cap L(G_2)=\{a^i b^i c^i | i \geq 0\}$, ami, ahogy a Bar-Hillel lemma segítségével bizonyítottuk nem környezetfüggetlen. ∎

57. Tétel. A környezetfüggetlen nyelvek halmaza nem zárt a komplementerképzésre.

Bizonyítás vázlat. a $\{wcw | w \in \{a, b\}^*\}$ nyelvről a Bar-Hillel lemmával beláthatjuk hogy nem környezetfüggetlen (lásd 7.11. példa - Bar-Hillel lemma 2. feladat), míg a komplementere környezetfüggetlen, sőt lineáris nyelv. ∎

7.34. példa - A jelölt másolatok nyelvének komplementere - Gyakorló feladat

Készítsünk környezetfüggetlen nyelvtant (vagy 2-fejű automatát) amely a $\{wcw | w \in \{a, b\}^*\}$ nyelv komplementerét fogadja el. ★

Tehát sem a lineáris, sem a környezetfüggetlen nyelvek halmaza nem zárt a metszet és a komplementerképzés műveletére.

7.7. A szóprobléma megoldása - szintaktikai elemzés

Ha egy adott környezetfüggetlen nyelvet generáló Greibach normálformájú nyelvtan alapján készítjük el a felismerő veremautomatát, akkor bármely $w \neq \lambda$ szó esetén az automata elfogadó számítás esetén minden lépésben az inputnak pontosan egy betűjét olvassa el. Ez azt jelenti hogy a w szó elfogadása $|w|$ lépésben történik, vagyis lineáris, sőt valós-időben. Mindezzel az az egy probléma van, hogy a

veremautomata egy nemdeterminisztikus eszköz (legalábbis a veremautomatáknak az az osztálya, amivel a teljes környezetfüggetlen nyelvosztály felismerhető). Tehát nemdeterminisztikusan egy szót végigolvasva, lineáris időben eldönthető a szóprobléma. A fejezet további részében determinisztikus algoritmusokat fogunk bemutatni a probléma megoldására.

A gyakorlati alkalmazásoknál általában nem elég egy formális nyelv felismerési problémáját csupán eldöntési problémának tekinteni, hanem egy $u \in L$ szóról azt is meg kell állapítanunk, hogy az hogyan vezethető le az adott nyelvtanban. A szintaktikai elemzés tehát az adott nyelv szavainak a nyelvtani szerkezetét hivatott feltárni. Programozási nyelvek esetén is lényeges szerepe van a szintaktikai elemzésnek, mert a programok értelmezése a nyelvtani szerkezet ismeretére épül. A következőkben a környezetfüggetlen nyelvek szintaktikai elemzésének lehetőségét vizsgáljuk.

7.7.1. A CYK algoritmus

Tegyük fel, hogy az adott környezetfüggetlen nyelvtan Chomsky-féle normálalakra van hozva. Példaként tekintsük a következő nyelvtant:

$G=(\{S, A, B, C, D\}, \{a, b\}, S, H)$,

ahol a H- beli szabályok a következők:

$S \to AB$,
$S \to CD$,
$S \to CB$,
$S \to SS$,
$A \to BC$,
$C \to DD$,
$B \to SC$,
$C \to b$,
$A \to a$,
$D \to BA$,
$B \to b$.

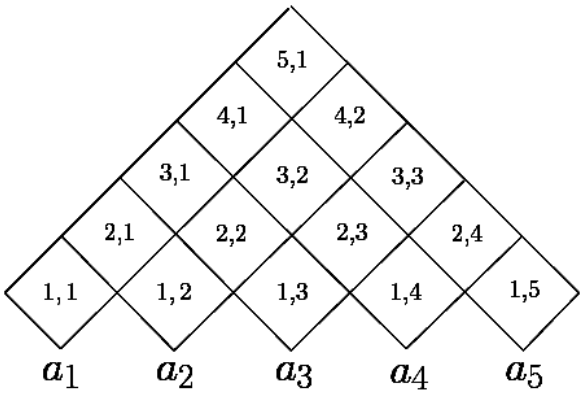
A Cocke-Younger-Kasami-féle (vagy röviden CYK-) (ejtsd: kuk, jánger, kasami) felismerő algoritmus ezeknek a szabályoknak az alapján egy tetszőleges bemenő szóhoz igyekszik megkonstruálni a megfelelő levezetési fát. A Chomsky-féle normálalak miatt ez - a végpontokba befutó élektől eltekintve - egy bináris fa lesz, amelynek a méretét a bemenő szó hosszúsága fogja meghatározni. Ha például a bemenő szó mindössze négybetűs, akkor az összes lehetséges levezetési fának a mélysége maximum 4.

Egy n hosszúságú bemenő szó esetében a levezetési fa összes lehetséges csúcsai egy olyan háromszög mátrixba rendezhetők, amelynek a legalsó sora n- elemű, és a többi sora eggyel kevesebb elemű, mint a közvetlen alatta levő sor. Egy adott fa természetesen nem használhatja fel az összes mezőjét ennek a mátrixnak, csak az n=2 esetben.

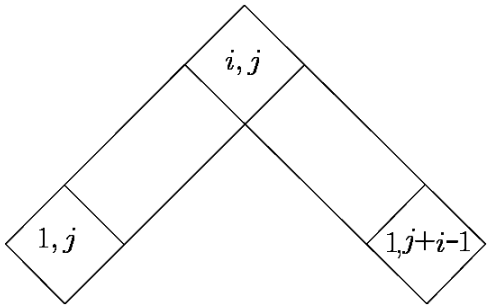
A felismerési algoritmus ennek a háromszög mátrixnak az elemeit határozza meg úgy, hogy minden szögponthoz beírja azokat a nemterminális jeleket, amelyekből a kérdéses szögpont alatti részháromszög végpontjainak megfelelő bemenő szórészlet levezethető. Ha a mátrix kitöltését befejeztük, akkor már csak azt kell megnéznünk, hogy a háromszög legfelső csúcsához be van-e írva az S. Ha igen, akkor az adott bemenő szó levezethető az S- ből, tehát hozzátartozik az adott nyelvtan által generált nyelvhez. Ellenkező esetben viszont biztos, hogy nem tartozik hozzá, mert a kitöltés során minden lehetőséget ki fogunk meríteni.

A szóban forgó háromszög mátrixot felismerési mátrixnak nevezzük, és minden elemének tulajdonképpen egy rekeszt feleltetünk meg, ahová a változókat beírjuk. (Egy-egy rekeszbe a nemterminálisok egy-egy halmaza fog kerülni, mégpedig pontosan azok, amelyekből a mező alatti háromszögnek megfelelő terminális szó levezethető.) Ezeket a rekeszeket a következő ábrán megadott módon indexeljük. (Az ábrán a felismerési mátrix indexelését n=5 esetre láthatjuk.) A kitöltést az

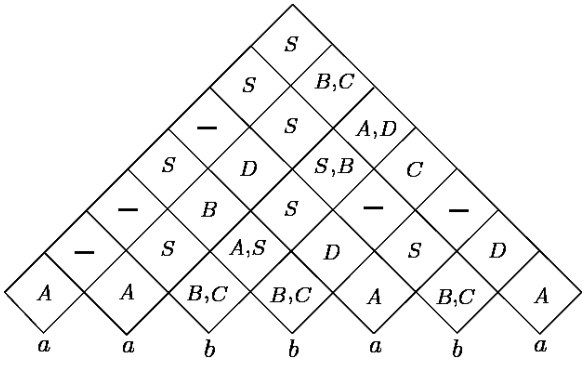
alsó (= első) sornál kezdve soronként balról-jobbra végezzük, a sorokat pedig alulról felfelé történő sorrendben töltjük ki.



Legyen a megvizsgálandó bemenő szó $w=a_1a_2\ldots a_n$. Az első sor i- edik rekeszébe beírjuk az B-t, ha szerepel egy $B \rightarrow a_i$ szabály a nyelvtanban. (Ugyanabba a rekeszbe általában több változót is beírhatunk.) A mátrix további sorainak a kitöltésekor, minden egyes rekesz tartalmának a meghatározása az adott rekesz alatti részháromszög befogói mentén található rekeszek tartalmának a felhasználásával történik. Pontosabban az (i, j) rekesz kitöltéséhez az $(1, j)$ rekeszben található változókat egyenként párosítjuk az $(i\text{-}1, j\text{+}1)$ rekeszben találhatókkal, majd ugyanezt tesszük a $(2, j)$ és $(i\text{-}2, j\text{+}2)$, …, és végül az $(i\text{-}1, j)$ és $(1, j\text{+}i\text{-}1)$ rekesz-párokkal. Ha egy ilyen párosítás alkalmával olyan párt találunk, amely szerepel egy szabály jobb oldalán (a megfelelő sorrendben), akkor a szabály bal oldalán álló változót felvesszük az (i, j) rekeszbe. Formálisan: A- t akkor írjuk be az (i, j) rekeszbe, ha van olyan $k\in\{1, \ldots, i\text{-}1\}$, hogy B szerepel a (k, j) rekeszeben miközben C szerepel a $(i\text{-}k, j\text{+}k)$ rekeszben, és $A \rightarrow BC\in H$. (A feldolgozás menetét a következő ábra szemlélteti.)



A példánkban az *aabbaba* bemenő szóhoz ily módon megkonstruált felismerési mátrixot az alábbi ábra szemlélteti.



A CYK algoritmus helyességének bizonyítása: Indukcióval belátjuk, hogy bármely (i, j) rekeszre egy A nemterminális pontosan akkor eleme a rekesznek, ha belőle levezethető az input $a_j\ldots a_{i+j-1}$ részszava.

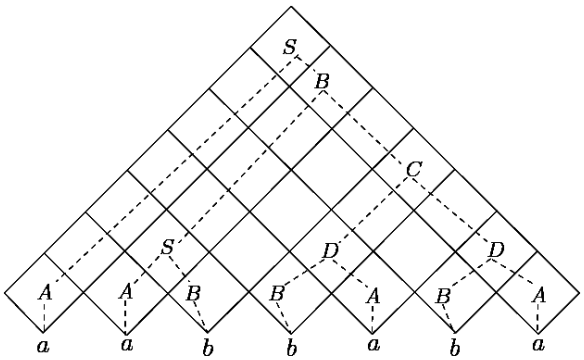
Az indukció alapját az $(1, j)$ rekeszek jelentik. Állítás: A pontosan akkor szerepel $(1, j)$ rekeszben ha $A\Rightarrow^* a_j$. Az algoritmus szerint akkor került az A az $(1, j)$ rekeszbe, ha a Chomsky-normálformájú

nyelvtanban szerepel az $A \rightarrow a_j$ alakú szabály. Egy Chomsky normálformájú nyelvtan esetén egy egybetűs szót pontosan akkor lehet levezetni egy adott nemterminálisból, ha azt egy lépésben, közvetlenül megtehetjük egy szabály segítségével. Ily módon az $(1, j)$ rekeszekre beláttuk állításunkat.

Tekintsük most az (i, j) rekeszt valamely $i>1$ értékre. Tegyük fel most, indukciós feltevésként, hogy minden (k, j) $(k<i)$ rekeszre igaz az állítás. Ha A szerepel az (i, j) rekeszben, akkor az algortimus működése miatt, csak úgy kerülhetett oda, ha van olyan $k \in \{1, \ldots, i\text{-}1\}$, hogy B szerepel a (k, j) rekeszben és C szerepel a $(i\text{-}k, j\text{+}k)$ rekeszben, valamint $A \rightarrow BC \in H$. Mivel $k<i$ és $i\text{-}k<i$ az indukciós feltevésünk miatt $B \Rightarrow^* a_j \ldots a_{j+k-1}$ és $C \Rightarrow^* a_{j+k} \ldots a_{i-k+j+k-1}$, vagyis $C \Rightarrow^* a_{j+k} \ldots a_{i+j-1}$. De hiszen éppen akkor vezethető le egy Chomsky normálalakú nyelvtanban egy egy betűnél hosszabb szó egy A nemterminálisból, ha van $A \rightarrow BC$ szabály, és a szó első része levezethető a B, a második része pedig a C nemterminálisból. Vagyis $A \Rightarrow BC \Rightarrow^* a_j \ldots a_{j+k-1} a_{j+k} \ldots a_{i-k+j+k-1}$ valamilyen k értékre, pontosan akkor ha A szerepel az (i, j) rekeszben. Egy nyelvtan által generált nyelvben pontosan azok a szavak szerepelnek amelyek a nyelvtan S mondatszimbólumából levezethetőek. Éppen ennek tesztelése (vagyis a legfelső rekeszben szerepel-e az S) adja meg a választ arra kérdésre, hogy a keresett u szó benne van-e a generált nyelvben. Az állítást bebizonyítottuk.

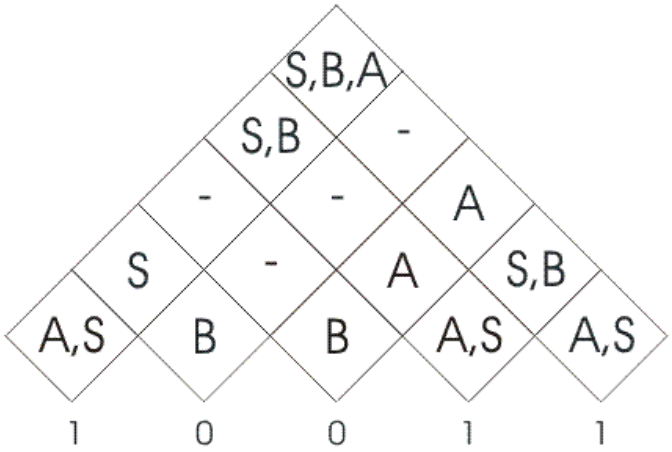
Mivel szintaktikai elemzésről beszélünk, nézzük meg, hogyan is tudunk a felimerési mátrix alapján egy levezetési fát előállítani. A CYK algoritmus alulról felfelé elemzést valósít meg, és az összes lehetséges levezetési fát/részfát tartalmazza. Módosítsuk egy kicsit az algoritmust: amikor egy (i, j) rekeszbe beírunk egy A nemterminális szimbólumot, akkor írjuk be az indexébe, hogy miért is írtuk oda (hogyan került be a mátrixba): A helyett írjunk $A_{B(k,\ j)C(i\text{-}k,\ j+k)}$- t ha a (k, j) rekeszben szereplő B és $(i\text{-}k, j\text{+}k)$ rekeszben szereplő C, valamint az $A \rightarrow BC$ szabály alapján írtuk be az A- t. A módosított algoritmus alapján egyszerű egy leveztési fa előállítása: nézzük meg az $(n, 1)$ (vagyis a legfelső) rekeszben szereplő S indexét, ebből kiolvasható, hogy mely $S \rightarrow AB$ alakú szabály alkalmazásakor került ide az S, márészt az is, hogy mely rekeszekben szereplenek az adott nemterminálisok, ezek indexeinek segítségével megállapíthatjuk ezek milyen szabály segítségével kerültek ide, és így tovább, amíg eljutunk az $(1, j)$ rekeszekig, ahova az $A \rightarrow a_j$ szabályok alapján írtunk be a megfelelő nemterminálisokat.

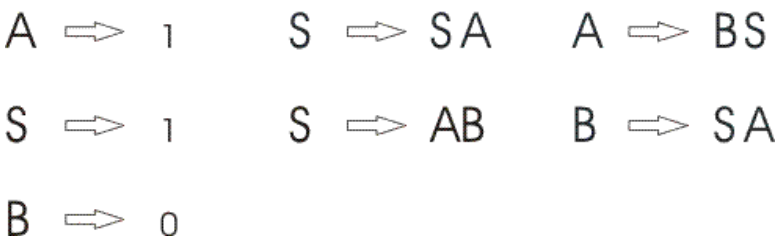
Az alábbi ábrán berajzoltunk egy levezetésfát. A levezetésben nem szereplő nemterminálisokat a jobb áttekinthetőség érdekében elhagytuk.



7.35. példa - Cocke-Younger-Kasami algoritmus 1. feladat

Tekintsük a következő grammatikát!
$G=(\{S,A,B\},\{0,1\},S,H)$,ahol H szabályai:
$\{S \rightarrow SA,\ S \rightarrow AB,\ A \rightarrow BS,\ B \rightarrow SA,\ A \rightarrow 1,\ S \rightarrow 1,\ B \rightarrow 0\}$
Bizonyítsuk be, hogy az 10011 szó benne van a grammatika
által generált nyelvben, majd adjuk meg az összes legbaloldalibb levezetéset!





Megoldás: Jelölje $V_{i,j}$ az (i,j) rekeszt.
1. levezetés: $S \Rightarrow SA \Rightarrow SAA \Rightarrow ABAA \Rightarrow 1BAA \Rightarrow 10AA \Rightarrow 10BSA \Rightarrow 100SA \Rightarrow 1001A \Rightarrow 10011$
Nézzük, hogyan kerültek a levezetésbe az új nemterminálisok!

$S \in V_{5,1}$, mert $S \in V_{4,1}$ és $A \in V_{5,5}$, ahol $S \to SA$-t használtuk. Kövessük a baloldali nemterminálist!
$S \in V_{4,1}$, mert $S \in V_{2,1}$ és $A \in V_{4,3}$, ahol $S \to SA$-t használtuk ismét. Most ismét a balodali nemterminálist követjük
$S \in V_{2,1}$, mert $A \in V_{1,1}$ és $B \in V_{2,2}$, ahol $S \to AB$-t használtuk.
$A \in V_{1,1}$, mert $A \to 1 \in H$
$B \in V_{2,2}$, mert $B \to 0 \in H$
$A \in V_{4,3}$, mert $B \in V_{3,3}$ és $S \in V_{4,4}$, ahol $A \to BS$-t használtuk.
$B \in V_{3,3}$, mert $B \to 0 \in H$
$S \in V_{4,4}$, mert $S \to 1 \in H$
$A \in V_{5,5}$, mert $A \to 1 \in H$

2. levezetés: $S \Rightarrow SA \Rightarrow ABA \Rightarrow 1BA \Rightarrow 10A \Rightarrow 10BS \Rightarrow 100S \Rightarrow 100SA \Rightarrow 1001A \Rightarrow 10011$

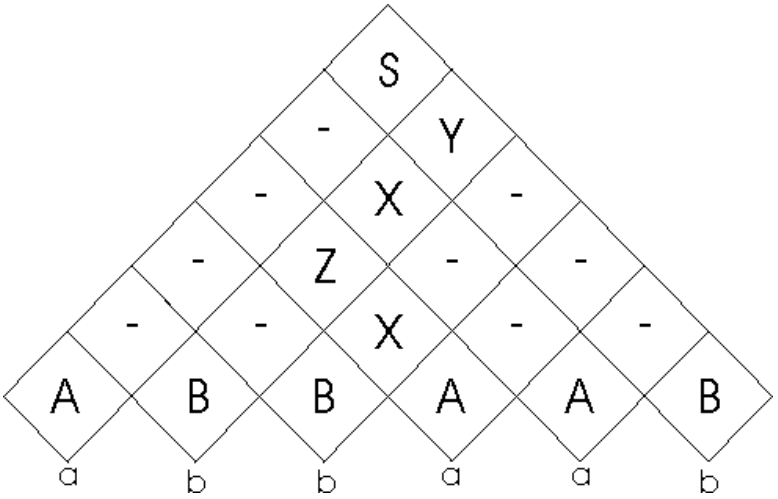
$S \in V_{5,1}$, mert $S \in V_{2,1}$ és $A \in V_{5,3}$, ahol $S \to SA$-t használtuk. Kövessük a baloldali nemterminálist!
$S \in V_{2,1}$, mert $A \in V_{1,1}$ és $B \in V_{2,2}$, ahol $S \to AB$-t használtuk. Kövessük a baloldali nemterminálist!
$A \in V_{1,1}$, mert $A \to 1 \in H$
$B \in V_{2,2}$, mert $B \to 0 \in H$
$A \in V_{5,3}$, mert $B \in V_{3,3}$ és $S \in V_{5,4}$, ahol $A \to BS$-t használtuk. Kövessük ismét a baloldali nemterminálist!
$B \in V_{3,3}$, mert $B \to 0 \in H$
$S \in V_{5,4}$, mert $S \in V_{4,4}$ és $A \in V_{5,5}$, ahol $S \to SA$-t használtuk. Kövessük a baloldali nemterminálist!
$S \in V_{4,4}$, mert $S \to 1 \in H$
$A \in V_{5,5}$, mert $A \to 1 \in H$

★

7.36. példa - Cocke-Younger-Kasami algoritmus 2. feladat

Tekintsük a G=($\{S,A,B,X,Y,Z\},\{a,b\},S,H$) grammatikát, ahol H szabályai:
$\{S \to AY,\ Y \to XB,\ X \to BA,\ X \to ZA,\ Z \to BX,$
$A \to a,\ B \to b\}$!
Benne van-e a G nyelvtan által generált nyelvben az *abbaab* szó?

Megoldás:

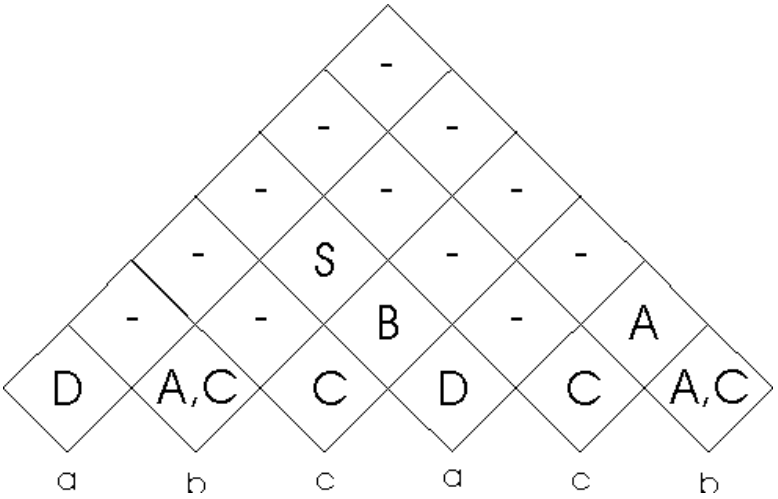


Mivel $S \in V_{6,1}$, ezért a abbaab szó benne van a G grammatika által generált nyelvben. ★

7.37. példa - Cocke-Younger-Kasami algoritmus 3. feladat

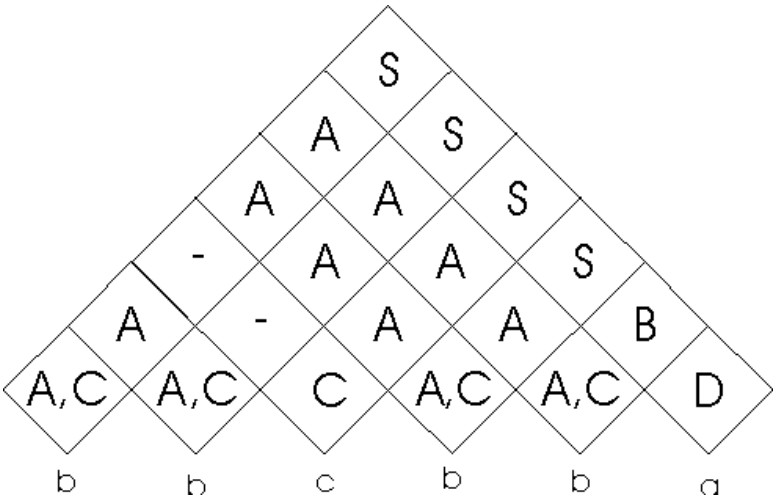
Tekintsük a G=($\{S,A,B,C,D\},\{a,b,c\},S,H$) grammatikát, ahol H szabályai:
$\{S \to AB,\ A \to CA,\ A \to SS,\ B \to CD,$
$A \to b,\ D \to a,\ C \to c,\ C \to b\}$.
Döntsük el, benne van-e a grammatika által generált nyelvben az *abcacb* és a *bbcbba* szó, majd ha lehet adjuk meg az összes legbaloldalibb levezetését!

Megoldás:



Mivel $S \notin V_{6,1}$, ezért az *abcacb* szó
nincs benne az adott grammatika által generált nyelvben.

Nézzük most a másik keresett szót:

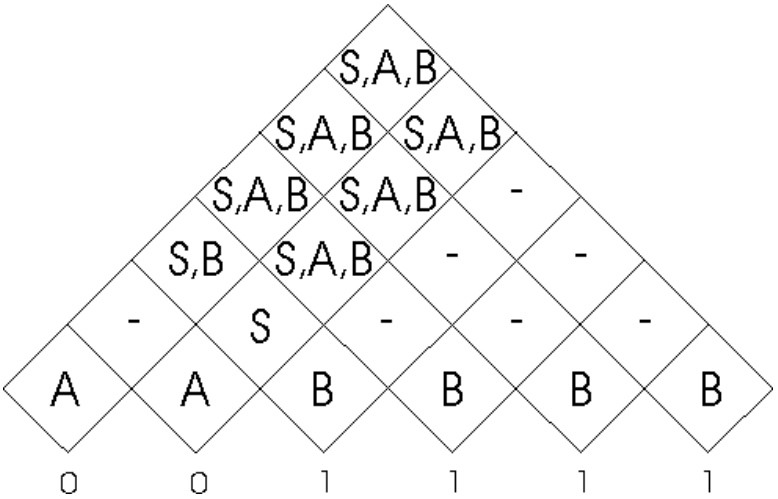


Mivel $S \in V_{6,1}$, ezért a *bbcbba* szó benne van az adott grammatika által generált nyelvben.
Az egyetlen legbaloldalibb levezetés pedig:
$S \Rightarrow AB \Rightarrow CAB \Rightarrow bAB \Rightarrow bCAB \Rightarrow bbAB \Rightarrow bbCAB \Rightarrow bbcAB \Rightarrow bbcbB \Rightarrow bbcbCD \Rightarrow$
$bbcbbD \Rightarrow bbcbba$ ★

7.38. példa - Cocke-Younger-Kasami algoritmus 4. feladat

Tekintsük a $G=(\{S,A,B\},\{0,1\},S,H)$ grammatikát ,ahol H szabályai:
$\{S \to AS,\ S \to SB,\ S \to AB,\ B \to SB,\ A \to SB,\ B \to AS,$
$A \to 0, B \to 1\}$.
Döntsük el CYK-algoritmus segítségével, hogy levezethető-e a nyelvtanban a 001111 szó!

Megoldás:



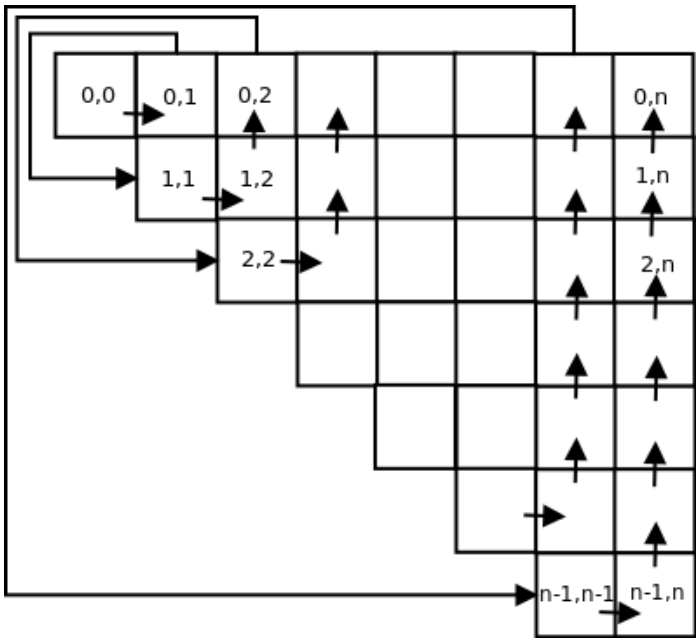
Tehát mivel $S \in V_{6,1}$, ezért a 001111 szó benne van az adott grammatika által generált nyelvben. ★

Az itt ismertetett algoritmus Chomsky-féle normálalakú nyelvtan esetén működött, a következő részben olyan algoritmust ismertetünk, amely bármilyen környezetfüggetlen nyelvtan esetén használható.

7.7.2. Az Earley-féle algoritmus

Ennél az algoritmusnál (az előző alfejezetben ismertetett algoritmushoz hasonlóan) az adott bemenő szóhoz felismerési mátrixot készítünk. Itt azonban a felismerési mátrix elemei nemterminálisok helyett úgynevezett pontozott szabályokat fognak tartalmazni. Legyen mondjuk $A \to pr$, ahol p, $r \in (N \cup T)^*$, egy tetszőleges szabálya az adott környezetfüggetlen nyelvtannak. Ekkor az $A \to p.r$ egy, az eredeti szabályhoz tartozó pontozott szabály lesz, ahol a pont lényegében azt jelzi, hogy az adott szabály tekintetében balról jobbra haladva meddig jutottunk el az elemzésben.

Eszerint ha az elemzéssel a bemenő szó j- edik betűjéig jutottunk el, miközben megállapítottuk, hogy léteznek olyan p, q, r, $t \in (N \cup T)^*$ szavak, melyekre $S \Rightarrow^* pAq$, $p \Rightarrow^* a_1 \ldots a_i$, $r \Rightarrow^* a_{i+1} \ldots a_j$ és $A \to rt \in H$. Ezt a körülményt fejezzük ki azzal, hogy az $A \to r.t$ pontozott szabályt beírjuk a felismerési mátrix i- edik sorának j- edik elemébe. A felismerési mátrixot (mely ismét egy háromszög mátrix) most az ábrán megadott módon indexeljük.



A felismerési mátrix most annyi sort és eggyel több oszlopot fog tartalmazni, mint ahány betűből a bemenő szó áll. A bemenő szó akkor és csak akkor tartozik a nyelvhez, ha a 0- dik sor n- edik elemében állni fog egy $S \to t.$ alakú pontozott szabály ($t \in (N \cup T)^*$). Az előbbiek értelmében ugyanis ez azt jelenti, hogy $S \to t \in H$, és $t \Rightarrow^* a_1 \ldots a_n$.

