BACH IVÁN

Formális nyelvek

Második, javított kiadás

Egyetemi tankönyv

TYPOTEX Kiadó ◊ Budapest, 2002

A könyv az illetékes kuratórium döntése alapján az Oktatási Minisztérium támogatásával a Felsőoktatási Tankönyvtámogatási Program keretében jelent meg (2000/362).

ISBN 9639132 92 6

Kedves Olvasó!
Önre gondoltunk, amikor a könyv előterjesztésén munkálkodtunk. Kapcsolatunkat szorosabbra fűzhetjük, ha belép a Typoklubba, ahonnan értesülhet új kiadványainkról, akcióinkról, programjainkról, és amelyet a www.typotex.hu címen érhet el. Honlapunkon megtalálhatja az egyes könyvekhez tartozó hibajegyzéket is, mert sajnos hibák olykor előfordulnak.

Kiadja a Typotex Elektronikus Kiadó Kft., az 1795-ben alapított Magyar Könyvkiadók és Könyvterjesztők Egyesülésének tagja
Felelős kiadó: Votisky Zsuzsa
Tördelő: Naszódi Mátyás
A borítót tervezte: Tóth Norbert
Terjedelem 14,8 (A/5) ív
Készült a pécsi Bornus Nyomdában
Felelős vezető: Borbély Tamás

Formális nyelvek

Annotáció

Egy megismételhetetlen emberi és szakmai
kapcsolat örökségeként ezt a munkámat

Frey Tamás

barátom emlékének ajánlom.
Mint a Budapesti Műszaki Egyetem tanára,
elsők között szorgalmazta a számítástudomány
oktatását.
Fájdalmasan korai halála csak az első lépések
megtételét tette lehetővé számára.
Ez a könyv az ő célkitűzéseit kívánja
szolgálni.

Tartalomjegyzék

Előszó

Ez a könyv a „Számítástechnikai Nyelvészet” című egyetemi jegyzetem alapján készült. Egy tankönyvnek szánt munka sohasem lehet, ha szabad egy szakmabeli kifejezést alkalmaznom, környezetfüggetlen. A környezetet itt a tanterem, a katedra, az egyik oldalon álló oktató, és a másik oldalon helyet foglaló egyetemi hallgatók sokasága jelenti.

Annak idején a jegyzetet szándékaim szerint abban a stílusban írtam, ami megfelelt az előadások hangulatának. Így aztán a közvetlenebb első személyben íródott a jegyzet. Sokak szerint egy könyv esetében a személytelenség kívántatik meg. Én nem követtem ezt a szemléletmódot, nemcsak azért, mert így a jegyzet könyvesítése egyszerűbbnek ígérkezett, hanem mert írásban is meg akartam tartani azt a közvetlenséget, amelyet az előadásokon szándékaim szerint gyakorolok.

Ez a könyv hosszú évek pedagógusi tapasztalatai után tulajdonképpen kollektív munka eredménye. Munkatársaim, akik velem együtt oktatták ezt a tárgyat, megjegyzéseikkel kimondatlanul is hozzájárultak a könyv megszületéséhez. Régebbi munkatársaim közül sokan már más feladatot kaptak, de a mostani munkatársi gárdát hadd soroljam itt fel köszönettel: Vitéz András, Naszódi Mátyás, Vaszil György, Csima Judit, Varró Dániel.

Külön köszönöm Naszódi Mátyás aktív segítségét: az ő munkája többek között az ábrák számítógépes komponálása.

Végül – last but not least – köszönetet szeretnék mondani annak a jóval több, mint ezer egyetemi hallgatónak, akik közül sokan nem csak passzív szereplői voltak az oktatásnak. Az ő igényességük szorított engem rá arra, hogy világosan, érthetően, az egyetemi hallgató számára emészthető formában íródjék az egyetemi jegyzet és ez a könyv. Bízom benne, hogy ez többé-kevésbé sikerült. A munka megjelenésekor az ő érdemük se maradjon megemlítetlenül.

Megkülönböztetett köszönet illeti a **TYPOTEX** kiadót, amiért beiktatta ennek a könyvnek a kiadását terveibe.

Bevezetés

Történeti áttekintés

Az emberi munka eredménye mindig valamilyen társadalmi igényt elégít ki. Ez a megállapítás igaz a tudományos élet területén is. Míg az elvont tudományok esetében az eredmény és az azt kiváltó társadalmi igény megfeleltetése gyakran nem triviális feladat, az alkalmazott tudományok művelői általában jól tudják, milyen gyakorlati céllal végzik kutató munkájukat.

A matematikai nyelvészetre gondolva ez a megfeleltetés teljesen egyértelmű.

A második világháború alatt és az azt követő években a kutatási és fejlesztési munkákban forradalmi változások következtek be. A műszaki és természettudományokban ugrásszerűen megnőtt a magasan kvalifikált szakemberek száma, ráadásul sokkal nagyobb hányaduk kezdett a kutatás és fejlesztés területén dolgozni, mint az azt megelőző időkben. Ez a változás természetesen az új műszaki és tudományos eredmények, vele együtt pedig a tudományos publikációk számának hirtelen növekedésével járt.

Korábban egy szakember viszonylag könnyen figyelemmel kísérhette azt a mintegy 5-6 folyóiratot, ahol tartalmas cikkeket remélhetett. A problémát tovább nehezítette, hogy a publikációk egy része, elsősorban az angolul beszélők számára „szokatlan" nyelven volt írva, mint az orosz, japán vagy akár a magyar.

Az információhoz való hozzáférés igénye hirtelen megnövelte a fordítandó cikkek számát olyannyira, hogy ezt a lényegében egysíkú, majdhogynem mechanikus munkát sem fordítói kapacitással, sem pénzzel nem lehetett győzni.

Ugyanakkor az egyik, éppen abban az időben, az 50-es évek elején kiteljesedő, és az információrobbanáshoz nem kis mértékben hozzájáruló új tudomány, a számítástechnika azt állította, hogy célja a gépies szellemi munka kiváltása. Magától értetődő volt tehát az a törekvés, hogy a fordítást, ezt a gépies szellemi munkát egy számítástechnikai eszköz, a számítógép végezze.

Minthogy azonban a számítógép csak szabatosan megfogalmazott feladat megoldására képes, szükség volt a fordítás mint feladat formális leírására. Ez volt tehát az a kiváltó társadalmi igény, amely az 50-es években szinte a semmiből egy új tudomány a matematikai nyelvészet megalkotását eredményezte. Az új tudomány születési évének az 1956 esztendőt tekinthetjük. Ekkor publikálta ugyanis a szakma atyja, és mindmáig egyik legnagyobb egyénisége *Noam Chomsky* munkásságának első eredményeit. A korai és látványos sikerek hatására igen nagy pénzek, dollármilliók áramlottak a szakmába, és ezzel együtt egy sereg kitűnő koponya kapcsolódott be ennek a divatszakmának a művelésébe.

Sajnos a kezdeti lendület – legalábbis ami az új eredményeket illeti – később alábbhagyott, jóllehet az anyagi és személyi feltételek továbbra is kedvezőek voltak. Végül is *Johnson* elnök 1967-ben összehívta a szakma tojásfejűit, vizsgálják meg, milyen perspektívája van a természetes nyelvek számítógépes fordításának, várható-e ezen a területen a közeljövőben sikeres áttörés.

A döntés a következő évben született meg: Véleményük szerint – ne feledjük akkor 1968-at írtak – a számítástechnika pillanatnyi fejlettsége nem ígér gyors eredményt, a feltételek még nem érettek meg az eredeti célkitűzés megvalósítására.

Ennek hatására az elsősorban állami pénzekből csordogáló dollárfolyam csapjait persze azonnal elzárták, és a téma művelői közül sokan más, jövedelmezőbb foglalkozás után néztek, csak a szakma legmegrögzöttebb hívei maradtak. Ezzel a téma a nagy érdeklődéssel kísért gyakorlati tudomány állapotából felemelkedett vagy lesüllyedt – a fogalmazás nézőpont kérdése – az elméleti tudományok gyanúsan tiszteletreméltó státusába.

Szerencsére a természetes nyelveknél sokkal egyszerűbbek a mesterséges nyelvek, nevezetesen azok, amelyek nem az emberek közötti kommunikációt, hanem az ember és a számítógép közötti kapcsolatteremtést szolgálják. Fogalmi körük erősen leszűkített, nyelvtani szabályaik egyszerűek, általában nem ismernek kivételt, végül – talán ez a legfontosabb ismérvük – mondataik egyértelműek.

A matematikai nyelvészet eredményei éppen ezért jóval könnyebben voltak alkalmazhatóak a gépi nyelvek fordításának területén, mind a fordítás elméletének megteremtésére, mind a fordítóprogramok gyakorlati megvalósítására. Így alakult ki azután egy tudomány, a matematikai nyelvészet gyakorlatibb aspektusú fattyúhajtása, a számítástechnikai nyelvészet. Ennek a tudománynak az elsajátításához kíván segítséget nyújtani ez a könyv.

Befejezésül néhány szó a természetes nyelvek gépi fordításáról. Az ominózus döntés óta több mint három évtized telt el. A számítástechnika fejlődése töretlen volt, és azóta lehetőségei alapvetően megváltoztak mind a műveleti sebesség, mind az igénybe vehető tárkapacitás tekintetében. Ennek fényében a 80-as évek óta egyre több szakember figyelme fordult ismét a természetes nyelvek gépi fordítása felé. Már túl vagyunk az első sikeresnek mondható próbálkozásokon. A természetes nyelvek gépi fordításának témája felébredt csipkerózsika álmából.

1. Formális nyelvek

1.1. A formális nyelvek definíciója

Az olvasó, akinek „polgári” előéletéből van valamilyen – és a maga nemében bizonyára helyes – képzete a *nyelv* szó jelentéséről, jobban teszi, ha egyelőre megfeledkezik ilyen irányú ismereteiről. Talán helyesebb lett volna a most sorra kerülő fogalom jelölésére a nyelv helyett valamilyen új szó bevezetése, ahogyan azt például a francia szaknyelv teszi, két szót használva a két fogalomra – *la langue* és *le langage*. Így azzal vigasztalódhatunk, hogy sem az angol, sem a német, sem az orosz nyelv nem disztingvál a két fogalom között, legalábbis elnevezés tekintetében nem. Hozzá kell azonban tennem, hogy a nyelv kétféle – számítástechnikai és köznapi – értelmezése között, az első pillanatban alapvetőnek tűnő különbség valójában nagyon kicsi. Remélem, a későbbiekben kitűnik majd, hogy a kétféle értelmezés milyen közel áll egymáshoz.

Elöljáróban itt nyelv alatt mindig írott nyelv értendő. A nyelv mondatai ennek megfelelően mindig írott szövegek. Ezek a szövegek egy adott véges szimbólum- illetve karakterkészlet elemeiből alkotott jelsorozatok.

Jelölje Σ a nyelv karakterkészletének véges halmazát. Ezt a halmazt a nyelv alfabetájának nevezzük.

Legyen Σ^* a Σ alfabetából alkotott véges, de nem korlátos hosszúságú jelsorozatok halmaza.

A talán túl tömör megfogalmazás „véges, de nem korlátos hosszúságú” bizonyos magyarázatot igényel. Tekintsük például a természetes számoknak, pontosabban decimális, tízes számrendszerben való ábrázolásuknak halmazát. Ezt a halmazt nyilván a decimális számjegyekből álló, és nem a 0 számjeggyel kezdődő véges, de nem korlátos hosszúságú jelsorozatok alkotják. A jelsorozatok valóban végesek, hiszen nincsen olyan természetes szám, amelyet véges hosszúságú jelsorozattal ne tudnánk leírni, ugyanakkor azonban a hossz nem korlátos, hiszen bármilyen nagy számot engedélyezünk is a jelsorozat hosszának, mindig található olyan természetes szám, amelynek leírására a megadottnál több karaktert tartalmazó jelsorozat szükséges.

Az alkalmazott Σ^* jelöléssel, illetve jelölési konvencióval kapcsolatosan a következőket kell tudnunk.

A Σ^i jelölés általában a Σ elemeiből alkotott, és pontosan i hosszúságú jelsorozatok halmazát jelöli. Ennek megfelelően

$$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i \qquad (1.1.)$$

valóban az összes lehetséges a Σ alfabetájából képezhető jelsorozatot adja. Az $i = 0$ eset a zérus hosszúságú, vagyis üres jelsorozatnak felel meg. Ennek jele ε.

A fentiekből nyilvánvaló, hogy nem üres Σ halmaz esetén a Σ^* halmaz megszámlálhatóan végtelen számosságú. A továbbiakban mindig feltételezzük, hogy a Σ halmaz nem üres.

Egy adott alfabetából képezhető összes lehetséges jelsorozatba természetesen beletartozik az üres jelsorozat, az ε is. Éppen ezért – az (1.1.) jelöléseivel összhangban – ε is eleme a Σ^* halmaznak.

Valamely – egy adott Σ alfabeta felett értelmezett – $\boldsymbol{L}$ nyelv a Σ^* halmaz egy tetszőleges részhalmaza.

$$\boldsymbol{L} \subset \Sigma^* \tag{1.2.}$$

Ha egy adott Σ^* halmazbeli w jelsorozat eleme az $\boldsymbol{L}$ nyelv által meghatározott részhalmaznak, akkor a w jelsorozat az $\boldsymbol{L}$ nyelv mondata.

$$w \in \boldsymbol{L} \tag{1.3.}$$

Egy adott Σ alfabeta feletti összes lehetséges nyelvek halmaza nyilván a Σ^* összes részhalmazából alkotott halmaz, vagyis a Σ^* halmaz hatványhalmaza. Minthogy viszont a Σ^* halmaz számossága megszámlálhatóan végtelen, így egy véges, de nem üres Σ alfabeta felett kontinuum számosságú nyelv létezik.

Ezen nyelvek között megkülönböztetett fontosságú és nevezetes nyelv az üres nyelv, amelynek egyetlen mondata, eleme sincsen, továbbá az a nyelv, amelynek egyetlen mondata az üres jelsorozat. A könyvben az előbbit $\boldsymbol{L}_0$, az utóbbit $\boldsymbol{L}_e$ jelöli. Így $\boldsymbol{L}_0 = \varnothing$ $\boldsymbol{L}_e = \{\varepsilon\}$

Felhívom az olvasó figyelmét, ügyeljen a két nyelv közötti különbségre.

A nyelvek vizsgálatánál alapvető fontosságú az a kérdés, miképpen lehet egy nyelvet megadni, jellemezni. Amennyiben egy nyelv véges számú mondatot tartalmaz, úgy a mondatok exhausztív felsorolásával a nyelv egyértelműen megadható.