Lássuk most az algoritmus pontos leírását. Ha a nyelvtan nem λ- mentes, akkor az Üresszó-lemmanál leírt módón készítsük el azon nemterminálisok U halmazát, amelyekből az üresszó levezethető. Az üresszavas szabályokat ennek az U halmaznak a segítségével fogjuk figyelembe venni. Tehát az algoritmusban a környezetfüggetlen nyelvtan λ- mentes részére szorítkozunk (vagyis nem használjuk az $A \to \lambda$ alakú szabályokat); viszont minden egyes $A \to r.Bt$ pontozott szabály esetén (A, $B \in N$, r, $t \in (N \cup T)^*$) ha ez a szabály bekerült a felismerési mátrix egy cellájába, és $B \in U$, akkor $A \to rB.t$ pontozott szbályt is beírjuk az adott cellába ezzel figyelembe véve a $B \Rightarrow^* \lambda$ esetet a szóbanforgó B- re. Ha a feladatunk az, hogy eldöntsük $\lambda \in L(G)$ teljesül-e, akkor ezt az $S \in U$ kérdés alapján válaszoljuk meg. A továbbiakban nézzük meg ha hosszabb szóra kell megoldanunk a feladatot. Jelöljük a bemenő szó i- edik betűjét a_i- vel, tehát legyen $w = a_1 \ldots a_n$. Legyen továbbá $G = (N, T, S, H)$ egy környezetfüggetlen nyelvtan, és jelöljük az előző ábra szerint indexelt felismerési mátrix i- edik sorának j- edik elemét $F_{i,j}$- vel. Legyen továbbá $U = \{A \mid A \in N, A \Rightarrow^* \lambda\}$, és H' a H λ- mentes része, vagyis $H' = \{A \mid A \to p \in H, p \neq \lambda\}$. Kezdetben a felismerési márix minden eleme üres, és az algoritmus mindegyikbe egy vagy több pontozott szabályt írhat be, vagy üresen is hagyhatja. A felismerési mátrix kitöltését oszloponként alulról fölfelé végezzük, kivéve a fődiagonálisban lévő elemeket, amelyeket az adott oszlopban utoljára töltünk ki. Az egyes oszlopokat balról jobbra vesszük sorra (ahogy az előző ábra mutatja). Az Earley-féle algoritmus lépései egy $w \neq \lambda$ input szóra a következőek. Mindegyik lépés

előtt informálisan megadjuk, hogy miről is szól az adott lépés, ezután formálisan (matematikailag pontosan) is megadjuk az adott lépést.

0. (Segédlépés: nem λ-mentes nyelvtanok esetére) minden egyes pontozott szabály beírása után alkalmazandó: Ha a pont után $B \in U$ nemterminális áll, akkor az adott szabályt felvesszük úgy is, hogy a pont már a B nemterminális után áll, az így beírt szabályt is viszgáljuk ez alapján.

1. (Kezdőlépés: az S- el kezdődő, vagyis az $S \to p$ alakú szabályok felvétele a legelső cellába kezdőponttal.) Minden $S \to p \in H'$ szabályra legyen $S \to .p \in F_{0,0}$, és legyen $j=0$.

2. (Főátlóbeli elemek kitöltése: ha az adott oszlopban a pont után szerepel B nemterminális, akkor felvesszük a B- vel kezdődő szabályokat kezdőponttal.) Ha $B \to p \in H'$ és van olyan $k \le j$, hogy $A \to r.Bt \in F_{k,j}$ valamely r, $t \in (N \cup T)^*$- ra, akkor legyen $B \to .p \in F_{j,j}$. Ekkor az újonnan beírt szabályokra is megvizsgáljuk az adott feltételek fennállását.

3. (Eggyel jobbra lépés.) Ha $j<n$, akkor növeljük j értékét eggyel, és legyen $i=j$-1.

4. (A baloldali szomszéd cellából szabálymásolás az oszlop terminálisnának átlépésével.) Legyen $A \to ra_j.t \in F_{i,j}$, ha $A \to r.a_j t \in F_{i,j-1}$.

5. (Cella hasonlítások: az oszlopban kitöltött legalsó cellát a kitöltendő cella balszomszédjával hasonlítjuk, aztán az oszlopban fölfelé haladunk az aktuális celláig, a sorban pedig balra a legelsőig; ha az oszlopban levő cellában van pontra végződő pontozott szabály, ami B nemterminálissal kezdődik, akkor a sorban levő cellából szabályt másolhatunk miközben átlépjük a B- t.) Legyen $A \to rB.t \in F_{i,j}$, ha van olyan k, melyre $i \le k<j$, és $A \to r.Bt \in F_{i,k}$, $B \to p. \in F_{k,j}$ valamely $p \in (N \cup T)^*$- ra.

6. (Ha a legutolsó cellát töltöttük ki, akkor w benne van a generált nyelvben pontosan akkor, ha ebben a cellában van $S \to p.$ alakú szabály; egyébként eggyel följebb lépünk és vissza a 4. lépésre, vagy ha legfelül vagyunk, akkor urás a főátlóra, és a 2. lépésre.) Ha $i=0$ és $j=n$, akkor $w \in L(G)$ pontosan akkor ha van olyan $p \in (N \cup T)^*$, hogy $S \to p. \in F_{0,n}$. Egyébként: ha $i>0$, akkor csökkentsük az i értékét eggyel, és térjünk vissza a negyedik lépésre. Ha $i=0$, akkor pedig térjünk vissza a második lépésre.

Ezzel az algoritmussal tehát megadtuk a felismerési mátrixot, amelyből a w szó levezetése viszonylag egyszerűen rekonstruálható, amennyiben $w \in L(G)$. Az algoritmus helyességének a bizonyításához be kell látnunk a következő tételt.

58. Tétel. Az Earley-féle algoritmussal nyert felismerési mátrixban egy $A \to p.r$ pontozott szabály (p, $r \in (N \cup T)^*$) pontosan akkor szerepel az $F_{i,j}$- ben, ha $A \to pr \in H$, $p \Rightarrow^* a_{i+1} \ldots a_j$, és van olyan $t \in (N \cup T)^*$ szó, amelyre $S \Rightarrow^* a_1 \ldots a_i A t$.

Bizonyítás. A bizonyítást először a λ- mentes nyelvtanokra végezzük el.

Először a feltétel szükségességét látjuk be. A bizonyítást a mátrix elemeire vonatkozó teljes indukcióval végezzük.

Az $F_{0,0}$ esetében az 1. és a 2. lépések szerint csak olyan pontozott szabályok jöhetnek szóba, ahol a jobb oldalon a pont előtti rész üres. Ha tehát $A \to .r$, akkor vagy $A=S$ és $S \to r \in H$, vagy $A \to r \in H$ és $S \Rightarrow^* At$ valamely t- re.

Az indukciót ezek után az (i, j) indexpároknak egy olyan elrendezésével végezzük, amely pontosan megfelel a mátrix kitöltési sorrendjének. A fődiagonális elemeit külön kell kezelnünk. Tegyük fel tehát, hogy a bizonyítandó állítás minden $F_{l,m}$- re teljesül, ha $m<j$, vagy $m=j$ és $i<l<j$.

Legyen most $A \to p.r \in F_{i,j}$. Ha $i \ne j$, akkor ezt a szabályt csak az algoritmus 4. vagy 5. lépésében írhattuk be ide. Az első esetben $p=p'a_j$ és $A \to p'.a_j r \in F_{i,j-1}$. De akkor az indukciós feltevésből $S \Rightarrow^* a_1 \ldots a_i A t$ következik valamilyen t szóra, és $p' \Rightarrow^* a_{i+1} \ldots a_{j-1}$, amiből az állításunk közvetlenül adódik. Ha viszont az 5. lépésben írtuk be a szabályt, akkor $p=p'B$ valamely $p' \in (N \cup T)^*$ szóra és B változóra úgy, hogy van egy olyan k index, melyre $A \to p'.Br \in F_{i,k}$, és $B \to q. \in F_{k,j}$ valamely t szóra.

Az indukciós feltevésből most $S \Rightarrow^* a_1 \ldots a_i A t$, valamint $p' \Rightarrow^* a_{i+1} \ldots a_k$, illetve $S \Rightarrow^* a_1 \ldots a_k B t$, és $q \Rightarrow^* a_{k+1} \ldots a_j$ adódik. Ezekből a $B \rightarrow q \in H$ figyelembevételével nyilván $p'B \Rightarrow^* a_{i+1} \ldots a_k a_{k+1} \ldots a_j$ amivel az állításunkat bebizonyítottuk.

Hátra van még az $i=j$ eset ($i>0$), vagyis a fődiagonálisban levő elemek esete. Az indukciós feltevés ebben az esetben azokra az $F_{l,m}$ elemekre vonatkozik, ahol $m<j$, vagy $m=j$ és $0<l<j$. A fődiagonális elemeibe csak az algoritmus 2. lépésében írhattunk be szabályokat. De itt is csak olyan $B \rightarrow .q$ alakú szabályok jöhetnek szóba, amelyekre van olyan $i \leq j$, hogy $A \rightarrow p.Br \in F_{i,j}$. Itt az $i=j$ eset is előfordulhat. Tekintsük azonban azt a szabályt, amelyet az algoritmus először helyez el az $F_{j,j}$- ben. Ebben az esetben az $i<j$ kell, hogy fennálljon. Az indukciós feltevésből tehát $S \Rightarrow^* a_1 \ldots a_i A t$ és $p \Rightarrow^* a_{i+1} \ldots a_j$ következik. Ugyanakkor nyilván $A \Rightarrow^* pBr$ is fennáll, amiből $S \Rightarrow^* a_1 \ldots a_j B r t$ következik, s ezzel állításunkat igazoltuk. Az indukciós feltevést most már minden új szabály elhelyezésénél alkalmazhatjuk az $F_{j,j}$- be beírt szabályokra is. Ha tehát egy $B \rightarrow .q$ szabályt az algoritmus azért vesz fel az $F_{j,j}$- be, mert $B \rightarrow q \in H$, és $A \rightarrow p.Br \in F_{j,j}$, akkor itt nyilván $p=\lambda$ és $S \Rightarrow^* a_1 \ldots a_j A t$, és ezért $S \Rightarrow^* a_1 \ldots a_j B r t$, amivel a bizonyítást befejeztük.

A feltétel elégségességének bizonyítását ugyancsak teljes indukcióval végezzük. Először nézzük az $F_{0,0}$- t. Tegyük fel, hogy $S \Rightarrow^* Bt$ teljesül valamely B változóra és $t \in (N \cup T)^*$ szóra, és $B \rightarrow q \in H$. Az $S \Rightarrow^* Bt$ levezetés lépésszámára vonatkozóan egy külön teljes indukciót végzünk. Ha a lépésszám nulla, akkor $B=S$ és $t=\lambda$, és ezért az algoritmus a $B \rightarrow .t$ szabályt már az első lépésben felveszi az $F_{0,0}$- ba. Tegyük fel, hogy az állítás k lépésre igaz, és legyen az $S \Rightarrow^* Bt$ levezetés lépésszáma $k+1$. Ekkor nyilván van olyan A nemterminális és t' szó, hogy $S \Rightarrow^* At'$, $A \Rightarrow Bt''$, ahol $t=t''t'$ és az $S \Rightarrow^* At'$ levezetés lépésszáma k. Ezért az indukciós feltevésből $A \rightarrow .Bt'' \in F_{0,0}$, tehát az algoritmus 2. lépése szerint $B \rightarrow .q \in F_{0,0}$.

Most tegyük fel, hogy az algoritmus elégséges volta igaz minden olyan $F_{l,m}$- re, ahol $m<j$, vagy $s=m$ és $i<l<j$. Legyen akkor $A \rightarrow pr \in H$, $p \Rightarrow^* a_{i+1} \ldots a_j$, és $S \Rightarrow^* a_1 \ldots a_i A t$ valamely t szóra. Ha most $p=p'a_j$, akkor az indukciós feltevés szerint $A \rightarrow p'.a_j r \in F_{i,j-1}$, és az algoritmus 4. lépése alapján $A \rightarrow p'a_j.r \in F_{i,j}$. Egyébként $p=p'B$, ahol $B \Rightarrow^* a_{k+1} \ldots a_j$ és $p' \Rightarrow^* a_{i+1} \ldots a_k$ valamilyen k- ra. Az indukciós feltevésből most $A \rightarrow p'.Br \in F_{i,k}$ és $B \rightarrow q. \in F_{k,j}$ adódik valamilyen q szóra, amelyre $q \Rightarrow^* a_{k+1} \ldots a_j$ és $B \rightarrow q \in H$. De akkor az algoritmus 5. lépése szerint $A \rightarrow p'B.r \in F_{i,j}$.

Végül vegyük a fődiagonális elemeit. Az indukciós feltevést most azokra az $F_{l,m}$ elemekre tesszük, amelyekre $m \leq j$ és $0<l<j$. Legyen tehát $A \rightarrow r \in H$ és $S \Rightarrow^* a_1 \ldots a_j A t$. Ez a levezetés nyilván átrendezhető úgy, hogy a változók közül a legbaloldalibbat helyettesítjük be először mindaddig, amíg az a_j- t meg nem kapjuk. Eszerint van egy olyan A' változó, hogy $S \Rightarrow^* a_1 \ldots a_k A' t' \Rightarrow^* a_1 \ldots a_k a_{k+1} \ldots a_j A t'' t'$, ahol $t''t' \Rightarrow^* t$. Ha itt $k<j$, akkor az indukciós feltevés szerint $A' \rightarrow a_{k+1} \ldots a_j.At'' \in F_{k,j}$, és így az algoritmus 2. lépése szerint $A \rightarrow .r \in F_{j,j}$. A $k=j$ esetben viszont az indukciós feltevést már alkalmazhatjuk az $A' \rightarrow At''$ szabályra és az $S \Rightarrow^* a_1 \ldots a_j A' t'$ levezetésre, hiszen ez a levezetés véges számú lépésből áll, tehát véges számú lépésben el kell jutnunk egy olyan esethez, ahol $k<j$. Az $A' \rightarrow .At'' \in F_{j,j}$- ből pedig az algoritmus 2. lépése szerint $A \rightarrow .r \in F_{j,j}$ következik, és ezzel a tétel bizonyítását befejeztük a λ- mentes esetre.

Ha a nyelvtan nem λ- mentes akkor az algoritmus 0. lépése ezt éppen úgy veszi figyelembe, ahogyan az Üresszó lemma bizonyításában konstruáltunk ekvivalens nyelvtant az eredetihez, vagyis minden olyan nemterminális esetén amiből az üresszó levezethető úgy is tekinhetjük a levezetésben, hogy az üres szót vezetjük le belőle. Ezek alapján nyilvánvaló, hogy az algoritmus az ilyen esetlekben is helyesen működik. ∎

Az algoritmus futása során az input szó négyzetével arányos méretű táblázat (a felismerési mátrix) kitöltése történik. Az egy mezőbe írható pontozott szabályok számának felső korlátja kiszámolható a H szabályhalmaz elemeinek száma és hosszai alapján. Az algoritmus lépéseit megvizsgálva láthatjuk, hogy a 2. és 5. lépésekben az adott cella kitöltéséhez maximum a szó hosszának megfelelő mennyiségű cellában, illetve cellapárban található pontozott szabályokat kell megvizsgálni. Így a program futásának ideje, hasonlóan a CYK algoritmushoz, az input szó hosszának köbével arányos.

Az Earley-féle algoritmus működését először a következő egyszerű példán szemléltetjük.

7.39. példa - Earley algoritmus 1.feladat

A G generatív nyelvtan legyen a következő formában definiált: $G=(\{S, A, B\}, \{a, +, \times, (,)\}, S, H)$, ahol H szabályhalmaz a következő: $\{S \to S+A, A \to A\times B, B \to (S), S \to A, A \to B, B \to a\}$.

Tekintsük most az $a\times a+a$ bemenő szó szintaktikai elemzését. Az algoritmus az első lépésben az $S \to .S+A$, $S \to .A$ szabályokat veszi fel az $F_{0,0}$- ba. Ezután a 2. lépés alapján kiegészíti az $F_{0,0}$- at az $A \to .A\times B$, $A \to .B$, $B \to .(S)$, $B \to .a$ szabályokkal. Utána j értéke 1 lesz, és az algoritmus a 4. lépésben az $F_{0,1}$- be felveszi a $B \to a.$ pontozott szabályt, mivel a bemenő szó első betűje a. Az 5. lépésben az $F_{0,1}$ kiegészül még az alábbi szabályokkal: $A \to B.$, $A \to A.\times B$, $S \to A.$, $S \to S.+A$.

Ezután a 2. lépés következik, ahol most $F_{1,1}$- et kell kitölteni. De az $F_{0,1}$- ben szereplő szabályokhoz nem található egyetlen olyan szabály sem, amely a feltételeket kielégíti, hiszen az $F_{0,1}$- beli szabályokban a pont után közvetlenül nem is fordul elő változó. Ezért az $F_{1,1}$ üres marad. Az algoritmus ezután a j=2 értékre hajtja végre a 4. lépést, melynek során az $F_{1,2}$ nyilván szintén üres marad, az $F_{0,2}$- be pedig az $A \to A\times.B$ szabály fog bekerülni. Az 5. lépésben mind az $F_{1,2}$, mind az $F_{0,2}$ változatlan marad. Az $F_{2,2}$- be ezután a 2. lépés során bekerülnek a $B \to .(S)$ és $B \to .a$ szabályok, mivel $A \to A\times.B \in F_{0,2}$.

A felismerési mátrix kitöltésének további menetét nem részletezzük, de a következő ábrán megadjuk a teljes mátrixot. Mivel ebben a mátrixban $S \to S+A.\in F_{0,5}$, a kérdéses bemenő szó benne van az $L(G)$- ben. A tételünk szerint ugyanis ez pontosan akkor áll fenn, ha $S \to S+A \in H$ és $S+A \Rightarrow^{*} a\times a+a$, továbbá van olyan t szó, amelyre $S \Rightarrow^{*} St$, mely utóbbi összefüggés már nem is lényeges most a mi szempontunkból.

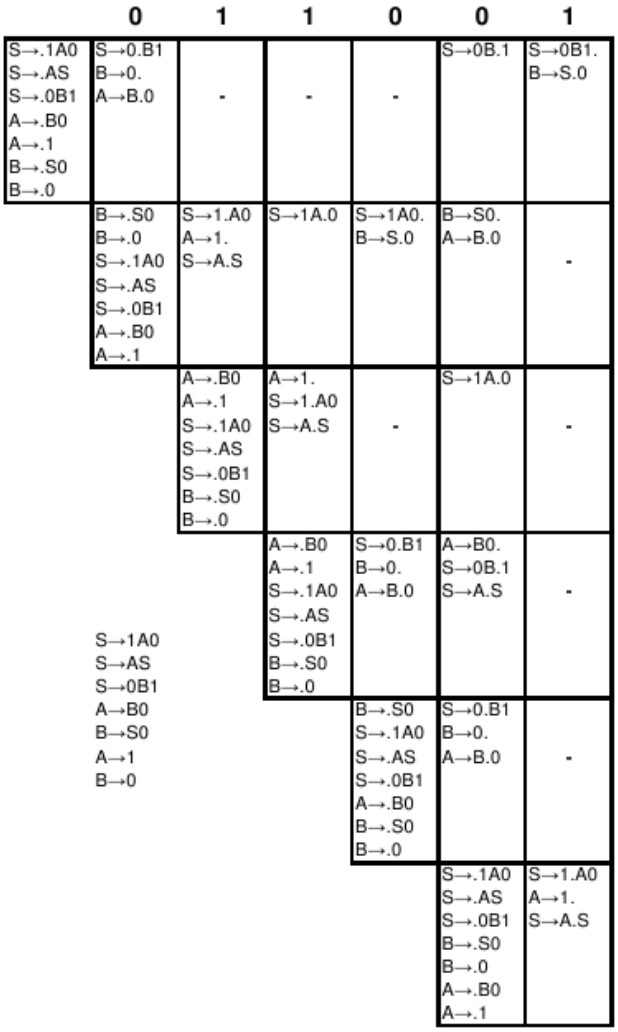
	a	X	a	+	a
S→.S+A S→.A A→.AXB A→.B B→.(S) B→.a	B→a. A→B. A→A.XB S→A. S→S.+A	A→AX.B	A→AXB. S→A. A→A.XB S→S.+A	S→S+.A	S→S+A. S→S.+A
	---	---	---	---	---
		B→.(S) B→.a	B→a.	---	---
			---	---	---
				A→.AXB A→.B B→.(S) B→.a	B→a. A→B. A→A.XB

★

7.40. példa - Earley algoritmus 2. feladat

Tekintsük a $G=(\{S,A,B\},\{0,1\},S,H)$ grammatikát, ahol H szabályai:
$\{S \to 1A0, S \to 0B1, A \to B0, B \to S0, A \to 1, B \to 0\}$.
Döntsük el Earley-algoritmus segítségével, hogy levezethető-e a nyelvtanban a 011001 szó!

Megoldás:

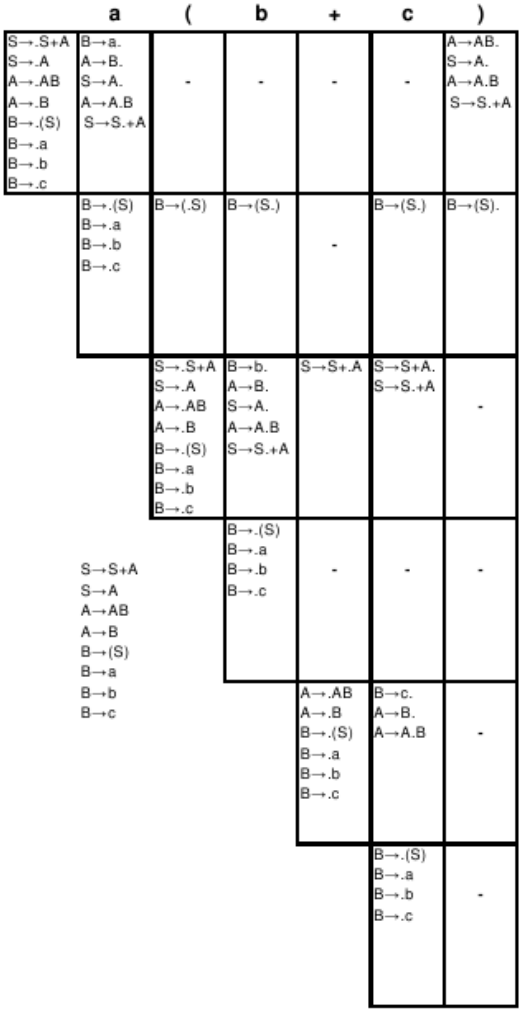


Mivel a jobbfelső "cella" tartalmaz *S*-ből kiinduló pontra végződő szabályt,
ezért a 011001 szó benne van az adott grammatika által generált nyelvben. ★

7.41. példa - Earley algoritmus 3. feladat

Tekintsük a $G=(\{S,A,B\},\{a,b,c,+,(,)\},S,H)$ grammatikát, ahol H szabályai:
$\{S \to S+A, S \to A,\ A \to AB,\ A \to B,\ B \to (S),\ B \to a,\ B \to b,\ B \to c\}$.
Döntsük el Earley-algoritmus segítségével, hogy levezethető-e a nyelvtanban az $a(b+c)$ szó!

Megoldás:

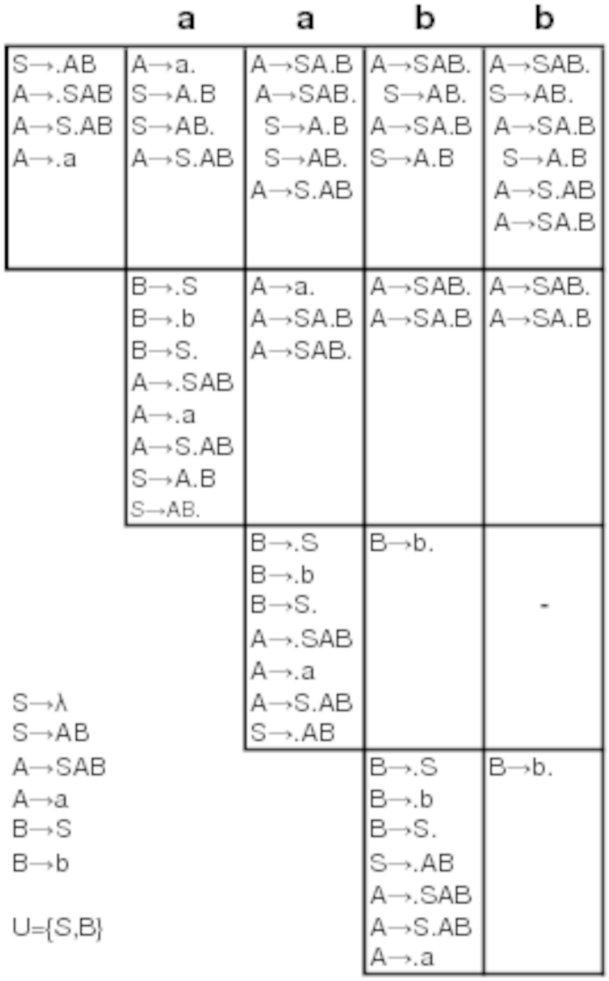


Mivel a jobbfelső "cella" tartalmaz *S*-ből kiinduló pontra végződő szabályt,
ezért az $a(b+c)$ szó benne van az adott grammatika által generált nyelvben. ★

7.42. példa - Earley algoritmus 4. feladat

Tekintsük a $G=(\{S,A,B\},\{a,b\},S,H)$ grammatikát, ahol H szabályai:
$\{S \to AB, S \to \lambda, A \to SAB, A \to a, B \to S, B \to b\}$.
Döntsük el Earley-algoritmus segítségével, hogy levezethető-e a nyelvtanban az *aabb* szó!

Megoldás:



Mivel a jobbfelső "cella" tartalmaz *S*-ből kiinduló pontra végződő szabályt, ezért az *aabb* szó benne van az adott grammatika által generált nyelvben. ★

7.8. Irodalmi megjegyzések

A természetes nyelvek leírásának céljából vezette be [Chomsky 1956] a környezetfüggetlen nyelvtanokat. Nem sokkal később a számítógépes nyelvek leírására készült el a BNF (Backus-Naur Form), lásd pl. [Backus 1959] ugyanezen nyelvosztály leírására. A Chomsky-féle normálalak [Chomsky 1959]-ben, a Greibach-féle normálforma [Greibach 1965]-ben jelent meg. A negatív zártsági tulajdonságok a metszet és a komplementerképzés műveletekre [Scheinberg 1960]-ban kerültek bizonyításra. A Bar-Hillel lemma [Bar-Hillel et al 1961]-ben jelent meg. Azóta több más iterációs lemma is megjelent, amelyek kialakításánál az egyik főcél az, hogy a nem környezetfüggetlen nyelveknek minél kisebb része teljesítse a feltételeiket. Ilyen „erősebb” iterációs lemma található,

pl. [Dömösi et al 1996]-ban. A veremautomata ötlete [Oettinger 1961] és [Schutzenberger 1963]-ból származik. A CYK algoritmust hárman egymástól függetlenül írták le: J. Cocke és T. Kasami nem publikálták, [Younger 1967]-ben jelent meg hivatalosan. Az Earley-féle algoritmus pedig [Earley 1970]-ben volt eredetileg publikálva. A környezetfüggetlen nyelvek szintaktikai elemzése iránt érdeklődőknek ajánljuk a magyar [Bach 2002] és [Fülöp 2005] tankönyveket.

8. fejezet - Környezetfüggő nyelvek

A környezetfüggetlen nyelvekkel kapcsolatosan sok elméleti eredmény ismert, és a gyakorlatban is jól alkalmazhatóak, ez köszönhető főleg annak, hogy a levezetési fa fogalma jól illeszkedik ezekhez a nyelvtanokhoz, illetve, amint láttuk a szóprobléma is hatékonyan megoldható. A világ viszont nem környezetfüggetlen, nagyon sok olyan jelenséget ismerünk, ami nem írható le környezetfüggetlen nyelvtanokkal, pl. a természetes nyelvek, az igaz ítéletlogikai formulák nyelve, a prímszámok halmaza stb.

8.1. Szóhossznemcsökkentő (monoton) nyelvtanok

Először az 1. típusú nyelvtanok és nyelvek általánosabb definícióját (lásd monoton nyelvtan [25]) vesszük elő emlékeztetőül.

16. Definíció. Egy G generatív nyelvtant hosszúságot nem csökkentőnek (vagy röviden monotonnak) nevezünk, ha minden $p \to q \in H$ szabályára $|p| \leq |q|$ teljesül, kivétel csak az $S \to \lambda$ alakú szabály lehet, de ekkor az S szimbólum nem szerepelhet egyik szabály jobboldalán sem. ★

9. *Megjegyzés.* Az $S \to \lambda$ alakú szabály csak az üresszó generálására használható, és pontosan akkor szerepel egy nyelvtan szabályai közt, ha a generált nyelv $L(G)$ tartalmazza az üresszót.

8.1. példa - Monoton nyelvtan

Legyen a $G=(\{S, A, B, C\}, \{a, b, c\}, S, \{S \to \lambda, S \to abc, S \to aABC, A \to aABC, A \to aBC, CB \to BC, aB \to ab, bB \to bb, bC \to bc, cC \to cc\})$. Egyszerűen bizonyítható, hogy G az $L(G)=\{a^n b^n c^n | n \geq 0\}$ nyelvet generálja. ★

Egyrészt, az előző fejezetben megmutattuk a Bar-Hillel lemma segítségével, hogy $L(G)$ nem környezetfüggetlen. Másrészt, az üresszó-lemma segítségével azt is láttuk, hogy minden környezetfüggetlen nyelv egyben környezetfüggő is; így a környezetfüggő és környezetfüggetlen nyelvek esetére a Chomsky hierarchia élességét bizonyítottuk. Már csak azt kell belátnunk, hogy a monoton nyelvtanok pontosan a környezetfüggő nyelvosztályt generálják.

Figyeljük meg, hogy egy monoton nyelvtanban pl. a $bcA \to Baa$ megengedett szabály, vagyis a terminális betűket is átírhatjuk, ha a közelükben nemterminális szimbólum áll, ez kicsit ellentmond a "terminális" elnevezésnek. Azt hogy a helyzet mégsem ennyire súlyos a következő alfejezetben fogjuk belátni: minden környezetfüggő nyelvet generálhatunk olyan monoton nyelvtannal is, amiben nem történhet terminális átírás.

8.2. Normálformák a környezetfüggő nyelvtanokhoz, a monoton és a környezetfüggő nyelvtanok ekvivalenciája

A normálformák itt is nagy jelentőséggel bírnak. Az első normálforma amivel megismerkedünk, a szabályok hosszát korlátozza: csak olyan terminális normális alakú monoton szabályokat engedünk meg, amelyekben a jobboldal hossza maximum kettő.

17. Definíció. Egy monoton nyelvtanról azt mondjuk, hogy Kuroda-féle normálalakban van, ha minden szabálya az alábbi alakok egyikében van: $A \to B$, $A \to BC$, $A \to a$, $AB \to CD$($a \in T$, A, B, C, $D \in N$). ★

59. Tétel. Minden monoton nyelvtanhoz van vele ekvivalens Kuroda normálformájú nyelvtan.

Bizonyítás. A bizonyítás konstruktív, minden G monoton nyelvtan esetén hatékonyan megkonstruálhatjuk azt a Kuroda normálformájú nyelvtant, amely az $L(G)\setminus\{\lambda\}$ nyelvet generálja: Legyen tehát adott $G=(N, T, S, H)$ monoton nyelvtan.

Az algoritmus első lépésével (ugyanúgy, mint pl. a Chomsky normálforma esetén) elérjük, hogy a terminális szimbólumok csak $A \to a$ alakú szabályokban forduljanak elő: ehhez vezessünk be minden terminális szimbólumhoz egy-egy a nyelvtanban még nem szereplő új nemterminális szimbólumot: tehát legyen $N'=\{X_a | a\in T\}$, miközben $N\cap N'=\emptyset$. Állítsuk elő a H' szabályrendszert a H- ból a következőképpen: a H minden szabályában szereplő minden a terminálist helyettesítsünk a neki megfelelő X_a újonnan bevezetett nemterminálissal és így másoljuk át a H'- be és adjuk még hozzá az $X_a \to a$ alakú szabályokat minden $a\in T$ -re. Ekkor $G'=(N\cup N', T, S, H')$ ekvivalens G- vel és benne terminálisok csak $X_a \to a$ alakú szabályokban fordulnak elő.

Második lépésben, ha $p \to q\in H$, ahol $p=A_1\ldots A_m$ és $q=B_1\ldots B_n$, akkor világos a nyelvtan definíciójából következően, hogy $m \le n$. Ekkor vegyük sorra a szabályokat és járjunk el a következő módon:

- ha $n \le 2$, akkor a szabály eleget tesz a Kuroda normálformának.

- ha $m=1$, $n>2$, akkor (ugyanúgy, mint a Chomsky normálformára alakításnál; lásd Chomsky-féle normálalak) helyettesítsük az $A \to B_1\ldots B_n$ szabályt $A \to B_1C_1$, $C_1 \to B_2C_2$, …, $C_{n-2} \to B_{n-1}B_n$ szabályokkal, ahol a C_1, …, C_{n-2} újonnan bevezetett, csak itt szereplő nemterminálisok.

- ha $m \ge 2$, $n>2$, akkor a C_1, C_2, …, C_{n-2} újonnan bevezetett, és csak itt szereplő, nemterminálisok segítségével hozzuk létre a következő szabályokat az eredeti szabály helyettesítésére: $A_1A_2 \to B_1C_1$, $C_1A_3 \to B_2C_2$, …, $C_{m-2}A_m \to B_{m-1}C_{m-1}$, $C_{m-1} \to B_mC_m$, …, $C_{n-2} \to B_{n-1}B_n$.

Az új nyelvtan Kuroda normálformájú és ekvivalens az eredetivel. ∎

8.2. példa - Kuroda normálforma 1.feladat

Adjunk meg a $G=(\{S,A,B\},\{x,y,z\},S,H)$ környezetfüggő nyelvtannal ekvivalens Kuroda-féle normálalakú nyelvtant, ahol
$H=\{\ S \to ABABx,\ ABA \to AyyyA,\ Ayyy \to Byyy,\ A \to z,\ A \to BB,\ B \to S,\ B \to x\ \}$.

Megoldás:
(I.) Első lépésben megadunk egy G_1 nyelvtant, ami ekvivalens a G nyelvtannal és terminális csak $X \to a$ alakú szabályban fordul elő ($X\in N$, $a\in T$).
Ehhez először a G nyelvtan minden olyan x_i terminális betűjéhez, amely szerepel olyan szabályban, ami nem normálalakú, új X_i nemterminálist vezetünk be.
Ezután a G_1 nyelvtan H_1 szabályhalmazát úgy kapjuk, hogy felvesszük az $X_i \to x_i$ szabályokat, valamint a H szabályhalmaz elemeit átvesszük úgy, hogy a szabályokban az x_i betűk azon előfordulásait, melyek nem normálalakú szabályban szerepelnek, X_i-re cseréljük.

Jelen esetben legyenek az új nemterminálisok az X és az Y, ekkor:
$G_1=(\{S,A,B,X,Y\},\{x,y,z\},S,H_1)$, ahol
$H_1=\{\ X \to x,\ Y \to y,\ S \to ABABx,\ ABA \to AyyyA,\ Ayyy \to Byyy,\ A \to z,\ A \to BB,\ B \to S,\ B \to x\ \}$

(II.) Második lépésben megadunk egy G_2 nyelvtant, ami ekvivalens
az eredeti nyelvtannal, normálformájú és nem szerepel benne $Y \to Y_1Y_2...Y_n$, $n \ge 3$ alakú szabály.
Ehhez a G_1 nyelvtanból indulunk ki.
A H_1 *szabályhalmaz* $Y \to Y_1Y_2...Y_n$, $n \ge 3$ alakú szabályait helyettesítjük új szabályokkal,
a többi szabályt pedig változtatás nélkül átvesszük a H_2 szabályhalmazba.
Minden $Y \to Y_1Y_2...Y_n$, $n \ge 3$ alakú szabályhoz
vezessünk be $Z_1,Z_2, \ldots ,Z_{n-2}$ nemterminálisokat, ahol a Z_i-ből pontosan az $Y_{i+1}\ldots Y_n$ szót fogjuk levezetni:

az összes $Y \rightarrow Y_1Y_2...Y_n$, $n \geq 3$ alakú szabályt helyettesítsük
a következő szabályokkal:
$Y \rightarrow Y_1Z_1$,
$Z_1 \rightarrow Y_2Z_2$,
.
.
.
$Z_{n-3} \rightarrow Y_{n-2}Z_{n-2}$,
$Z_{n-2} \rightarrow Y_{n-1}Y_n$.

Jelen esetben:
$G_2=(\{S,A,B,X,Y,Z_1,Z_2,Z_3\},\{x,y,z\},S,H_2)$, ahol
$H_2=\{ X \rightarrow x, Y \rightarrow y, S \rightarrow AZ_1, Z_1 \rightarrow BZ_2, Z_2 \rightarrow AZ_3, Z_3 \rightarrow BX, ABA \rightarrow AYYYA, AYYY \rightarrow BYYY, A \rightarrow z, A \rightarrow BB, B \rightarrow S, B \rightarrow x \}$

(III.) Harmadik lépésben megadjuk az eredeti nyelvtannal ekvivalens G' Kuruda-féle normá alakú nyelvtant.
Ehhez a G_2 nyelvtanból indulunk ki.
A H_2 szabályhalmaz $X_1X_2 ... X_n \rightarrow Y_1Y_2 ... Y_m$, $n \geq 2$, $m \geq 3$ alakú szabályait helyettesítjük új szabályokkal, a többi szabályt pedig változtatás nélkül tvesszük a H' szabályhalmazba.
Minden $X_1X_2 ... X_n \rightarrow Y_1Y_2 ... Y_m$, $n \geq 2$, $m \geq 3$ alakú szabályhoz vezessünk be
$Z_1,Z_2, ...,Z_{m-2}$ új nemterminálisokat!
Ezek után az
$X_1X_2 ... X_n \rightarrow Y_1Y_2 ... Y_m$, $n \geq 2$, $m \geq 3$ alakú szabályokat helyettesítsük a következő szabályokkal:
$X_1X_2 \rightarrow Y_1Z_1$,
$Z_1X_3 \rightarrow Y_2Z_2$,
.
.
.
$Z_{n-2}X_n \rightarrow Y_{n-1}Z_{n-1}$,
$Z_{n-1} \rightarrow Y_nZ_n$,
.
.
.
$Z_{m-3} \rightarrow Y_{m-2}Z_{m-2}$,
$Z_{m-2} \rightarrow Y_{m-1}Y_m$.

Jelen esetben:
$G'=(\{S,A,B,X,Y,Z_1,Z_2,Z_3,Z_4,Z_5,Z_6,Z_7,Z_8\},\{x,y,z\},S,H')$, ahol
$H'=\{ X \rightarrow x, Y \rightarrow y, S \rightarrow AZ_1, Z_1 \rightarrow BZ_2, Z_2 \rightarrow AZ_3, Z_3 \rightarrow BX, AB \rightarrow AZ_4, Z_4A \rightarrow YZ_5, Z_5 \rightarrow YZ_6, Z_6 \rightarrow YA, AY \rightarrow BZ_7, Z_7Y \rightarrow YZ_8, Z_8Y \rightarrow YY, A \rightarrow z, A \rightarrow BB, B \rightarrow S, B \rightarrow x \}$ ★

8.3. példa - Kuroda normálforma 2.feladat

Adjunk meg a $G=(\{S,A,B\},\{a,b,c\},S,H)$ környezetfüggő nyelvtannal ekvivalens Kuroda-féle normálalakú nyelvtant, ahol
$H=\{ S \rightarrow BaB, BaB \rightarrow BaBa, A \rightarrow SaS, A \rightarrow c, B \rightarrow AbbA, B \rightarrow c \}$.

Megoldás:
(I.) Legyenek az új nemterminálisok az X és az Y, ekkor:
$G_1=(\{S,A,B,X,Y\},\{a,b,c\},S,H_1)$, ahol
$H_1=\{ X \rightarrow a, Y \rightarrow b, S \rightarrow BXB, BXB \rightarrow BXBX, A \rightarrow SXS, A \rightarrow c, B \rightarrow AYYA, B \rightarrow c \}$

(II.)
$G_2=(\{S,A,B,X,Y,Z_1,Z_2,Z_3,Z_4\},\{a,b,c\},S,H_2)$, ahol
$H_2=\{ X \to a,\ Y \to b,\ S \to BZ_1,\ Z_1 \to XB,\ BXB \to BXBX,\ A \to SZ_2,\ Z_2 \to XS,\ A \to c,$
$B \to AZ_3,\ Z_3 \to YZ_4,\ Z_4 \to YA,\ B \to c \}$
(III.)
$G'=(\{S,A,B,X,Y,Z_1,Z_2,Z_3,Z_4,Z_5,Z_6\},\{a,b,c\},S,H')$, ahol
$H'=\{ X \to a,\ Y \to b,\ S \to BZ_1,\ Z_1 \to XB,\ BX \to BZ_5,\ Z_5B \to XZ_6,\ Z_6 \to BX,$
$A \to SZ_2,\ Z_2 \to XS,\ A \to c,\ B \to AZ_3,\ Z_3 \to YZ_4,\ Z_4 \to YA,\ B \to c\}$ ★

8.4. példa - Kuroda normálforma 3.feladat

Adjunk meg a $G=(\{S,A,B\},\{a,b\},S,H)$ környezetfüggő nyelvtannal ekvivalens
Kuroda-féle normálalakú nyelvtant, ahol
$H=\{ S \to AB,\ AB \to ABAA,\ ABA \to ABAABB,\ B \to S,\ A \to a,\ B \to b,\ A \to SA \}$.

Megoldás:
(I.)
Mivel a G nyelvtan már normálalakban van, ezért $G_1=G$.
(II.)
Mivel a G_1 nyelvtanban nincs $Y \to Y_1Y_2...Y_n$, $n \geq 3$ alakú szabály,
ezért $G_2=G_1$.
(III.)
$G'=(\{S,A,B,Z_1,Z_2,Z_3,Z_4,Z_5,Z_6\},\{a,b\},S,H')$, ahol
$H'=\{ S \to AB,\ AB \to AZ_1,\ Z_1 \to BZ_2,\ Z_2 \to AA,\ AB \to AZ_3,\ Z_3A \to BZ_4,\ Z_4 \to AZ_5,$
$Z_5 \to AZ_6,\ Z_6 \to BB,\ B \to S,\ A \to a,\ B \to b,\ A \to SA\}$ ★

8.5. példa - Kuroda normálforma 4.feladat

Adjunk meg a $G=(\{S,A,B\},\{0,1\},S,H)$ hosszúság nemcsökkentő nyelvtannal ekvivalens Kuroda-féle normálalakú nyelvtant, ahol
$H=\{ S \to SAS,\ SA \to B0B0S,\ S \to 1,\ A \to S0S,\ B0B0 \to 0S0S \}$.

Megoldás:
(I.) Legyen az új nemterminális a C, ekkor:
$G_1=(\{S,A,B,C\},\{0,1\},S,H_1)$, ahol
$H_1=\{ C \to 0,\ S \to SAS,\ SA \to BCBCS,\ S \to 1,\ A \to SCS,\ BCBC \to CSCS \}$
(II.)
$G_2=(\{S,A,B,C,Z_1,Z_2\},\{0,1\},S,H_2)$, ahol
$H_2=\{ C \to 0,\ S \to SZ_1,\ Z_1 \to AS,\ SA \to BCBCS,\ S \to 1,\ A \to SZ_2,\ Z_2 \to CS,\ BCBC \to CSCS \}$
(III.)
$G'=(\{S,A,B,C,Z_1,Z_2,Z_3,Z_4,Z_5,Z_6,Z_7\},\{0,1\},S,H')$, ahol
$H'=\{ C \to 0,\ S \to SZ_1,\ Z_1 \to AS,\ SA \to BZ_3,\ Z_3 \to CZ_4,\ Z_4 \to BZ_5,\ Z_5 \to CS,$
$S \to 1,\ A \to SZ_2,\ Z_2 \to CS,\ BC \to CZ_6,\ Z_6B \to SZ_7,\ Z_7C \to CS \}$ ★

8.6. példa - Kuroda normálforma 5.feladat

Adjunk meg a $G=(\{S,B\},\{a,b,c\},S,H)$ hosszúság nemcsökkentő nyelvtannal ekvivalens Kuroda-féle normálalakú nyelvtant, ahol
$H=\{ S \to abc,\ S \to aSBc,\ cB \to Bc,\ bB \to bb \}$.

Megoldás:
(I.) Legyenek az új nemterminálisok az A, E és a C, ekkor:
$G_1=(\{S,B,A,E,C\},\{a,b,c\},S,H_1)$, ahol
$H_1=\{\ A \rightarrow a,\ E \rightarrow b,\ C \rightarrow c,\ S \rightarrow AEC,\ S \rightarrow ASBC,\ CB \rightarrow BC,\ EB \rightarrow EE\ \}$
(II.)
$G_2=(\{S,B,A,E,C,Z_1,Z_2,Z_3\},\{a,b,c\},S,H_2)$, ahol
$H_2=\{\ A \rightarrow a,\ E \rightarrow b,\ C \rightarrow c,\ S \rightarrow AZ_1,\ Z_1 \rightarrow EC,\ S \rightarrow AZ_2,\ Z_2 \rightarrow SZ_3,$
$Z_3 \rightarrow BC,\ CB \rightarrow BC,\ EB \rightarrow EE\ \}$
(III.)
Mivel a G_2 nyelvtan már Kuroda-féle normálalakban van, ezért $G'=G_2$. ★

8.7. példa - Kuroda normálforma 6.feladat

Adjunk meg a $G=(\{S,A,B\},\{d,e\},S,H)$ hosszúság nemcsökkentő nyelvtannal ekvivalens Kuroda-féle normálalakú nyelvtant, ahol
$H=\{\ S \rightarrow BeBe,\ BeBe \rightarrow dAdA,\ eB \rightarrow dede,\ Bd \rightarrow SAS,\ A \rightarrow ede\ \}$.

Megoldás:
(I.) Legyenek az új nemterminálisok a D és az E, ekkor:
$G_1=(\{S,A,B,D,E\},\{d,e\},S,H_1)$, ahol
$H_1=\{\ D \rightarrow d,\ E \rightarrow e,\ S \rightarrow BEBE,\ BEBE \rightarrow DADA,\ EB \rightarrow DEDE,\ BD \rightarrow SAS,\ A \rightarrow EDE\ \}$
(II.)
$G_2=(\{S,A,B,D,E,Z_1,Z_2,Z_3\},\{d,e\},S,H_2)$, ahol
$H_2=\{\ D \rightarrow d,\ E \rightarrow e,\ S \rightarrow BZ_1,\ Z_1 \rightarrow EZ_2,\ Z_2 \rightarrow BE,\ BEBE \rightarrow DADA,\ B \rightarrow DEDE,$
$BD \rightarrow SAS,\ A \rightarrow EZ_3,\ Z_3 \rightarrow DE\ \}$
(III.)
$G_2=(\{S,A,B,D,E,Z_1,Z_2,Z_3,Z_4,Z_5,Z_6,Z_7,Z_8\},\{d,e\},S,H')$, ahol
$H'=\{\ D \rightarrow d,\ E \rightarrow e,\ S \rightarrow BZ_1,\ Z_1 \rightarrow EZ_2,\ Z_2 \rightarrow BE,\ BE \rightarrow DZ_4,\ Z_4B \rightarrow AZ_5,\ Z_5E \rightarrow DA,$
$EB \rightarrow DZ_6,\ Z_6 \rightarrow EZ_7,\ Z_7 \rightarrow DE,\ BD \rightarrow SZ_8,\ Z_8 \rightarrow AS,\ A \rightarrow EZ_3,\ Z_3 \rightarrow DE\ \}$ ★

A Kuroda-féle normálformát a Révész György nevéhez fűződő észrevétellel tovább alakíthatjuk.

Révész-trükk

Egy $AB \rightarrow CD$ alakú monoton szabály helyettesíthető a következő négy szabállyal, ahol A' és B' újonnan bevezetett nemterminálisok:

$AB \rightarrow A'B$
$A'B \rightarrow A'B'$
$A'B' \rightarrow CB'$
$CB' \rightarrow CD$

Figyeljük meg, hogy az újonnan bevezetett szabályok mindegyike eleget tesz a környezetfüggő nyelvtanoknál előírt megszorításoknak. Ez alapján kimondhatjuk a következő tételt.

60. Tétel. Minden monoton nyelvtanhoz van vele ekvivalens, aminek szabályai csak $A \rightarrow B$, $A \rightarrow BC$, $A \rightarrow a$, $AB \rightarrow AC$, $AB \rightarrow CB$ alakúak lehetnek ($a \in T$, A, B, $C \in N$).

Ezzel egyúttal azt is bizonyítottuk, hogy minden monoton nyelvtannal generálható nyelv generálható környezetfüggő nyelvtannal is. Mivel a másik irányú tartalmazás a nyelvtantípusok definíciójából adódóan nyilvánvaló (vagyis minden környezetfüggő nyelvtan egyben monoton is), ezért :

5. Következmény. A monoton és a környezetfüggő nyelvtanok által generált nyelvek osztálya megegyezik (így, most már jogosan hívhatjuk a monoton nyelvtanok által generált nyelveket környezetfüggőnek).

További normálformákat mutatunk a környezetfüggő nyelvosztályhoz (bizonyítások nélkül).

Cremers normálforma (lánc szabályok kiküszöbölése):

61. Tétel. Minden környezetfüggő nyelvtanhoz van vele ekvivalens, melynek szabályai csak a következő alakúak lehetnek:

$A \to a, A \to BC, AB \to CD$.

Révész-féle egyoldali normálforma:

18. Definíció. Egy nyelvtan Révész-féle egyoldali normálformájú, ha szabályai csak $A \to a$, $A \to B$, $A \to BC$, $AB \to AC$, $AB \to BA$ alakúak lehetnek ($a \in T$, A, B, $C \in N$). ★

62. Tétel. Minden környezetfüggő nyelvtanhoz van vele ekvivalens Révész-féle egyoldali normálformájú nyelvtan.

10. *Megjegyzés.* Révész-féle egyoldali normálformájú nyelvtanban csak egyoldali környezetfüggés megengedett, viszont permutációs szabály (helycsere) használható (ami nem tesz eleget a környezetfüggő definíciónak).

Felmerülhet a kérdés, hogy vajon mindkét típusú nem környezetfüggetlen szabályra szükség van-e ahhoz, hogy (minden) környezetfüggő nyelvet generálni tudjunk.

A permutációs nyelvek esetében környezetfüggetlen szabályok mellé $AB \to BA$ alakú szabályokat véve nem környezetfüggetlen, környezetfüggő nyelveket is generálhatunk. (lásd, a következő alfejezetben: Permutációs nyelvek). Az előző kérdésre a választ a következő egyoldali normálforma adja meg:

19. Definíció. Egy környezetfüggő nyelvtanról azt mondjuk, hogy Penttonen normálformában van, ha minden szabályára illik az alábbi alakok egyike $A \to a$, $A \to BC$, $AB \to AC (a \in T, A, B, C \in N)$. ★

63. Tétel. Minden környezetfüggő nyelvtannal van ekvivalens olyan, amely Penttonen normálalakú.

A normálformák bevezetése során egyrészt a szabályok hosszának korlátozása volt fontos, másrészt a különböző típusú rekurziók kiküszöbölése, pl. egy (egyszerű és nem teljesen jól megírt) számítógépprogram könnyen végetlen ciklusba kerülhet, egy jól megírt program pedig sokkal összetettebb kell, hogy legyen, ha valamilyen rekurziót engednek meg a nyelvtan szabályai, hiszen az előforduló rekurziók némelyikét külön kezelni kell.

A környezetfüggetlen esetben a Chomsky-féle normálalak egyik nagy előnye, hogy a láncszabályok által okozott rekurzió (pl. $A \to B$, $B \to C$, $C \to A$) nem okozhat gondot, - minden lépésben vagy nő a mondatforma hossza, vagy terminálist vezetünk be, - ezért működik jól pl. a CYK algoritmus. A láncszabályok a környezetfüggő esetben is kiküszülhetőek, ahogy a Cremers és a Penttonen-féle normálformák sem tartalmaznak ilyet.

A Greibach normálforma Greibach normálforma a balrekurziót küszöböli ki, ezért ugyancsak fontos lehet néhány szóelemző algoritmus számára (legbalodali levezetés esetén).

Környezetfüggő esetben viszont, az eddig általunk legerősebbnek vélt normálforma is megenged rekurziót: a nem környezetfüggetlen szabályok esetén a mondatforma hossza nem változik.

Környezetfüggő rekurziónak nevezzük azt, ha csupán $AB \to AC$ alakú szabályok segítségével egy levezetés "körbe mehet", vagyis vannak olyan E és F nemterminálisok, hogy $EF \Rightarrow^{+} EF$, vagyis valahány (nem nulla) lépés után ismétlődés léphet fel. Egy ilyen egyszerű ismétlődést generálhat pl. az $AB \to AC$ és $AC \to AB$ szabályok együttes jelenléte.

Egy nyelvtant rekurzió-mentes normálformájúnak hívunk, ha Penttonen normálformájú és környezetfüggő rekurzió nincs benne. A következő tétel Nagy Benedek és Varga Péter nevéhez fűződik:

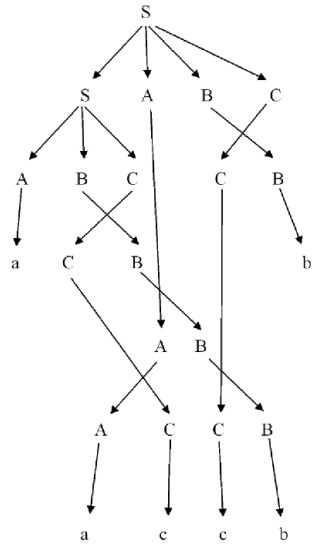
64. Tétel. Minden környezetfüggő nyelvtanhoz van vele ekvivalens, ami rekurzió-mentes normálformájú.

8.3. Permutációs nyelvek

Nézzük tehát az $A \to a$, $A \to B$, $A \to BC$, $AB \to BA$ alakú szabályokkal rendelkező nyelvtanokat: segítségükkel nem generálható minden környezetfüggő nyelv, de generálhatóak nem környezetfüggetlen nyelvek is, tehát az ilyen típusú szabályokkal generálható Permutációs nyelvek halmaza a Chomsky hierarchiában szigorúan a környezetfüggetlen és a környezetfüggő nyelvcsaládok közt helyezkedik el.

8.8. példa - Permutációs nyelvtan

Legyen G=({S, A, B, C}, {a, b, c}, S, {$S \to SABC$, $S \to ABC$, $AB \to BA$, $BA \to AB$, $AC \to CA$, $CA \to AC$, $BC \to CB$, $CB \to BC$, $A \to a$, $B \to b$, $C \to c$}), ekkor $L(G)$= {w | a w szóban az a, b és c betűk száma megegyezik (és nem 0) }. Belátható, hogy $L(G)$ nem környezetfüggetlen. ★



Mivel a permutációs szabályok ($AB \to BA$) alkalmazása esetén a mondatformában nem változik meg a szereplő (terminális és nemterminális) betűk száma, a többi szabály pedig környeztfüggetlen, minden levezetésnek van olyan megfelelője, amiben a permutációs szabályokat nem alkalmazzuk, a többi szabályt pedig az eredeti levezetésben található sorrendben alkalmazzuk. Az ilyen levezetés végén kapott szó betűekvivalens az eredetileg levezetett szóval. (Betűekvivalens két szó, ha minden betűből pont ugyanannyit tartalmaznak; két nyelv pedig akkor ha bármelyik nyelv bármely szavához van vele betűekvivalens szó a másik nyelvben.) Viszont a permutációs szabályok (használata) nélküli nyelvtan környezetfüggetlen nyelvet generál: tehát a permutációs nyelvtan által generált nyelvben mindig van egy az eredeti nyelvvel betűekvivalens környezetfüggetlen résznyelv. Az $a^n b^n c^n$ nyelvben minden szó csak saját magával betűvekvivalens, így nem teljesül rá az előzőekben ismertetett feltétel, tahát nem permutációs nyelv.

Ide tartozó nyitott probléma: vajon az $A \to B$ alakú láncszabályok kiküszöbölhetőek-e, vagyis generálható-e minden permutációs nyelv láncszabályok nélkül?

A szóprobléma megoldására hatékony algoritmus létezésének kérdése is nyitott kérdés, vagyis jelenleg nem tudjuk, hogy melyik bonyolultsági osztályhoz tartozik ez a probléma.

A permutációs nyelvtanok generáló ereje növekszik, ha kettő hosszúságú permutációs szabályok helyett három hosszúságúakat is megengedünk (pl. $ABC \to CBA$), vagyis az így kapott $Perm_3$ nyelvosztály szigorúan a $Perm$=$Perm_2$ és a környezetfüggő nyelvek osztálya között helyezkedik el a hierarchiában.

A permutációs nyelvek nem zártak a reguláris nyelvekkel való metszetképzésre (a példaként bemutatott ugyanannyi a, b és c betűt tartalmazó szavakból álló nyelv metszete a reguláris $a^* b^* c^*$ nyelvvel éppen $a^n b^n c^n$, a környezetfüggetlen nyelvek pedig zártak a reguláris nyelvekkel vett

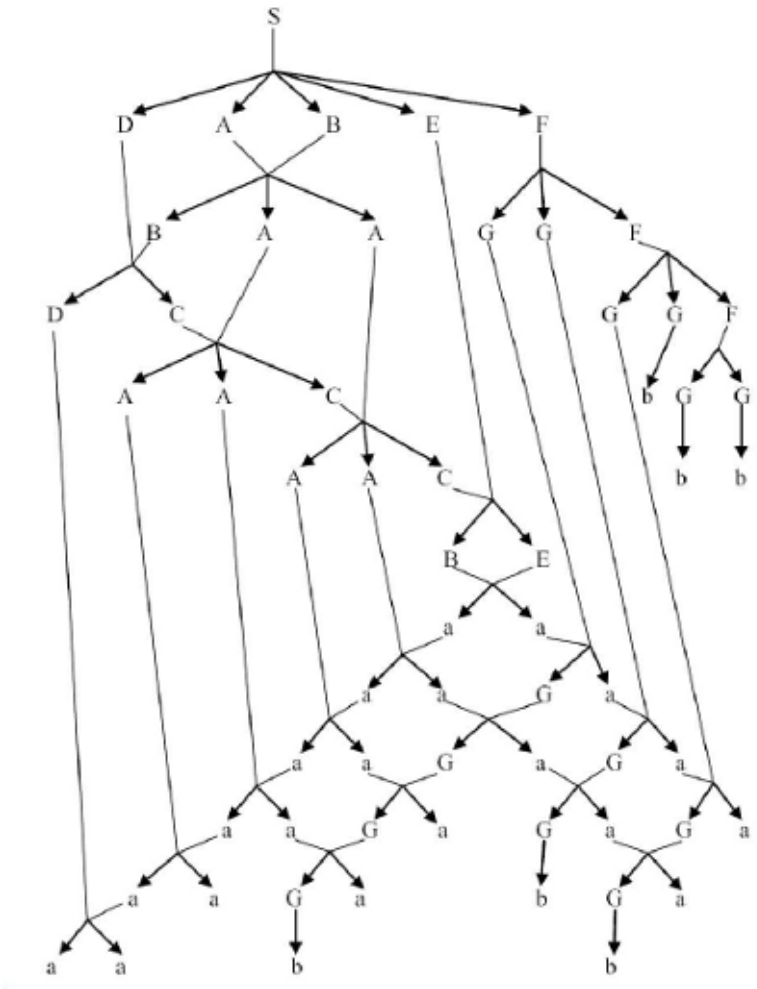
metszetképzésre). Egy permutációs és egy reguláris nyelv metszeteként előálló nyelvek osztálya a *Perm-reg* nyelvosztály szigorúan a *Perm* és a környezetfüggő nyelvek osztálya között helyezkedik el a hierarchiában.