Természetesen a gyakorlat szempontjából csakis azok a nyelvek érdekesek, amelyeknek végtelen sok mondatuk van. A programozási nyelvekre gondolva ugyanis mit kezdhetnénk egy olyan nyelvvel, amelyen csak véges sok programot lehet írni. A végtelen sok mondatot tartalmazó nyelvek, pontosabban ezen nyelvek egyik osztályának leírására egy új matematikai objektum, egy új formális eszköz, a nyelvtan, a grammatika szolgál.

1.2. Grammatikák

Mielőtt a grammatikák formális leírásának részleteiben elmerülnénk, jellemezzük néhány szóval a grammatikát mint számítástechnikai fogalmat.

Mindenekelőtt azok a megjegyzések, amelyeket a nyelv köznapi és számítástechnikai értelmezésének első pillanatra alapvetőnek tűnő eltéréséről, de valójában mélyenjáró azonosságáról mondottunk, betűről betűre igaz a nyelvtanokra is. Mint az a későbbiekből – remélem – kitűnik, ez akkor is így van, ha a

következőkben leírtak aligha emlékeztetik az olvasót az általános-, illetve középiskolában a nyelvtanról tanultakra, jóllehet ugyanarról a fogalomról van szó.

Formálisan a grammatikát egy négyes határozza meg

$$\mathbf{G} = (\mathbf{N}, \Sigma, \mathbf{P}, S) \tag{1.4.}$$

ahol – **N** a grammatikai szimbólumok véges halmaza,
– Σ az alfabeta, a nyelv karakterkészletének már jól ismert véges halmaza,
– **P** az úgynevezett levezetési szabályok összessége, ami szintén egy véges halmazt alkot,
– *S* a mondatszimbólum.

Az újonnan bevezetett fogalmak értelme rövidesen világos lesz.

Nagyon lényeges, hogy a grammatikai és nyelvi szimbólumok között különbséget tudjunk tenni.

A nyelvtanok, grammatikák azért alkalmasak egy nyelv definiálására, mert segítségükkel egy nyelv mondatai levezethetőek, generálhatóak.

Egy levezetés mondatszerű formáknak nevezett jelsorozatok egymásutánja. A mondatszerű forma abban különbözik a mondattól, hogy abban nem csak a nyelv szimbólumai, vagyis a Σ elemei, hanem grammatikai szimbólumok is szerepelhetnek. Legyen

$$\gamma_0 \Rightarrow \gamma_1 \Rightarrow \ldots\ldots \Rightarrow \gamma_n \tag{1.5.}$$

egy levezetés. A levezetések sorozatában az egyik mondatszerű formából a következőt a levezetési szabályok segítségével származtathatjuk. A $\Rightarrow$ szimbólum azt jelenti, hogy a baloldali mondatszerű formából a nyelvtan levezetési szabályai szerint a jobboldali pontosan egy lépésben levezethető. Ha külön ki akarjuk hangsúlyozni, hogy melyik grammatika szerinti levezetésről van szó, a grammatika azonosító jelét indexszerűen a $\Rightarrow$ szimbólum alá írhatjuk. Amennyiben a levezetés nem egyetlen lépésben történt, hanem meg nem határozott tetszőleges számú lépésben, akkor a $\Rightarrow$ szimbólum fölé tett csillag * jelzi ezt. Így, ha azt akarjuk leírni, hogy a γ_0 mondatszerű formából a γ_n mondatszerű forma a ***G*** grammatika segítségével előre meg nem határozott számú lépésben levezethető, akkor ennek jelölése:

$$\gamma_0 \overset{*}{\Rightarrow}_G \gamma_n \tag{1.6.}$$

A levezetési szabályok – használatos még rájuk a produkciós szabály, helyettesítési szabály, újraírási szabály elnevezés is, amely elnevezések ezen szabályok egy-egy tulajdonságára vonatkoznak – a következő alakúak:

$$\alpha \rightarrow \beta \tag{1.7.}$$

ahol a nyíl a szabály bal-, illetve jobboldalát elválasztó szimbólum, amely különbözik mind a grammatikai, mind a nyelvi szimbólumoktól, míg α és β két jelsorozat, amely mind grammatikai, mind nyelvtani szimbólumokat tartalmazhat. A szabály baloldalán álló jelsorozat, tehát α legalább egy grammatikai szimbólumot kell hogy tartalmazzon, a jobboldali jelsorozat tetszőleges.

Egy mondatszerű forma esetében egy levezetési szabály akkor alkalmazható, ha a mondatszerű formának van olyan – nem szükségképpen valódi – részsorozata, amely megegyezik a levezetési szabály baloldalával. Ez esetben a szabály alkalmazása abban áll, hogy az említett részsorozatot a szabály jobboldalával helyettesítjük, a mondatszerű formának ezt a részét újraírjuk. Az alkalmazás módja magyarázza a korábban említett elnevezéseket.

Így ha a γ_i mondatszerű formából az $\alpha \rightarrow \beta$ szabály segítségével a γ_{i+1} egy lépésben levezethető, akkor szükségképpen:

$$\gamma_i = \gamma'\alpha\gamma'' \Rightarrow \gamma'\beta\gamma'' = \gamma_{i+1} \qquad (1.8.)$$

Itt γ' és γ'' tetszőleges, esetleg üres jelsorozat.

A helyettesítési szabályok alkalmazásának módja utasítást ad arra vonatkozóan, hogyan haladjunk mondatszerű formáról mondatszerű formára.

De hogyan kapjuk meg az első mondatszerű formát?

Megállapodás szerint minden levezetés egy különleges, egyetlen speciális grammatikai szimbólumból, a mondatszimbólumból álló mondatszerű formával kezdődik. Ez az a kályha, ahonnan minden levezetés elindul.

Amennyiben a levezetés során olyan jelsorozatot kapunk, amely csak nyelvi szimbólumokat tartalmaz, vagyis a mondatszerű forma egyben mondat is, akkor a levezetés szükségképpen befejeződik, terminálódik, hiszen grammatikai szimbólum hiányában egy levezetési szabály sem alkalmazható. A kapott jelsorozat pedig a nyelv egy mondata.

Amennyiben a mondatszerű forma tartalmaz grammatikai szimbólumot, akkor a levezetést – ha lehetséges – folytatni kell, a mondat generálása még nem fejeződött be, nem terminálódott. Ha annak ellenére, hogy a mondatszerű forma még tartalmaz grammatikai szimbólumot, de nem találunk olyan helyettesítési szabályt, amelyet alkalmazhatnánk, akkor ez a generálás, ez a levezetési kísérlet sikertelen volt.

A fentiek indokolják az irodalomban, de ebben a könyvben is a grammatikai illetve nyelvi szimbólumokra alkalmazott nemterminális illetve terminális szimbólum elnevezést.

Ismerkedjünk meg néhány, a számítástechnikai irodalomban általánosan használt konvencióval.

A nemterminális szimbólumokat a latin ábécé elejéről vett nagybetűk jelölik – *A, B, C*.

A terminális szimbólumokra a latin ábécé elejéről vett kisbetűket használjuk – *a, b, c*.

Egyetlen, közelebbről meg nem határozott szimbólumot latin nagybetűvel jelölünk, de az ábécé végéről – *X, Y, Z*. A jelölésből ilyenkor nem derül ki, hogy terminális vagy nemterminális szimbólumról van-e szó.

A terminális szimbólumokból álló jelsorozatokat az ábécé végéről vett latin kisbetűk – *x, y, z* – jelentik.

Az olyan jelsorotokat, amelyek tartalmazhatnak mind terminális, mind nemterminális szimbólumokat is görög kisbetűk jelölik – α, β, γ.

A mondatszimbólumot az angol *sentence* (mondat) szó kezdőbetűje *S* jelöli.

A következőkben a formulák jelölései – kivételes esetektől eltekintve – megfelelnek a fenti konvenciónak. Eltérés esetén külön felhívom erre a tényre az olvasó figyelmét.

Ennyi bevezetés után gyorsan és könnyen túleshetünk a nyelvtanok formális leírásán. Mint említettük a grammatikát formálisan egy négyes írja le.

$$\mathbf{G} = (N, \Sigma, \mathbf{P}, S)$$

Az egyes jelölések értelmét már korábban tisztáztuk. Érdemes megfigyelni, hogy az ábécé közepéről vett betűk a konvenciókat nem érintik.

A terminális és nemterminális szimbólumok közötti különbségtétel nagyon fontos. Éppen ezért a két halmaznak – az N és a Σ halmaznak diszjunktnak kell lennie, tehát

$$N \cap \Sigma = \varnothing \tag{1.9.}$$

A levezetési szabályok alakját már ismerjük, sőt arról is történt említés, hogy a baloldalnak mindig kell nemterminális szimbólumot tartalmaznia. Ez a követelmény formálisan a következőképpen írható fel:

$$\alpha \to \beta \quad \text{ahol} \quad \alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^* \quad \text{és} \quad \beta \in (N \cup \Sigma)^* \tag{1.10}$$

vagyis az α jelsorozat mindenképpen tartalmaz legalább egy nemterminális szimbólumot, amelyet – akárhány, ezt jelzi a kitevőben szereplő * jel – tetszőleges, tehát akár N akár Σ halmazbeli szimbólum előzhet meg, illetve követhet.

A β tetszőleges, így akár zérus hosszúságú, terminális és nemterminális szimbólumokból álló jelsorozat.

Természetesen a mondatszimbólum eleme a nemterminális szimbólumok halmazának, így

$$S \in N \tag{1.11.}$$

Egy adott **G** grammatika által definiált nyelv jelölése $\boldsymbol{L}(\mathbf{G})$, és ez a nyelv a Σ^* azon elemeit tartalmazza, amely jelsorozatok a **G** grammatika segítségével tetszőleges számú lépésben generálhatóak:

$$\boldsymbol{L}(\mathbf{G}) = \{ w \in \Sigma^* \mid S \overset{*}{\Rightarrow}_{\mathrm{G}} w) \tag{1.12.}$$

A korábbiak alapján a fenti formula könnyen értelmezhető.

Gyakorlásképpen adjunk meg egy nyelvet ezzel a formális leírással. Legyen

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

ahol $\mathbf{P} = \{ S \to aSBC,\ S \to abC,\ CB \to BC,\ bB \to bb,\ bC \to bc,\ cC \to cc \}$

Vezessük le a nyelv egy mondatát. A levezetés mondatszerű formáiban – a levezetés követését megkönnyítendő – aláhúzással emeltem ki azokat a részeket, amelyeket az adott lépésben újraírtunk.

$$S \Rightarrow a\underline{S}BC \Rightarrow aab\underline{CB}C \Rightarrow aa\underline{bB}CC \Rightarrow aab\underline{bC}C \Rightarrow aabbcC \Rightarrow aabbcc$$

A fenti példánkban a generálás sikeres volt. A levezetés végén eljutottunk a nyelv egy mondatához. Ez nem szükségszerű. Adódhat helyzet, amikor a menetközben kapott mondatszerű forma tovább nem vezethető, nincsen egyetlen olyan szabály sem, amely alkalmazható volna, jóllehet a mondatszerű forma még tartalmaz nemterminális szimbólumot. Ekkor a generálási kísérlet fiaskóval végződött.

Ugyanezt a grammatikát használva lássunk erre is egy példát:

$$S \Rightarrow a\underline{S}BC \Rightarrow aa\underline{bC}BC \Rightarrow aabcBC \Rightarrow ?$$

A levezetés lefulladt, ezen az úton a nyelv egyetlen mondata sem generálható.

A grammatika egyébként, mint arról könnyen meggyőződhetünk az

$$\boldsymbol{L}(\mathbf{G}) = \{\, a^{\mathbf{i}}\, b^{\mathbf{i}}\, c^{\mathbf{i}} \;\mid\; \mathbf{i} > \mathbf{0} \,\}$$

nyelvet generálja. Adott esetben a nyelv mondatainak száma valóban végtelen, mégpedig megszámlálhatóan végtelen.

1.3. Miért grammatika a grammatika – Egy illusztratív példa

A régóta jól ismert fogalmak, mint nyelv, nyelvtan, grammatika formális definíciója, mint azt elöljáróban beharangoztam, látszatra valóban nem sok rokonságot mutat az iskolában az ezekről a fogalmakról tanultakkal. Ezt az ellentmondást oldjuk most fel egy demonstratív példával, amely ráadásul még új fogalmak megismerésére is módot ad.

A példa a magyar nyelv egy nagyon korlátozott és nagyon leszűkített részhalmaza lesz. Sajnos az imént nagy gonddal kidolgozott konvencióink felhasználása itt nem célszerű, ezért erre a kivételes alkalomra néhány új konvencióval élünk.

A nemterminális szimbólumokat, amelyek ebben az esetben szemlátomást valóban grammatikai szimbólumok, csúcsos zárójelbe tett, a szimbólum nyelvtani szerepére utaló szöveggel jelöljük.

Még egy új jelölést, pontosabban jelölési egyszerűsítést vezetünk be. Olyan esetekben, amikor több levezetési szabály baloldala azonos, ezeket a szabályokat összevonva adjuk meg. A csupán egyszer megadott baloldalt követően a jobboldalakat egymás után a ***vagy*** olvasatú **|** jellel elválasztva írjuk le. Ezt a jelölést a későbbiekben máskor is alkalmazzuk. A levezetési szabályok bal- és jobboldalának elválasztására is a $\rightarrow$ helyett egy új jelet a ::= szimbólumot alkalmaztuk (Backus–Naur jelölés). jobboldalának elválasztására is a $\rightarrow$ helyett egy új jelet a ::= szimbólumot alkalmaztuk (Backus–Naur jelölés).

Példa grammatikánk levezetési szabályai a következők lesznek:

<mondat> ::= <alanyi rész> <állítmányi rész>
<alanyi rész> ::= <főnévi rész> <határozó>
<állítmányi rész> ::= <tárgyi rész> <igei rész:>
<főnévi rész> ::= <névelő> <jelzők> <főnév>
<névelő> ::= ε | a | az | egy
<jelzők> ::= <jelző> | <jelző> <jelzők>
<jelző> ::= ε | hideg | meleg | fehér | fekete | nagy | kis
<főnév> ::= kutya | macska | hús | egér | sajt | tej | víz
<határozó> ::= ε | nappal | éjjel | reggel | este
<tárgyi rész> ::= <főnévi rész>t
<igei rész> ::= eszik | iszik

Természetesen itt a kutya egyetlen terminális szimbólum, és nem jelsorozat. Ezért is kell ragaszkodnunk a terminális szimbólum elnevezéshez, mivel a terminális karakter kifejezés félreértések forrása lehet.

A fenti minimagyar nyelv grammatikájával néhány korrekt magyar mondat generálható.

Az alábbiakban erre mutatunk be egy példát, arra kérve az olvasót, hogy a kissé hosszadalmas levezetést türelemmel nézze át.