8.4. Levezetési gráfok, levezetési fák a környezetfüggő nyelvtanokhoz

A környezetfüggő nyelvek esetén is ábrázolhatjuk a megfelelő nyelvtanokban a levezetéseket gráfok segítségével. Monoton esetben vegyük a következő példát:

8.9. példa - Levezetési gráf monoton esetben

Legyen $(\{S, A, B, C, D, E, F, G\}, \{a, b\}, S, H)$ egy monoton nyelvtan, ahol $H=\{S \to DABE, S \to DABEF, F \to GG, F \to GGF, G \to b, aG \to Ga, BE \to aa, Aa \to aa, Da \to aa, AB \to BAA, DB \to DC, CA \to AAC, CE \to BE\}$. A következő ábrán egy példaleveztést láthatunk: az *aaabaabbaabbb* szó levezetési gráfja:

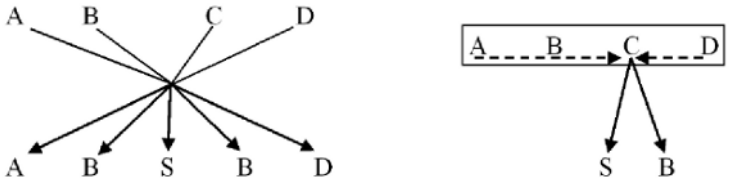


★

A monoton nyelvtanok esetén a levezetési gráf páros gráf, kétféle csúcstípussal: a levezetési fáknál szokásos betűcímkézett (nemterminális, terminális, illetve csak az üresszó levezetése esetén a λ); illetve szabálycímkézett csúcsok (itt a címkéket általában elhagyhatjuk, hiszen a befutó és kifutó élek által mutatott csúcsok címkéiből egyértelműen felírható az alkalmazott levezetési szabály).

Környezetfüggő nyelvtan esetén ez a gráf nagyon redundáns, ugyanis az összes környezetszimbólum megismétlésre kerül. Ezen segíthetünk, ha nem ismételjük meg őket, hanem speciális "környezet-

élek" (illetve környezetdoboz) segítségével jelöljük meg a gráfban, hogy az adott nemterminális azért helyettesíthető a megadott módon, mert a megfelelő környezet rendelkezésre áll. Ily módon csak a hagyományos levezetési fáknál megszokott csúcstípusra van szükségünk, viszont kétféle éltípusra. Példaként az $ABCD \rightarrow ABSBD$ szabály kétféle ábrázolása látható a következő ábrán:

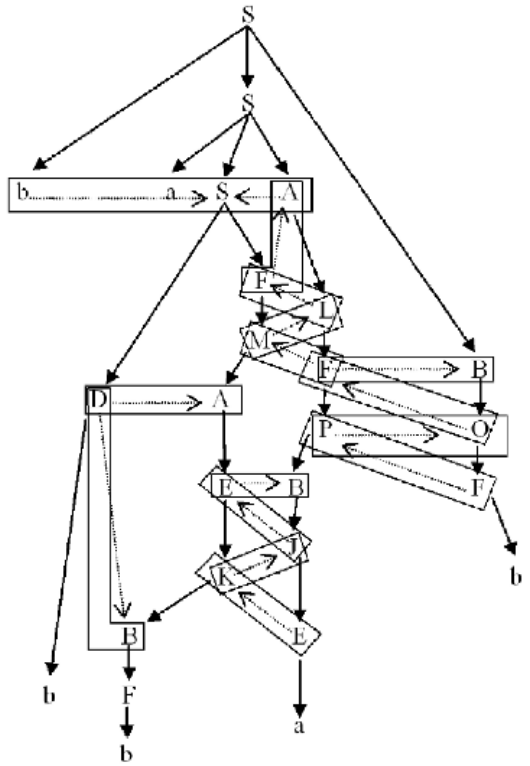


8.10. példa - Levezetési gráf környezetfüggő nyelvtanban

Legyen adott a $G=(\{S, A, B, C, D, E, F, G, I, J, K, L, M, O, P\}, \{a, b, c\}, S, H)$ környezetfüggő nyelvtan, ahol a H szabályhalmaz:

$H=\{S \rightarrow aSA, S \rightarrow bSB, abS \rightarrow abCE, baSA \rightarrow baDFA, EA \rightarrow EG, EG \rightarrow IG, IG \rightarrow IE, IE \rightarrow AE, EB \rightarrow EJ, EJ \rightarrow KJ, KJ \rightarrow KE, KE \rightarrow BE, FA \rightarrow FL, FL \rightarrow ML, ML \rightarrow MF, MF \rightarrow AF, FB \rightarrow FO, FO \rightarrow PO, PO \rightarrow PF, PF \rightarrow BF, CA \rightarrow CE, CB \rightarrow CF, DA \rightarrow DE, DB \rightarrow DF, C \rightarrow a, D \rightarrow b, E \rightarrow a, F \rightarrow b\}$.

A következő ábrán egy levezetést láthatunk.



★

Még speciálisabb a helyzet, ha csak Penttonen normálformájú nyelvtant engedünk meg, ekkor a környezet mindig csak egy baloldali szomszédos nemterminális lehet. Erre az esetre definiáljuk formálisan a környezetfüggő "levezetési fát":

20. Definíció. Legyen $G=(N, T, S, H)$ Penttonen normálformájú nyelvtan. Egy q mondatformához a levezetési fát a következőképpen írhatjuk le: irányított csúcscímkézett (a címkék az $(N \cup T)$ halmazbeliek) gráf. A levezetési élek (folytonosak az ábrán) hagyományos értelemben vett fa szerkezetet adnak a gráfnak (pontosan ahogyan a környezetfüggetlen esetben). A gyökér címkéje S.

Minden nem levél elem címkéje nemterminális. A levélelemek címkéit balról jobbra összeolvasva q- t kapjuk. Legyen A egy nem levélelem címkéje és legyen r a gyerekeinek a címkéi balról jobbra összeolvasva. Ekkor a következő feltételeknek kell teljesülnie:

- Ha környezetél (az ábrán szaggatott) nem érkezik be az adott A címkéjű csúcsba, akkor $A \to r$ alakú leveztési szabályt tartalmaz a G nyelvtan H szabályrendszere.

- Ha pontosan egy környezetél érkezik az adott A címkéjű csúcsba, akkor legyen annak a balszomszéd ágon levő csúcsnak ahonnan a környezetél indul a címkéje C, ekkor $CA \to Cr \in H$. ★

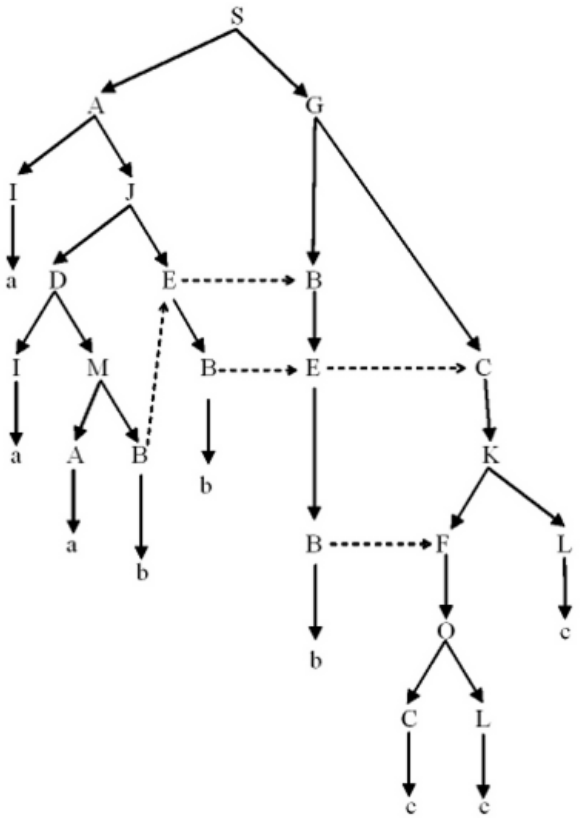
Minden környezetél két szomszédos ágat köt össze (balról jobbra), vagyis pontosan akkor mehet egy α (C címkéjű) csúcsból környezetél egy β (A címkéjű) csúcsba, ha a levezetési fában γ az a legalacsonyabban levő csúcs, amely mind az α, mind a β csúcsot dominálja (a legutolsó közös őse mindkettőnek); α a γ baloldali részfájának legjobboldali ágán van, míg β a γ jobboldali részfájának legbalodali ágán helyezkedik el (lásd a következő ábrát is). Ezen kívül a következő feltételnek is teljesülnie kell: a δ csúcs egyik őséből sem indulhat környezetél az ε csúcs egyik leszármazottjához sem, ha δ csúcsból indul környezetél az ε csúcsba. Egy csúcsból több környzezetél is indulhat, de maximum egy érkezhet.

8.11. példa - Környezetfüggő levezetési fa

Legyen adott a ($\{S, A, B, C, D, E, F, G, I, J, K, L, M, O\}$, $\{a, b, c\}$, S, H) környezetfüggő nyelvtan Penttonen normálformában:

H=$\{S \to AG, G \to BC, A \to IJ, J \to DE, EB \to EE, EC \to EK, K \to FL, D \to IM, M \to AB, BE \to BB, BF \to BO, O \to CL, A \to a, B \to b, C \to c, D \to a, E \to b, F \to c, I \to a, L \to c\}$.

Egy példalevezetést láthatunk a következő ábrán.



★

Figyeljük meg hogy a levezetési gráf szerkezete milyen egyszerű, a struktúrája az ebben a fejezetben korábban ismertetett gráfoknál tényleg sokkal közelebb áll a fához.

Az előzőek alapján, kicsit pongyolán, a következő feltételt mondhatjuk a környezetélekre: egy környezetél nem metszhet semmilyen más élt a gráfban.

Úgy is felfoghatjuk, hogy egy adott ágon levő csúcsra a balszomszéd ág "árnyékot" vet, azok a nemterminálisok jöhetnek szóba környezetél indítására egy adott csúcsba, melyeknek ott lehet az árnyéka. Egy levezetési fát befejezettnek nevezünk, ha minden levéleleme terminális.

8.4.1. Legbaloldalibb levezetés és átfogalmazása

Mint ahogy a környezetfüggetlen nyelvtanok esetén láttuk a legbalodali levezetés létezése fontos szerepet játszott, hiszen a veremautomata egy adott levezetési fa legbalodali levezetését szimulálta. A legbaloldalibb levezetést ott mondatformára definiáltuk, mindig a benne szereplő első (vagyis legbalodali) nemterminálisra alkalmaztunk egy alkalmazható levezetési szabályt. A környezetfügetlen esetben ez a definíció egybeesik azzal, hogy a levezetési fában mindig a legbaloldalibb még be nem fejezett ágon folytatjuk a levezetést, vagyis a fa továbbépítését.

Ismert, hogy bármilyen generatív nyelvtan (tehát bármilyen 0. típusú nyelvtan) esetén, ha csak legbaloldalibb levezetést engedünk meg (vagyis mindig csak olyan szabályt alkalmazhatunk, ami a legelső szereplő nemterminálisnál alkalmazható, és ott alkalmazzuk), akkor a generált nyelv környezetfüggetlen lesz.

Ezek alapján tehát ahhoz, hogy a generáló ereje egy (nem környezetfüggetlen) nyelvtannak ne csökkenjen, szükséges a legbalodali levezetés fogalmának általánosítása (átfogalmazása).

21. Definíció. Legbaloldalibb levezetésnek (vagy legbalodali konstrukciónak) nevezzük környezetfüggő esetben a (Penttonen normálformájú nyelvtan esetén a) levezetési fa felépítésének legbalodali módját: minden lépésben a legbalodali levélelemnél folytatjuk a leveztést (a levezetési fa megkonstruálását) innen induló levezetési él(ek) bevezetésével, esetlegesen felhasználva egy a gráfban már jelenlevő baloldali szomszédos csúcsot környezetél segítségével (a korábbiakban leírt feltételek betartásával). ★

65. Tétel. Legyen adott egy $G=(N, T, S, H)$ Penttonen normálformájú nyelvtan. Egy w szóhoz pontosan akkor létezik befejezett levezetési fa, illetve legbaloldalibb levezetés, ha $w \in L(G)$.

8.5. Árnyék-veremautomata

Ahogy az eredeti legbaloldalibb levezetés alapján a veremautomatát megkonstruálhattuk, úgy készítjük el az árnyék-veremautomatát a környezetfüggő legbaloldalibb levezetés esetére. Az árnyék-veremautomata vermében a hagyományos veremszimbólumok mellett, árnyékszimbólumok is lehetnek. Az automata olvashat a szalagról egy betűt, látja a veremben levő legfelső veremszimbólumot, illetve a közvetlenül ezen levő árnyékszimbólumhoz is hozzáférhet.

22. Definíció. Az (állapotnélküli nemdeterminisztikus) árnyék-veremautomata egy $SPDA=(T, Z \cup Z', X_0, d)$ rendezett négyes, ahol T az input ábécé, Z a veremábécé, $Z'=\{X' \mid X \in Z\}$ az árnyékszimbólumok halmaza, X_0 a kezdő veremszimbólum, $d:(T \cup \{\lambda\}) \times Z \times Z' \rightarrow (Z \cup Z')^*$ pedig az átmenetfüggvény; ahol a d átmenetfüggvény a következő átmeneteket írhatja le:

- $\lambda, A' \in d(a, A, \lambda)$, ez az átmenet csak akkor lehetséges ha nincs árnyékszimbólum az A felett a veremben, ekkor az automata elolvassa az a- t a szalagról és törli a verem tetején levő A- t, vagy kicseréli az árnyékára (A').

- $BC, BCA' \in d(\lambda, A, \lambda)$, ezekben az átmenetekben az A- ra rátesszük a C- t és a B- t (a B lesz a legfelső veremszimbólum), és az A- t töröljük, vagy az árnyékára cseréljük.

- $C, CB', A'C, A'CB' \in d(\lambda, B, A')$, ezekben az átmenetekben az A' árnyékszimbólumnak pont a legfelső verem szimbólumon, B- n kell lennie; ekkor B- re rátesszük a C- t (a C lesz a legfelső veremszimbólum) és a B- t töröljük, vagy az árnyékára cseréljük, illetve az A'- t is törölhetjük. ★

Az árnyék-veremautomata egy konfigurációja (w, z), ahol $w \in T^*$ a még feldolgozandó input, $z \in (Z \cup Z')^*$ pedig a verem aktuális tartalma. Akkor fogadunk el egy w szót, ha a (w, X_0) konfigurációból a megadott átmenetek alapján véges sok lépésben elérhető a (λ, λ) konfiguráció, vagyis elfogy az input és a verem kiürül. Belátható, hogy az automata a Penntonen normálforma alapján értelmezett legbaloldalibb levezetést szimulálja, így bizonyítható a következő tétel.

66. Tétel. Az állapotnélküli nemdeterminisztikus árnyék-veremautomaták pontosan a környezetfüggő nyelvek osztályát fogadják el.

8.12. példa - Árnyék-veremautomata és működése

Legyen $SPDA=(\{a,b,c\},\{S,A,B,C,D,E,F,G,I,J,K,L,M,O\} \cup \{S',A',B',C',D',E',F',G',I',J',K',L',M',O'\}, S, d)$, ahol d a következőképpen definiált:

$\lambda,A' \in d(a,A, \lambda)$,
$\lambda,B' \in d(b,B, \lambda)$,
$\lambda,C' \in d(c,C, \lambda)$,
$\lambda,D' \in d(a,D, \lambda)$,
$\lambda,E' \in d(b,E, \lambda)$,
$\lambda,F' \in d(c,F, \lambda)$,
$\lambda,I' \in d(a,I, \lambda)$,
$\lambda,L' \in d(c,L, \lambda)$,
$AG,AGS' \in d(\lambda,S, \lambda)$,
$BC,BCG' \in d(\lambda,G, \lambda)$,
$IJ,IJA' \in d(\lambda,A, \lambda)$,
$DE,DEJ' \in d(\lambda,J, \lambda)$,
$FL,FLK' \in d(\lambda,K, \lambda)$,
$IM,IMD' \in d(\lambda,D, \lambda)$,
$AB,ABM' \in d(\lambda,M, \lambda)$,
$CL,CLO' \in d(\lambda,O, \lambda)$,
$E,EB',E'E,E'EB' \in d(\lambda,B, E')$,
$K,KC',E'K,E'KC' \in d(\lambda,C, E')$,
$B,BE',B'B,B'BE' \in d(\lambda,E, B')$,
$O,OF',B'O,B'OF' \in d(\lambda,F, B')$.

Ekkor az *aaabbbccc* input szón lehetséges az *SPDA* következő futása: a konfigurációkat adjuk meg, az árnyékszimbólumokat (piros és lila), illetve az átmenetfüggvényben felhasznált inputbetűket (kék) és a verem tetején felhasznált betűket (kék veremszimbólum, illetve lila árnyékszimbólum) emeltük ki.

(aaabbbccc, S)
(aaabbbccc, AG)
(aaabbbccc, I JG)
(aabbbccc, JG)
(aabbbccc, DEG)
(aabbbccc, IMEG)
(abbbccc, MEG)
(abbbccc, ABEG)
(bbbccc, BEG)
(bbccc, B′EG)
(bbccc, BE′G)
(bccc, B′E′G)
(bccc, B′E′BC)
(bccc, B′EC)
(bccc, BE′C)
(ccc, B′E′C)
(ccc, B′ K)
(ccc, B′FL)
(ccc, OL)
(ccc, CLL)
(cc, LL)
(c, L)
(λ, λ)

Ez alapján a szót az automata elfogadja. ★

8.6. Lineárisan korlátozott automata

A következőkben egy másik, a szakirodalom által jól ismert olyan automata modellt mutatunk be, amely pontosan a környezetfüggő nyelveket fogadja el. A lineárisan korlátolt automatának (Linear Bounded Automaton (LBA), vagy lineárisan korlátozott Turing-gép) több változata is ismert, ezek közül ismertetünk egyet. Az automata egy véges vezérlővel rendelkezik és egy szalaggal, amelyen kezdetben az input szó áll. Az automata a működése során két lényeges eltérést mutat az eddig tárgyalt automatákhoz képest: az egyik, hogy a szalagon a fej előre és hátra is mozoghat, a másik, még lényegesebb, hogy nem csak olvashatja a szalagot, de annak tartalmát át is írhatja, ennek megfelelően nem olvasó, hanem író-olvasó fejről beszélünk.

23. Definíció. Az $LBA=(Q, T, V, q_0, \#, d, F)$- t (nemdetrminisztikus) lineárisan korlátozott automatának hívjuk, ha Q az állapotok véges halmaza, T az inputábécé, $V \supseteq T$ a szalagábécé, q_0 a kezdőállapot, $\# \in V \setminus T$ speciális jel a szalag azon részén, ahol nincs input (tulajdonképpen a szóköz jelnek feleltethető meg), d az átmenetfüggvény, $F \subseteq Q$ pedig a végállapotok halmaza. A d átmenetfüggvény (leképezés) alakja a következő: a $(Q \times (V \setminus \{\#\}))$ halmazból képez a $(Q \times V \times \{Bal, Jobb, Helyben\})$ részhalmazaiba, illetve $(Q \times \{\#\})$ halmazból képez a $(Q \times \# \times \{Bal, Jobb, Helyben\})$ részhalmazaiba. ★

Az automata egy konfigurációját (u, q, av) –ként írhatjuk le, ahol $u, v \in V^*$ a szalagon levő információ a fejtől balra, illetve jobbra, $q \in Q$ az aktuális állapot, $a \in V$ pedig a fej által éppen látott szimbólum a szalagon. Kezdetben az input # jelek között szerepel az input szalagon és a fej az input első betűjére van pozícionálva, vagyis a kezdeti konfiguráció (λ, q_0, aw') ahol az input $w=aw'$. (Üresszó input esetén $(\lambda, q_0, \#)$ a konfiguráció kezdetben.)

Az automata vezérlője (a d leképezés által leírt módon) az aktuális állapot és az éppen olvasott szimbólum alapján nemdeterminisztikusan választ a lehetséges átmenetek közül: $(q', b, m) \in d(q, a)$

jelentése: ha az automata q állapotban van és a szimbólumot lát a szalagon, akkor átmehet q' állapotba, miközben a szalagon a helyére b- t ír és a fej az $m \in \{Bal, Jobb, Helyben\}$ által megadott irányba lép egyet a szalagon (vagy értelemszerűen *Helyben* esetén helyben marad). Akkor mondjuk, hogy az automata elfogadott egy input szót, ha van az átmeneteknek olyan véges sorozata a szó feldolgozása során, hogy az automata végállapotba jut. Az elfogadott szavak halmaza adja az elfogadott (vagy felismert) nyelvet.

A definícióból látható, hogy a szalagon levő # jelek nem írhatóak felül, ennek megfelelően az automata csak az eredeti input által elfoglalt területet használhatja számolásra.

67. Tétel. A lineárisan korlátozott automatákkal elfogadott nyelvek osztálya megegyezik a környezetfüggő nyelvek osztályával.

A bizonyítás ötletét röviden mutatjuk be: Ha egy nyelv környezetfüggő, akkor monoton nyelvtannal generálható, vagyis a mondatforma hossza a levezetés egyik lépésében sem haladja meg a levezett szó hosszát. Az adott nyelvtan alapján megszerkeszthető egy olyan lineárisan korlátozott automata, amely éppen a szabályok alkalmazását szimulálja visszafelé. Ha adott egy lineárisan korlátozott automata, akkor készíthető egy olyan analitikus nyelvtan, amely pont az automatát szimulálja, a nemterminális éppen a fej helyét jelöli, és tárolja az aktuális állapotot. Ez alapján a duális, generatív nyelvtan is elkészíthető, ami monoton. ∎

Itt jegyezzük meg, hogy a nemdeterminisztikus lineárisan korlátozott automata az amely a környezetfüggő nyelvosztály elfogadására alkalmas, az pedig, hogy a determinisztikus lineárisan korlátozott automaták által felismert nyelvek osztálya valódi részhalmaza-e ennek egy e jegyzet megírása idején is fennálló nevezetes megoldatlan probléma.

Igazolható, hogy az ismertetett modellel ekvivalensek, vagyis ugyanezt a nyelvosztályt fogadják el azok a lineárisan korlátozott automaták, ahol az automatának egy előre rögzített k konstansszor annyi szalagpozíció (tárhely) áll rendelkezésre működése során, mint az input hossza (lásd Tárhely tétel [223]). Továbbá az is ismert (Savitch tétele (ejtsd: szévics)), hogy determinisztikus automatával, négyzetesen korlátolt tárral (vagyis, ahol az inputszó hosszának négyzetével arányos a megengedetten felhasználható szalagterület) minden környezetfüggő nyelv felismerhető.

8.7. A környezetfüggő nyelvek tulajdonságai

A környezetfüggő nyelvek halmaza tartalmaz több olyan nyelvet is, amely nem tesz eleget a konstansnövekmény elvének: pl. : $\{a^p \mid p$ prímszám $\}$, $\{a^k \mid k$ négyzetszám $\}$ vagy $\{a^k \mid k=2^i$ valamely i természetes számra $\}$. A környezetfüggő nyelvek osztályában szereplő nyelvekre így nincs általános pumpáló/iterációs lemma.

8.7.1. A szóprobléma

A környezetfüggő nyelvekre a szóprobléma eldönthető, vagyis meg lehet adni olyan algoritmust, ami véges idő alatt eldönti, hogy egy adott szó szerepel-e az adott monoton nyelvtan által generált nyelvben. Mivel a mondatforma hossza egy levezetés során nem csökkenhet, adott hosszig generálható, és így felsorolható az összes generálható mondatforma. Így ha az adott w szó hossza n, akkor $n+1$ hosszig felsorolva (levezetve) az összes levezethető szót eldönthető, hogy w eleme-e a generált nyelvnek, formálisan tehát:

68. Tétel. Bármely $G=(N, T, S, H)$ környezetfüggő grammatikáról és tetszőleges $w \in T^*$ szóról eldönthető, hogy $w \in L(G)$ fennáll-e.

Bizonyítás. Vegyük észre, hogy tetszőleges 1-típusú $G=(N, T, S, H)$ nyelvtan esetén az esetleges $S \to \lambda$ szabálytól eltekintve minden egyes szabály baloldalának hossza legfeljebb akkora mint a jobboldalé. Ráadásul, tekintettel arra, hogy az $S \to \lambda$ szabály fellépésekor S nem fordulhat elő egyetlen szabály jobboldalán sem, minden olyan esetben, amikor egy $S \Rightarrow W_1 \Rightarrow \ldots \Rightarrow W_t \Rightarrow w$ levezetés első lépésében nem az $S \to \lambda$ szabályt alkalmazzuk, az $S \to \lambda$ szabályt egyetlen

további lépésben sem tudjuk alkalmazni. Tehát egy $S \Rightarrow W_1 \Rightarrow \ldots \Rightarrow W_t \Rightarrow w$ levezetés vagy $S \Rightarrow \lambda$ alakú, vagy pedig $|S| \leq |W_1| \leq \ldots \leq |W_t| \leq |w|$ fenn fog állni. Tehát a levezetés egyetlen lépése sem eredményezhet $|w|$- nél hosszabb $(N \cup T)^*$- beli szót (mondatformát). Nyilvánvaló, hogy ha egy szó G- ben levezethető, akkor levezethető oly módon, hogy a levezetés során nincs ismétlődő mondatforma. Képletben, ha $(S=)W_0 \Rightarrow W_1 \Rightarrow \ldots \Rightarrow W_t \Rightarrow w$ mellett $W_i = W_j$ telesül valamely $0 \leq i < j \leq t$ párra (a levezetés utolsó lépéseként adódó T^*- beli szó a levezetés definíciója értelmében nem ismétlődhet), akkor $(S=)W_0 \Rightarrow \ldots \Rightarrow W_i \Rightarrow W_{j+1} \Rightarrow \ldots \Rightarrow W_t \Rightarrow w$ is fennáll (ahol $j=t$ esetén a $W_{j+1} \Rightarrow \ldots \Rightarrow W_t$ lépések értelemszerüen elmaradnak). Az ilyen ismétlődéseket nem tartalmazó levezetések száma véges. Nevezetesen, (eltekintve az egy lépéses $S \Rightarrow \lambda$ esettől) nem nagyobb mint a $N \cup T$ ábécé feletti, $|w|$-nél nem hosszabb szavakból álló ismétlődés nélküli szósorozatok hossza, vagyis $\sum_{i=1}^{|w|} |N \cup T|^i$ Vagyis, ha el akarjuk dönteni a $w \in L(G)$ kérdést, $w=\lambda$ esetén meg kell vizsgálnunk, hogy $S \rightarrow \lambda$ szerepel-e a szabályok között, $w \neq \lambda$ esetén pedig meg kell vizsgálnunk azt, hogy az ismétlődés nélküli, $|w|$- nél hosszabb szavakat nem tartalmazó (véges sok, $\sum_{i=1}^{|w|} |N \cup T|^i$-nál nem nagyobb számú) levezetések közt van-e olyan, melynek az utolsó lépéseként w adódik. Ha igen, $w \in L(G)$, különben pedig $w \notin L(G)$. Ezzel bizonyításunk végéhez értünk. ∎

Itt hívjuk fel a figyelmet, hogy a rekurzió-mentes normálformájú nyelvtanok esetén eleve kizárható, hogy ismétlődő mondatforma lépjen fel egy levezetés során. Ha a lehetséges levezetéseket a mesterséges intelligenciában állapottér gráfnak nevezett gráffal reprezentáljuk, akkor a rekurzió-mentes normálforma esetén a gráf tehát körmentes lesz, így akár egy visszalépéses kereső, aminek lépésszámkorlátja a mondatforma hosszának függvénye, is képes a szóprobléma eldöntésére. A szóproblémára a környezetfüggő esetben nincs (nem ismert) hatékony algoritmus, a probléma, hogy tetszőleges környezetfüggő nyelvet generáló (akár valamely ismertetett normálformában levő) nyelvtan generál-e egy adott szót általánosan PSPACE-teljes probléma (vagyis hiába elég legfeljebb négyzetes tár a determinisztikus, illetve lineáris tár a nemdeterminisztikus esetben, a probléma a legbonyolultabb problémák közt van, melyek megoldásához polinomiális tárra van szükség (lásd Bonyolultsági osztályok)).

8.7.2. Zártsági tulajdonságok

69. Tétel. A környezetfüggő nyelvek osztálya zárt a reguláris műveletekre.

Bizonyítás. Konstruktív: legyen adott L_1 és L_2 környezetfüggő nyelvek. Legyen adva (N_1, T, S_1, H_1) és (N_2, T, S_2, H_2) két Kuroda normálformájú nyelvtan, amely L_1 és L_2 üresszómentes részét generálja, ahol N_1 és N_2 diszjunktak.

Konstruáljuk meg a $(N_1 \cup N_2 \cup \{S\}, T, S, H)$ nyelvtant, ahol S új szimbólum, nem szerepel sem N_1, sem N_2 elemei közt, H pedig a következőképpen definiált: $H=H_1 \cup H_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}$, illetve legyen $S \rightarrow \lambda$ benne a H- ban pontosan akkor ha az L_1 és L_2 nyelvek legalább egyike tartalmazza az üresszót. Könnyen belátható, hogy az új nyelvtan megfelel a monoton nyelvtan definíciójának és éppen L_1 és L_2 unióját generálja.

Tekintsük most a $(N_1 \cup N_2 \cup \{S\}, T, S, H)$ nyelvtant, ahol S új szimbólum, nem szerepel sem N_1, sem N_2 elemei közt, H pedig a következőképpen definiált:

$H=H_1 \cup H_2 \cup \{S \rightarrow S_1 S_2\}$, ha sem L_1 sem L_2 nem tartalmazza az üresszót,

$H=H_1 \cup H_2 \cup \{S \rightarrow S_1 S_2, S \rightarrow S_1\}$ ha L_2 tartalmazza az üresszót, de L_1 nem,

$H=H_1 \cup H_2 \cup \{S \rightarrow S_1 S_2, S \rightarrow S_2\}$ ha L_1 tartalmazza az üresszót és L_2 nem,

$H=H_1 \cup H_2 \cup \{S \rightarrow S_1 S_2, S \rightarrow S_1, S \rightarrow S_2, S \rightarrow \lambda\}$ ha az L_1 és L_2 nyelvek mindegyike tartalmazza az üresszót.

Könnyen belátható, hogy az új nyelvtan megfelel a monoton nyelvtan definíciójának és éppen L_1 és L_2 konkatenációját generálja. Legyen most L környezetfüggő nyelv és legyen adva (N_1, T, S_1, H_1) és

(N_2, T, S_2, H_2) két Kuroda normálformájú nyelvtan, amelyek L üresszómentes részét generálják, ahol N_1 és N_2 diszjunktak.

Tekintsük most a ($N_1 \cup N_2 \cup \{S, S'\}$, T, S, H) nyelvtant, ahol S és S' új szimbólumok, egyikük sem szerepel sem N_1, sem N_2 elemei közt, és $H=H_1 \cup H_2 \cup \{S \rightarrow \lambda, S \rightarrow S_1, S \rightarrow S_1S_2, S \rightarrow S_1S_2S', S' \rightarrow S_1, S' \rightarrow S_1S_2, S' \rightarrow S_1S_2S'\}$. Az így megkonstruált nyelvtan monoton és az L^* nyelvet generálja. ∎

70. Tétel. A környezetfüggő nyelvek osztálya zárt a halmazműveletekre: az unió műveleten kívül a metszet és komplementer műveletekre nézve is.

8.8. Növekvő környezetfüggő nyelvek

E fejezetben végezetül a környezetfüggő nyelvek egy speciális osztályát mutatjuk be.

Növekvő környezetfüggő (GCS: Growing Context-Sensitive) egy $G=(N, T, S, H)$ nyelvtan, ha S nem szerepelhet egyetlen szabály jobb oldalán sem, és ha minden (nem $S \rightarrow p$ alakú, ahol $p \in (N \cup T)^*$) szabályára teljesül, hogy ha $p \rightarrow q \in H$, akkor $|p| < |q|$.

8.13. példa - Egy növekvő környezetfüggő nyelv

Legyen $G=(\{S, A, B, F, H, I, L\}, \{a\}, S, \{S \rightarrow a, S \rightarrow aa, S \rightarrow aaaa, S \rightarrow FHAL, FH \rightarrow FBH, BHA \rightarrow BBBH, BHL \rightarrow BBIL, IL \rightarrow IAL, BIA \rightarrow IAAA, FBI \rightarrow FHAA, FH \rightarrow aaH, aHA \rightarrow aaaH, aHL \rightarrow aaaa, IL \rightarrow Iaa, BIa \rightarrow Iaaa, FBa \rightarrow aaaa\})$. Ez a nyelvtan növekvő környezetfüggő és pontosan azokat a szavakat generálja az $\{a\}$ egybetűs ábécé felett, amelyek hossza a 2- nek valamilyen egész kitevős hatványa. Ez a nyelv köztudottan nem tesz eleget a konstansnövekmény tulajdonságnak. ★

A növekvő környezetfüggő nyelvek osztálya szigorúan részhalmaza a környezetfüggő nyelvek osztályának, ezekre a nyelvekre a szóprobléma determinisztikus polinomiális időben eldönthető. A növekvő környezetfüggő nyelvek osztálya halmazelméleti szempontból nem összemérhető a permutációs nyelvekkel a tartalmazási (részhalmaz) relációra nézve.

8.9. Irodalmi megjegyzések

A Kuroda normálforma bevezetése [Kuroda 1964]-ben jelent meg. A környezetfüggő egyéb normálformákkal kapcsolatos eredmények nagy része az 1970-es évekből származik: a Cremers-féle normálforma [Cremers 1973], a Penttonen-féle [Penttonen 1974] jelent meg. A Révész-féle egyoldali normálforma megtalálható [Révész 1983,1989] könyvekben. A rekurziómentes normálforma [Nagy, Varga 2009]-ben jelent meg. A levezetési fák és a legbaloldalibb levezetés [Nagy 2006b, 2010b], az árnyékveremautomata bevezetése pedig [Nagy 2006c, 2010c] cikkekben történt. A lineárisan korlátozott automata bevezetése és annak bizonyítása, hogy éppen a környezetfüggő nyelvosztályt tudják elfogadni szintén [Kuroda 1964]-ben jelent meg. A permutációs nyelvekkel kapcsolatos eredmények [Mäkinen 1985], [Nagy 2009a] jelentek meg. Annak a bizonyítása, hogy a 3 hosszúságú permutációs szabályokkal több nyelv generálható, mint a csak 2 hosszúságúakat felhasználva [Nagy 2010a]-ban található. A permutációs és reguláris nyelvek metszeteként előálló nyelveket vizsgálja a [Nagy 2009b, 2011a]. A növekvő környezetfüggő nyelvekkel kapcsolatos eredményeket pl. [Buntrock, Otto 1995] tartalmaz.

9. fejezet - Rekurzívan felsorolható nyelvek és Turing-gépek

Ebben a fejezetben a mondatszerkezetű nyelvtanok által generált nyelvosztályt fogjuk megvizsgálni, vagyis azon nyelvek osztályát, amelyek generatív nyelvtannal generálhatóak.

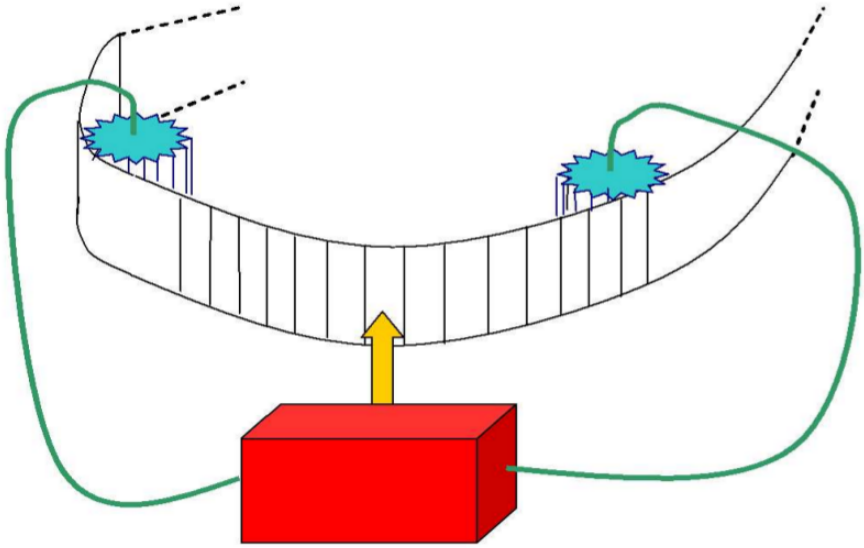
A mondatszerkezetű nyelvtanok esetén a levezetéseket gráffal tudjuk szemléltetni hasonlóan a környezetfüggő, illetve a monoton esethez. Ezeket a továbbiakban nem részletezzük. Megjegyezzük viszont, hogy a gráf kinézete speciális lehet, ha különböző normálformát követelünk meg a nyelvtantól.

A fejezetben, ahogy a címben is szerepel, központi szerepet játszik a Turing-gép fogalma.

9.1. A Turing-gép

A *Turing-gép* (TM: Turing Machine) fogalmát Alan Turing (ejtsd: tyuring) vezette be bizonyos automatikusan végrehajtható számítások tanulmányozására 1936-ban, jóval az első programvezérlésű elektronikus számítógépek megjelenése előtt. Egyszerűsége ellenére a Turing-gép elég jó modellnek bizonyult bizonyos számítógépekkel kapcsolatos vizsgálatokban, így például a számítógépek számítási kapacitásának elvi korlátai kutatásában.

A Turing-gép egy potenciálisan végtelen *szalagmemóriával* és egy *író-olvasó fejjel* ellátott véges automata. A szalagmemória *pozíciókra* van osztva, s minden egyes pozíció mint memória-egység az úgynevezett *szalagábécé* pontosan egy betűjének tárolására képes. Kezdetben a Turing-gép egy specifikált kezdőállapotában van, s a szalagon egy véges hosszúságú *input szó* helyezkedik el. Az eddig tárgyalt automatákhoz hasonlóan ez a modell is szekvenciális működésű. Működésének kezdetekor a Turing-gép író-olvasó feje az input szó első betűjén áll. Az input szó előtti és utáni (végtelen sok) szalagpozíció egy speciális betűvel, a *szóközzel* (üres betűvel) van feltöltve, ami nem tévesztendő össze az üresszóval. Többek között azért is, hogy az input szó elkülöníthető lehessen a szalag többi részén tárolt mindkét irányban végtelen számú szóköztől, feltételezzük, hogy az input szó utolsó betűje nem lehet szóköz. Az input szó tehát az író-olvasó fej alatti betűtől (jobbra haladva) tart a szalag utolsó nem üres betűjéig. Speciálisan, üres input szó is elképzelhető. Ez esetben a szalag minden egyes pozíciója szóközzel van feltöltve, és az író-olvasó fej ezek egyikére mutat. (Utolsó szóköztől különböző betű pedig ekkor értelemszerűen nincs.) A Turing-gép diszkrét időskála mentén, elkülönített időpillanatokban hajt végre egy-egy elemi műveletet, mely az író-olvasó fej alatti betű olvasásából, ezen betű felülírásából, a belső állapot változtatásából, s az író-olvasó fej egy pozícióval való balra avagy jobbra mozgatásából, vagy éppen a fej helybenhagyásából áll. Amennyiben a Turing-gép eljut egy végállapotba, megáll.



Formálisan, a Turing-gép egy $TM=(Q, T, V, q_0, \#, d, F)$ rendezett hetes, ahol Q a gép *belső állapotainak* (véges) halmaza, $q_0 \in Q$ a *kezdő állapot, V* a *szalagábécé*, $T \subseteq V$ az *inputábécé* $\# \in (V \setminus T)$ a *szóköz betű, F*$\subseteq Q$ a *végállapotok halmaza,* $d: Q \times V \to 2^{Q \times V \times \{Bal,\ Jobb,\ Helyben\}}$ a gép *mozgásfüggvénye,* mint szokásos, $2^{Q \times V \times \{Bal,\ Jobb,\ Helyben\}}$ jelöli a $Q \times V \times \{Bal, Jobb, Helyben\}$ halmaz összes részhalmazainak halmazát.

Ha tehát a *TM* Turing-gép egy $q \in Q$ állapotában van és az író-olvasó fej alatt valamely $a \in V$ jel áll, akkor- ha $d(q, a)$ nem üres - a $d(q, a)$- beli hármasok egyikeszolgáltatja a gép operáció utáni új állapotát, a szalagjelet felülíró szimbólumot (mely nem feltétlen különböző a felülírt szimbólumtól), illetve az elmozdulás irányát. Ha $d(q, a)=\emptyset$, azt úgy interpretáljuk, hogy ha a gép a q állapotban azíró-olvasó fej alatt az a betűt találja, további működésétfelfüggeszti (megáll).

Megjegyezzük, hogy bár a gép szalagját mindkét irányban végtelennek tekintjük, mindig csak véges sok #- tól különböző jel lehet rajta.

Vegyük észre, hogy a környezetfüggő nyelveknél definiált és tárgyalt lineárisan korlátozott automata (lásd LBA), tulajdonképpen a Turing-gép egy olyan változata, ahol a számításra fordítható szalagterület az input által lefoglalt területre korlátozódik. Ily módon a számítás bonyolultsága van korlátozva (lásd Bonyolultsági osztályok).

Az egyszerűbb írásmód kedvéért a továbbiakban fel fogjuk tételezni azt az egyébként megfelelő írásmóddal elérhető, ám jelentéktelen megszorítást, hogy $Q \cap V=\emptyset$. Egy *pillanatnyi konfiguráció* (u, q, av) alakú, ahol $a \in V \cup \{\lambda\}$, $u, v \in V^*$, $q \in Q$ és $u \in V^+$ esetén u nem kezdődhet, $v \in V^+$ esetén pedig v nem végződhet szóközzel. (Természetesen $u=\lambda$, $v=\lambda$, $uv=\lambda$ bármelyike előfordulhat.) Az (u, q, av) konfiguráció azt jelenti, hogy a gép q belső állapotban van, miközben a feje éppen egy a jel felett áll, miközben a szalag "értelmes tartalma" éppen uav, vagyis a szalagon u áll a fejtől balra és v jobbra (természetesen a bevezető, illetve folytató szóköz jeleket nem számítva).

Ha $q=q_0$ és $u=\lambda$ akkor *kezdőkonfigurációról*, ha pedig $q \in F$, akkor *végkonfigurációról* beszélünk. (Sok esetben nem is szokták elkülöníteni a Turing-gép végállapotait a többi állapottól. Ilyenkor végkonfiguráció alatt azokat a (u, q, av), konfigurációkat szokásérteni, ahol $d(q, a)=\emptyset$. Ilyenkor a számítás "eredménye" a szalagról olvasható le a megálláskor.) Amennyiben valamely (u, q, av), $q \in Q \setminus F$ pillanatnyi konfiguráció esetén $\{(q, a, Helyben)\}=d(q, a)$, akkor azt mondjuk, hogy a *TM* Turing-gép a tekintett pillanatnyi konfigurációban *egyszerű végtelen ciklusba esik.*

Minden egyes $W=(ub, q, av)$, $q \in Q$, $u, v \in V^*$, $a \in V$, $b \in (V \setminus \{\#\}) \cup \{\lambda\}$ pillanatnyi konfigurációhoz (ahol ha $ub \neq \lambda$, akkor ub nem kezdődhet, ha pedig $v \neq \lambda$, akkor av nem végződhet szóközzel) rendeljük hozzá a következőképp definiált $\varphi(W)$ kifejezést.

$\varphi(W)=$	$ubqav$,	ha $a, b\in V\setminus\{\#\}$
	$ubq\#$,	ha $b\in V$, $a=\#$, $v=\lambda$
	$\#qav$,	ha $ub=\lambda$, $a\in V\setminus\{\#\}$
	$\#q\#$,	ha $ub=v=\lambda$, $a=\#$.

Világos, hogy ez a hozzárendelés kölcsönösen egyértelmű. Mondjuk azt, hogy a W_1 pillanatnyi konfigurációból a W_2 pillanatnyikonfiguráció (TM- *ben*) *közvetlenül* (*vagy egy lépésben*) *levezethető*(jelekben $W_1\vdash_{TM}W_2$, vagy ha egyértelmű mely *TM* gépről van szó, egyszerűbben $W_1\vdash W_2$),ha $W_1=(ub, q, av)$ eseténa következő feltételek valamelyike teljesül.

(I) $W_2=(u, q', ba'v)$ és $(q', a', Bal)\in d(q, a)$,

(II) $W_2=(uba', q', v)$ és $(q', a', Jobb)\in d(q, a)$,

(III) $W_2=(ub, q', a'v)$ és $(q', a', Helyben)\in d(q, a)$.

A W pillanatnyi konfigurációból a W' pillanatnyi konfiguráció(a *TM* -ben) *levezethető* (jelekben $W\vdash_{TM}^{*}W'$, vagy röviden $W\vdash^{*}W'$), ha van a pillanatnyikonfigurációknak olyan W_0, …, W_n sorozata, hogy $W_i\vdash W_{i+1}$, $i=0$, …, n-1 mellett $W_0=W$, $W_n=W'$. Speciálisan, minden W pillanatnyi konfigurációrafeltesszük $W\vdash^{*}W$ fennállását. Szokásosan, ha ki akarjuk hangsúlyozni,hogy $W\vdash^{*}W'$ és $W\neq W'$, használjuk a $W\vdash^{+}W'$ jelölést is. A *TM* Turing gép által elfogadottnyelven értjük az

$L(TM)=\{w\in T^{*}\mid(\lambda, q_0, w)\Rightarrow^{*}W, W$ végkonfiguráció $\}$

nyelvet.

11. *Megjegyzés.* Jelölje C_{TM} a $TM=(Q, T, V, q_0, \#, d, F)$ Turing-gép összes pillanatnyi konfigurációinak halmazát. Vegyük észre, hogy a fentiekben tekintett *TM* Turing-gép által elfogadott nyelv egybeesik a következő *G* nyelvtan által elfogadott nyelvvel: $G=(Q, T, q_0, H)$, ahol $H=H_1\cup H_2$ és $H_1=\{q\to\lambda\mid q\in F\}$, $H_2=\{bqa\to\varphi(W)\mid q\in Q, b, a\in(V\setminus\{\#\})\cup\{\lambda\}, W\in C_{TM}, (b, q, a)\vdash W\}$.

Tekintsünk egy $TM=(Q, T, V, q_0, \#, d, F)$ és egy $TM'=(Q', T', V', q'_0, \#', d', F')$ Turing-gépet.Akkor mondjuk, hogy *TM izomorf TM'*- vel, haaz állapotok és a szalagábécé betűinek alkalmas kölcsönösen egyértelmű átjelölésével a két gép meg fog egyezni. Formálisan, ha létezik olyan $\varphi_1:Q\to Q'$ és $\varphi_2:V\to V'$ kölcsönösen egyértelmű leképezés-pár, hogy fennállnak a következők:

(i) $\varphi_1(q_0)=q'_0$, $\varphi_2(\#)=\#'$;

(ii) minden $q\in Q$- ra $q\in F$ akkor és csak akkor ha $\varphi_1(q)\in F'$;

(iii) valahányszor $(p, b,$ Irány $)\in d(q, a)$, mindannyiszor $(\varphi_1(p), \varphi_2(b),$ Irány $)\in d'(\varphi_1(q), \varphi_2(a))$ és viszont.

Érvényes a következő tétel:

71. Tétel. A mondatszerkezetű nyelvek osztálya egybeesik a Turing gépek által elfogadott nyelvek osztályával.

Amennyiben a mozgásfüggvény képhalmaza a $Q\times V\times\{Bal, Jobb, Helyben\}$ halmaz (és nem annak részhalmazainak halmaza), akkor determinisztikus Turing gépről beszélünk, és érvényes a következő tétel.

72. Tétel. A determinisztikus Turing gépek által elfogadott nyelvek osztálya egybeesik a nemdeterminisztikus Turing gépek által elfogadott nyelvek osztályával.

A Turing-gépeknek több változata is ismert, most ezek közül mutatunk be néhányat.

Van olyan definíció, ahol a fej a $\{Bal, Jobb\}$ irányokba léphet, és nem maradhat helyben. Belátható, hogy egy ilyen Turing-gép szimulálni tudja az eddigiekben ismertetett változatnak a fejet helyben

hagyó lépéseit is: pl. egyet balra lép a fej, és egy olyan állapotba kerül az automata, amiben bármit is olvas a szalagon, azt nem változtatja meg, viszont jobbra visszalép és az eredeti automata állapotának megfelelelő állapotba kerül.

Ugyancsak szokásos a csak *egyirányban végtelen szalagú Turing-gép* használata, amelyik szintén képes az általános változat szimulációjára. Ekkor a szalag első karaktere egy speciális jel, amiből a gép rájön, hogy erre nem mehet tovább a fej. Ekkor egy olyan speciális állapotba kerül, aminek hatására jobbra lép, először leírja azt a jelet, ami eredetileg a speciális szimbólum helyére írt volna (ha a szalag mindkétirányban végtelen lenne), majd az itt olvasott karaktert eggyel jobbra, és így tovább, vagyis a szalag teljes (értelmes) tartalmát eggyel jobbra másolja, ezután (észlelve a felhasznált tárterület jobb szélét), a fej vissza mozog a baloldalra, aholis a gép folytatja az eredetileg tervezett számítását.

Sokszor az egyszerűbb leírás kedvéért többszalagos Turing-gépet használunk:

$TM_k=(k, Q, T, V, q_0, \#, d, F)$ k- szalagos Turing-gép, ahol $k\in\mathbb{N}$ természetes szám, ennyi szalagja van aTuring-gépnek; és a d átmenetfüggvény alakja a következő: $d:Q\times V^k \rightarrow Q\times V^k\times\{Bal, Jobb, Helyben\}^k$.

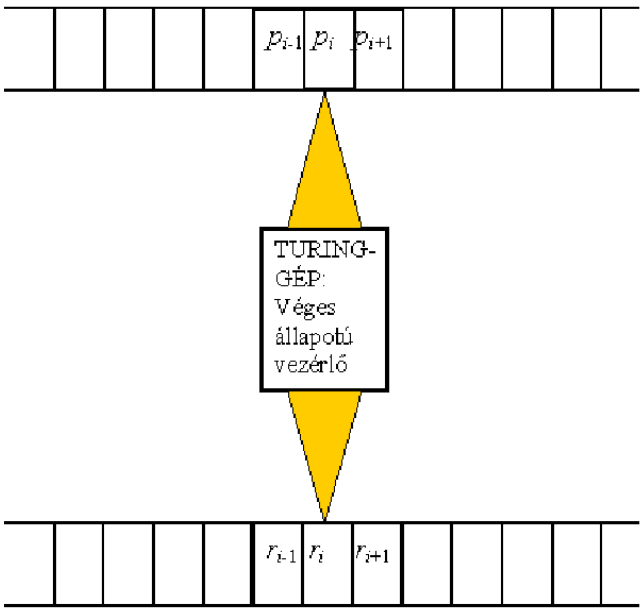
Működését tekintve a többszalagos Turing-gép egy lépésben olvashat/írhat egyszerre több szalagra is. Kezdő konfigurációban az egyik szalagon (input-szalag) van a feldolgozandó adat, a többi szalag pedig üres. Többszalagos gépek esetén szokás egy szalagot az outputnak is fenntartani, ekkor a számítás végén azon a szalagon olvasható az eredmény, illetve sokszor az input szalag csak olvasható.

Minden többszalagos Turing-gép működése szimulálható egyszalagos Turing-géppel, vagyis egyszalagos Turing-gép is el tudja végezni azt a számítást amit egy többszalagos Turing-gép.

Megkülönböztethetünk kiszámító és eldöntő Turing-gépeket a következő definíció alapján.

Amennyiben a Turing-gép célja adott függvény kiszámítása a megadott bemenő értékekkel, akkor a gép a megállásakor az (output)szalagon a megfelelő eredményt hagyja. Ezzel szemben vannak olyan számítások, amikor a választ egy igen-nem kérdésre keressük, ezesetben eldöntő Turing-gépről beszélünk. Az eldöntő Turing-gépekkel lehet pl. egy L nyelvet elfogadtatni a következőképpen: bemenet egy $w\in T^*$ szó, a Turing-gép számításának eredménye pontosan akkor "igen" ha $w\in L$ teljesül. (Ugyanez a hatás érhető el, ha csak akkor engedjük végállapotba jutni a gépet, ha elfogad.)

A következő ábrán egy kétszalagos Turing-gép vázlata látható.



A továbbiakban, ha mást nem mondunk, Turing-gép alatt determinisztikus Turing-gépet értünk. A példákban a fej mozgását jelentő szavakat azok kezdőbetűivel rövidítjük.

9.1. példa - Turing gépek 1.feladat

Adjunk meg olyan Turing-gépet, amely az $\{a,b\}$ feletti tükörszavakat ismeri fel!

Megoldás:

Állapodjunk meg abban, hogy a kezdő konfigurációban az input szó első betűjén áll az író/olvasó fej, és az input szó előtt és után "#" van.

Ha a Turing gép q_i állapotban t-t olvas, v-t ír, átmegy q_j állapotba és m irányba tovább lépteti a fejet, azaz $(q_j,v,m)\in d(q_i,t)$, akkor jelöljük ezt (q_i,t,v,q_j,m) ötössel! A fej mozgásának irányát pedig annak kezdőbetűjével rövidítjük.

Legyen a $T=(\{q_0,q_1,q_2,q_3,q_4,q_5,q_a\},\{a,b\},\{a,b,\#\},q_0,\#,\delta,\{q_a\})$, ahol δ a következő:

$(q_0,a,\#,q_1,J)$ Ha az első betű a, akkor q_1 állapotba megy a gép
$(q_0,b,\#,q_2,J)$ Ha az első betű b, akkor q_2 állapotba megy a gép
$(q_0,\#,\#,q_a,J)$ Ha az első betű # (az üresszó van a szalagon), akkor q_a állapotba megy a gép

(q_1,a,a,q_1,J)
(q_1,b,b,q_1,J)
$(q_1,\#,\#,q_3,B)$ Végig megy a gép az input szón, majd annak utolsó betűjére áll q_3 állapottal

(q_2,a,a,q_1,J)
(q_2,b,b,q_1,J)
$(q_2,\#,\#,q_4,B)$ Végig megy a gép az INPUT szón, majd annak utolsó betűjére áll q_4 állapottal

$(q_3,a,\#,q_5,B)$ Ha az utolsó betű a, akkor töröljük, és mehet a gép a szó elejére q_5 állapottal

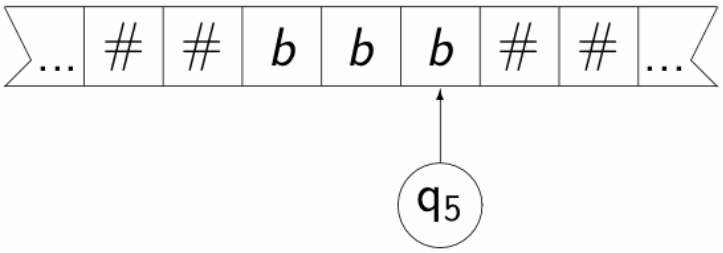
$(q_4,b,\#,q_5,B)$ Ha az utolsó betű b, akkor töröljük, és mehet a gép a szó elejére q_5 állapottal

$(q_3,\#,\#,q_a,B)$
$(q_4,\#,\#,q_a,B)$ Ha az input szó páratlan hosszú volt, akkor elfogadjuk

(q_5,a,a,q_5,B)
(q_5,b,b,q_5,B)
$(q_5,\#,\#,q_0,J)$ Újra a szó elejére és a kezdőállapotba megyünk!

A tükörszavakat felismerő Turing-gépet működés közben mutatja a következő ábra:

Tükörszavak



A Turing gép az $\{a, b\}$ feletti tükörszavakat ismeri fel.

$$T = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_a\}, \{a, b\}, \{a, b, \#\}, q_0, \#, \delta, \{q_a\}).$$

★

9.2. példa - Turing gépek 2.feladat

Adjunk meg olyan Turing-gépet, amely az $L=\{A^{2^n} \mid n \geq 0\}$ nyelvet ismeri fel!

Megoldás:

Ezt a feladatot más oldalról közelítjük meg, mint elsőre az logikusnak tünne!
Első körben az A-k számát ábrázoljuk bináris formában, majd megnézzük,
hogy a kapott szám csak 1 darab 1-es számjegyet és utána esetleg 0-kat tartalmaz-e.
$T=(\{q_0,q_1,q_2,q_3,q_4,q_a\},\{A\},\{A,X,0,1,\#\},q_0,\#,\delta,\{q_a\})$, ahol δ a következő:

(q_0,A,X,q_1,B) Az első A-t átírjuk X-re, majd eggyel balra lépünk

$(q_1,\#,1,q_2,J)$
$(q_1,0,1,q_2,J)$
$(q_1,1,0,q_1,B)$ A legbaloldalabbi X előtti bináris számot növeljük 1-gyel, majd q_2 állapotba megyünk

$(q_2,1,1,q_2,J)$
$(q_2,0,0,q_2,J)$
(q_2,X,X,q_2,J)
(q_2,A,A,q_0,H) Megkeressük a legbaloldalabbi A-t, majd átmegyünk a kezdőállapotba

$(q_2,\#,\#,q_3,B)$ Nincs több A, a fejtől balra csak X-ek és egy bináris számunk van. Átmegy a gép q_3 állapotba.

$(q_3,X,\#,q_3,B)$ Töröljük az X-eket

$(q_3,0,0,q_3,B)$ A 0-kon átmegyünk q_3 állapottal

$(q_3,1,1,q_4,B)$ Ha egy 1-est találunk, átmegyünk q_4-be

$(q_4,\#,\#,q_a,H)$ Ha az 1-es előtt # van, akkor elfogadjuk a szót! ★

9.3. példa - Turing gépek 3.feladat

Adjunk meg olyan Turing-gépet, amely az $L=\{a^n b^n c^n | \ n \geq 0\}$ nyelvet ismeri fel!

Megoldás:
Az első *a*-t átírjuk *A*-vá, majd más állapotba megyünk, az első *b*-ig.
Az első *b*-t átírjuk *B*-vé, majd más állapotba megyünk, az első *c*-ig.
Az első *c*-t átírjuk *C*-vé, majd más állapotba megyünk visszafelé az első *a*-ig.
Az *A,B,C* betűkön csak tovább megyünk mindig.
Ha a végén csak # marad, akkor felismeri a gép a szót!
Legyen $T_3=(\{q_0, q_1, q_2, q_3, q_a\},\{a,b,c\}, \{a,b,c,A,B,C,\#\},q_0,\#,\delta,\{q_a\})$, ahol δ a következő:

$(q_0, \#, \#, q_a, B)$ üresszón állunk q_0-nál, akkor a beolvasott szót felismeri a gép

(q_0, a, A, q_1, J) a legbaloldalibb *a*-t átírjuk *A*-vá, és átmegyünk q_1-be

(q_0, B, B, q_0, J)
(q_0, C, C, q_0, J) a *B* és *C* betűkön csak jobbra átmegyünk q_0-lal

(q_1, a, a, q_1, J)
(q_1, B, B, q_1, J) az *a*-kon és a *B*-ken jobbra haladva átmegyünk az első *b*-ig

(q_1, b, B, q_2, J) a legbaloldalibb *b*-t átírjuk *B*-vé, és átmegyünk q_2-be

(q_2, b, b, q_2, J)
(q_2, C, C, q_2, J) a *b*-ken és a *C*-ken jobbra haladva átmegyünk az első *c*-ig

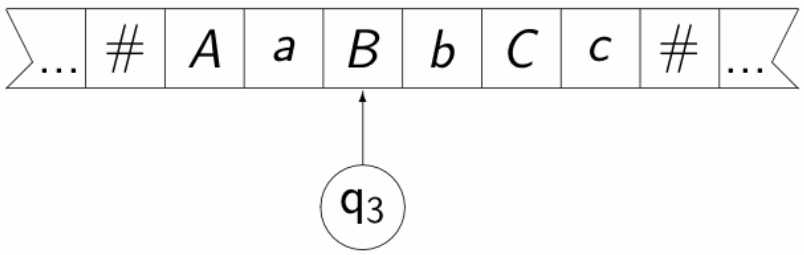
(q_2, c, C, q_3, B) a legbaloldalibb *c*-t átírjuk *C*-vé, és átmegyünk q_3-ba, majd balra lépünk egyet

(q_3, a, a, q_3, B)
(q_3, b, b, q_3, B)
(q_3, c, c, q_3, B)
(q_3, B, B, q_3, B)
(q_3, C, C, q_3, B) q_3 állapottal elmegyünk a legbaloldalibb *A*-ig, majd jobbra lépünk 1-et

(q_3, A, A, q_0, J) az első *A*-t követő karakteren állunk kezdőállapotban.