<mondat> ⇒ <alanyi rész><állítmányi rész> ⇒
<főnévi rész><határozó><állítmányi rész> ⇒
<névelő><jelzők><főnév><határozó><állítmányi rész> ⇒
a <jelzők><főnév><határozó><állítmányi rész> ⇒
a <jelző><jelzők><főnév><határozó><állítmányi rész> ⇒
a nagy <jelzők><főnév><határozó><állítmányi rész> ⇒
a nagy <jelző><főnév><határozó><állítmányi rész> ⇒
a nagy fehér <főnév><határozó><állítmányi rész> ⇒
a nagy fehér kutya <határozó><állítmányi rész> ⇒
a nagy fehér kutya reggel <állítmányi rész> ⇒
a nagy fehér kutya reggel <tárgyi rész><igei rész> ⇒
a nagy fehér kutya reggel <főnévi rész>t<igei rész> ⇒
a nagy fehér kutya reggel <névelő><jelzők><főnév>t<igei rész> ⇒
a nagy fehér kutya reggel <jelzők><főnév>t<igei rész> ⇒
a nagy fehér kutya reggel <jelző><főnév>t<igei rész> ⇒
a nagy fehér kutya reggel meleg <főnév>t<igei rész> ⇒
a nagy fehér kutya reggel meleg húst <igei rész> ⇒
a nagy fehér kutya reggel meleg húst eszik

Úgy tűnik sikerült ezzel a nem túl nagyigényű nyelvtanunkkal egy épkézláb magyar mondatot levezetnünk.

Sajnos a magyar nyelv szabályai azért nem annyira egyszerűek. Így aztán nyelvtanunk generálja az alábbi mondatot is:

az fehér egér hideg sajtt eszik

Itt a kissé archaikus az névelő használata még úgy ahogy elfogadható, de a sajtt alak a sajt szónak sem nem tárgyesete, sem nem múlt ideje. Ennek ellenére nézzük el grammatikánk talán túlzott nagyvonalúságát, és tekintsük a kifogásolt mondatot is, a korábban levezetett, és valóban minden igényt kielégítő mondattal együtt korrekt magyar mondatnak.

Hogyan vélekedjünk azonban az olyan mondatról, mint

a fekete tej kutyat iszik

Vajon ez is korrekt magyar mondat?

Feltétlenül! A mondat ugyanis egy apró, és megállapodásunk szerint bocsánatos pongyolaságtól eltekintve minden tekintetben eleget tesz a magyar nyelv szabályainak. A matematikai nyelvészet szóhasználatával jól formált, vagy szintaktikusan helyes mondat.

Egy mondat nyelvtani, szintaktikai helyességéből semmi következtetést sem vonhatunk le a mondat igazság tartalmára. Valljuk be ugyanis, hogy ennek a mondatnak nincs nagy igazság tartalma, sőt őszintén szólva szemenszedett badarság.

Az a tény, hogy mondat igazság tartalma hamis, semmiképpen sem érinti a mondat helyes magyarságát. Hiszen ha létezne olyan nyelv, amelyen csakis igaz kijelentéseket lehet tenni! Sőt épp ellenkezőleg. Csakis azért tehetünk megállapításokat egy mondat igaz vagy hamis voltára vonatkozóan, mert a mondat szintaktikusan helyes, és így érthető.

Az olyan, a fenti grammatika szavaiból kialakított jelsorozat, mint

kutya hús reggel fekete eszik víz

szintaktikusan természetesen helytelen, számunkra emészthetetlen, és így fel sem merülhet az a kérdés, vajon a fenti szöveg igaz-e vagy hamis.

Annak feltétele, hogy egy mondat értelméről, szemantikájáról beszélhessünk az, hogy a mondat szintaktikusan helyes mondat legyen.

1.4. A *Chomsky*-féle nyelvosztályok

A generatív nyelveket tehát grammatikák segítségével jellemezhetjük. A grammatikákat meghatározó helyettesítési szabályok bonyolultsága alapján *Chomsky* a nyelveket osztályokba sorolta.

Chomsky négy nyelvosztályt definiált. Ezeket számokkal jelölte, így van **0**-ás, **1**-es, **2**-es és **3**-as nyelvosztály.

Az egyes nyelvosztályokban a helyettesítési szabályok alakjára vonatkozóan *Chomsky* az osztály sorszámának növekedésével egyre szigorúbb megkötéseket írt elő. Ezek szerint legkevésbé kötött nyelvtanú a **0**-ás nyelvosztály. Itt semmiféle külön megkötés nincsen. Természetesen az az előírás, hogy minden produkciós szabály baloldala tartalmazzon legalább egy nemterminális szimbólumot, – mint minden grammatikánál, – itt is érvényes.

Lássuk most az egyes nyelvosztályokra megadott korlátozásokat. Felsorolásunkban a nagyobb sorszámoktól haladunk a kisebbek felé.

A **3**-as nyelvosztály nyelvtanaiban csak kétféle szabálytípus engedélyezett. Ezek:

$$A \to a \qquad \text{illetve} \qquad A \to aB \tag{1.13.}$$

A helyettesítési szabály baloldala mindig egyetlen nemterminális, jobb oldalán pedig vagy egyetlen terminális szimbólum, vagy egyetlen terminális és egyetlen nemterminális szimbólumból álló jelsorozat.

Az ilyen alakú grammatikákat reguláris, pontosabban jobbreguláris nyelvtanoknak nevezzük. Amennyiben a második szabálytípusnál a két jobboldali szimbólum sorrendjét felcseréljük, vagyis a nemterminális szimbólumot követi a terminális szimbólum, akkor balreguláris grammatikáról beszélünk.

Ezek a nyelvtanok generálják a reguláris nyelveket. Itt a különbségtételnek nincs értelme, hiszen – mint azt később igazoljuk – a kétféle nyelvtan ugyanazt a halmazt generálja.

A **2**-es nyelvosztály helyettesítési szabályainak alakja:

$$A \to \alpha \tag{1.14.}$$

ahol α tetszőleges, mind terminális mind nemterminális szimbólumokat tartalmazható jelsorozat. A szabály baloldala itt is egyetlen, szükségképpen nemterminális szimbólum. Ezt úgy interpretálhatjuk, hogy egy adott A nemterminális szimbólum mindig helyettesíthető az α jelsorozattal, függetlenül attól mi a nemterminális környezete.

Minthogy a szabályok alkalmazása a környezettől, kontextustól független, ennek a nyelvosztálynak a nyelveit környezetfüggetlen, vagy angol rövidítésük alapján **CF** (*Context Free*) nyelveknek nevezzük.

Az **1**-es nyelvosztály helyettesítési szabályainak korlátait kétféle módon is jellemezhetjük. Az első megadási mód szerint a levezetési szabályok alakja:

$$\beta A \gamma \to \beta \alpha \gamma \tag{1.15.}$$

amit úgy interpretálhatunk, hogy az $A \to \alpha$ szabály csakis a β-γ környezetben alkalmazható. Ez az értelmezés magyarázza ezen nyelvosztály elnevezését, ezek a nyelvek a környezetfüggő, vagy angol rövidítésük alapján **CS** (*Context Sensitive*) nyelvek.

Az **1**-es másik lehetséges meghatározási módja szerint a szabályok alakja:

$$\alpha \rightarrow \beta \qquad \text{ahol} \qquad |\alpha| \leq |\beta| \tag{1.16.}$$

Jelsorozatok esetében a két függőleges vonal, amely máshol az abszolút érték jele, itt a jelsorozat hosszát adó operátor. A fenti megkötés tehát azt fejezi ki, hogy a helyettesítési szabályok jobboldala nem lehet rövidebb a baloldalnál. Ezen tulajdonság alapján ezeket a nyelveket nem csökkentő nyelveknek is nevezik.

Az **1**-es nyelvosztályra megadott két látszólag teljesen független meghatározás valójában, mint azt látni fogjuk, ugyanazt a halmazt definiálja.

A **0**-ás nyelvosztályban alkalmazható szabályokra nézve, mint már említettük, nincsen korlátozás.

Természetesen egy adott grammatika akkor tesz eleget egy nyelvosztály követelményeinek, ha a grammatika valamennyi szabálya megfelel a feltételeknek.

Vegyük észre, hogy a nyelvosztály sorszámát növelve a helyettesítési szabályok korlátozásai olyan értelemben súlyosbodnak, hogy egyúttal eleget tesznek az előző nyelvosztály követelményeinek is. Így például a **3**-as osztályú nyelvtan a **2**-es és az **1**-es nyelvosztály megkötéseit is kielégíti.

A figyelmes olvasónak bizonyára feltűnt, hogy jóllehet következetesen nyelvosztályokról beszéltünk, ezeket a nyelvtan levezetési szabályaira vonatkozó korlátozásokkal jellemeztük.

Pedig a nyelv és a nyelvtan két dolog. Egy nyelvtanhoz egy és csakis egy nyelv tartozik, nevezetesen az, amelyiket a nyelvtan generál. Egy nyelvnek viszont több nyelvtana is lehet, sőt az is megeshet, hogy különböző nyelvtanai különböző nyelvosztályok megkötéseinek tesznek eleget.

Az alkalmazott terminológia nem a szerző pongyolasága, hanem a matematikai nyelvészet általánosan elterjedt szóhasználata. Persze meg kell mondanunk, mit is értünk egy nyelvnek egy nyelvosztályhoz való tartozásán.

Egy nyelv hovatartozását az a legegyszerűbb nyelvtan határozza meg, amely a nyelvet generálni képes. Az egyszerűséget természetesen itt a nyelvosztályok jelentik, és nyilván a **3-**as nyelvosztály a legegyszerűbb.

Egy adott nyelv esetén tehát két dolgot kell megvizsgálnunk. Meg kell állapítanunk, hogy a generáló nyelvtan melyik nyelvosztályba tartozik, majd igazolnunk kell, hogy a nyelvnek nincsen egyszerűbb, azaz magasabb sorszámú nyelvosztályba tartozó nyelvtana.

Az első kérdésre a válasz roppant könnyű, csak végig kell olvasnunk a helyettesítési szabályokat.

A második probléma már fogas kérdés. Adott konkrét nyelv esetében néha intuitíve igazolható, hogy a szóban forgó nyelvtannál a nyelvnek egyszerűbb nyelvtana nem lehet. Általánosságban azonban a kérdést nem lehet megválaszolni.

Az irodalom ebben a tekintetben nem bakafántoskodik. A nyelvet szemrebbenés nélkül abba a nyelvosztályba tartozónak véli, amely a nyelvet generálja.

Nem törődik azzal, hogy egy nyelvet esetleg „jogtalanul" minősített a **2**-es nyelvosztályba, amikor pedig létezik **3**-as osztályú nyelvtana.

A nyelvek osztályba sorolásánál ez a könyv is – mint minden számítástechnikai publikáció – az előbbiekben ismertetett gyakorlatot követi.

1.5. A tartalmazás problémája – Eljárások és algoritmusok

A tartalmazás problémája a számítástechnikai nyelvészet egyik alapvető kérdése. Megfogalmazása a következő: Legyen adott egy nyelv grammatikájával, és egy Σ^* halmazbeli jelsorozat. Eldöntendő, hogy a megadott jelsorozat eleme-e a nyelvtanával meghatározott nyelvnek, generálható-e a szóban forgó jelsorozat az adott nyelvtannal.

Ennek kapcsán vizsgáljuk meg, hogyan keressünk választ egy szabatosan megfogalmazott kérdésre. Olyan műveletsorozatot, lépések egymásutánját kell megszerkesztenünk, amelynek végrehajtása révén választ kapunk kérdésünkre.

Például, ha az a feladatunk, hogy egy szám prímszám voltát megállapítsuk, akkor erre olyan műveletsorozatot tervezhetünk, amely a vizsgált számot minden nála kisebb számmal elosztja. Amennyiben az osztás mindenkor eredményez maradékot, akkor a szám prímszám, ha egyszer is sikerül a számot maradék nélkül osztanunk, akkor a szám nem prímszám.

Nem állítom azt, hogy ez egy hatékony megoldás, viszont nagyon könnyű lépésekre bontani, így nem sértem meg az olvasót azzal, hogy ennek részleteiben elmerülnék.

Ami a problémánk megoldására vezető lépéssorozatot illeti, két alapvető esetet kell megkülönböztetnünk. Az első esetben a választ véges számú lépés után mindenképpen megkapjuk, míg a második esetben sajnos nem garantálható, hogy véges lépésben, tehát valaha is választ kapjunk kérdésünkre.

Az első esetben algoritmusról, a másodikban eljárásról, procedúráról beszélünk. Ennek illusztrálására lássunk egy szabatosan meghatározott feladatot.

Tételezzük fel, hogy egész együtthatós egyenletek, polinomok gyökeit akarjuk meghatározni.

Tudjuk azt, hogy a racionális számok megszámlálhatóan végtelen sokan vannak. Generáljuk tehát sorban a racionális számokat, és helyettesítsük be ezeket egyenletünkbe.

Természetesen nem állítom, hogy ez egy hatékony módszer, a gondolatkísérletnek kizárólag illusztratív szerepe van.

Amennyiben első fokú egyenlettel van dolgunk, amelynek megoldása racionális szám, akkor bizonyosak lehetünk abban, hogy előbb-utóbb a megoldást

generálni fogjuk, és így véges sok lépésben megkapjuk a megoldást. Ne csüggedjünk tehát, ha már hosszabb ideje generáljuk hiábavalóan a racionális számokat. Folytassuk munkánkat abban a biztos tudatban, hogy állhatatosságunk előbb-utóbb megtermi a maga gyümölcsét, vagyis adott esetben az egyenlet megoldását.

Nagyobb fokszámú egyenletek, például másodfokú egyenletek esetében már nem lehetünk biztosak a dolgunkban. Itt nem tudhatjuk az időleges sikertelenség okát. Lehet, hogy az egyenletnek van racionális gyöke, csak ez a szám még nem került elő a generálás során. Ebben az esetben érdemes továbbfolytatni a generálást. Előfordulhat viszont az az eset, hogy nincsen racionális gyök, és ilyenkor bármeddig generáljuk is a racionális számokat, sohasem kapunk eredményt.

Módszerünk tehát az elsőfokú egyenletek esetében algoritmus volt, vagyis véges sok lépés után mindig kaptunk eredményt, magasabb fokszámú egyenletek esetében viszont csak procedúra.

Ebből természetesen nem lehet azt a következtetést levonni, hogy magasabb fokszámú egyenletek esetében nem szerkeszthető algoritmus. Így például a másodfokú egyenletek közismert megoldó képlete nagyon hatékony algoritmust szolgáltat, és választ ad abban az esetben is, ha a gyökök nem racionálisak, sőt akkor is, ha komplex számok. Itt pusztán arról van szó, hogy a megoldási módszert „ügyetlenül” választottuk meg.

A huszadik század harmincas éveinek egyik meglepő és nagy jelentőségű tudományos felismerése volt, hogy bizonyos problémákra nem szerkeszthető algoritmus, vagyis nem minden esetben kaphatunk választ kérdésünkre. Az ilyen feladatokra azt mondjuk, hogy algoritmikusan eldönthetetlenek.