Az $L=\{a^n b^n c^n | \ n \geq 0\}$ nyelvet felismerő Turing-gép működés közben látható:

$a^n b^n c^n$



A Turing gép az $L = \{a^n b^n c^n \mid n \geq 0\}$ nyelv szavait ismeri fel.

$T = (\{q_0, q_1, q_2, q_3, q_a\}, \{a, b, c\}, \{a, b, c, A, B, C, \#\}, q_0, \#, \delta, \{q_a\})$.

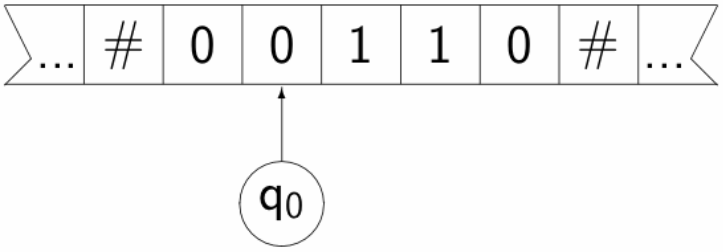
★

9.4. példa - Turing gépek 4. feladat

Készítsünk olyan Turing-gépet, amely egy bináris szám kettes komplemensét állítja elő.

Megoldás:

Kettes komplemens



A Turing gép az adott bináris szám kettes komplemensét adja meg.

$$T = (\{q_0, q_1, q_v\}, \{0, 1\}, \{0, 1, \#\}, q_0, \#, \delta, \{q_v\}),$$

$\delta(q_0, 1) = (q_0, 0, J)$,
$\delta(q_0, 0) = (q_0, 1, J)$,
$\delta(q_0, \#) = (q_1, \#, B)$,
$\delta(q_1, 1) = (q_1, 0, B)$,
$\delta(q_1, 0) = (q_v, 1, H)$.

★

9.2. A Turing-gépek megállási problémája és az algoritmikusan eldönthetetlen feladatosztályok kapcsolata

A matematikában és az informatikában gyakran foglalkozunk igen/nem problémaosztályokkal, vagyis olyan problémák osztályával, amelyeknél a megoldás mindig vagy "igen," vagy "nem." Ezeknél azt vizsgáljuk, hogy létezik-e algoritmus (kiszámítási mód) az illető osztály összes problémájának megoldására. *Church tézise* szerint egy Turing-géppel ekvivalens erejű számítási modellel, a *parciális rekurzív függvénnyel* kiszámítható feladatok tekinthetők effektíve kiszámíthatónak. Ezek szerint *a Turing-gép a lehető legáltalánosabb számítási eszköz, azaz minden, ami effektíve kiszámítható, kiszámítható Turing-géppel is.* Ez utóbbi átfogalmazást *Church-Turing tézisnek* is hívják a szakirodalomban. A Church tézist, miszerint *az effektíve kiszámítható függvények osztálya megegyezik a parciális rekurzív függvények osztályával,* Alonzo Church fogalmazta meg 1936-ban. (Még azt is hozzátette, hogy aki ezzel nem értene egyet, annak ez a tézis legyen kihívás.) Ezt a tézist, illetve annak ekvivalens megfelelőit, így a Church-Turing tézist is ma a szakemberek többsége elfogadja. Tézisről, tehát egy olyan nem bizonyított állításról van szó, melynek igazolása formális matematikai eszközökkel nem lehetséges. Érvényességét a gyakorlat verifikálja.

A Church-Turing tézis értelmében egy igen/nem problémaosztályt *megoldhatónak* hívjuk, ha létezik olyan rögzített algoritmus (Turing-gép), mely az osztály tetszőleges problémája mint bemenő adat esetén eredményként megadja a helyes "igen" vagy "nem" választ. 1936-ban Turing azt az akkor meglepő eredményt kapta, hogy létezik ebben az értelemben megoldhatatlan feladatosztály. Az eredeti bizonyítás helyett mi Minsky 1967-ben közölt bizonyítását tárgyaljuk.

Akkor mondjuk, hogy egy Turing-gép valamely input szó hatására *megáll,* ha az input szó eleme a tekintett Turing-gép által felismert nyelvnek, azaz az input szóhoz tartozó kezdő konfigurációból kiindulva eljut egy végkonfigurációba. Mondjuk azt, hogy a *Turing-gépek megállási problémája megoldható,* ha létezik olyan *TM'* Turing-gép, melynek egy alkalmas kódolási algoritmussal egy tetszőleges *TM* Turing-gép leírását és a *TM* gép egy *w* input szavának kódolt alakját input szóként megadva megáll, s megállva a *TM'* szalagján olyan szó keletkezik, melyre alkalmas dekódolási eljárást alkalmazva egyértelműen megállapítható, hogy $w \in L_{TM'}$, avagy sem. Más szóval, egyértelműen megállapítható, hogy a leírt *TM* Turing-gép a kérdéses *w* szó mint input szó hatására el tud-e jutni egy végkonfigurációba, avagy sem. Ha ilyen *TM'* Turing-gép nem létezik, akkor mondjuk azt, hogy *a Turing-gépek megállási problémája megoldhatatlan.*

73. Tétel. (Turing tétel) A Turing-gépek megállási problémája megoldhatatlan.

Bizonyítás. Először eltekintünk a bekódolás kérdésének vizsgálatától, s feltesszük, hogy létezik olyan általános kódolási algoritmus, mely tetszőleges *TM* Turing-gép és annak *w* input szava esetén megad a gép egy $l(TM)$ leírását és a *w* által meghatározott alkalmas $c(l(TM), w)$ kódolt alakot mint egy rögzített ábécé feletti szót.

A bizonyítás indirekt. Tegyük fel, hogy létezik olyan *TM'* Turing-gép, mely a $c(l(TM), w)$ szót input szóként megkapva

(a) megáll és ekkor az "IGEN" válasz kódolt alakja olvasható a szalagján, ha *TM* megálla *w* inputra,

(b) megáll és ekkor a "NEM" válasz kódolt alakja olvasható a szalagján, ha *TM* nem állmeg a *w* inputra.

Ha *TM'* létezik, megszerkeszthető az a *TM''* Turing-gép, mely az $l(TM)$ input szó hatására előállítja a $c(l(TM), l(TM))$ input szót,

majd ezután erre az input szóra a *TM'* Turing-gép működését utánozza egyetlen módosítással: valahányszor a *TM'* igen megállást ér el, a *TM''* gép egyszerű végtelen ciklusba esik. Figyelembe véve a *TM'* eredeti viselkedését, kapjuk:

TM '' az $l(TM'')$ input szó hatására pontosan akkor áll meg, ha *TM* '' a $l(TM'')$ input szó hatására nem áll meg. Ez nyilvánvaló ellentmondás, amivel (feltételezve, hogy létezik az univerzális kódolási eljárás,) a tétel igazolást nyert. ∎

74. Tétel. Létezik univerzális algoritmus, mely izomorfizmustól eltekintve egyértelműen megadja bármely Turing-gép és annak input szava leírását.

Bizonyítás. A bizonyítás során alkalmazott meggondolást Gödel-számozásnak, a kódolt alakot pedig a Turing-gép Gödel számának hívjuk. Megjegyezzük, hogy ismeretesek ezen módszernél jóval hatékonyabb univerzális kódolási módszerek is. (Az ismertetett módszert Gödel eredetileg axiómarendszerek vizsgálatára használta fel.)

Ismeretes, hogy minden természetes szám sorrendtől eltekintve egyértelműen felírható prímhatvány tényezős alakba (azaz páronként különböző prímszámok hatványainak szorzataként). Ezt a tényt fogjuk felhasználni. Tekintsünk egy $TM=(Q, T, V, q_0, \#, d, F)$ Turing-gépet, s legyen $q_0, \ldots, q_{m-1}$ az állapotok egy olyan (kezdőállapottal kezdődő) felsorolása, ahol alkalmas $0 \le k \le m-1$ mellett $\{q_0, \ldots, q_k\}=Q\setminus F$. Jelölje továbbá $a_1, \ldots, a_n$ a szalagábécé betűinek egy felsorolását, s végül legyen $D_1=(q_0, a_1), \ldots, D_{k+1}=(q_k, a_1), D_{k+2}=(q_0, a_2), \ldots, D_{2(k+1)}=(q_k, a_2), \ldots, D_{n(k+1)}=(q_k, a_n)$.

(Megtesszük azt a nem túl lényeges megjegyzést, hogy a $q_0, \ldots, q_k$ és az $a_1, \ldots, a_n$ felsorolások már egyértelműenmeghatározzák a $D_1, \ldots, D_{n(k+1)}$ felsorolást. Utóbbi alkalmazását csupán a módszer könnyebb megértése kedvéért vezettük be.)

Képezzük az *TM* Turing-géphez és ezekhez az elrendezésekhez a

$$g_{TM} = 2^m \cdot 3^{k+1} \cdot 5^n \cdot 7^{u_1} \cdot 11^{v_1} \cdot 13^{w_1} \cdot \ldots \cdot p_{n(k+1)+1}^{u_{n(k+1)}} \cdot p_{n(k+1)+2}^{u_{n(k+1)}} \cdot p_{n(k+1)+3}^{u_{n(k+1)}}$$

(tetszőlegesen rögzített számrendszerbeli, például tízes számrendszerbeli) természetes számot, aholis 2 hatványa az állapotok számát, 3 hatványa a nem-végállapotok számát, 5 hatványa a szalagábécé betűinek számát jelöli, s minden további $p_{i+1}^{u_i}$, $p_{i+2}^{v_i}$, $p_{i+3}^{w_i}$, $i=1, \ldots, n(k+1)$ prímhatvány-hármas esetén, aholis alkalmas $q_s \in \{q_0, \ldots, q_k\}(=Q\setminus F)$, $a_t \in \{a_1, \ldots, a_n\}$ mellett $D_i=(q_s, a_t)$ áll fenn, $(u_i, v_i, w_i)=(0, 0, 0)$ ha $d(q_s, a_t)$ nincs értelmezve, ha pedig $d(q_s, a_t)$ értelmezve van, akkor $d(q_s, a_t)=(q_{s'}, a_{t'},$ Merre) esetén $u_i=s'+1$, $v_i=t'$, továbbá, mondjuk, $w_i=1$ ha Merre=*Bal*, $w_i=2$ ha Merre=*Jobb*, illetve $w_i=3$ ha Merre=*Helyben*. Az így meghatározott g_{TM} számot a *TM* Turing-gép *Gödel-számának* fogjuk hívni. Amennyiben a w input szó $w=a_{i_1}\ldots a_{i_f}$ alakú, a $c(l(TM), w)$ kódolt alak legyen (a rögzített számrendszerbeli)

$$g_{TM}(p_{n(k+1)+4})^{i_1}\ldots(p_{n(k+1)+d})^{i_f}$$

természetes szám. Látható, hogy a szóban forgó kódolás (izomorfizmustól eltekintve) egyértelmű, s ha a nyert $c(l(TM), w)$ természetes szám például tízes számrendszerbenvan megadva, akkor egy alkalmas tizenegy bemenő jeles gépnek inputként megadható (tizenegyedik bemenő jel a szóköz jel). ∎

9.2.1. Univerzális Turing-gép

Ebben a részben a Turing-gépeknek egy speciális fajtájával, az ún. univerzális Turing-géppel fogunk foglalkozni.

Az univerzális Turing-gép többféleképpen is megadható, mi itt most az egyik ilyen megvalósítást mutatjuk be.

Az *UTM* Turing-gépet univerzálisnak nevezzük a Turing-gépek osztályára nézve, ha minden *TM* Turing-géphez van olyan $v \in T^*$, hogy minden $s \in T^*$- ra a *TM* futásának eredménye az *s* inputon megegyezik az *UTM* futásának eredményével a *vXs* inputon (ahol $X \in V\setminus T$).

A *TM* "programja" *v* (az *UTM* nyelvén), *s* pedig egy tetszőlegesszó. Ha *TM* az *s* inputot kapja, akkor ugyanazt csinálja, mint *UTM* a *v* programmal az *s*- en.

Az univerzális Turing-gép tulajdonképpen egy általános, elvont számítógép, ami minden Turing-gépet képes szimulálni, vagyis elvileg a programjának megfelelően feldolgozni az input szót. Ez azt jelenti, hogy van olyan gép, ami minden kiszámítható függvényt ki tud számolni.

Az alábbiakban egy példát adunk az univerzális Turing-gépre.

Tulajdonképpen a *d* átmenetfüggvényt kell megadnunk ami a *Q*×*V* véges értelmezési tartományon és a *Q*×*V*×{*Bal*, *Jobb*, *Helyben*} véges értékkészleten van értelmezve, illetve egy másik Turing-gép leírásának (program) kódolását.

Legyen *TM* valamilyen Turing-gép, melynek *n* darab belső állapotavan, és a szalagábécéje *m* betűt tartalmaz. Tegyük fel, hogy *TM* -etkódolt formában adtuk meg (jelölje [TM] ezt a leírást). A továbbiakban vázlatosan ismertetjük, hogy hogyan tudjuk modellezni *TM* működését egy *UTM* univerzális Turing-géppel. Tegyük fel, hogy *TM* a *v* inputon (ennek kódja [*v*])dolgozik. Először azt kell megadnunk, hogy adott [TM] és [*v*] esetén ezeket az információkat miképpen tároljuk az *UTM* szalagján (azaz mi lesz *UTM* kezdőkonfigurációja), majd pedig azt, hogy ezek hatására *UTM* hogyan fog működni.

Jelöljünk ki a szalagon egy mezőt, és írjunk ebbe a mezőbe egy rögzített *X* jelet a szalagábécéből. A szalagnak a kiválasztottmezőtől jobbra eső felét három részre osztjuk. Az első részt nevezzük pufferterületnek; ez közvetlenül az *X* jel utánkezdődik, legalább *n*+*m*+2 mezőt tartalmaz, és ezek mindegyikébe 0van írva. A pufferterülettől jobbra eső szalagrész legelső mezőjébe egy *Y* jelet teszünk a szalagábécéből, utána beírjuk *TM* kódolt formáját, [TM]-et, három 0-t téve a végére. A szalagnak ezen részét nevezzük *TM* kódolási területének. A harmadik rész a második résztől jobbra helyezkedik el. Az első mezőjébe egy rögzített *Z* jelet írunk a szalagábécéből, majd a *v* bemeneti szó [*v*] kódolása következik. A szalagnak e három részen kívül eső mezői (kezdetben) üresek.

A pufferterület arra szolgál, hogy mialatt *TM* valamelyik lépését szimuláljuk, ide másolhassuk *TM* pillanatnyi belső állapotának, illetve az éppen leolvasott *TM* -beli szalagjelnek a kódját. Az *Y* jel általában az előtt a rendezett ötös előtt fog állni, amely azt határozza meg, hogy milyen belső állapotban van *TM* , milyen szalagjelet olvasunk éppen, mivel kell ezeket kicserélni (új állapot és új szalagjel), s eközben milyen irányban mozduljon el *TM* olvasófeje a szalagon. A *Z* jel az *TM* szalagjára felírt jelekközül jelöli ki azt, amelyiket éppen olvasunk.

Az *UTM* -ben lezajló számítási folyamatot, mellyel a *TM* Turing-gép működését szimuláljuk az *v* bemeneti szóval, olyan szakaszokrabonthatjuk, melyek sorra megfelelnek a *TM* egyes konfigurációi közötti átmeneteknek.

Az *UTM* működésének egy ilyen szakasza az alábbi módon zajlik le. Az *UTM* univerzális Turing-gép először a pufferterület elejére másolja azt az 1-esekből álló blokkot, amely közvetlenül az *Y* jel utánkövetkezik - nevezzük ezt *Y*- blokknak -, majd a végére odaír mégegy *X* jelet. Ezután kitörli *Y*- t, és jobbra haladva megkeresi a *Z*- t tartalmazó mezőt. Amikor ezt megtalálta, akkor a *Z* utánkövetkező 1-esekből álló blokkot (*Z*- blokkot) is átmásolja apufferterületre az előbb beírt második *X* jel után, majdvisszaírja *Y*- t a *TM* kódolt leírása, [TM] elé. Így apufferterületre az aktuális belső állapot és a szalagról éppen beolvasott jel kódja került. A következő lépésekben *UTM* az *Y* jelután következő két 1-es blokkot hasonlítja össze a pufferterületen levőkkel. Ezáltal azt ellenőrzi, hogy a *TM* gép soron következő konfigurációátmenetét az a rendezett ötös határozza-e meg, amelynek kódja az *Y* jel után van leírva. Ha a blokkokmegegyeznek, ez azt jelenti, hogy megtaláltuk a keresett ötöst. Ha nem, akkor *UTM* áthelyezi az *Y* jelet a következő rendezett ötöskódolása elé, majd újrakezdi a blokkok összehasonlítását. Abban az esetben, ha a *TM* leírásában szereplő ötösök közül egyik sem felel meg, *UTM* leáll a működésével (az eredeti *TM* is ugyanezt tenné a *v* inputra). Ha viszont megtaláljuk a keresett ötöst, akkor *UTM* kitörli a pufferterületet, majd az *Y* jelet átteszi az ötösben szereplőharmadik elem elé. Ezután kicseréli a *Z* után következő blokkotaz *Y* utáni blokkal, majd *Y*- t jobbra mozdítja el a rendezettötös negyedik elem elé. Miután *UTM* leolvasta ezt a negyedik elemet is, mely a *TM* olvasófejének elmozdulási irányát határozza meg, *UTM* átteszi a jelet az elem mögé, az ötödik elem elé. Attól függően, hogy a negyedik blokkban két vagy csak egy darab 1-est talált-e, az *UTM* egy blokkal jobbra vagy egy blokkal balra tolja el *Z*- t. Ha *Z* eredetileg a szalagszó bal szélén volt, és *TM* -nek balra kellettlépnie, akkor *UTM* a szó kódolását jobbra tolja, és egy üres mező kódjelét írja be a *Z* után. Ha pedig *Z* a szalagszó jobb szélénállt, s jobbra kellene elmozgatni, akkor *UTM* a szó

végére írja egy üres mező kódját. Amikor tehát mindezzel végeztünk, az *Y* jelután álló 1-es blokk a *TM* aktuális belső állapotát jelzi, a *Z* utáni blokk pedig azt a szalagjelet, amelyet *TM* -nek a következő lépésben be kellene olvasnia. Minden készen áll tehát arra, hogy a *TM* következő lépését szimuláló szakasz megkezdődhessen.

Az *UTM* működésének egyes szakaszai így *TM* egy-egy lépését modellezik. *UTM* ezeken kívül még a következőket hajtja végre: a munka legelején a szalag mindhárom részében a 0-kat a saját üres-jeleire cseréli, a munka végeztével pedig, olyankor, amikor *TM* leállna, *UTM* még ellenőrzi, hogy *TM* -nek végállapota-e az az állapot, amelyben megállt, és ettől függően kerül saját maga is végállapotba, ill. nem végállapotba.

Minden Turing-gép működése szimulálható olyan Turing-géppel, amiben a szalagábécé bináris, vagyis $\Sigma=\{0, 1\}$.

Az Univerzális Turing-gép létezése azt mutatja, hogy elvileg konstruálható olyan számítási eszköz, amely programozható és mindent ki tud számítani, ami kiszámítható. A gyakorlati megvalósulás felé a következő lépés a Neumann elv, amit a következő alfejezetben ismétlünk át.

Az absztrakt számítógép után lássuk a valódi gépek milyen ezzel nagyon rokon elveken működnek.

9.2.1.1. A Neumann-elv

A hagyományos számítógépek atyjának tekinthetjük Neumann Jánost, aki sok más tudományos tevékenysége mellett, a klasszikus számítógépek működésének alapelveit is megadta.

Ezek az elvek, amelyeknek megfelelően épült a legtöbb számítógép, a következőek:

- A program legyen a belső memóriában (tárolt program elve): A programot alkotó utasítások kifejezhetők számokkal, azaz adatként kezelhetők. Ezek a belső memóriában tárolhatók, mint bármelyik más adat. Ezáltal a számítógép önállóan képes működni, hiszen az adatokat és az utasításokat egyaránt a memóriából veszi elő. A memória a numerikus adatokkal együtt tárolja a programot is, a vezérlőegység pedig végrehajtja az utasítások sorozatát.
- A számítógép használja a kettes számrendszert és legyen teljesen elektronikus: A kettes számrendszert és a rajta értelmezett aritmetikai ill. logikai műveleteket könnyű megvalósítani kétállapotú áramkörökkel (pl.: 1- magasabb feszültség, 0 - alacsonyabb feszültség).
- A számítógép legyen soros (szekvenciális) működésű: A gép az egyes utasításokat egymás után, egyenként hajtsa végre.
- A számítógépnek legyen belső memóriája: A számítógép gyors működése miatt nincs lehetőség arra, hogy minden egyes lépés után a kezelő beavatkozzon a számítás menetébe. A belső memóriában tárolhatók az adatok és az egyes számítások részeredményei, így a gép bizonyos műveletsorokat automatikusan el tud végezni.
- A számítógép legyen univerzális: A számítógép különféle feladatainak elvégzéséhez nem kell speciális berendezéseket készíteni. Ugyanis Turing bebizonyította, hogy az olyan gép, amely el tud végezni néhány alapvető műveletet, elvileg bármilyen számítás elvégzésére is alkalmas (Turing-gép).

Láthatjuk, hogy a Turing-gép jó összhangban van a Neumann elvvel és ezért méltán tekinthető a számítógépek elméleti modelljének.

9.3. A szóprobléma - rekurzív és rekurzívan felsorolható nyelvek

A 0-típusú nyelvek esetén a $w \in L$ reláció általában algoritmikusan nem eldönthető. Ez a fejezetezt a problémakört vizsgálja meg részletesebben. (Lásd a A Turing-gépek megállási problémája és az algoritmikusan eldönthetetlen feladatosztályok kapcsolata fejezetet is.)

Egy L nyelvet *rekurzív*nak nevezünk, ha a $w \in L$ tartalmazásiprobléma algoritmikusan eldönthető.

Egy L nyelvet *rekurzívan felsorolható*nak nevezünk, ha van olyan eljárás, amely az összes $w \in L$ szót valamilyen sorrendben (esetleg ismétlésekkel) felsorolja.

12. *Megjegyzés.* Minden rekurzív nyelv nyilván rekurzívan felsorolható. Nem kell ugyanis mást tennünk, mint rendre megvizsgálni az összes $w \in T^*$ szót alkalmazva rájuk az eldöntési algoritmust, és egy w szót beleveszünk a felsorolásba, ha igen választ kapunk, egyébként elhagyjuk.

75. Tétel. Egy L nyelv akkor és csak akkor rekurzív, ha mind az L mind az $\overline{L}$ rekurzívan felsorolható.

Bizonyítás. Ha L rekurzív, akkor a $w \in L$ probléma algoritmikusan eldönthető, akkorugyanez áll az $\overline{L}$ nyelvre is, hiszen $w \in L$ akkor és csak akkor teljesül, ha $w \notin \overline{L}$. Eszerint ugyanazt az eldöntési algoritmust használhatjuk az $\overline{L}$- re, azzal a különbséggel, hogy amit L esetén elfogadtunk azt most nem, és fordítva.

Másik irány: Tegyük fel, hogy mind az L mind az $\overline{L}$ rekurzívan felsorolható. Kombináljuk az L és az $\overline{L}$ felsorolását biztosító eljárásokat úgy, hogy váltakozva hol az egyikkel, hol a másikkal állítunk elő egy-egy szót, miáltal egy olyan $w_0, w_1, \ldots$ felsorolást kapunk, ahol $w_{2i} \in L$, $w_{2i+1} \in \overline{L}$ minden i=0, 1, 2… értékre. Mivel afelsorolás teljes, ezért a w keresett szónak valahol elő kell fordulnia, így csak azt kell eldöntenünk, hogy páros vagy páratlan pozícióban fordul-e elő, így tulajdonképpen egy döntési algoritmust adtunk meg az L- re.∎

76. Tétel. Minden 1-típusú nyelv rekurzív, és minden 0-típusú nyelv rekurzívan felsorolható.

77. Tétel. Van olyan rekurzív nyelv, amely nem 1-típusú.

Bizonyítás. Minden 1-típusú nyelvtant megadhatunk úgy, hogy felsoroljuk a szabályait. Feltehetjük, hogy a nyelvtanban terminálisok csak $A \to a$ alakú szabályban fordulnak elő ($A \in N$, $a \in T$). A nemterminális ábécét a szabályok implicit módon definiálják. Ezek után tekintsük azt az 1-típusú nyelvtant, amelynek terminális ábécéje T={0, 1}. Kódoljuk a nemterminális jeleket a 01, 011, 0111, … jelsorozatokkal, ahol az S kezdőszimbólumnak mindig a 01 kód feleljen meg. Kódoljuk továbbá a 0 és az 1 terminális jeleket 00 és 001 szavakkal, a $\to$ és a # (elválasztó-) jeleket 0011, illetve 00111 szavakkal. Ezek alapján minden nyelvtant kifejezhetjük egy $\{0, 1\}^*$- beli szóval. Rendezzük az összes $\{0, 1\}^*$- beli szót valamilyen módon (ezzel a nyelvtanokat is sorbarendeztük).

Legyen w_i a $\{0, 1\}^*$ megadott rendezés értelmében az i- edik eleme a $\{0, 1\}^*$- nak és definiáljuk az L nyelvet úgy, hogy

$L=\{w_i \mid w_i \notin L(G_i)\}$

ahol G_i az i- edik nyelvtant jelenti. Az L nyelv rekurzív, mert a $w \in L$ véges sok lépésben eldönthető. Ugyanis egy adott w- ről eldönthetjük, hogy hányadik eleme a rendezett halmaznak. Legyen ez i, akkor meghatározzuk a G_i- t, ami úgy történik, hogy rendre előállítjuk a $\{0, 1\}^*$- beli szavakat (célszerűenugyanabban a rendezési sorrendben, mint amit az előbb használtunk), és mindegyikről egyenként eldöntjük, hogy ez a kódolás megfelel-e egy 1-típusú nyelvtannak vagy sem. Ez szintén megtehető véges lépésben. Miután meghatároztuk a G_i- t, eldönthetjük, hogy $w \in L(G_i)$ teljesül-e, ami szinténvéges számú lépésben megtehető.

Most belátjuk, hogy L nem 1-típusú. Ha ugyanis az volna, akkor volna egy olyan G_j a fenti felsorolásban, amelyre $L=L(G_j)$. Tekintsük a $\{0, 1\}^*$ j- edik elemét, amit w_j- vel jelöltünk. Ha $w_j \in L(G_j)$, akkor az L definíciója értelmében $w_j \notin L=L(G_j)$ ami ellentmondás. Ha $w_j \notin L(G_j)$, akkor $w_j \in L=L(G_j)$ szintén ellentmondás, tehát L nem 1-típusú.∎

78. Tétel. A rekurzívan felsorolható nyelvek osztálya megegyezik a 0-típusú nyelvek osztályával.

Az angol Recursively Enumerable név alapján jelöljük e nyelvosztályt RE-vel.

Amint láttuk a Turing gépek leírása rekurzívan felsorolható, vagyis csak megszámlálhatóan végtelen sok Turing gép, és ennek megfelelően megszámlálhatóan végtelen sok rekurzívan felsorolható nyelv létezik.

Ugyancsak megszámlálhatóan végtelen sok szót tartalmaz T^* bármilyen véges, nemüres T esetén.Ezzel szemben egy megszámlálhatóan végtelen halmaz lehetséges részhalmazainak száma nem megszámlálható végtelen, hanem több annál.

Tehát egy adott T ábécé feletti nyelvek száma több, mint ahányat Turing-géppel el lehet fogadni;vagy érdekesebben hangzó megfogalmazással: több, mint amennyit fel lehet sorolni.

Ezek alapján az itt tárgyalt nyelvosztályokra a következő hierarchia teljesül:

$CS \subsetneq R \subsetneq RE \subsetneq AL$,

ahol *CS* a környezetfüggő nyelvek, *R* rekurzív nyelvek, *RE* a rekurzívan felsorolható nyelvek osztálya, *AL* pedig az összes nyelv osztályát jelenti.

Itt emlékezzünk vissza arra, hogy a jegyzet elején tárgyalt Markov-féle algoritmus (lásd Markov-féle normál algoritmus [15]) ugyancsak univerzális számítási modell, vagyis minden rekurzívan felsorolható nyelv elfogadtatható ilyen algoritmusokkal, és mivel formális modellről van szó, pontosan ezek fogadtathatóak el ily módon.

9.3.1. Bonyolultsági osztályok

Tehát a rekurzív függvények, illetve rekurzív nyelvek osztálya megadja a kiszámíthatóság határát, az ezen kívül eső függvények, illetve nyelvek esetén nem garantált, hogy valaha is befejeződik a számítás egy adott inputra és megáll a Turing-gép.

Viszont a rekurzív osztályon belül sem egyformák a problémák, vannak amelyek egyszerűen megoldhatóak, és vannak amelyek bonyolultak. Ebben az alfejezetben röviden áttekintjük a legfontosabb *bonyolultsági fogalmakat*.

Ha *TM* egy determinisztikus Turing-gép, $v \in V^k$ input (vagyis $v \in V^*$ és $|v|=k$), a számítás *időigényének* a $t_{TM}(v)$ függvényt nevezzük, mely a következő alakú:

$t_{TM}(v)=max\{n, |v|)\}=max\{n, k\}$, ha *TM* a v inputon n lépésután megáll, egyébként pedig ∞.

Egy konkrét számítási folyamat időigényének meghatározása után olyan időigény-fogalmat vezetünk be, amely egy egész problémára, és nem csak annak egyes példányaira vontkozik. Az időigényt a bemenet hosszának függvényében fogjuk megadni.

Azt mondjuk, hogy egy determinisztikus *TM* Turing-gép időigénye (legfeljebb) $f(n)$, ha *TM* futási ideje egyetlen n hosszú bementre sem több $f(n)$- nél. Ha egy $f(n)$ időigényűTuring-gép elfogad egy L nyelvet,akkor azt mondjuk, hogy $L \in TIME(f(n))$.