Itt természetesen mindig egy feladatosztályról van szó. Ennek egyes konkrét reprezentánsai, feladatai esetleg megoldhatóak, véges sok lépésben kapunk kérdésünkre választ, annak ellenére, hogy általánosságban a feladatosztály algoritmikusan eldönthetetlen, vagyis minden esetben választ adó algoritmus nem szerkeszthető.

Sajnos a tartalmazás kérdése, a számítástechnika ezen alapvető feladata algoritmikusan eldönthetetlen. Szerencsére azért a helyzet ennyire nem sötét, az **1**-es nyelvosztály, és így természetesen a **2**-es és **3**-as nyelvosztály esetében szerkeszthető algoritmus a tartalmazás feladatára.

Az alábbiakban megadjuk egy ilyen algoritmus alapgondolatát.

Legyen adott egy *w* jelsorozat, és egy **G** grammatikájával adott meghatározott nyelv. Feltételezzük, hogy a nyelvtan kielégíti az **1**-es nyelvosztály követelményeit.

A megoldás kulcsa az a felismerés, hogy a nyelvtan nem csökkentő tulajdonságú. Generáljuk ugyanis az összes lehetséges levezetést abban a reményben, hogy előbb-utóbb eljutunk a *w* jelsorozathoz. Ha a generálás során egy mondathoz érünk, azt mindig összevetjük a vizsgált jelsorozattal. Amennyiben a két jelsorozat azonos a vizsgált jelsorozat mondata a nyelvnek.

Viszont valahányszor egy mondatszerű forma hossza meghaladja a *w* jelsorozat hosszát, akkor a további próbálkozásokat ezen az ágon abbahagyhatjuk, hiszen a nyelvtan nem csökkentő jellege miatt ebből a mondatszerű formából a *w* jelsorozat már biztosan nem vezethető le.

Minthogy egy adott hosszon belül az összes lehetséges mondatszerű formák száma véges, véges számú lépésben vagy megtaláljuk a szóban forgó jelsorozatot, vagy valamennyi levezetési utunk „befuccsol", és így kizárhatjuk a tartalmazást. Annak megoldása, hogy valamennyi levezetési utat bejárjunk, egyszerű technikai probléma.

Az összes lehetséges levezetés módszere a **0**-ás nyelvosztály esetében is alkalmazható. Amennyiben az azonosítandó jelsorozat eleme a nyelvnek, vagyis levezethető, akkor a generálás során előbb-utóbb megkapjuk ezt a jelsorozatot, hiszen ennek levezetése is benne van az összes lehetséges levezetésben. A probléma onnan adódik, hogy most nem hagyhatjuk abba a levezetést a túl hosszú mondatszerű formák esetében, hiszen a nyelv tartalmaz csökkentő szabályokat (ha nem így lenne, akkor a nyelv **1**-es osztályú lenne), és így a hosszúra nyúlt mondatszerű formák visszasoványodhatnak az elemzett jelsorozat hosszára. Éppen ezért itt az előbbi módszer nem algoritmus, hanem csak procedúra.

Abból, hogy az összes lehetséges levezetések módszere a **0**-ás nyelvosztály esetében nem szolgáltat algoritmust, hanem csak procedúrát, önmagában még nem következik, hogy a tartalmazás feladata a **0**-ás nyelvosztályban algoritmikusan eldönthetetlen, hanem csupán azt jelzi, hogy ezen az úton a feladat nem algoritmizálható. Mint később kitűnik, a feladat a **0**-ás nyelvosztályra valóban algoritmikusan eldönthetetlen, de ez külön igazolást igényel.

Az algoritmikus eldönthetetlenség kérdésére, és ezen belül a tartalmazás feladatára nagy fontossága miatt még visszatérünk. Itt csak utalni szeretnék arra, hogy egy feladat, pontosabban egy feladatosztály algoritmikusan eldönthetetlen voltának megállapítása olyan újszerű apparátust igényel, amely még nincs birtokunkban.

Megjegyzem, hogy a tartalmazás kérdése sajnos a számítástechnikai nyelvészet nem egyetlen olyan lényeges problémája, amelyről kiderül majd, hogy algoritmikusan eldönthetetlen.

1.6. Nyelvek és automaták

A számítástechnikai nyelvészet egyik alapvető, ha nem a legalapvetőbb problémáját, a tartalmazás kérdését az előző pontban tulajdonképpen megoldottuk.

Ez az általános algoritmus azonban messzemenően nem optimális, és a matematikai nyelvészet erre a feladatra, a tartalmazás kérdésének megválaszolására új matematikai objektumokat definiált, az automatákat.

Az automata persze ugyanolyan matematikai objektum, mint volt a nyelv és a nyelvtan, de szerencsére az automata már első látásra is jobban hasonlít a „civil" életből magunkkal hozott automata fogalom képzetére.

Az itt szereplő automatáknak – mint azt említettük – az a kizárólagos feladatuk, hogy választ adjanak a tartalmazás kérdésére. Minthogy a különböző nyelvosztályokhoz különböző bonyolultsági szintű nyelvtanok tartoznak, nem meglepő, hogy az egyes nyelvosztályokhoz különböző, egyre bonyolultabb automata szükségeltetik.

Minél korlátozottabbak a nyelvtan helyettesítési szabályai, annál egyszerűbb szerkezetű automata elegendő a tartalmazás feladatának megoldásához.

Bár ezek az automaták matematikai objektumok, mégis beszélhetünk „szerkezeti" felépítésükről. Tulajdonképpen itt olyan valóságos, tehát fizikai elemekből felépített automatát specifikálunk, amely pontosan ugyanúgy viselkedik, mint az absztrakt matematikai objektum.

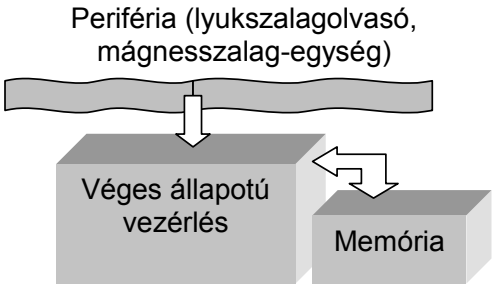
Egy automata – és ebben a tekintetben a matematikai és a köznapi értelemben vett automata között nincs különbség – mindig valamilyen meghatározott állapotban van. Például egy telefonautomata lehet kiinduló állapotban, lehet abban az állapotban, amikor a beszélőt már levettük, és pénzbedobásra vár, a következő állapot az lesz, amikor a tárcsázó hangra várunk, és így tovább. Épp így a matematikai automatának is van egy véges állapothalmaza, amelyből minden helyzetben egy és csakis egy érvényes. Amennyiben az automatát elemekből összetettnek képzeljük, akkor az automatának a véges állapothalmaz kezelésére alkalmas részét véges vezérlésnek nevezzük.

Minthogy automatánknak egy jelsorozatot, egy szöveget kell értelmeznie, kell legyen valamilyen input perifériája. Ez vagy egy olvasó, vagy egy író-olvasó berendezés. Az automata bonyolultságától függ, hogy megelégedhetünk-e csak olvasással, vagy mind olvasásra, mind írásra szükségünk van. Az előbbi esetben a perifériát mint lyukszalag olvasót, az utóbbiban mint mágnesszalag egységet képzelhetjük el.

A szalag, pontosabban a szalagon lévő értékes információ hossza az olvasó egység esetében triviális. Író-olvasó egységet alkalmazva korlátozhatjuk a szalag hosszát, de megengedhetünk nem korlátos szalaghosszt is.

Végül az automatának lehet memóriája is. A memória jelenléte, és hozzáférési módja szintén befolyásolja az automata bonyolultságát. Az író-olvasó

berendezéssel ellátott automaták természetesen felhasználhatják információk tárolására a bemenő szalagot is. Valóban az ilyen automaták élnek is ezzel a lehetőséggel.



1.1. ábra

Az 1.1. ábrán feltüntettük egy ilyen, a matematikai automatát modellező fizikai automata elvi rajzát.

Mint látni fogjuk, a négy különböző nyelvosztálynak négy különböző automataosztály felel meg. Ezek az automaták alkalmasak lesznek arra, hogy a tartalmazás kérdésére a választ – ha ez egyáltalában lehetséges – megadják.

2. Reguláris nyelvek

2.1. Reguláris nyelvek és véges automaták

A reguláris nyelveket a **3**-as osztályba tartozó grammatikák generálják. Az ilyen nyelveket elfogadó, a tartalmazás kérdésére választ adó automataosztály a véges automaták osztálya.

A véges automaták a legegyszerűbbek azon automaták közül, amelyek az előző fejezetben ismertetett általános automatából származtathatóak. A véges automata csak olvasni tud, tehát modellezésénél lyukszalag olvasót képzelhetünk el, és nincsen memóriája.

Elnevezését onnan kapta, hogy állapottere véges. Tulajdonképpen a névválasztás nem mondható túl szerencsésnek, hiszen valamennyi, a későbbiekben tárgyalt automata szintén véges állapotterű. Az elnevezést talán az indokolhatja, hogy erről az automatáról semmiféle más jellegzetesség nem mondható el.

A véges automatát, mint matematikai objektumot egy ötös jellemez:

$$\mathbf{M}\ (\ Q, \Sigma, \delta, q_0, F) \tag{2.1.}$$

ahol
- Q az automata állapotainak véges halmaza,
- Σ az elemzendő jelsorozat alfabetája,
- δ az automata mozgási szabályainak halmaza, amely szintén véges. A mozgási szabályok az automaták világában ugyanolyan meghatározó szerepet játszanak, mint a helyettesítési szabályok a grammatikák esetében. Ismertetésükre rögvest visszatérünk.
- q_0 az induló állapot. Az automata minden jelsorozat elemzését a q_0 állapotból kiindulva végzi. Minthogy q_0 szintén egy automataállapot

$$q_0 \in Q \tag{2.2.}$$

- F az elfogadó állapotok halmaza. Ez a halmaz az összes állapotok halmazának részhalmaza, így

$$F \subset Q \tag{2.3.}$$

A mozgási szabályok mondják meg, hogy egy adott állapotban egy adott karakter beolvasásának hatására milyen új állapotot vesz fel az automata. Így például

$$\delta(\ A, a\ ) = B \tag{2.4.}$$

mozgási szabály akkor alkalmazható, ha az automata az A állapotban van, és az olvasott karakter a. Az új állapot B lesz.

A mozgási szabályok összessége tulajdonképpen egy leképezés, amely az automataállapotok és az alfabeta karaktereinek direkt szorzatából álló halmazt képezi le az automataállapotok halmazára:

$$Q \times \Sigma \Rightarrow Q \tag{2.5.}$$

Erre a leképezésre semmiféle külön megkötést nem teszünk, így nem kívánjuk meg sem azt, hogy teljes, sem azt, hogy egyértelmű legyen. Ezek szerint lehetnek olyan állapot-karakter párok, amelyekre nincs mozgási utasítás, ugyanakkor olyan párok is előfordulhatnak, amelyekre egyidejűen egynél több mozgási szabály is vonatkozik. Ilyenkor az automata bármelyik mozgási szabályt követheti, az automata valamelyik szabály által meghatározott állapotba megy át.

Amennyiben minden állapot–karakter párhoz legfeljebb egy mozgási szabály tartozik, akkor az automata működése egyértelműen meghatározott, az automata determinisztikus. Ha minden állapot–karakter pár esetében van mozgási szabály, akkor az automata teljesen specifikált.

Mindannyiszor, amikor egy mozgási szabályt alkalmazva egy karaktert figyelembe vettünk, mintegy elolvastunk, a jelsorozat következő karaktere lesz érvényes. Fizikai automatánkban ezt úgy jellemezhetjük, hogy minden mozgás alkalmával az olvasófej alatt a szalag egy karakternyit elmozdul. Egy mozgássorozat eredményeképpen tehát a teljes jelsorozatot elolvashatjuk.

Az automata az elemzett jelsorozatot elfogadhatja, vagy visszautasíthatja. Az elfogadásnak két, egyidejűleg teljesítendő feltétele van. Az automatának a szöveget végig kell olvasnia, és az utolsó karakter elolvasása után az automatának elfogadó állapotba kell kerülnie. Az első feltételre azért van szükség, mivel nem kötöttük ki a leképezés teljes voltát, és így előfordulhat, hogy egy bizonyos szituációban, egy bizonyos állapot-karakter páros esetében az automata nem tud továbblépni, mivel nincsen alkalmazható mozgási szabály. Ilyenkor az automata anélkül áll meg, hogy a jelsorozatot végigolvasta volna.

A fentiek szerint ezt a helyzetet visszautasításnak tekintjük. A visszautasításnak ez a formája természetesen csak nem teljesen specifikált automaták esetében fordulhat elő.

Az automata mozgási folyamatát mint úgynevezett konfigurációk egymásutánját követhetjük nyomon. Valamely automata konfigurációja tulajdonképpen pillanatfelvétel az automata működéséről, amely mindazt az információt tartalmazza, amelynek alapján az automata további működése meghatározható

$$\boldsymbol{k}_0 \mapsto \boldsymbol{k}_1 \mapsto \ldots \mapsto \boldsymbol{k}_i \mapsto \ldots \mapsto \boldsymbol{k}_n \qquad (2.6.)$$

A $\mapsto$ jel egy operátor, amely az egymásból egyetlen mozgással, egyetlen mozgási szabály alkalmazásával elérhető konfigurációkat választja el. Amennyiben ki akarjuk hangsúlyozni, hogy melyik automata mozgási szabályairól van szó, akkor a $\mapsto$ műveleti jelhez indexként odaírjuk az automata nevét. Ha nem egyetlen, hanem tetszőleges számú lépésben elérhető konfigurációt jelölünk, akkor a már ismert módon kitevőben csillagot teszünk az operátor fölé, $\mapsto^*$.

Véges automaták esetében a konfiguráció az automata állapotát, és a még el nem olvasott jelsorozatot tartalmazza. A már elolvasott jelsorozat ugyanis csak

az olvasás eredményeképpen kiadódó állapot útján befolyásolja az automata működésének további menetét.

Itt tehát egy konfiguráció:

$$\boldsymbol{k} = (q, w) \tag{2.7.}$$

ahol – q egy automata állapot, tehát $q \in Q$

– w pedig tetszőleges jelsorozat, tehát $w \in \Sigma^*$.

A fentiek szerint a $\delta(A, a) = B$ mozgási szabály abban a konfigurációban alkalmazható, ahol az állapot A, és a még elolvasandó jelsorozat első karaktere a.

Egy automata $\boldsymbol{L}(\mathbf{M})$ nyelve alatt azon jelsorozatok halmazát értjük, amelyeket az automata elfogad. Eddigi jelöléseinkkel:

$$\boldsymbol{L}(\mathbf{M}) = \{ w \mid (q_0, w) \overset{*}{\mapsto}_{\mathrm{M}} (p, \varepsilon) \cap p \in F \} \tag{2.8.}$$

Az összefüggés szerint a nyelv elemei azok a w jelsorozatok, amelyek a kezdőállapottal olyan konfigurációt alkotnak, amely az **M** automata mozgási szabályait alkalmazva tetszőleges számú lépésben áttér olyan konfigurációra, ahol az olvasandó jelsorozat üres – magyarán az automata a teljes w jelsorozatot már elolvasta – és az állapot elfogadó állapot, vagyis az F halmaz eleme.