$TIME(f(n))$ egy bonyolultsági osztály, ami tartalmazza azokat a nyelveket, amelyek $f(n)$ időben eldönthetőek.

Formálisan tehát a *TM* Turing-gép *időbonyolultsága* (*maximális időbonyolultsága*):

$t_{TM}(n)=\{t_{TM}(w)$, ahol $|w| \leq n\}$.

A tárbonyolultság a számítások közben a szalagokon előforduló leghosszabb sztring hossza (vagyis a nem szóköz jelek száma).

Az eddig itt tárgyalt Turing-gépek determinisztikus működésűek (ahogy napjaink számítógépei is azok).

A Turing-gépeknek a nemdeterminisztikus verzióját is említettük már. A nemdeterminisztikus verzióban a d "függvény" tulajdonképpennem egyértelműen adja meg az új állapot, új szalagjel, fejmozgás hármas(oka)t (vagyis a d a $Q \times V \times \{Bal, Jobb, Helyben\}$ halmaz részhalmazaiba képez).Itt a kiszámíthatóságot úgy definiáltuk, hogy van olyan számítási sorozat amely az eredményt szolgáltatja.

Számítási "erejét" tekintve a nemdeterminisztikus verzió sem tud többet a determiniztikus változatoknál, vagyis minden ami nemdeterminisztikusan kiszámítható kiszámítható determinisztikusan is. (A másik irány triviális.) A nemdeterminisztikus változat számítási sebessége, hatékonysága viszont lehet jobb a determinisztikusénál. Ez azt jelenti, hogy pl. "találgatással" hamarabb találhatunk megoldást. A következő részben röviden kitekintünk arra, hogy milyen problémák milyen költséggel oldhatók meg determinisztikus, illetve nemdeterminisztikus Turing-gép segítségével.

A továbbiakban néhány fontos bonyolultsági osztályt említünk meg. A bonyolultsági osztályok meghatározásához szükségünk lesz a számítási módra, amely lehet determinisztikus vagy nemdeterminisztikus. Ezenkívül arra, hogy mely erőforrást (idő, tár: szalag) korlátozzuk.

Mint az előző részben bevezettük, a determinisztikus módon, $f(n)$ időkorlátozással számolóTuring gépekkel kiszámítható nyelvek a $TIME(f(n))$ bonyolultsági osztályt alkotják.Általában az $f(n)$ nemnegatív egészekhez nemnegatív egészeket rendelő függvénytől megköveteljük, hogy monoton növekvő legyen. Ha $f(n)=c$ egy $c\in\mathbb{N}$ konstansra, akkor a nyelv bármely v szavátmaximum $|v|+c$ lépésben eldönti a TM Turing-gép. Szokásos a lineáris függvény($f(n)=cn$), illetve az n tetszőleges polinómjának használata(ahol n a bemeneti szó hossza).Azon nyelvek (problémák) unióját, amelyekhez van olyan determinisztikus Turing-gép, ami ezek szavait valamilyen polinomfüggvénnyel megadható időben eldönti (kiszámítja), P bonyolultági osztálynak nevezzük.

Legyen most TM egy nemdeterminisztikus Turing-gép. Azt mondjuk, hogy TM $f(n)$ időben eldönti/ felismeri az L nyelvet, ha bármely $v\in L$ szóval indítvaa kezdőkonfigurációból van olyan számítás, amely elfogadja v- t legfeljebb $f(n)$ lépés után ($n=|v|$).

Azon nyelvek unióját, amelyekhez van olyan nemdeterminisztikus Turing-gép, ami a ezek szavait valamilyen polinomfüggvénnyel megadható időben eldönti (kiszámítja), NP bonyolultági osztálynak nevezzük.

A számítógéptudomány egyik legfontosabb nem tisztázott problémája a P és NP bonyolultsági osztályok viszonyának eldöntése, vagyis mivel $P \subseteq NP$, ezért a kérdés $P\equiv?NP$ alakba írható. Általában elfogadott az a feltételezés, hogy a két osztály nem egyezik meg, egyelőre azonban nem ismert olyan feladat (nyelv) ami NP-ben van és bizonyítottan nincs P-ben.

Minden NP-beli nyelvre (problémára) igaz, hogy minden szavára létezik egy "tömör bizonyíték" (ami polinomiális időben ellenőrizhető) arra, hogy az adott szó benne van a nyelvben.

Az NP osztály rengeteg természetes és a gyakorlatban is fontos számítási problémát tartalmaz. Például sok tervezési probléma (utak, kiértékelések, egyenletek megoldásai, VLSI tervrajzok) ilyen. Amikor optimális (egy adott feltételt kielégítő) megoldást keresünk, akkor a keresett objektum maga lesz a bizonyíték. Ezek a bizonyítékok gyakran fizikai objektumok vagy azok matematikai absztrakciói, amelyek nem túl nagyok a probléma méretéhez képest és a feltételek is gyakran polinomiális időben ellenőrizhetőek. Azokat a problémákat nevezzük NP-teljesnek, amelyek legkevésbé feltételezhetőek, hogy egyben P-beliek is. Egy L problémát NP-teljesnek nevezünk, ha abból, hogy $L\in$ P az következik, hogy P = NP. Ez azt jelenti, hogy ha valaki determinisztikusan polinomiális időben tud megoldani egy NP-teljes problémát, akkor minden NP-beli problémát meg lehet oldani polinomiális időben determinisztikusan is.

9.3.1.1. Egy NP-teljes probléma: a SAT

A SAT (angol: satisfiability, kielégíthetőség szóból) probléma alapvető szerepet játszik a bonyolultságelméletben, ugyanis ez az egyik legismertebb NP-teljes probléma. A feladat maga röviden a következő: adott egy propozicionális logikai formula, döntsük el, hogy kielégíthető-e (vagyis lehet-e a propozicionális változóknak úgy igaz-hamis értéket adni, hogy a formula igaz legyen). A feladatnak többféle speciális megfogalmazása is létezik, és használt mind az elméletben, mind a gyakorlatban. Az első speciális alak, amikor a formula konjunktív normálformában adott. A feladat így is NP-teljes. A következő, még speciálisabb alak, amikor a konjunktív normálforma mellett az is adott, hogy az egyes tagok hány literált tartalmaz(hat)nak. A feladat ilyen megszorítással történő megfogalmazását nevezik n- SAT problémának. Az n- SAT $n \geq 3$ esetén NP-teljes, míg a 2-SAT probléma P-beli. A tömör

bizonyíték ez esetben egy kielégítő kiértékelés, amely megadja mely Boole-változó értéke legyen igaz és melyek legyenek hamisak. ★

További fontos bonyolultsági osztályok a

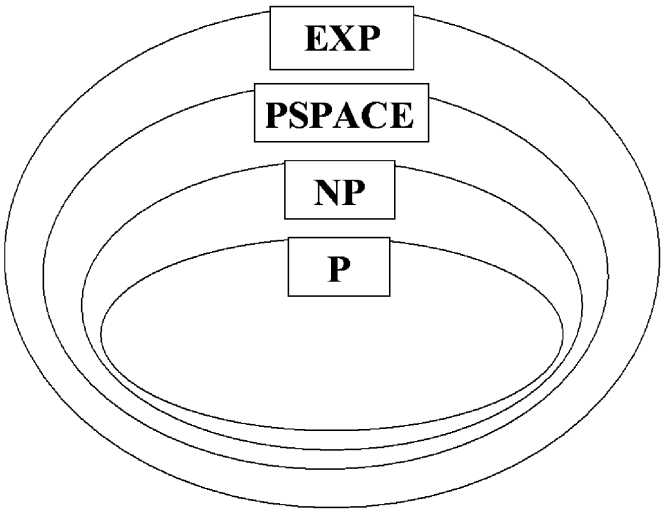
PSPACE: determinisztikus Turing-géppel polinomiális szalagigénnyel kiszámolható problémák osztálya;

NPSPACE: nemdeterminisztikus módon polinomiális szalagigénnyel kiszámolható problémák osztálya;

EXP: determinisztikusan exponenciális időben kiszámolható problémák osztálya.

Ismert, hogy a PSPACE és az NPSPACE problémaosztály egybeesik. Itt jegyezzük meg, hogy a környezetfüggő nyelvosztályra a szóprobléma PSPACE-teljes, viszont nemdeterminisztikusan lineáris tárhelyen is megoldható. Mint jeleztük a környezetfüggő nyelveknél a Szóprobléma fejezetben, az viszont nem ismert, hogy determinisztikusan is megoldható-e lineáris tárhelyen.

Az alábbi ábrán a fent említett bonyolultsági osztályok egymáshoz képesti viszonya látható.



9.4. Normálformák a mondatszerkezetű nyelvtanokhoz

A monoton nyelvtanok definíciójából kitűnik, hogy valójában csak a törlések (rövidítések) lehetősége az amivel a mondatszerkezetű nyelvtanok többet tudhatnak.

Ez valóban így is van, vagyis ha megengedünk akár egyetlen $uBv \rightarrow uv$, ($B \in N$, u, $v \in (N \cup T)^*$) alakú szabályt, vagy egy (N, T, S, H) nyelvtanban az $S \rightarrow \lambda \in H$ használata esetén megengedjük, hogy S szerepeljen szabályok jobb oldalán is, akkor a generáló erő megnő, nem környezetfüggő nyelvek is generálhatóak. Ide kapcsolható a következő eredmény, melyet bizonyítás nélkül közlünk:

79. Tétel. Minden rekurzívan felsorolható nyelv generálható olyan szabályokkal, amelyek mindegyike környezetfüggő ($uBv \rightarrow urv$, $B \in N$, u, v, $r \in (N \cup T)^*$, $r \neq \lambda$), kivéve egyetlen $A \rightarrow \lambda$ alakú szabályt.

A Penttonen normálformát is kiterjeszthetjük ez alapján mondatszerkezetű nyelvtanokhoz:

24. Definíció. Egy mondatszerkezetű nyelvtant Penttonen normálformájúnak hívunk, ha minden szabálya az alábbi formájúak egyike: $A \rightarrow a$, $A \rightarrow BC$, $AB \rightarrow AC$, $A \rightarrow \lambda$ (A, B, $C \in N$, $a \in T$). ★

80. Tétel. Minden rekurzívan felsorolható nyelv generálható Penttonen normálformájú mondatszerkezetű nyelvtannal.

Szorosan ide kapcsolódik a következő tétel is, amely azt mutatja, hogy a környezetfüggő és a mondatszerkezetű nyelvek osztálya között nem is olyan nagy a különbség.

81. Tétel. Legyen $L \subseteq T^*$ egy rekurzívan felsorolható nyelv, ekkor van olyan L' környezetfüggő nyelv, hogy T kiegészíthető egy új betűvel: $V=T \cup \{c\}$ ($c \notin T$), hogy $L' \subseteq c^* L$ és minden $w \in L$ szóra van olyan $n \geq 0$, hogy $c^n w \in L'$.

Vagyis, ha egy *LBA* lineárisan korlátozott automatának adjuk oda egy tetszőleges rekurzívan felsorolható nyelv szavait, akkor minden egyes szóra a tárhelyet eléggé megnövelve, *LBA* el tudja fogadni pontosan a nyelv szavait (illetve azok meghosszabbított, de információt nem tartalmazó változatait). Természetesen a megnövelt tárhely igényű automata már nem feltétlenül lineárisan korlátozott. A következő tétel ezzel összefüggésben hatékonyan használható annak eldöntésére, hogy egyes nyelvek környezetfüggőek.

82. Tétel. (Tárhely tétel) Ha egy adott G mondatszerkezetű nyelvtanra teljesül, hogy létezik olyan k konstans, hogy benne bármely tetszőleges $w \in L(G)$ nemüres szót levezetve van olyan levezetés amiben a mondatforma hossza soha nem hosszabb a $k \cdot |w|$ értéknél, akkor $L(G)$ környezetfüggő.

Ebben a fejezetben több fontos további normálformát mutatunk be (bizonyítás nélkül), amelyek a mondatszerkezetű és egyéb nyelvosztályok kapcsolatára is rávilágítanak.

9.4.1. Révész-féle normálalak

A következő normálalak a Kuroda-féle normálalak Révész-féle észrevétellel kiegészített alakjának általánosítása a mondatszerkezetű nyelvtanokra.

83. Tétel. Minden rekurzívan felsorolható nyelv generálható olyan nyelvtannal, amelyben a szabályok alakja a következő hétféle alakúak lehetnek:

$S \to \lambda$,

$A \to a$,

$A \to B$,

$A \to BC$,

$AB \to AC$,

$AB \to CB$,

$AB \to B$,

ahol $a \in T$, A, B, $C \in N$,

és az S mondatszimbólum nem fordul elő egyik szabály jobboldalán sem.

9.4.2. Geffert-féle normálformák

V. Geffert bizonyította, hogy a törlő (rövidítő) szabályok és a környezetfüggő (monoton) nyelvtanoknak az tulajdonsága, hogy egy szabály bal oldalán egynél több betű is állhat, akár egy közös szabályalakkal is figyelembe vehető. Az itt bemutatandó normálformák közös tulajdonsága, hogy bennük a környezetfüggetlen szabályok mellett csak olyan szabályok jelennek meg, melyek jobboldala az üresszó.

84. Tétel. Minden rekurzívan felsorolható nyelv generálható olyan (N, T, S, H) nyelvtannal, amelyben 5 nemterminális van ($N=\{S,A,B,C,D\}$), minden szabály $S \to v$ alakú (vagyis környezetfüggetlen és a startszimbólum van a bal oldalon); és még két szabály: $AB \to \lambda$, $CD \to \lambda$.

85. Tétel. Minden rekurzívan felsorolható nyelv generálható olyan (N, T, S, H) nyelvtannal, amelyben 4 nemterminális van ($N=\{S,A,B,C\}$), minden szabály $S \to v$ alakú (vagyis környezetfüggetlen alakú és a startszimbólum van a bal oldalon); és még két szabály: $AB \to \lambda$, $CC \to \lambda$.

86. Tétel. Minden rekurzívan felsorolható nyelv generálható olyan (N, T, S, H) nyelvtannal, amelyben 3 nemterminális van ($N=\{S,A,B\}$), minden szabály $S \to v$ alakú (vagyis környezetfüggetlen és a startszimbólum van a bal oldalon); és még két szabály: $AA \to \lambda$, $BBB \to \lambda$.

87. Tétel. Minden rekurzívan felsorolható nyelv generálható olyan (N, T, S, H) nyelvtannal, amelyben 3 nemterminális van ($N=\{S,A,B\}$), minden szabály $S \to v$ alakú (vagyis környezetfüggetlen és a startszimbólum van a bal oldalon) egy kivételével: $ABBBA \to \lambda$.

88. Tétel. Minden rekurzívan felsorolható nyelv generálható olyan (N, T, S, H) nyelvtannal, amelyben 4 nemterminális van ($N=\{S,A,B,C\}$), minden szabály $S \to v$ alakú (vagyis környezetfüggetlen és a startszimbólum van a bal oldalon) egy kivételével: $ABC \to \lambda$.

9.5. Zártsági tulajdonságok

Ebben a fejezetben a rekurzívan felsorolható nyelvek zártsági tulajdonságait vizsgáljuk meg.

89. Tétel. A rekurzívan felsorolható nyelvek osztálya zárt a reguláris műveletekre nézve.

Bizonyítás. A bizonyítás során tételezzük fel, hogy $L_1=L(G_1)$, $L_2=L(G_2)$, ahol $G_1=(N_1, T, S_1, H_1)$ és $G_2=(N_2, T, S_2, H_2)$, $N_1 \cap N_2=\emptyset$; valamint terminális csak $A \to a$ alakú szabályban fordul elő ($A \in N_1 \cup N_2$, $a \in T$). A bizonyítást az egyes műveletekre külön-külön végezzük el.

- *Unió.* S legyen egy olyan nemterminális, amelyeddig nem szerepelt sem N_1- ben, sem N_2- ben. A

  $G_\cup=(N_1 \cup N_2 \cup \{S\}, T, S, H_1 \cup H_2 \cup \{S \to S_1, S \to S_2\})$

  generatív nyelvtan ekkor az $L_1 \cup L_2$ nyelvet generálja.A levezetés során első lépésként alkalmazható $S \to S_1$, illetve $S \to S_2$ szabály választásával eldöntjük, hogy melyik nyelvből szeretnénk generálni; ezután pedig, mivel N_1 és N_2 diszjunkt halmazok, csak a megfelelő H_1, illetve H_2- beli szabályok alkalmazhatóak a levezetés során.

- *Konkatenáció.* Legyen ismét $S \notin N_1 \cup N_2$. Ekkor a

  $G_\cdot=(N_1 \cup N_2 \cup \{S\}, T, S, H_1 \cup H_2 \cup \{S \to S_1S_2\})$

  nyelvtan az L_1L_2 nyelvet generálja, hiszenaz elsőlépésben generált S_1S_2 mondatformában S_1- ből egy L_1- beli szó, S_2- ből pedig egy L_2- beli szó generálható egymástól függetlenül, hiszen $N_1 \cap N_2=\emptyset$, terminálisok pedig nem fordulnak elő szabályok bal oldalán.

- *Kleene-csillag* (*konkatenáció lezárása*). A következőkben egy tetszőleges rekurzívan felsorolható L nyelvhez elkészítjük azt a G_* nyelvtant, amely az L^* nyelvet generálja. Legyenek adottak $G_1=(N_1, T, S_1, H_1)$ és $G_2=(N_2, T, S_2, H_2)$ Révész normálformájú nyelvtanok, amelyekre $N_1 \cap N_2=\emptyset$ és $L(G_1)=L(G_2)=L \setminus \{\lambda\}$. (Ha $\lambda \notin L$, akkor G_1, illetve G_2 L-et generálja, és nem szerepel benne $S_1 \to \lambda$ (illetve $S_2 \to \lambda$) szabály; ha pedig $\lambda \in L$, akkor ez csak az $S_1 \to \lambda$ (illetve $S_2 \to \lambda$) szabályokkal lehetséges, ekkor egyszerűen elhagyjuk ezeket a szabályokat, így generálva az L üresszómentes részét.) Legyen $S \notin N_1 \cup N_2$ új (modat)szimbólum. Tekintsük most a $G_*=(N_1 \cup N_2 \cup \{S, S'\}, T, S, H)$ nyelvtant, ahol $H=H_1 \cup H_2 \cup \{S \to \lambda, S \to S_1, S \to S_1S_2, S \to S_1S_2S\}$. Az így megkonstruált nyelvtan éppen az L^* nyelvet generálja, amit teljes indukcióval bizonyíthatunk.

∎

A fejezet hátralevő részében további halmazműveleteket vizsgálunk.

90. Tétel. A rekurzívan felsorolható nyelvek osztálya zárt a metszetképzésre.

Bizonyítás. Vegyük azokat a Turing-gépeket, amelyek elfogadják az L_1 és L_2 nyelveket olymódon, hogy a szalagnak csak az input által elfoglalt, illetve ettől jobbra eső részét használják. Készítsük

most el azt a Turing gépet, amely először egy másolatot készít az inputról (vagyis a w inputból a $w\#w$ új inputot készíti el), majd a jobboldali másolaton végrehajtja (szimulálja) az L_1- et elfogadó gép számítását, elfogadás esetén letörli (vagyis szóközökkel felülírja) a szalagon hagyott részszámításokat csak a baloldali w- t meghagyva, amelyre az L_2- t elfogadó gép működését hajtja végre. Világos, hogy pont akkor fogja az így megkonstruált Turing-gép elfogadni az inputot, ha mindkét nyelvhez tartozó Turing-gép külön-külön is elfogadná. ∎

Hasonlóan, megfelelő Turing-gépek segítségével, bizonyíthatóak a rekurzív nyelvekre a zártsági tulajdonságok:

91. Tétel. A rekurzív nyelvek osztálya zárt a reguláris (unió, konkatenáció, Kleene-csillag) és a halmaz- (unió, metszet, komplementer) műveletekre.

Korábban már volt szó a rekurzív és a rekurzívan felsorolható osztályok különbözőségéről (lásd Szóprobléma - rekurzív és rekurzívan felsorolható nyelvek).

6. Következmény. A rekurzívan felsorolható nyelvek halmaza nem zárt a komplementerképzésre.

9.6. Irodalmi megjegyzések

A Turing-gép koncepciója [Turing 1936]-ban jelent meg. Ugyanebben az időben jelentek meg más hasonló kifejezőerejű számítási modellek: A. Church, S.C. Kleene, illetve E. Post munkájaként. A számításelmélettel kapcsolatban ajánljuk magyar nyelven a [Demetrovics et al 1989] tankönyvet, míg angolul a [Minsky 1967], [Hopcroft, Ullman 1979] és [Sipser 2005] könyveket. A bonyolultsági osztályokkal kapcsolatosan a [Papadimitriu 1995], illetve a [Rónyai et al 1998] könyveket ajánljuk. Az NP-teljes problémákról részletesen szól [Garey, Johnson 1979]. A Révész-féle normálalak megtalálható [Révész 1989]-ben, amíg a Geffert-féle normálformák [Geffert 1988]-ban jelentek meg.

10. fejezet - Nyelvtanrendszerek (Grammatikarendszerek)

A klasszikus formális nyelvek és automaták elméletben a nyelvek és az automaták klasszikus számítási eszközöket modelleztek. Ezek központosítottak voltak – a számításokat egy központi ágens (egység) végezte. Így a klasszikus formális nyelvek elmélete szerint egy nyelvet egy grammatika készít, vagy egy automata ismer fel. A modern számítástudományban az elosztott számítások fontos szerepet játszanak. Az elosztott rendszerek megfigyelése számítógépes hálózatokban, elosztott adatbázisokban, stb., olyan fogalmakhoz vezetett, mint párhuzamosság, konkurencia és kommunikáció. A nyelvtanrendszerek elméletében vizsgált formális rendszerek az elosztott számítások szintaktikai modelljei, melyekben ezek a fogalmak értelmezhetőek és tanulmányozhatóak.

A nyelvtanrendszer nyelvtanok egy csoportja, amelyben a nyelvtanok meghatározott protokoll szerint közösen dolgoznak egyetlen nyelv létrehozásán. Több indok szól ilyen generatív mechanizmus létrehozása mellett, többek közt, az elosztás modellezése, a generáló erő növelése, a (leírási) bonyolultság csökkentése. A kritikus rész itt az együttműködési protokoll. A nyelvtanrendszerek elméletét tekinthetjük egyfajta formális együttműködési protokollok elméletének. A központi probléma a meghatározott protokollokat használó rendszerek működésének leírása, és a különböző protokollok a megfigyelt rendszerek különböző tulajdonságaira gyakorolt hatásának elemzése.

A nyelvtanrendszerek kétféle alapvető osztályba sorolhatóak: szekvenciálisak vagy párhuzamos működésűek, ezek alapján megkülönböztetjük a kooperatív elosztott (angolul: cooperating distributed, röviden CD) és a párhuzamos kommunikáló (angolul: parallel communicating, röviden PC) rendszereket. Ebben a fejezetben röviden ismertetjük ezen rendszereket.

10.1. Nyelvtanok kooperatív elosztott rendszerei – CD nyelvtanrendszerek

(Cooperating Distributed Grammar Systems – CD grammar systems)

A nyelvtanok kooperatív elosztott rendszere szekvenciális működésű: egymást követő lépéseken alapul. A rendszert alkotó összes nyelvtan egy adott, közös mondatformán dolgozik. Minden egyes időpillanatban csak egyetlen nyelvtan aktív, ami átírja az aktuális mondatformát. Azokat a kérdéseket, hogy melyik komponens lehet aktív egy adott pillanatban, és mikor válik inaktívvá egy aktív (műveleteket végző) nyelvtan – az aktuális mondatformát átadva a többi, a rendszert alkotó nyelvtan valamelyikének – az együttműködési protokoll határozza meg.

Példák megállási feltételekre (inaktívvá válásra). Egy lépés egy levezetési szabály alkalmazását jelenti.

- Az aktív komponensnek pontosan k lépést kell végrehajtania.
- Az aktív komponensnek legalább k lépést kell végrehajtania.
- Az aktív komponensnek legfeljebb k lépést kell végrehajtania.
- Az aktív komponensnek annyi lépést kell végrehajtania, amennyit tud.
- Az aktív komponens tetszőleges számú lépést hajthat végre.

A rendszer által generált nyelv az így generált terminális szavak nyelve lesz.

A CD nyelvtanrendszerek felépítése egy iskolai tábla használatához hasonlítható, ha azt probléma-megoldáshoz használják. A közös mondatforma az iskolai tábla tartalma (a közös adatszerkezet, amely a megoldandó probléma aktuális állapotát tartalmazza). A nyelvtanok a tudásforrások (ágensek, feldolgozó egységek, eljárások, különböző képességű diákok, stb.) amelyek hozzájárulnak a probléma megoldásához azzal, hogy megváltoztatják a tábla tartalmát a képességeiknek megfelelően. Az

együttműködési protokollban van kódolva a tudásforrások irányítása (például a sorrend, amelyben a tudásforrások hozzájárulhatnak a megoldáshoz).

25. Definíció. A (környezetfüggetlen) nyelvtanok kooperatív elosztott rendszere egy n rendű ($n \geq 1$) rendszer, felépítése: $CD=(N, T, S, H_1, \ldots, H_n)$ Ahol N és T diszjunkt ábécék, $V=N \cup T$, $S \in N$, és $H_1, \ldots, H_n$ pedig környezetfüggetlen levezetési szabályok véges halmazai. Az N elemei a nemterminálisok, a T elemei a terminálisok; S a mondatszimbólum, $H_1, \ldots, H_n$ pedig a rendszer komponensei. ★

Az iskolai tábla példájánál maradva a komponensek a táblánál a problémát megoldó ágenseknek felelnek meg. A szabályok megfeleltethetőek az ágensek által végrehajtott műveleteknek, ezek eredménye lehet a tábla tartalmának, vagyis a mondatformának a megváltoztatása.

- Az S axióma a táblán található probléma kezdeti állapotának formális megfelelője.
- A T ábécé tartalmazza azokat a betűket, amelyek megfelelnek az olyan tudás-részleteknek, amelyek elfogadhatóak megoldásként, illetve a megoldás részeiként.
- A nemterminálisok "kérdésekként" értelmezhetőek, amelyekre választ keresünk. Egy komponens által feltett kérdések, és egy másik által átírtak felfoghatóak az egyik komponens által feltett kérdésnek, amelyre egy másik komponens válaszol. Így a komponensek kommunikálhatnak a megoldás pillanatnyi állapotába illesztett üzenetekkel, mintegy a mondatformába (a tábla tartalmába) kódolva.

Ha konkrétan olyan grammatikát akarunk definiálni, ami egy CD nyelvtanrendszer része, akkor felírhatjuk a CD- t ilyen formában: $CD=(N, T, S, G_1, \ldots, G_n)$, ahol $G_i=(N, T, S, H_i)$, $1 \leq i \leq n$.

26. Definíció. Legyen $CD=(N, T, S, H_1, \ldots, H_n)$ egy CD nyelvtanrendszer. Ekkor

1. Minden egyes $i \in \{1, \ldots, n\}$- re az i- edik komponens * módbeli levezetését $\Rightarrow_i^*$- gal jelöljük, és a következőképpen definiáljuk: $p \Rightarrow_i^* q$ akkor és csakis akkor, ha $p \Rightarrow_{G_i}^* q$.

2. Minden egyes $i \in \{1, \ldots, n\}$- re az i- edik komponens termináló levezetését $\Rightarrow_i^t$- vel jelöljük, és a következőképpen definiáljuk: $p \Rightarrow_i^t q$ akkor és csakis akkor, ha $p \Rightarrow_i^* q$ és nincs olyan $r \in (N \cup T)^*$ amire $q \Rightarrow_i r$ ($r \neq q$).

3. Minden egyes $i \in \{1, \ldots, n\}$- re az i- edik komponens k lépéses levezetését $p \Rightarrow_i^{=k} q$- val jelöljük, és a következőképpen definiáljuk: $p \Rightarrow_i^{=k} q$ akkor és csakis akkor, ha van olyan $r_1, \ldots, r_{k+1} \in (N \cup T)^*$ amely $p=r_1$, $q=r_{k+1}$, és minden egyes j- re ($1 \leq j \leq k$), $r_j \Rightarrow_i r_{j+1}$.

4. Minden egyes $i \in \{1, \ldots, n\}$- re az i- edik komponens legfeljebb k lépéses levezetését $p \Rightarrow_i^{\leq k} q$- val jelöljük, és a következőképpen definiáljuk: $p \Rightarrow_i^{\leq k} q$ akkor és csakis akkor, ha $p \Rightarrow_i^{=j} q$ valamely $j \leq k$ értékre.

5. Minden egyes $i \in \{1, \ldots, n\}$- re az i- edik komponens legalább k lépéses levezetését $p \Rightarrow_i^{\geq k} q$- val jelöljük, és a következőképpen definiáljuk: $p \Rightarrow_i^{\geq k} q$ akkor és csakis akkor, ha $p \Rightarrow_i^{=j} q$ valamely $j \geq k$ értékre.

★

A *- módbeli levezetés, $p \Rightarrow_i^* q$, olyan működési formát jelöl, amelyben az ágensek addig dolgoznak a táblánál, ameddig akarnak. A t- módbeli levezetés annak a stratégiának felel meg, amelyben egy ágensnek addig kell hozzájárulnia a megoldási folyamathoz a táblánál, ameddig csak tud (maximálisan kihasználva a hatáskörét). Akkor beszélünk $=k$ levezetési módról, ha k egymást követő közvetlen levezetési lépésben használjuk fel az i- edik komponens szabályait, ez k műveletet jelent egy ágens számára a táblánál. A $\geq k$ levezetési mód időkorlátnak felel meg, mivel egy ágens maximum k- lépést hajthat végre. A $\geq k$ levezetési módban legalább k lépést kell végrehajtania az ágensnek. A levezetési módok a fentiek alapján feltételeznek bizonyos minimális kompetenciát az ágensek részéről.

Legyen $D=\{*, t\}\cup\{\geq k, =k, \geq k \mid k \text{ pozitív egész}\}$.