Definiáljunk most egy automatát az előbbiekben leírt módon:

$\mathbf{M}(\{S, A, B, C\}, \{a, b, c\}, \delta, S, \{C\})$

ahol $\delta = \{ (S, a) = A, (A, a) = A, (A, b) = B, (B, b) = B, (B, c) = C, (C, c) = \mathrm{C}\}$

Mint arról az olvasó könnyen meggyőződhet ez a leképezés nem teljes, ugyanakkor egyértelmű.

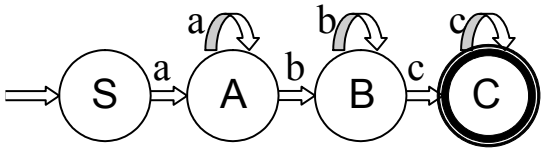
A véges automatákat egy gráffal szemléltethetjük, ami nagyon megkönnyíti tanulmányozásukat. A gráf irányított, és csomópontjai megfelelnek az automata állapotainak. Amennyiben egy állapotból egy adott karakter beolvasásának hatására az automata egy másik állapotba megy át, akkor a két állapotnak megfelelő csomópontokat egy, a régi állapotból az új állapotba mutató, és az olvasott karaktert mint nevet viselő éllel kötjük össze.

Így például, ha a mozgási szabályok között szerepel

$$\delta(A, a) = B$$

akkor az A csomópontból a B csomópontba egy a jelű él vezet.

A kiindulási állapotot kis nyíllal, az elfogadó állapotokat pedig kettős körrel jelölve tehetjük az ábrázolást egyértelművé.



2.1. ábra

A 2.1. ábra a fent definiált véges automatát tünteti fel. A működés követésére helyezzünk egy érmét a kezdőállapotra, és minden mozgásnál csúsztassuk

azt el a gráf élei mentén az új állapotba. Amennyiben a jelsorozat elolvasása után az érem egy elfogadó állapoton áll, az automata a jelsorozatot elfogadta. Könnyű belátni, hogy az ábrázolt automata az

$$a^i b^j c^k \qquad \mathbf{i, j, k > 0}$$

alakú jelsorozatokat fogadja el.

Az **M** automata $\boldsymbol{L}(\mathbf{M})$ nyelve azon jelsorozatok összessége, amelyeket az automata elfogad. A fenti összefüggés így az automata nyelvét adja.

Az automaták által elfogadott nyelvek éppen a **3**-as nyelvosztály elemei, tehát a véges automaták pontosan azokat a nyelveket fogadják el, amelyeket a reguláris nyelvtanok generálnak.

Ennek az alapvető és fontos állításnak igazolására minden véges automatához rendelünk egy reguláris nyelvtant, és minden reguláris nyelvtanhoz egy véges automatát olymódon, hogy az automata által elfogadott nyelv a nyelvtan által generált nyelv legyen.

Emlékeztetőül írjuk fel ismét a reguláris nyelvtanokban engedélyezett két szabálytípust:

$$A \to aB \qquad A \to a$$

Legyen adott egy véges automata. A hozzárendelt nyelvtan karakterkészlete azonos kell legyen az automata nyelvének karakterkészletével. Nem véletlen, hogy mindkettőt Σ jelöli.

Feleltessünk meg az automata minden állapotának egy nemterminális szimbólumot. Ezen belül a kezdőállapotnak a mondatszimbólum feleljen meg.

Az automata minden mozgási szabályához rendeljünk egy levezetési szabályt a következőképpen:

$$\delta(A, a) = B \qquad A \to aB \tag{2.9.}$$

Az automata elfogadó állapotainak megfeleltetett nemterminális szimbólumokhoz rendeljünk egy úgynevezett ε-szabályt, amelynek jobboldala az üres jelsorozat, ε:

$$A \to \varepsilon \tag{2.10}$$

Az ilyen szabályok szemantikája betűről betűre megegyezik az eddig tárgyalt szabályokéval. Ez annyit jelent, hogy a baloldalon szereplő nemterminálist a mondatszerű formában a jobboldallal, vagyis adott esetben az üres jelsorozattal kell helyettesíteni. Az ilyen ε-szabály alkalmazásakor tehát, a mondatszerű formából az adott nemterminális nyom nélkül eltűnik, elenyészik.

Az ilyen módon származtatott nyelvtan ugyanazt a nyelvet generálja, amelyet az automata elfogad. Mielőtt azonban ennek igazolásában elmerülnénk, végezzük el előbb a fordított műveletet, vagyis készítsünk valamely reguláris nyelvtan alapján véges automatát.

Az egységes tárgyalás kedvéért alakítsuk át kissé a nyelvtant, és vezessünk be egy új elenyésző nemterminálist. Legyen ennek neve *E*. Minthogy ez a nemterminális elenyészhet, egészítsük ki a nyelvtant ennek ε-szabályával:

$$E \to \varepsilon$$

Az *E* elenyésző nemterminálisnak a nyelvtanban nincs is más olyan szabálya, amelyben az a baloldalon szerepelne. Így ha egyszer az *E* nemterminális bekerül a mondatszerű formába, szükségszerűen azonnal megsemmisül.

Ezután egészítsük ki a második típusú levezetési szabályokat az imént bevezetett elenyésző nemterminálissal:

$$A \to a \qquad \text{helyett legyen} \qquad A \to aE \tag{2.11.}$$

Az ε-szabályokon kívül így nyelvtanunknak most már csak első típusú szabályai lesznek. Ezzel az átalakítással a generált nyelv nyilván nem változik, csupán az történt, hogy az eredeti nyelvtan utoljára alkalmazott, és a mondatszerű formát mondattá alakító második típusú szabály funkcióját két lépésben oldjuk meg. Először a (2.11.) szerint beírjuk az elenyésző nemterminálist, majd azt elemésztjük.

Itt minden nemterminálisnak feleltessünk meg egy automata állapotot, ezen belül a mondatszimbólum megfelelője a kezdőállapot legyen. Az állapotok közül azok lesznek elfogadó állapotok, amelyek ε-szabállyal bíró nemterminálisnak felelnek meg.

A helyettesítési szabályok átírása mozgási szabályokká az alábbi minta szerint történik:

$$A \to aB \qquad \delta(A, a) = B \tag{2.12.}$$

Az automaták és a nekik megfeleltetett reguláris nyelvtanok, illetve az automaták nyelvének és a reguláris nyelvtanok által generált nyelv azonossága a következőképpen látható be.

Először azt igazoljuk, hogy amennyiben a nyelvtan segítségével levezethető a *wA* mondatszerű forma, akkor létezik az automatának olyan mozgássorozata, amely a *w* bemenet hatására az automatát az *A* állapotba viszi át.

Ennek az állításnak a megfordítása is igaz, vagyis ha az automata a *w* jelsorozat elolvasása után az *A* állapotba kerül, akkor a mondatszimbólumból a nyelvtan segítségével levezethető a *wA* mondatszerű forma.

Eddigi jelöléseinkkel ezt a következőképpen írhatjuk le formális módon:

$$S \overset{*}{\Rightarrow}_{\mathrm{G}} wA \qquad \Leftrightarrow \qquad (q_0, w) \overset{*}{\mapsto}_{\mathrm{G}} (A, \varepsilon) \tag{2.13.}$$

Állításunkat teljes indukcióval igazolhatjuk. Tételezzük fel, hogy a (2.13.) összefüggés az **n** hosszúságú jelsorozatokra igaz. Legyen egy **n+1** hosszúságú jelsorozat alakja $w = va$, ahol $|v| = \mathbf{n}$, és *a* a jelsorozat utolsó karaktere. Minthogy feltételezésünk szerint az **n** hosszúságú *v* jelsorozatra állításunk igaz, így ha ennek hatására az automata a *B* állapotba mehet át, akkor a nyelvtan képes a *vB* mondatszerű forma generálására.

Ha most a következő karakter olvasásakor az automata az *A* állapotba kerül, vagyis létezik $\delta(B, a) = A$ alakú mozgási szabálya, akkor a (2.9.) megfeleltetés miatt a nyelvtan képes az $B \rightarrow aA$ helyettesítésre, és így generálhatja a *vaA* mondatszerű formát.

Fordítva, ha a nyelvtan alkalmas a *vB* mondatszerű formából a *vaA* mondatszerű forma származtatására, akkor a (2.12.) megfeleltetés miatt az automata a *va* jelsorozat elolvasása után felveheti az *A* állapotot.

Az, hogy ez a feltételezés igaz **n =1** esetére, triviális.

Amennyiben az automata állapota elfogadó, azaz a beolvasott jelsorozat az automata nyelvének egy mondata, akkor ennek az állapotnak megfelelő nemterminális elemészthető, és így a nyelvtan ugyanazt a mondatot képes generálni. Ez az okfejtés visszafele is helytálló, így a két nyelv, a reguláris nyelvtan által generált és a véges automata által elfogadott azonos.

Amennyiben az új ε-szabályok bevezetését nem találjuk rokonszenves megoldásnak, akkor ezt megkerülhetjük. Amikor a nyelvtanból készítünk automatát, akkor az automatában definiálnunk kell egy olyan, a nyelvtan egyik nemterminálisának sem megfeleltetett új *E* elfogadó állapotot. A második típusú helyettesítési szabályokhoz tartozó nyilakat ebbe az új automata állapotba irányítjuk.

$$A = a \qquad \delta(A, a) = E \tag{2.14.}$$

Vegyük észre, hogy az ily módon definiált elfogadó állapot funkcióját a korábbi konstrukcióban az elenyésző nemterminálisnak megfeleltetett automata állapot látta el.

Amikor az automatához szerkesztünk egy reguláris nyelvtant, megtehetjük, hogy az automata elfogadó állapotaiba vezető nyilakhoz nem egyetlen, hanem két, egy első és egy második típusú levezetési szabályt rendelünk.

Így, ha a *B* állapot elfogadó állapot, akkor a

$$\delta(A, a) = B$$

mozgási szabályhoz az alábbi két helyettesítési szabályt rendeljük:

$$A \rightarrow aB \qquad A \rightarrow a \tag{2.15.}$$

Így minden olyan esetben, amikor az automata elfogadó állapotba kerül, tehát a beolvasott jelsorozat mondat is lehet, a második típusú szabály segítségével a nyelvtan a nemterminálist megölve mondatot generálhat. Így az esetek nagy részében szerkeszthető olyan, az automata nyelvével azonos nyelvet generáló reguláris nyelvtan, amely ε-szabályokat nem tartalmaz. Probléma akkor adódik, amikor a kezdőállapot egyben elfogadó állapot is. Míg minden más állapotba ugyanis csakis mozgás útján juthatunk el, amikor a fenti (2.15.) szerinti átírás alkalmazható, addig a kezdőállapotba „beleszületünk” és emiatt a mozgással elérhető állapotok esetében bevált módszer nem alkalmazható.

Egyébként amennyiben a kezdőállapot is elfogadó állapot, akkor az üres jelsorozat, az ε is eleme a nyelvnek. Ez a mondat pedig csökkentő, tehát ε-szabály nélkül nem állítható elő. Így, ha mást nem is, de egy

$$S \to \varepsilon \tag{2.16.}$$

alakú szabályt mindenképpen be kell vennünk a nyelvtan szabályai közé.

Annyit mindenesetre rögzíthetünk, hogy amennyiben az üres jelsorozat nem eleme a nyelvnek, akkor az ε-szabály nélküli reguláris nyelvtannal generálható.

Hogyan kell értékelnünk azt a körülményt, hogy a nem csökkentő első osztályú nyelvtanok részhalmazaként említett reguláris nyelvtanok csökkentő szabályt tartalmaznak. Erre vonatkozóan a későbbiekben részletes és kielégítő magyarázattal fogok szolgálni. Most egyelőre hunyjunk szemet e felett a látszólagos anomália felett.

Ezzel egyrészt a véges automaták és az előbbi szabályok szerint konstruált reguláris nyelvtanok, másrészt a reguláris nyelvtanok és a belőlük származtatott automaták megfeleltetésének helyességét igazoltuk, vagyis a két nyelvosztály, a véges automaták és a reguláris nyelvek nyelvosztályának azonosságát bebizonyítottuk.

Törlesszük most egy régi adósságunkat, és mutassuk meg, hogy a jobbreguláris és balreguláris nyelvtanok ugyanazt a nyelvosztályt generálják. Itt elegendő, ha a balreguláris nyelvtanok és a véges automaták ekvivalenciáját látjuk be, hiszen a jobbreguláris nyelvek és a véges automaták nyelveinek azonosságát már igazoltuk.

Vegyük észre, hogy míg a jobbreguláris nyelvtanok szokásos írásmódunknak megfelelően, vagyis balról jobbra generálják a mondatokat, addig a balreguláris nyelvtanok jobbról balra generálnak. Az írási és olvasási irány persze csak konvenció kérdése, hiszen például a semita népek jobbról balra írnak és olvasnak.

Azt hiszem ezek után elegendő, ha csak az átírási szabályokat ismertetem, a szabatos igazolást átengedem a szorgalmas olvasónak.

A két nyelvtani szabálytípus átírása a következő:

$$A \to Ba \qquad \delta(B, a) = A \tag{2.17.}$$

$$A \to a \qquad \delta(q_0, a) = A \tag{2.18.}$$

Az eredeti nyelvtan mondatszimbólumának az automata egyetlen elfogadó állapota felel meg, míg az újonnan bevezetett q_0 állapot lesz az automata kiindulási állapota.

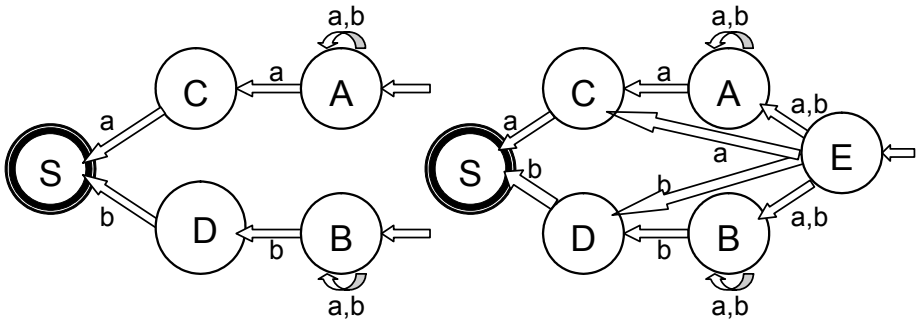
Amennyiben a nyelvtanban volna ε-szabály, akkor annak baloldalán álló nemterminális, pontosabban a neki megfelelő automata állapot itt értelemszerűen kezdőállapot lesz.

Természetesen bármilyen legyen is a nyelvtan, egy automatának nem lehet egynél több kezdőállapota, így ezeket egyesítenünk kell. A keveredés elkerülésére vezessünk be egy új kezdőállapotot, és minden az eredeti kezdőállapot jelöltekből kiinduló nyilat duplikáljunk olymódon, hogy a nyíl kiindulása az új kezdőállapot legyen.

Példaképpen szerkesszünk véges automatát az alábbi balreguláris nyelvtanhoz.

$$S \to Ca \mid Db \qquad C \to Aa \qquad D \to Bb$$
$$A \to Aa \mid Ab \mid \varepsilon \qquad B \to Ba \mid Bb \mid \varepsilon$$

A 2.2. ábra baloldalán a (2.17.) és (2.18.) összefüggések alapján szerkesztett és történetesen két kezdőállapottal bíró „torzszülött” automata van feltüntetve, az ábra jobboldalán az állapotok már közösítve vannak.



2.2. ábra

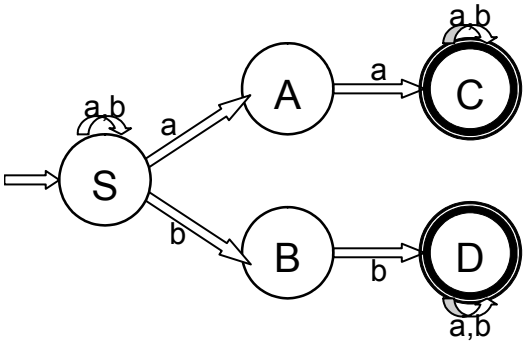
Az átalakítás megfordításánál, amikor egy véges automatából kell balreguláris nyelvtant készítenünk, akkor nehézséget okozhat, ha az automatának több elfogadó állapota van. Ilyenkor az automatát egyenértékű – ugyanazt a nyelvet elfogadó – és csak egyetlen elfogadó állapottal bíró automatává kell átalakítani.

Ezt úgy érhetjük el, hogy egy új, és egyetlen elfogadó állapotot vezetünk be. Ezzel egyidejűen az eredeti elfogadó állapotok elfogadó jellegét megszüntetjük.

Ezek után valamennyi, eredetileg elfogadó állapotba vezető nyilat duplikálunk olymódon, hogy a nyilak kiindulási állapotát nem változtatjuk meg, a célállapot azonban az új elfogadó állapot lesz. Amennyiben az olvasott karakter a jelsorozat utolsó eleme volt, akkor a duplikált nyíl segítségével az új elfogadó állapotba kell lépnünk, viszont ha ez nem az utolsó karakter, akkor az eredeti élen haladunk tovább. Persze ez az átalakítás még akkor sem eredményez determinisztikus automatát, ha az eredeti automata determinisztikus volt.

Arról, hogy egy automata több mozgási alternatíva esetében hogyan viselkedik, más szóval, hogyan kell kezelni a nemdeterminisztikus automatákat, a következő fejezetben részletesen szólunk.

Most példaképpen készítsünk egy véges automatához balreguláris nyelvtant, még akkor is, ha néhány kérdés még tisztázásra vár.



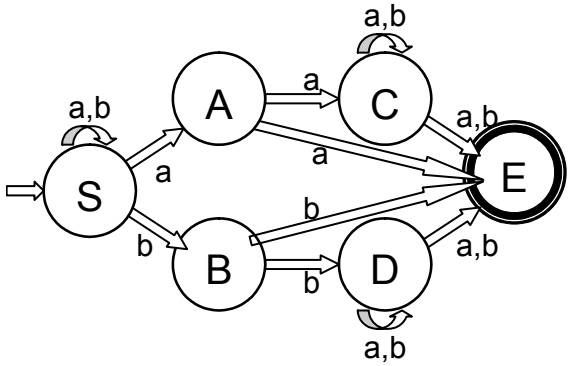
2.3. ábra

A 2.3. ábra automatájának két elfogadó állapota van. Az automata nyelve azokat az *a* és *b* karakterekből jelsorozatokat tartalmazza, amelyben akár két *a* karakter, akár két *b* karakter egymás után áll.

Az ismertetett átírási szabályokat alkalmazva a jobbreguláris grammatika a következő lesz:

$S \to aS$	$S \to bS$	$S \to aA$	$S \to bB$
$A \to aC$	$A \to a$		
$B \to bD$	$B \to b$		
$C \to aC$	$C \to bC$	$C \to a$	$C \to b$
$D \to aD$	$D \to bD$	$D \to a$	$D \to b$

A balreguláris grammatika származtatására alakítsuk át az automatát olymódon, hogy csak egyetlen elfogadó állapotot tartalmazzon. Ezt a módosítást tünteti fel a 2.4. ábra. Sportszerűen csakis az általunk ismertetett, és minden esetben alkalmazható szabályok szerint jártunk el, jóllehet itt sokkal egyszerűbb megoldás is kínálkozna.



2.4. ábra

Az átírási szabályok megfordításával könnyen megszerkeszthetjük balreguláris nyelvtan helyettesítési szabályait.

$$E \to Aa \qquad E \to Bb \qquad E \to Ca$$
$$E \to Cb \qquad E \to Da \qquad E \to Db$$
$$C \to Aa \qquad C \to Ca \qquad C \to Cb$$
$$D \to Bb \qquad D \to Da \qquad D \to Db$$
$$A \to Sa \qquad A \to a \qquad B \to Sb \qquad B \to b$$
$$S \to Sa \qquad S \to a \qquad S \to Sb \qquad S \to b$$

Ezzel demonstráltuk, hogy a jobb- és balreguláris nyelvtanok ugyanazt a nyelvosztályt generálják.

Felhívjuk az olvasó figyelmét, hogy egy reguláris nyelvtanban csak egyféle, vagy $A \to aB$ vagy $A \to Ba$ alakú szabályok lehetnek. A kétféle szabály egyidejű alkalmazása kivezet a reguláris nyelvek osztályából.

Álljon itt erre egy példa. Az alábbi nyelvtan

$$S \to aX \qquad X \to Sb \qquad X \to b$$

könnyen igazolhatóan az $a^i b^i$ nyelvet generálja. Ez a nyelv viszont – mint azt később bizonyítjuk – nem reguláris nyelv.

2.2. Determinisztikus és nemdeterminisztikus véges automaták

Ha az automata minden állapot–karakter párjához legfeljebb egy mozgási szabály tartozik, akkor az automata determinisztikus. Ilyenkor a mozgási szabályok értelmezése nyilvánvaló és egyértelmű.

Hogyan kell azonban értelmezni az automata működését, ha bizonyos állapot–karakter párokhoz egynél több mozgási szabály tartozik, az automata nemdeterminisztikus? Az általános definíciót, bár egy ízben már megadtuk, fontossága miatt megismételjük.

Egy automata akkor fogad el egy jelsorozatot, ha létezik olyan mozgássorozat, amelynek során az automata a teljes jelsorozatot elolvassa, és utána elfogadó állapotban áll meg.

A nemdeterminisztikus automaták esetében kellemetlen, hogy valamennyi mozgási lehetőséget egyidejűen kell figyelemmel kísérnünk, hiszen nem tudhatjuk, melyik mozgássorozat lesz sikeres, melyik vezet végül eredményre.

Ugyanakkor ez a definíció nemdeterminisztikus automatáknál is egyértelművé teszi a tartalmazás kérdését.

Az olyan automatákkal, a sztochasztikus automatákkal, ahol a több lehetőség közül az automata valamilyen törvényszerűség szerint mindig csak egyet vesz igénybe, tehát ahol előfordulhat, hogy egy automata egy jelsorozatot egyszer elfogad máskor meg nem, ez a könyv nem foglalkozik.

Az automaták „nem determinizmusát” egy számmal jellemezhetjük. Megadjuk a változatok számát annál az állapot–karakter párnál, amely az ilyen variánsokban leggazdagabb. Ez a szám determinisztikus automaták esetében pontosan egy, nemdeterminisztikusoknál egynél nagyobb.

A determinisztikus automata így felfogható, mint a nemdeterminisztikusok speciális esete, olyan nemdeterminisztikus automata, amelynek nem determinizmusa egy. Ebből viszont az is következik, hogy a determinisztikus automaták nyelvei a nemdeterminisztikus automaták, vagyis az összes véges automata nyelveinek részhalmazát alkotják.

Kérdés persze, hogy ez a részhalmaz valódi részhalmaz-e, létezik-e olyan a **3**-as nyelvosztályba tartozó nyelv, amelyet nem lehet determinisztikus automatával elfogadni.

A válasz nemleges, vagyis a determinisztikus automaták által elfogadott nyelvek halmaza megegyezik az összes lehetséges véges automaták által elfogadott nyelvek halmazával.

Ezt úgy igazoljuk, hogy megadunk egy mindig alkalmazható algoritmust, amely megmondja, hogyan lehet egy nemdeterminisztikus automatához egy vele ekvivalens, vagyis ugyanazt a nyelvet elfogadó determinisztikus automatát szerkeszteni.

Mint említettem a véges automaták gráfos megjelenítése alkalmas az automaták működésének követésére. Abban maradtunk, hogy egy érmét, például egy tízforintost teszünk a kiindulási állapotra, és a mozgások alkalmával ezt az érmét mindig az új érvényes állapotra csúsztatjuk át.

Determinisztikus automaták esetén, amikor minden helyzetben legfeljebb egyetlen mozgási lehetőség van, ez a pénzérme az egyetlen lehetséges állapotot mutatja.

Amennyiben automatánk nemdeterminisztikus, bizonyos helyzetekben több mozgási lehetőség is adódhat. Annak érdekében, hogy egyidejűen valamennyi mozgási lehetőséget nyomon követhessük, helyezzünk most érmét minden olyan állapotra, amely a mozgási szabályok útján elérhető.

Ennek semmilyen elvi akadálya sincsen, megvalósítása pusztán pénzkérdés, amennyiben most egynél több tízforintos érmére lesz szükségünk.

Természetesen ezt a gondolatot rekurzív módon minden lépésben véghez kell vinnünk. Egy adott helyzetben az összes megjelölt, tehát az addig elérhető állapotot megvizsgáljuk, és meghatározzuk, hogy a most beolvasott karakter hatására ezekből az állapottokból milyen állapotok érhetőek el.

Így figyelemmel tudjuk kísérni az összes lehetséges változat viselkedését, ami célkitűzésünk volt. Lényeges különbség azonban, hogy míg a determinisztikus automaták esetében egyetlen állapotot adtunk meg mint elérhető állapotot, addig itt egy állapothalmaz lesz az eredmény.

Ha most egy olyan új automatát definiálunk, amelynek állapottere az eredeti automata állapotterének hatványhalmaza, akkor nyomon tudjuk követni a nemdeterminisztikus automata lehetséges állapotait, mégpedig determinisztikus módon.

Legyen adott egy nemdeterminisztikus automatánk, és definiáljunk hozzá egy új, vele ekvivalens determinisztikus automatát.

A régi és az új automata karakterkészlete nyilván azonos.

Az új automata állapotai a régi automata állapotterének részhalmazai.

Az új automata azon állapotai lesznek az elfogadó állapotok, amelyek az eredeti automata állapotok olyan részhalmazának felelnek meg, amelyben legalább egy állapot az eredeti automata elfogadó állapota.

Jelölje $\mathbf{M}(Q, \Sigma, \delta, q_0, F)$

az eredeti feltételezésünk szerint nemdeterminisztikus automatát, míg

$$\mathbf{M}'(P, \Sigma, \delta', p_0, F')$$

a belőle szerkesztett determinisztikus automatát. Mint tisztáztuk a két automata alfabetája szükségképpen azonos, így megnevezésük is egyforma.

A determinisztikus automata állapotterét az eredeti automata állapotterének részhalmazai alkotják. Ezek szerint

$$P \subset 2^Q \tag{2.19a}$$

vagyis az új állapottér az eredeti állapottér hatványhalmazának részhalmaza. Részhalmaz, hiszen nem minden állapotkombináció fordul elő, így például az üres halmaz, amely definíció szerint eleme a hatványhalmaznak, nem eleme az új állapottérnek.

$$p_0 = \{q_0\} \tag{2.19b}$$

hiszen induláskor, anélkül, hogy karaktert olvasnánk be, csakis az eredeti kezdőállapot érhető el. Felhívom a gondos olvasó figyelmét, hogy az új kezdőállapot nem azonos a régi kezdőállapottal, hanem a régi állapottérnek egy olyan részhalmaza, amely egyedül az eredeti kezdőállapotot tartalmazza.

$$F' = \{ p_i \mid p_i \cap F \neq \varnothing\} \tag{2.19c}$$

mint tisztáztuk, az eredeti állapottér azon részhalmazai lesznek az új állapottér elfogadó állapotai, amelyek tartalmaznak eredeti elfogadó állapotot.

Végül az új automata mozgási szabályait az alábbiak alapján lehet származtatni:

$$\delta'(R, a) = \{ \bigcup_{q \in R} \delta(q, a) = T \quad R, T \in 2^Q \} \tag{2.19d}$$

Ezen formális leírás azt mondja, hogy az új állapottér R állapotában az a karakter beolvasásának hatására az új állapottér T állapotába kerülünk. Mind az R, mind a T az eredeti állapottér részhalmaza. A T által definiált részhalmazt úgy állapíthatjuk meg, hogy vesszük az R részhalmazába tartozó eredeti állapotokat, megnézzük, hogy ezekből az állapotokból az eredeti mozgási szabályok szerint milyen állapotokba tudunk eljutni, és képezzük ezeknek az unióját.

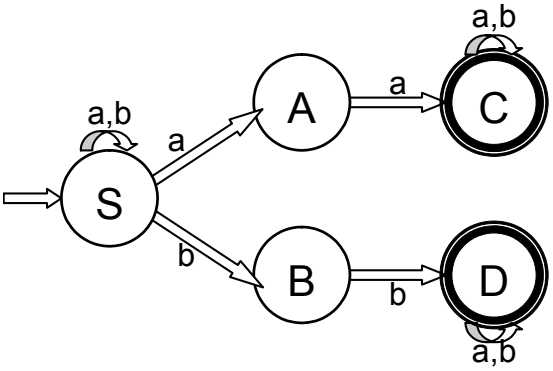
Világos, hogy az eredeti és az új most már bizonyosan determinisztikus automata nyelve azonos, és bármilyen is legyen az eredeti automata, az új automatát a leírt módszerrel mindig származtatni tudjuk. Ezzel igazoltuk, hogy a determinisztikus automaták valamennyi, a nemdeterminisztikus automaták által elfogadott nyelvet képesek elfogadni.

Sajnos a hatványhalmaz képzése nagyon megnöveli a halmaz számosságát. Szerencsére általában a helyzet nem ennyire kritikus, ugyanis a hatványhalmaz nem minden elemének van szerepe az új automatában.