27. Definíció. Egy $CD=(N, T, S, H_1, \ldots, H_n)$ nyelvtanrendszer által generált nyelv valamely $f\in D$ levezetési módban: $L_f(CD)=\{w\in T^* \mid S\Rightarrow_{i_1}^{f} p_1\Rightarrow_{i_2}^{f} p_2\ldots\Rightarrow_{i_m}^{f} p_m=w,\ m\geq 1,\ 1\leq i_j\leq n,\ 1\leq j\leq m\}$. ★

Az előző definícióval számos nyelvet társítottunk egy CD-rendszerhez, felhasználva a különböző D-beli megállási feltételeket. A H_i egy komponense elkezdhet dolgozni (futása engedélyezett lesz) egy p mondatformán, amikor p tartalmazza egy H_i- beli szabály baloldalát. Az, hogy melyik engedélyezett komponens kapja meg az éppen aktuális mondatformát, egy nemdeterminisztikus választás alapján dől el. Elképzelhetünk különböző kezdőfeltételeket is, például, ha egy komponens csak akkor lesz engedélyezett egy mondatformán való munkára, ha bizonyos feltételek teljesülnek, esetleg egy külső vezérlő egység (például egy gráf, vagy verem meghatározva a komponensek engedélyezési sorrendjét) irányít.

10.1. példa - CD nyelvtanrendszer

Legyen $CD=(\{S, A, B, C, D\}, \{a, b, c\}, S, \{S\to S, S\to AC\}, \{A\to aBb, C\to cD\}, \{B\to aAb, D\to cC\}, \{A\to ab, B\to ab, C\to c, D\to c\})$. Könnyen belátható, hogy ez a nyelvtanrendszer $=2$, ≥ 2 és t-módban az $\{a^nb^nc^n \mid n>0\}$ nyelvet, míg $\leq k$ (bármely $k\geq 1$ esetén), $=1$ és $*$ módban a $\{a^nb^nc^m \mid n, m>0\}$ nyelvet generálja; ha pedig $k>2$ akkor $\geq k$ módban az üres nyelv az eredmény. ★

10.2. példa - CD nyelvtanrendszer

Legyen $CD=(\{S, A\}, \{a\}, S, \{S\to AA\}, \{A\to SS\}, \{A\to a, S\to a\})$. Könnyen belátható, hogy ez a nyelvtanrendszer t- módban a $\{a^{2^n} \mid n\geq 0\}$ nyelvet generálja. ★

Ahogy az előző példákban láthattuk a környezetfüggetlen nyelvtanok generáló ereje megnő, ha őket CD-rendszerben használjuk: sem a $\{a^nb^nc^n \mid n>0\}$, sem a $\{a^{2^n} \mid n\geq 0\}$ nyelv nem környezetfüggetlen, sőt a második nyelv nem is konstansnövekményű (nem szemilineáris).

10.3. példa - CD nyelvtanrendszerek - Gyakorló feladat

Adjunk meg olyan CD-rendszert, amely valamilyen módban a $\{ww \mid w\in\{a, b\}^*\}$ nyelvet tudja generálni. ★

10.2. Nyelvtanok párhuzamos kommunikáló rendszerei – PC nyelvtanrendszerek

(Parallel Communicating Grammar Systems – PC grammar systems)

Ebben az alfejezetben generatív nyelvtanok PC rendszereibe nyújtunk rövid betekintést.

28. Definíció. Egy PC nyelvtanrendszer egy n rendű ($n\geq 1$) rendszer $PC=(N, K, T, (S_1, H_1), \ldots, (S_n, H_n))$, ahol N és T a nemterminális és a terminális ábécék, $K=\{Q_1, \ldots, Q_n\}$ kérdőszimbólumok, az i-edik komponenshez Q_i, (N, T és K páronként diszjunkt halmazok), a H_i halmazok levezetési szabályok véges halmazai ($N\cup T\cup K$) ábécé felett úgy, hogy K-beli elem nem állhat szabály bal oldalán. Valamint a komponensek mondatszimbólumai $S_i\in N$ (minden $1\leq i\leq n$ esetében). ★

29. Definíció. Adott egy PC nyelvtanrendszer $PC=(N, K, T, (S_1, H_1), \ldots, (S_n, H_n))$. A $(p_1, \ldots, p_n)$ és $(q_1, \ldots, q_n)$ két elem n-esre, ahol $p_i, q_i\in(N\cup T\cup K)^*$ minden $1\leq i\leq n$- re és $p_1\notin T^*$- ra, a közvetlen levezethetőség fennáll, ezt $(p_1, \ldots, p_n)\Rightarrow(q_1, \ldots, q_n)$ alakba írjuk, ha a következő két eset egyike fennáll:

1. Minden i értékre ($1\leq i\leq n$) $p_i\in(N\cup T)^*$, továbbá $p_i\Rightarrow q_i$ a H_i egy levezetési szabálya segítségével, vagy $p_i=q_i$ ha $p_i\in T^*$.

2. Amennyiben van olyan i, amelyre p_i tartalmaz K- beli elemet, akkor minden ilyen i- re járjunk el a következőképpen: legyen $p_i=r_{i,1}Q_{i,1}r_{i,2}Q_{i,2}\ldots r_{i,k}Q_{i,k}r_{i,k+1}$ (valamilyen $k \geq 1$ mellett), ahol $r_{i,j} \in (N \cup T)^*$ minden j- re ($1 \leq j \leq k+1$). Ha $p_{i,j} \in (N \cup T)^*$ minden j értékre ($1 \leq j \leq k+1$), akkor legyen $q_i=r_{i,1}p_{i,1}r_{i,2}p_{i,2}\ldots r_{i,k}p_{i,k}r_{i,k+1}$, és legyen $q_{i,j}=S_j$ minden j értékre ($1 \leq j \leq k+1$).

Minden egyéb esetben legyen $q_i=p_i$.★

A ($p_1, \ldots, p_n$) elem n- est a PC rendszer konfigurációjának hívjuk. Tehát a ($p_1, \ldots, p_n$) konfigurációból közvetlenül levezethető a ($q_1, \ldots, q_n$) a következő két esetben:

Az első eset, amikor egyik aktuális mondatforma sem tartalmaz K- beli szimbólumot, a levezetés komponensenként zajlik, kivéve azokat a p_i mondatformákat, amik csak terminálisokat tartalmaznak, ugyanis azok változatlanul maradnak.

A második eset, amikor kérdőszimbólum fordul elő valamely mondatformában. Ekkor a következő kommunikációs lépés zajlik le: a Q_i minden előfordulásának helyére p_i kerül (feltéve hogy p_i nem tartalmaz kérdőszimbólumot). Pontosabban egy kérdőszimbólumot tartalmazó mondatforma csak akkor módosul, ha benne minden kérdőszimbólum olyan mondatformára utal, amely nem tartalmaz kérdőszimbólumot. Egy kommunikációs lépésben p_j- vel helyettesítjük a Q_j szimbólumot (a Q_j kérdést megválaszoltuk), és a j- edik komponens elölről kezdi a számítást az S_j axiómájából (vagyis a kezdőszimbólumából). A kommunikációs lépésben nem zajlik komponensenkénti levezetés, levezetési szabályokat nem alkalmazhatunk, ha valamely mondatforma tartalmaz kérdőszimbólumot. Ha egy kérdőszimbólumot nem tudunk megválaszolni egy adott lépésben, előfordulhat, hogy a következő lépésben meg tudjuk válaszolni.

Fontos, hogy a rendszerben nem lehet olyan szabály, aminek a baloldalán kérdőszimbólum fordul elő, így a kérdések csak kommunikációs lépéssel válaszolhatóak meg, a levezetés csak így folytatódhat. Ugyancsak fontos, hogy nincs definiálva a közvetlen levezetés arra az estre, ha $p_1 \in T^*$. A $\Rightarrow$ közvetlen levezetés tranzitív és reflexív lezártját $\Rightarrow^*$- al jelöljük és, szokásos módon, levezetésnek nevezzük.

Egy PC rendszer működése során a következő holtponti szituációk jöhetnek létre:

1. Nincs kérdőszimbólum, de valamely $p_i \notin T^*$ mondatformára nincs alkalmazható levezetési szabály.

2. Körbe-kérdezés jön létre, vagyis $p_{i,1}$- ben szerepel $Q_{i,2}$, $p_{i,2}$- ben viszont $Q_{i,3}$, és így tovább, amíg valamely $p_{i,k}$ a $Q_{i,1}$- et tartalmazza. Ilyenkor sem kommunikációs lépés, sem komponensenkénti levezetés nem alkalmazható.

30. Definíció. Adott egy PC nyelvtanrendszer $PC=(N, K, T, (S_1, H_1), \ldots, (S_n, H_n))$. Ekkor az általa generált nyelv $L(PC)=\{w \in T^* \mid (S_1, S_2, \ldots, S_n) \Rightarrow^* (w, p_2, \ldots, p_n)$ valamely $p_i \in (N \cup T \cup K)^*$ mondatformákra ($2 \leq i \leq n$) $\}$. ★

Tehát kiindulunk a mondatszimbólumokból és komponensenkénti levezetési, illetve kommunikációs lépésekkel haladunk, amíg az első komponens nem terminál (vagy holtpontra nem jut a rendszer). Az első komponens tehát megkülönböztetett szereppel bír, mesternek nevezzük.

31. Definíció. Ha egy $PC=(N, K, T, (S_1, H_1), \ldots, (S_n, H_n))$ PC nyelvtanrendszerben csak az első (mester) komponens vezethet be kérdőszimbólumot, akkor központosított PC rendszerről beszélünk, egyébként a PC rendszer nem központosított. ★

Központosított rendszer esetén az általunk fentebb ismertetett második típusú holtpont nem állhat elő.

Egy PC rendszert akkor nevezünk lineárisnak, környezetfüggetlennek, stb., ha minden komponense lineáris, környezetfüggetlen, stb.

Visszatérve az iskolai táblához, a PC rendszert a következő típusú problémamegoldásnak tekinthetjük: adott a csoportvezető (a tanár), aki a táblánál dolgozik, és a többiek (csapattagok), akik részszámításokat végeznek (pl. a saját füzetükben). A CD-rendszerek "egyenlő" tagjaival ellentétben, itt hierarchikus a felépítés, ami centralizált rendszer esetén még szembetűnőbb: csak a tanárnak van

joga kérdezni, ekkor a részszámítások eredményeit behelyettesíti a saját maga által készített (központi) számításba.

10.4. példa - PC nyelvtanrendszer

Legyen $PC=(\{S_1, S_2\}, \{Q_1, Q_2\}, \{a, b\}, (S_1, \{S_1 \to S_1, S_1 \to Q_2Q_2\}), (S_2, \{S_2 \to aS_2, S_2 \to bS_2, S_2 \to \lambda\}))$. Könnyen belátható, hogy $L(PC)=\{ww \mid w \in \{a, b\}^*\}$. ★

Láthatjuk, hogy egy PC rendszernek a generáló ereje meghaladhatja a komponensek generáló erejét: az előző példában környezetfüggetlen szabályokkal generáltunk nem környezetfüggetlen nyelvet.

10.5. példa - PC nyelvtanrendszer gyakorló feladat

Készítsünk olyan környezetfüggetlen PC rendszert, amely az $\{a^nb^ma^nb^m \mid n, \ m \geq 1\}$ nem környezetfüggetlen nyelvet generálja. ★

A CD és PC rendszereket nem csak generatív nyelvtanokra, de egyéb formális számítási modellekre, mint pl. automatákra, is definiálhatjuk (az automataelméletnél ismertetett átlátszóbetűs automaták (lásd Átlátszóbetűs felismerő automata) pl., mint nagyon speciális újrainduló automaták CD-rendszere volt eredetileg definiálva, később készült el az átlátszóbetűs átfogalmazás). Itt jegyezzük meg ismét, hogy vannak olyan nyelvtan-, illetve automatarendszerek, ahol egy aktív komponens után valamilyen további adatszerkezet, pl. verem, sor segít a következő aktív komponens kiválasztásában (akár determinisztikus módon).

10.3. Irodalmi megjegyzések

A nyelvtanrendszerek alap monográfiája a [Csuhaj-Varjú et al 1994], de más, a formális nyelvek elméletét összefoglaló szerkesztett művekben is található olyan fejezet, amely e témakör jó leírását adja, pl. [Rozenberg, Salomaa 1997], [Martín-Vide et al 2004]. A CD rendszerekkel kapcsolatos első eredmények [Csuhaj-Varjú, Dassow 1990]-ben találhatók. A PC rendszerekkel kapcsolatban régi, illetve újabb eredményekért lásd pl. [Vaszil 1997], [Csuhaj-Varjú, Vaszil 2001], [Csuhaj-Varjú, Salomaa 2001] cikkeket. Eredetileg újrainduló automaták CD rendszereiként lettek definiálva az átlátszóbetűs automaták is [Nagy, Otto 2010a, 2010b, 2010c, 2011a].

11. fejezet - Irodalomjegyzék

[Afonin, Golomazov 2009] Sergey Afonin, Denis Golomazov: Minimal Union-Free Decompositions of Regular Languages. LNCS 5457, LATA 2009, 83-92.

[Aho 1968] A. V. Aho: Indexed Grammars - An Extension of Context-Free Grammars, Journ. of ACM, 15 (1968), 647-671.

[Amar, Putzolu 1964] V. Amar, G.R. Putzolu: On a Family of Linear Grammars. Information and Control 7(3) (1964), 283-291.

[Amar, Putzolu 1965] V. Amar, G.R. Putzolu: Generalizations of Regular Events. Information and Control 8(1) (1965), 56-63.

[Babcsányi 2007] Babcsányi István: Automaták, nyelvek, kódok, BME, Matematika Intézet, Algebra Tanszék, 2007 (elektronikus jegyzet: http://www.math.bme.hu/~babcs/aut.pdf).

[Bach 2002] Bach Iván: Formális nyelvek, Typotex kiadó, 2002 (elektronikusan: http://www.typotex.hu/download/formalisnyelvek.pdf).

[Backus 1959] J.W. Backus: The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM coference, Proc. Intl. Conf. on Information Processing (1959), UNESCO, 125-132.

[Bar-Hillel et al 1961] Y. Bar-Hillel, M. Perles, E. Shamir: On formal properties of simple phrase structure grammars, Z. Phonetik. Sprachwiss. Komm., 14 (1961) 143-172.

[Bel-Enguix et al 2008] G. Bel-Enguix, M.D. Jiménez-López, C. Martín-Vide (eds.): Recent Developments in Formal Languages and Applications. Springer, Berlin, 2008.

[Buntrock, Otto 1995] G. Buntrock, F. Otto: Growing context-sensitive languages and Church-Rosser languages, 12th STACS, Lecture Notes in Computer Science 900 (1995), 313-324.

[Chomsky 1956] N. Chomsky: Three Models for the Description of Language. IRE Transactions on Information Theory IT-2, no. 3 (September 1956): 113-24. (Reprinted in Readings in Mathematical Psychology 2, edited by R. Luce, R. Bush, E. Galanter, 105-24. New York: Wiley and Sons, 1965.)

[Chomsky 1959] N. Chomsky: On Certain Formal Properties of Grammars. Information and Control 2 (June 1959): 137-67. (Reprinted in Readings in Mathematical Psychology 2, edited by Luce, Bush, Galanter, 125-55. New York, Wiley and Sons, 1965.)

[Cremers 1973] Armin B. Cremers: Normal Forms for Context-Sensitive Grammars, Acta Informatica 3 (1973) 59-73.

[Csuhaj-Varjú et al 1994] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Paun: Grammar Systems: A grammatical approach to distribution and cooperation. Gordon and Breach Science Publishers, Topics in Computer Mathematics 5, Yverdon, 1994.

[Csuhaj-Varjú, Dassow 1990] Erzsébet Csuhaj-Varjú, Jürgen Dassow: On Cooperating/Distributed Grammar Systems. Elektronische Informationsverarbeitung und Kybernetik 26 (1/2) (1990) 49-63.

[Csuhaj-Varjú, Salomaa 2001] Erzsébet Csuhaj-Varjú, Arto Salomaa: Networks of Language Processors: Parallel Communicating Systems. Current Trends in Theoretical Computer Science, Entering the 21th Century, World Scientific (2001) 791-810.

[Csuhaj-Varjú, Vaszil 2001] E. Csuhaj-Varjú, Gy. Vaszil: On context-free parallel communicating grammar systems: synchronization, communication, and normal forms. Theoretical Computer Science 255 (2001), 511-538.

[Dassow, Paun 1989] J. Dassow, Gh. Paun: Regulated Rewriting in Formal Language Theory. EATCS Monographs in Theoretical Computer Science 18, Springer, 1989.

[Dassow et al 1994] J. Dassow, M. Ito, Gh. Paun: On the Subword Density of Languages, SEA Bull. Math., 18 (1994), 49-61.

[Demetrovics et al 1989] Demetrovics János, Jordan Denev, Radiszlav Pavlov: A számítástudomány matematikai alapjai, Tankönyvkiadó, Budapest, 1989.

[Dömösi 1995] Dömösi Pál: Formális Nyelvek és Automaták, egyetemi jegyzet, Debrecen, 1995.

[Dömösi et al, 1996] P. Dömösi, M. Ito, M. Katsura, C. Nehaniv: New pumping property of context-free languages, Combinatorics, Complexity and Logic, Proc. International Conference on Discrete Mathemtics and Theoretical Computer Science - DMTCS'96, Springer, Singapore, 187-193.

[Dömösi et al 2004] Dömösi P., Fazekas A., Horváth G., Mecsei Z.: Formális nyelvek és automaták, egyetemi jegyzet, MobiDiák, 2004.

[Dömösi, Nehaniv 2005] P. Dömösi, C. Nehaniv: Algebraic theory of automata networks: an introduction, SIAM Monographs on Discrete Mathematics and Applications 11, 2005.

[Earley 1970] J. Earley: An efficient context-free parsing algorithm, Communications of the Association for Computing Machinery (ACM), 13 (1970) 94-102.

[Ésik et al 2006] Zoltán Ésik, Carlos Martín-Vide, Victor Mitrana (eds.): Recent Advances in Formal Languages and Applications, Springer 2006.

[Fülöp 2005] Fülöp Zoltán: Formális nyelvek és szintaktikus elemzésük, Polygon jegyzettár (3. kiadás, 2005)

[Garey, Johnson 1979] M.R. Garey, D.S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, 1979.

[Geffert 1988] Viliam Geffert: Context-Free-Like Forms for the Phrase-Structure Grammars. LNCS 324, MFCS 1988, 309-317.

[Greibach 1965] Sheila A. Greibach: A New Normal-Form Theorem for Context-Free Phrase Structure Grammars. J. ACM 12(1) (1965) 42-52.

[Harrison 1978] Michael A. Harrison: Introduction to Formal Language Theory, Addison-Wesley, 1978.

[Hayashi 1973] T. Hayashi: On Derivation Trees of Indexed Grammars - An Extension of the uvwxy - Theorem, Publ. RIMS, Kyoto Univ., 9 (1973), 61-92.

[Hopcroft, Ullman 1979] J. E. Hopcroft, J. D. Ullmann: Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, Massachusetts, Menlo Park, California, London, Amsterdam, Don Mils,Ontario, Sidney, 1979.

[Horváth et al 2004] Horváth G., Mecsei Z., Nagy B.: Formális nyelvek és automaták gyakorlati összefoglaló, egyetemi jegyzet, Mobidiák, Debrecen, 2004.

[Horváth, Nagy 2010] Horváth G., Nagy B.: Pumping lemmas for linear and nonlinear context-free languages, Acta Univ. Sapientiae, Informatica, 2/2 (2010), 194-209.

[Huffman 1954] D.A. Huffman: The synthesis of sequential switching circuits, J. Franklin Inst. 257 (1954), 161-190 és 275-303.

[Kleene 1956] S.C. Kleene: Representation of events in nerve nets and finite automata, in: C.E. Shannon, J. McCarthy: Automata Studies, Princeton Univ. Press, (1956), 3-42.

[Kuroda 1964] S.-Y. Kuroda: Classes of languages and linear-bounded automata, Information and Control 7 (1964), 207-223.

[Leupold, Nagy 2010] P. Leupold, B. Nagy: 5' → 3' Watson-Crick automata with several runs, Fundamenta Informaticae 104 (2010) 71-91.

[Lindenmayer 1968] A. Lindenmayer: Mathematical models for cellular interaction in development I. Filaments with one-sided inputs. Journal of Theoretical Biology 18 (1968) 280-289.

[Linz 2006] P. Linz: An introduction to formal languages and automata, Fourth edition, Jones and Bartlett Publishers 2006.

[Mäkinen 1985] E. Mäkinen: On permutative grammars generating context-free languages, BIT 25 (1985), 604-610.

[Martín-Vide et al 2004] C. Martín-Vide, V. Mitrana, Gh. Paun, (eds.): Formal Languages and Applications. Springer, Berlin, 2004.

[Mealy 1955] G.H. Mealy: A method for synthesizing sequential circuits, Bell System Technical Journal 34 (1955), 1045-1079.

[Minsky 1967] M. Minsky: Computation: Finite and Infinite Machines, Prentice-Hall 1967.

[Moore 1956] E.F. Moore: Gedanken experiments on sequential machines, in: C.E. Shannon, J. McCarthy: Automata Studies, Princeton Univ. Press, (1956), 129-153.

[Nagy 2004] Nagy B.: A Normal Form for Regular Expressions, DLT'04, Eighth International Conference on Developments in Language Theory, Auckland, New Zealand, (CDMTCS-252) 10 pages (2004).

[Nagy 2005] Nagy B.: Új elvű számítógépek (Bevezetés az új számítási modellekbe és a nem-klasszikus "számítógépek" tudományába), egyetemi jegyzet, Mobidiák, Debrecen, 2005.

[Nagy 2006a] Nagy B.: Union-free languages and 1-cycle-free-path-automata, Publicationes Mathematicae Debrecen 68 (2006), 183-197.

[Nagy 2006b] Nagy B.: Left-most derivation and shadow-pushdown automata for context-sensitive languages (2006), Proceedings of the 10th WSEAS International Conference on Computers, Athens, Greece, 962-967.

[Nagy 2006c] Nagy B.: Shadow-Pushdown Automata, WSEAS Transactions on Computers 5/11 (2006), 2565-2570.

[Nagy 2008]: Nagy B.: On 5' → 3' sensing Watson-Crick finite automata, DNA 13, Revised Selected Papers, Lecture Notes in Computer Science - LNCS 4848 (2008), 256-262.

[Nagy 2009a] Nagy, B.: Languages generated by context-free grammars extended by type $AB \rightarrow BA$ rules, Journal of Automata, Languages and Combinatorics 14 (2009), 175-186.

[Nagy 2009b] Nagy, B.: Permutation languages in formal linguistics, IWANN 2009, Lecture Notes in Computer Science 5517 (2009), 504-511.

[Nagy 2010a] Nagy, B.: On a hierarchy of permutation languages, in: (editors: M. Ito, Y. Kobayashi, K. Shoji) Automata, Formal Languages and Algebraic Systems, World Scientific, Singapore (2010), 163-178.

[Nagy 2010b] Nagy, B.: Derivation trees for context-sensitive grammars, in: (editors: M. Ito, Y. Kobayashi, K. Shoji) Automata, Formal Languages and Algebraic Systems, World Scientific, Singapore (2010), 179-199.

[Nagy 2010c] Nagy B.: An automata-theoretic characterization of the Chomsky-hierarchy, TAMC 2010, Lecture Notes in Computer Science - LNCS 6108 (2010), 361-372.

[Nagy 2010d] Nagy B.: On Union-complexity of Regular Languages, CINTI 2010, 11th IEEE International Symposium on Computational Intelligence and Informatics, 177-182.

[Nagy 2010e] Nagy, B.: Pumping lemmas for special linear languages, ICAI 2010, Eger, Hungary, vol. II. 73-81.

[Nagy 2011a] Nagy B.: Linguistic power of permutation languages by regular help, in: Linguistics, Biology and Computer Science: Interplays, Cambridge Scholars (2011), 135-152.

[Nagy 2011b] Nagy, B.: A class of 2-head finite automata for linear languages, TRIANGLE (2011), megjelenés alatt

[Nagy 2011c] Nagy, B.: On a hierarchy of 5' → 3' sensing Watson-Crick finite automata languages, Journal of Logic and Computation (Oxford University Press), 2011, közlésre elfogadva

[Nagy, Otto 2010a] Nagy B., F. Otto: CD-Systems of Stateless Deterministic R(1)-Automata Accept all Rational Trace Languages, LATA 2010, Lecture Notes in Computer Science - LNCS 6031 (2010), 463-474. Springer, Heidelberg

[Nagy, Otto 2010b] Nagy B., F. Otto: On CD-systems of stateless deterministic R-automata with window size one, Kasseler Informatikschriften 2010, 2, Kassel University, Germany

[Nagy, Otto 2010c] Nagy B., F. Otto: CD-Systems of Stateless Deterministic R(1)-Automata Governed by an External Pushdown Store, Kasseler Informatikschriften 2010, 4, Kassel University, Germany

[Nagy, Otto 2011a] Nagy B., F. Otto: An automata-theoretical characterization of context-free trace languages, SOFSEM 2011, Lecture Notes In Computer Science - LNCS 6543 (2011), 406-417.

[Nagy, Otto 2011b] Nagy B., F. Otto: Finite-State Acceptors with Translucent Letters, ICAART 2011 - 3rd International Conference on Agents and Artificial Intelligence, BILC 2011 - 1st International Workshop on AI Methods for Interdisciplinary Research in Language and Biology, 3-13.

[Nagy, Otto 2011c] Nagy B., F. Otto: CD-Systems of Stateless Deterministic R(1)-Automata Governed by an External Pushdown Store, RAIRO - Theoretical Informatics and Applications, RAIRO-ITA, 2011, megjelenés alatt

[Nagy, Otto 2011d] Nagy B., F. Otto: On CD-systems of stateless deterministic R-automata with window size one, Journal of Computer and System Sciences - JCSS, megjelenés alatt

[Nagy, Varga 2009] Nagy B., Varga P.: A New Normal Form for Context-Sensitive Grammars, SOFSEM 2009: (35th Conference on Current Trends in) Theory and Practice of Computer Science, Spindleruv Mlyn, Czech Repulic, Volume II, 60-71. Iteration-free Normal Form for Context-Sensitive Grammars, Tech. Rep., Dept. Math. and Fac. Inf., Univ. Debrecen, 2007/8.

[Oettinger 1961] A.G. Oettinger: Automatic syntactic analysis and the pushdown store, Proc. Symposia on Applied Math. 12 (1961), AMS, Providence, RI.

[Papadimitriou 1995] Christos H. Papadimitriou: Számítási bonyolultság, Addison-Wesley, 1995., magyar fordítás: Novadat Bt. 1999.

[Paun et al 1998] Gheorge Paun, Grzegorz Rozenberg, Arto Salomaa: DNA Computing - New Computing Paradigms. Springer-Verlag, 1998.

[Peák 1988] Peák István: Bevezetés az automaták elméletébe I., Tankönyvkiadó, Budapest, 1988.

[Peák 1989] Peák István: Bevezetés az automaták elméletébe II., Tankönyvkiadó, Budapest, 1989.

[Peák 1990] Peák István: Bevezetés az automaták elméletébe III., Tankönyvkiadó, Budapest, 1990.

[Penttonen 1974] M. Penttonen: One-sided and two-sided context in formal grammars. Information and Control 25 (1974), 371-392.

[Prusinkiewicz, Lindenmayer 1990] P. Prusinkiewicz, A. Lindenmayer: The Algorithmic Beauty of Plants, Springer-Verlag, 1990, 1996. (elektronikusan: http://algorithmicbotany.org/papers/#abop)

[Rabin, Scott 1959] M.O. Rabin, D. Scott: Finite automata and their decision problems, IBM J. Res. Dev. 3 (1959), 114-125.

[Révész 1983] Gy. E. Révész: Introduction to Formal Languages, McGraw-Hill, New York, St Louis, San Francisco, Auckland, Bogota, Hamburg, Johannesburg, London, Madrid, Mexico, Montreal, New Delhi, Panama, Paris, Sao Paulo, Singapore, Sydney, Tokyo, Toronto, 1983.

[Révész 1989] Révész György: Bevezetés a formális nyelvek elméletébe, Tankönyvkiadó, Budapest, 1989.

[Rónyai et al 1998] Rónyai L., Ivanyos G., Szabó R.: Algoritmusok, Typotex 1998.

[Rosenfeld 1979] A. Rosenfeld: Picture Languages, Formal Models for Picture Recognition, Academic Press, New York, 1979.

[Rozenberg, Salomaa 1997] G. Rozenberg, A. Salomaa (eds.): Handbook of formal languages, Springer, Berlin, Heidelberg, 1997. 3 volumes.

[Salomaa 1973] A. Salomaa: Formal Languages, Academic Press, New York, London, 1973.

[Salomaa 1981] A. Salomaa: Jewels of formal language theory, Computer Science Press, Rockville, Maryland, 1981.

[Scheinberg 1960] S. Scheinberg: Note on the boolean properties of context-free languages, Information and Control 3 (1960), 372-375.

[Schutzenberger 1963] M.P. Schutzenberger: On context-free languages and pushdown automata, Information and Control 6 (1963), 246-264.

[Sempere, García 1994] J. M. Sempere, P. García: A characterization of even linear languages and its application to the learning problem, Proc. Second International Colloquium, ICGI-94, Lecture Notes in Artificial Intelligence, 862 (1994) 38-44.

[Sipser 2005] M. Sipser: Introduction to the Theory of Computation, Course Technology, 2005.

[Shallit 2009] J. Shallit: A Second Course in Formal Languages and Automata Theory, Cambridge University Press, 2009.

[Shyr 1991] H.J. Shyr: Free Monoids and Languages, National Chung-Hsing University, Taichung, Taiwan, R.O.C., Ho Min Book Company, 1991.

[Turing 1936] A.M. Turing: On computable numbers with an application to the Entscheidungsproblem, Proc. Londom Math. Society 2 (1936), 230-265. és 544-546.

[Vaszil 1997] Gy. Vaszil: Various communications in PC grammar systems, Acta Cybernetica 13 (1997), 173-196.

[Younger 1967] D.H. Younger: Recognition and parsing of context-free languages in time n^3, Information and Control 10 (1967), 189-208.