A gyakorlatban úgy járhatunk el, hogy az eredeti nemdeterminisztikus automatát elemezve meghatározzuk az állapottér azon részhalmazait, amelyek előfordulhatnak mint egyidejűen elérhető állapotok. Ezeket és csakis ezeket a részhalmazokat vesszük aztán bele az új automata állapotterébe. Az egész folyamat kezdete az új automata kezdőállapota lesz, amely egyedül az eredeti automata kezdőállapotát tartalmazó részhalmaz. Ebből kiindulva állapíthatjuk meg, milyen részhalmazok figyelembe vételére van szükség.

Az algoritmust legjobban egy példán szemléltethetjük. Alakítsuk át a 2.3. ábrán feltüntetett, és szemmel láthatóan nemdeterminisztikus automatát, szabatosabban szerkesszünk egy az automatával ekvivalens, de determinisztikus automatát.

Az áttekinthetőség biztosítására tüntessük fel újra a szóban forgó ábrát.



2.5. ábra

Az egyszerűség kedvéért jelöljük az új automata állapotait egyetlen indexelt betűvel.

Az új automata kezdőállapota – mint azt megbeszéltük – egyedül a régi automata kezdőállapotát tartalmazó részhalmaz lesz:

$$q_0 = \{ S \}$$

Ebből az állapotból az *a* karakter beolvasásának hatására mind az *S* mind az *A* állapot elérhető. Ily módon az új automata következő állapota az előbbi két állapotot tartalmazó részhalmaz lesz:

$$q_1 = \{ S, A \}$$

Hasonlóan a *b* karakter beolvasása esetén

$$q_2 = \{ S, B \}$$

adódik.

Ha most a q_1 állapotból elérhető állapotokat keressük, akkor azokat az állapotokat kell egy részhalmazba összefognunk, amelyek az új állapotból, vagyis a q_1 halmaz bármelyik állapotából elérhetőek. Így tehát az új q_1 állapot és az *a* karakter esetében a következő állapotot alkotó halmaz elemei azok az eredeti automata állapotok lesznek, amelyek az *a* karakter beolvasásával vagy az *S* állapotból vagy az *A* állapotból elérhetőek. Minthogy az *S* állapotból az *S* és az *A* állapot, az *A* állapotból pedig a *C* állapot érthető el, a következő új állapot:

$$q_3 = \{ S, A, C \}$$

Ennek mintájára kaphatjuk meg az új automata többi állapotát is.

$q_4 = \{ S, B, D \}$ $q_5 = \{ S, B, C \}$ $q_6 = \{ S, A, D \}$

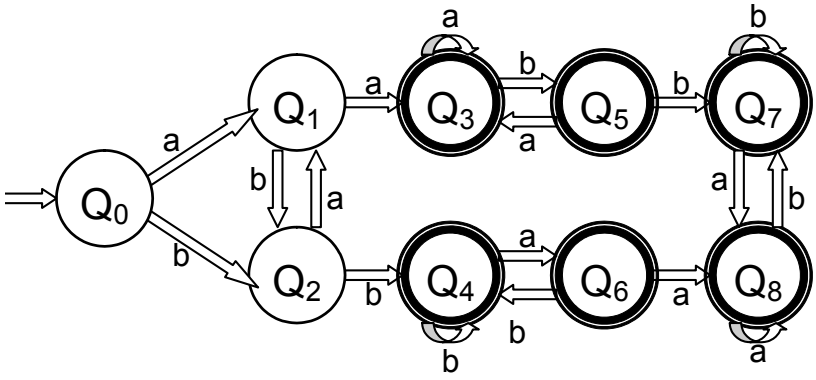
$q_7 = \{S, B, C, D \}$ $q_8 = \{ S, A, C, D \}$

Az ekvivalens determinisztikus automatát mutatja a 2.6. ábra. Az új automata elfogadó állapotai azok az új állapotok lesznek, ahol a részhalmaznak van olyan eleme, amely az eredeti automatában elfogadó állapot volt.

Két megjegyzés.

Amennyiben nem a 2.6. ábrán bemutatott automatából indultunk volna ki, hanem annak módosított, csupán egyetlen elfogadó állapotot tartalmazó változatából, akkor is ugyanerre a determinisztikus automatára jutottunk volna. Ez, ha nem is véletlen, de nem is szükségszerű.

A determinisztikus automatának 9 állapota van, ami lényegesen kevesebb, mint az 5 elemű eredeti állapottér hatványhalmazának $2^5 = 32$ lehetséges változata. Bár példánkban az újonnan szerkesztett automata terebélyesebb az eredetinél, ez nem szükségszerű. Előfordulhat, hogy a determinisztikus automata kevesebb állapotot tartalmaz, mint a nemdeterminisztikus elődje.



2.6. ábra

Minthogy tetszőleges nemdeterminisztikus automatához a fenti módon megszerkeszthetjük annak determinisztikus párját, igazoltuk azt az állításunkat, hogy a determinisztikus automaták nyelvei teljesen lefedik a **3**-as nyelvosztályt.

A determinisztikus automaták mintájára beszélhetünk determinisztikus nyelvtanokról és determinisztikus nyelvekről is. Egy reguláris nyelvtan akkor determinisztikus, ha a neki megfelelő véges automata determinisztikus. Egy nyelv akkor determinisztikus, ha determinisztikus nyelvtannal generálható.

Fenti eredményünket ezek után másképpen is megfogalmazhatjuk. A determinisztikus nyelvtanok alkalmasak a teljes **3**-as nyelvosztály generálására. Ennek következtében minden reguláris nyelv determinisztikus nyelv.

2.3. Minimálautomata

Ugyanúgy, ahogyan több nyelvtan generálhatja ugyanazt a nyelvet, egy nyelvnek több automatája is lehet. Korlátozzuk vizsgálatunkat determinisztikus és teljesen specifikált automatákra.

Az már ismeretes, hogyan lehet egy nemdeterminisztikus automatából vele ekvivalens determinisztikusat készíteni. Vizsgáljuk most azt, hogyan lehet egy nem teljesen specifikált automatát teljesen specifikálttá tenni.

Ha egy automata nem teljesen specifikált, az annyit jelent, hogy van olyan állapot–karakter pár, amelyre nincsen mozgási szabály. Vezessünk be egy új, csapdának nevezett állapotot. Ezt általában a csapda angol megfelelőjének a *trap* szónak kezdőbetűje, *T* jelöli. Ha valamelyik állapot–karakter párhoz nincsen mozgási szabály rendelve, akkor egészítsük ki szabályainkat egy olyan mozgással, amely erre az állapot–karakter párra az automatát a csapdaállapotba viszi át. Ezt természetesen valamennyi ilyen állapot–karakter párra megtesszük. A csapdaállapotból bármilyen karakter beolvasására a csapdaállapotba megyünk át, vagyis tulajdonképpen helyben maradunk. Ezek szerint, ha egyszer beleestünk a csapdába, akkor már nem tudunk belőle kimászni.

Teljesen világos, hogy az ily módon teljesen specifikálttá tett automata ugyanazt a nyelvet fogadja el, mint az eredeti. A kiindulási automata ugyanis bizonyos jelsorozatokat azért nem tudott elfogadni, mert mozgási szabály híján nem tudta azt végigolvasni. Most, mint minden teljesen specifikált automata, az új automata minden jelsorozatot végigolvas. Azonban abban a helyzetben, amikor az eredeti automata megállt, az új automata a csapdaállapotot veszi fel, ahonnan nem tud szabadulni, és minthogy ez az állapot visszautasító állapot, a mondatot nem fogadja el.

Ezek szerint bármilyen legyen is az eredeti automata, könnyen készíthetünk belőle determinisztikus teljesen specifikált automatát.

Keressük meg egy nyelv determinisztikus és teljesen specifikált automatái közül a legegyszerűbbet. Az egyszerűség mértékéül az állapotok száma szolgál. Ha egy automatának kevesebb állapota van, egyszerűbb. Ilyen legegyszerűbb automata természetesen létezik. A nyelvet elfogadó automaták mind véges állapotúak, így ezek között szükségképpen van minimális állapotszámú.

A probléma itt abban áll, hogy hogyan lehet ilyen minimális állapotszámú automatát megszerkeszteni, és unikális-e egy ilyen minimális állapotszámú automata. A második kérdésre adott válasz: minden nyelvhez lényegében egyetlen ilyen automata tartozik. Ennek neve minimálautomata.

A fentiek igazolására az automata valamennyi állapotához rendeljünk egy nyelvet. Ezt a nyelvet az az automata fogadja el, amely megegyezik az eredeti automatával, pusztán a kezdőállapotot helyeztük át a szóban forgó állapotra. Az eredeti kezdőállapothoz természetesen így az automata eredeti nyelve tartozik.

Értelmezzünk ezek után egy $\mathfrak{R}$ relációt az automata állapotai között. A és B állapotra akkor igaz az $\mathfrak{R}$ reláció, ha a két állapothoz rendelt két nyelv azonos. Ez a reláció ekvivalenciareláció. Ugyanis az A állapothoz ugyanaz a nyelv tartozik, mint az A állapothoz, vagyis $A\mathfrak{R}A$, tehát a reláció reflexív. Ha az A és B állapotok nyelvei azonosak, más szóval $A\mathfrak{R}B$, akkor természetesen a B és A állapotok nyelvei is azonosak, vagyis

$$A\mathfrak{R}B \quad \Rightarrow \quad B\mathfrak{R}A, \tag{2.23.}$$

tehát a reláció szimmetrikus. Végül, ha az A és B állapotokhoz valamint a B és C állapotokhoz tartozó nyelvek azonosak, akkor az A és C állapotokhoz ugyanaz a nyelv tartozik. Formálisan:

$$A\mathfrak{R}B \cap B\mathfrak{R}C \;\Rightarrow\; A\mathfrak{R}C \tag{2.24.}$$

tehát a reláció tranzitív.

Ezek szerint ez a reláció az automata állapotterét diszjunkt ekvivalenciaosztályokra osztja. Azonos ekvivalenciaosztályba azok az állapotok tartoznak, amelyeknek nyelve azonos. Különböző ekvivalenciaosztályhoz tartozó állapotok nyelve különböző.

Fogalmazzunk kicsit árnyaltabban. Amennyiben az A és B állapot ugyanabban az ekvivalenciaosztályban van, akkor nincsen olyan jelsorozat, amely különbséget tudna tenni a két állapot között. Ugyanis ha egy jelsorozat benne van a két állapot közös nyelvében, akkor a két állapotból indulva és ezt a jelsorozatot beolvasva mindkét esetben elfogadást kapunk eredményül. Ha viszont a jelsorozat nincsen benne a közös nyelvben, akkor mindkétszer elutasítunk. Amennyiben két állapot, C és D különböző ekvivalenciaosztályban, vagyis a hozzájuk rendelt két nyelv különböző, akkor mindig található olyan jelsorozat – hiszen egyébként a két nyelv azonos lenne – amely az egyik nyelvében benne van, a másikában viszont nincsen. Ezt a jelsorozatot a két automata állapotból indítva az egyik esetben elfogadjuk, a másikban visszautasítjuk. Tehát, ha a két állapot nincsen egy ekvivalenciaosztályban, akkor mindig létezik olyan jelsorozat, amely a két állapot között disztingvál.

Miután kiderült, hogy az azonos ekvivalenciaosztályba tartozó állapotokat semmiképpen sem lehet megkülönböztetni, logikus az a megoldás, amely csak egyetlen állapotot alkalmaz a teljes ekvivalenciaosztály helyett. Egyesítsük tehát az ekvivalenciaosztályok állapotait gondosan ügyelve arra, hogy gráf éleit megtartsuk.

Ezzel, ha volt olyan ekvivalenciaosztály, amely több állapotot tartalmazott, az automata állapotainak számát csökkentettük. Ez az automata lesz a minimálautomata. Ennek az automatának tehát nincs két azonos ekvivalenciaosztályhoz tartozó állapota.

Az a megállapítás, hogy a minimálautomata ugyanazt a nyelvet fogadja el, mint az eredeti, triviális. Hiszen, ha az eredeti automatában ugyanahhoz az ekvivalenciaosztályhoz tartozó két állapotba érkezünk, a továbbiakban ugyanazt az eredményt kapjuk, bármit is olvassunk be ezután.

Miután a minimálautomatát definiáltuk, felmerül a kérdés, hogyan lehet ezt az automatát egy adott automata alapján megszerkeszteni. A megoldás kulcsa az azonos ekvivalenciaosztályba tartozó állapotok megtalálása. Mint tudjuk, az ilyen állapotokat semmilyen jelsorozattal sem lehet megkülönböztetni. Ha tehát sorra vesszük az összes lehetséges jelsorozatot, és nem tapasztalunk különbséget, akkor a két állapot ekvivalenciáját sikerült megállapítanunk. Sajnos ez a módszer hosszadalmas eljárásnak ígérkezik, hiszen a lehetséges jelsorozatok száma, ami nem más, mint a Σ^* halmaz számossága, megszámlálhatóan végtelen.

Annak érdekében, hogy ezt a feladatot véges számú lépésben oldjuk meg, vezessünk be az állapotok között egy új relációt, pontosabban egy reláció családot. Két állapot legyen **i** ekvivalens, ha legfeljebb **i** hosszúságú jelsorozattal nem különböztethető meg. Az ekvivalencia eredeti meghatározása szerint két állapotot akkor mondunk ekvivalensnek, ha azokat bármely tetszőleges hosszúságú jelsorozattal sem tudjuk megkülönböztetni. Így az eredeti ekvivalencia az **i** ekvivalenciákból az $\mathbf{i} \to \infty$ átmenettel áll elő.

Ha ismerjük egy állapottér **i** ekvivalens ekvivalenciaosztályait, akkor ennek alapján az **i+1** ekvivalenciaosztályokat könnyen megállapíthatjuk. Nyilván ami nem volt **i** ekvivalens, nem lesz **i+1** ekvivalens sem. Azt kell csupán megvizsgálnunk, hogy az olyan **i** ekvivalens osztályokban, ahol egynél több állapot szerepelt, a különböző állapotokból egy újabb karakter beolvasására különböző **i**-ekvivalenciaosztályokba jutunk-e vagy sem. Ha útjaik eltérnek, akkor ezt a két **i** ekvivalens állapotot különböző (**i+1**)-ekvivalenciaosztályba kell sorolnunk. Ez azt jelenti, hogy ebből az ekvivalenciaosztályból több ekvivalenciaosztályt kell formálnunk.

Amennyiben egy újabb karakter figyelembe vételével nem növekszik meg az ekvivalenciaosztályok száma, akkor nyilván ez további karakterek esetében sem fog bekövetkezni, ami annyit jelent, hogy ez az **i** ekvivalencia azonos a jelző nélküli, vagyis az eredetileg definiált ekvivalenciával, így a mostani felosztás megegyezik az eredeti definíció szerinti ekvivalenciaosztályokkal.

Induláskor vegyük a **0** ekvivalenciát, vagyis tekintsük azt az esetet, amikor egyetlen jelsorozatot sem engedélyezünk, tehát csak ránézés alapján vagyunk képesek két állapot megkülönböztetésére. Ilyenkor csak az elfogadó és visszautasító állapotok között tudunk disztingválni, tehát két ekvivalenciaosztályunk lesz, az egyikbe a visszautasító, a másikba az elfogadó állapotokat soroljuk. Ezután növeljük az engedélyezett jelsorozatok hosszát mindaddig, amíg tovább már nem növekszik az ekvivalenciaosztályok száma. Ez a jelenség előbb utóbb szükségszerűen bekövetkezik, hiszen az állapotok száma véges, és az egyetlen állapotot tartalmazó ekvivalenciaosztályt tovább bontani már nem lehet. Így véges számú lépésben tudjuk meghatározni az eredetileg definiált ekvivalenciaosztályokat.

Mutassuk be ezt a módszert egy példán, a 2.6. ábra automatáján.

Kiindulásul írjuk fel a két **0-**ekvivalenciaosztályt. Az egyikbe a visszautasító, a másikba az elfogadó állapotok kerülnek:

$$\{ q_0, q_1, q_2\} \qquad \{q_3, q_4, q_5, q_6, q_7, q_8\}$$

A új karakter beolvasásánál végzett vizsgálatnál az éppen érvényes ekvivalenciaosztályok alapul vételével egy osztály állapotait egyenértékűeknek tekintjük. Két állapot nem egyenértékű volta abból tűnik ki, hogy van olyan karakter, amely a két állapotból két különböző osztályba visz.

Példánk automatájában egy karaktert alkalmazva kiderül, hogy a q_0, q_1 és q_2 állapotok nem egy ekvivalenciaosztályba tartoznak. Ugyanis például az *a* karakter a q_0 állapotból a visszautasító, míg a q_1 állapotból az elfogadó állapotok osztályába viszi át az automatát.

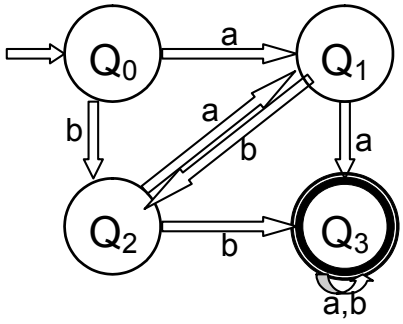
Ezek szerint az **1-**ekvivalenciaosztályokban a fenti három állapot különkülön képez egy-egy ekvivalenciaosztályt.

Az elfogadó állapotok osztályában ilyen finomítás nem következik be, mivel az osztály bármelyik állapotából bármely karakter osztályon belüli állapotba visz.

Ezek után az **1**-ekvivalenciaosztályok a következők:

$$\{q_0\} \quad \{q_1\} \quad \{q_2\} \qquad \{q_3, q_4, q_5, q_6, q_7, q_8\}$$

A következő karakter beolvasása – mint arról az olvasó könnyen meggyőződhet – nem hoz változást az osztályok felosztásában. Így a vizsgálatot befejezhetjük, megállapítva, hogy az elfogadó állapotok ekvivalensek, tehát egyetlen, a következő ábrán q_3 állapottal jelölt állapotba közösíthetőek. Ez lesz tehát a minimálautomata, és ezt mutatja a 2.7. ábra.

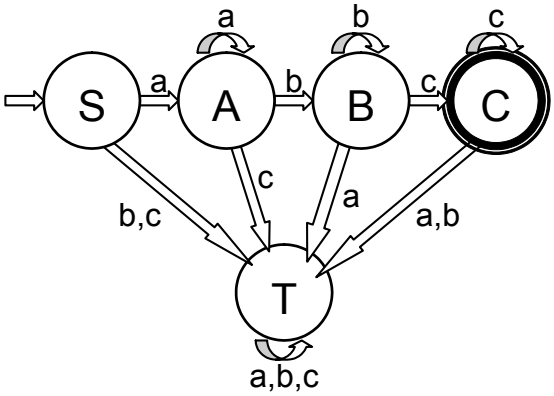


2.7. ábra

Minthogy determinisztikus és teljesen specifikált automatából indultunk ki, a minimálautomata is determinisztikus és teljesen specifikált lesz.

Néhány megjegyzés. A minimálautomata mindig determinisztikus és teljesen specifikált, tehát ha egy nemdeterminisztikus automatának keressük a minimálautomatáját, akkor előbb szerkesszünk az eredetivel egyenértékű determinisztikus és teljesen specifikált automatát, és azt redukáljuk.

Eddig is volt már dolgunk nem teljesen specifikált automatákkal. Ilyen volt például a 2.1. ábrán feltüntetett automata is. A 2.8. ábra ennek az automatának teljesen specifikált változatát mutatja be. Azt kell biztosítanunk, hogy minden állapotból, minden karakterre legyen kimenő él. Ahol ilyet a természet, vagyis az eredeti automata nem adott, ott nekünk kell behúzni egy, a csapdába vezető élt.



2.8. ábra

Így tetszőleges automatához megszerkeszthetjük a hozzátartozó minimálautomatát. Azt azonban még igazolnunk kell, hogy az ily módon megszerkesztett automata lényegében – izomorfiától eltekintve – unikális, azaz ez az egyetlen ilyen kis állapotszámmal bíró, az adott nyelvet elfogadó és teljesen specifikált automata.

Tételezzük ugyanis fel, hogy létezik egy másik automata, amely ugyanazt a nyelvet fogadja el, és állapotainak száma nem nagyobb a minimálautomatáénál. Nevezzük ezt a hipotetikus automatát ellenautomatának.

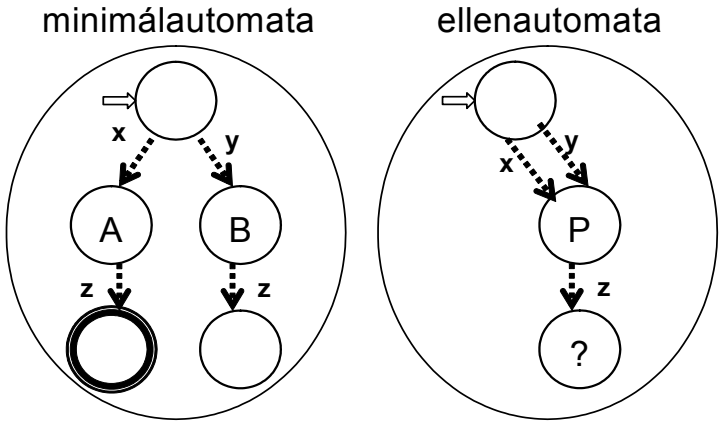
Legyen *P* az ellenautomata egy tetszőleges állapota. Tekintsük most azokat a jelsorozatokat, amelyek az ellenautomata kezdőállapotából kiindulva az ellenautomatát a *P* állapotba viszik át. Ezen jelsorozatok hatására a minimálautomata is saját kezdőállapotából kiindulva egyetlen állapotba – nevezzük ezt *A* állapotnak – kerül.

Amennyiben ez nem így lenne, ellentmondásra jutnánk abban a tekintetben, hogy a két automata ugyanazt a nyelvet fogadja el. Ha ugyanis az ellenautomatát a *P* állapotba vivő jelsorozatok elemzésekor kiderülne, hogy ezek a jelsorozatok a minimálautomatát egynél több állapotba juttatják, akkor létezne két olyan jelsorozat, mondjuk *x* és *y*, amely az ellenautomatát mindkét esetben a *P* állapotba, a minimálautomatát azonban két különböző, mondjuk, *A* és *B* állapotba viszi át.

A minimálautomata állapotai azonban megkülönböztethetőek, hiszen különböző ekvivalenciaosztályhoz tartoznak. Így szükségképpen található olyan *z* jelsorozat, amelyre a minimálautomata az *A,* illetve a *B* állapotból indítva eltérően viselkedik, egyszer elfogad, másszor visszautasít.

Ebből viszont következik, hogy a két automata nem fogadhatja el ugyanazt a nyelvet. Az *xz* és az *yz* jelsorozatokra ugyanis az ellenautomata azonosan, a minimálautomata pedig eltérően reagál.

Ezek szerint az ellenautomata egy állapotához a minimálautomatának csakis egy állapota tartozhat. Ez a megállapítás természetesen az ellenautomata minden állapotára helytálló.



2.9. ábra

Ha az ellenautomata több különböző állapotához ugyanaz a minimálautomata állapot tartozna, akkor az ellenautomata állapotainak a száma, ellentétben

feltételezésünkkel, felülmúlná a minimálautomata állapotainak számát. Ezt csak úgy kerülhetjük el, ha az ellenautomata különböző állapotainak a minimálautomata más-más állapota felel meg.

Ezek szerint az ellenautomata és a minimálautomata állapotai között kölcsönösen egyértelmű leképezés definiálható, más szóval a két automata izomorf, vagyis az állapotok elnevezésétől eltekintve azonos.

Ezzel a minimálautomata unicitására vonatkozó állításunkat igazoltuk.

Felhívom az olvasó figyelmét arra, hogy a minimálautomata nem a nyelvtanhoz, nem az azt megvalósító véges automatához, hanem a nyelvhez tartozik. Mindig hangsúlyoztam, hogy míg minden nyelvtanhoz és így automatához is egyetlen nyelv tartozik, addig egy nyelvnek számtalan nyelvtana illetve automatája lehet. A minimálautomatával más a helyzet. Minden reguláris nyelvhez – izomorfiától eltekintve – egy és csakis egy minimálautomata tartozhat.

Legyen adva két reguláris nyelvtanunk. Szeretnénk eldönteni, hogy a két nyelvtan ekvivalens-e, ugyanazt a nyelvet generálja-e vagy sem. Ennek a problémának egyik lehetséges megoldása, hogy megszerkesztjük a két automatát, azokból a két miniálautomatát, és ha a minimálautomaták izomorfak, a két nyelv azonos.

A későbbiekből kiderül, hogy két nyelvtan ekvivalenciáját általánosságban csak akkor lehet eldönteni, ha a két nyelvtan reguláris.

2.4. A két irányban mozgó véges automata

Nem hangsúlyoztuk ki külön, de természetesnek vettük, hogy a véges automata olvasáskor olyan „fizikai” mozgást végez, amelynek eredményeképpen folyamatosan és balról jobbra olvasunk. A mozgás során így az olvasófej alá egy új, sohasem látott karakter kerül.

Ez nem szükségszerű. Bonyolultabbá tehetjük az automata működését, ha megengedjük, hogy az egyes mozgásoknál a lyukszalag továbbítása tekintetében három lehetőség legyen. A jobbra lépés helyett balra is léphetünk, vagy esetleg helyben is maradhatunk.

Definiáljunk egy háromelemű halmazt:

$$\{ \boldsymbol{j}, \boldsymbol{h}, \boldsymbol{b} \} \tag{2.23.}$$

ahol ***j*** jobbra, ***h*** helyben és ***b*** balra jelentésű, és az automata mozgása során lehetséges lépéslehetőségeket jelöli. A mozgási szabályok így az állapothalmaz és a karakterhalmaz direkt szorzatát az állapothalmaz és a mozgási lehetőségeket kifejező fenti halmaz direkt szorzatára képezik le:

$$Q \times \Sigma \Rightarrow Q \times \{ \boldsymbol{j}, \boldsymbol{h}, \boldsymbol{b} \} \tag{2.24.}$$

Természetes, hogy az eddig tárgyalt egyszerű, egy irányban mozgó véges automaták a két irányban mozgók speciális esetei. Ha a mozgási irány tekintetében mindig a ***j*** lehetőséget választjuk, vagyis mindig az olvasottól jobbra álló lesz a következő karakter, akkor kapjuk az egyszerű véges automatákat.

Minthogy az egyszerű véges automaták a két irányban mozgók részhalmazát alkotják, az egyszerű automaták nyelve, vagyis a **3**-as nyelvosztály nyelvei a két irányban mozgó automaták nyelveinek részhalmaza.

Ismét felmerül a kérdés valódi-e ez a részhalmaz? A válasz ismét nemleges.

A bizonyítás, ha kissé bonyolultabb is, emlékeztet a determinisztikus és nemdeterminisztikus automaták ekvivalenciájának igazolására. Itt is kidolgozunk egy algoritmust, amelynek segítségével minden kétirányú automatához egy vele egyenértékű egyszerű véges automatát szerkeszthetünk.

Mielőtt a részletekre térnénk, ez a hely tűnik legalkalmasabbnak arra, hogy néhány szót szóljunk az automata fogalmáról.

Matematikai értelemben automata alatt mindig egy problémamegoldóképességre gondolunk, amelyet legjobban azzal a feladatosztállyal lehet jellemezni, amelynek megoldására az automata hivatott.

A véges automaták esetében ez a feladatosztály a reguláris nyelvek tartalmazási feladata. Egy automataosztály ilyenformán történő definiálása azonban túl elvont, és gyakorlatilag használhatatlan lenne.

Ezért általános az a módszer, amely az automatákat matematikai leírásukkal és fizikai modelljükkel adja meg. Kiderült azonban, hogy több hasonló felépítésű, de azért mégis különböző automata ereje, feladatmegoldó képessége azonos. A követett eljárás szerint ezen automaták közül egyet — lényegében önkényesen — kiválasztunk, és azt mondjuk, ez az automata.

A továbbiakban aztán bizonyítjuk, hogy a többi automataváltozat egyenértékű a normatívának elfogadott, mintegy kodifikált automatával.

A véges automaták világában a matematikai nyelvészet választása szerint ez az automata a nemdeterminisztikus és nem teljesen specifikált automata volt. A mozgási szabályok által definiált leképezésre ugyanis nem tettünk semmilyen megkötést, nem követeltük meg sem az egyértelműséget, sem a teljességet.

Megjegyzem a normatíva kiválasztása nem egységes, hiszen például az automataelméletben a determinisztikus és teljesen specifikált automatát tekintik a véges automata mintapéldányának.

Most egy új, első látásra feltétlenül nagyobb erejűnek látszó automata változatról bizonyítjuk be, hogy nem tud többet, mint a normatívának választott változat.

Mindenekelőtt az általánosság megsértése nélkül bátran korlátozhatjuk magunkat olyan automatákra, amelyek vagy a jobboldali, vagy a baloldali

karaktert veszik elő, helyben azonban nem maradnak. Ezzel a mozgási lehetőségek halmaza a {***j, b***} halmazra szűkül.

Ezt valóban megengedhetjük, hiszen ha az automata helyben marad, akkor a következő mozgást könnyen megállapíthatjuk. Az új állapot ugyanis ismeretes, a beolvasott karakter, mivel a szalag helyben maradt, a régi. A mozgásokat így addig követhetjük egyszerű módon nyomon, amíg a szalag jobbra vagy balra ki nem mozdul.

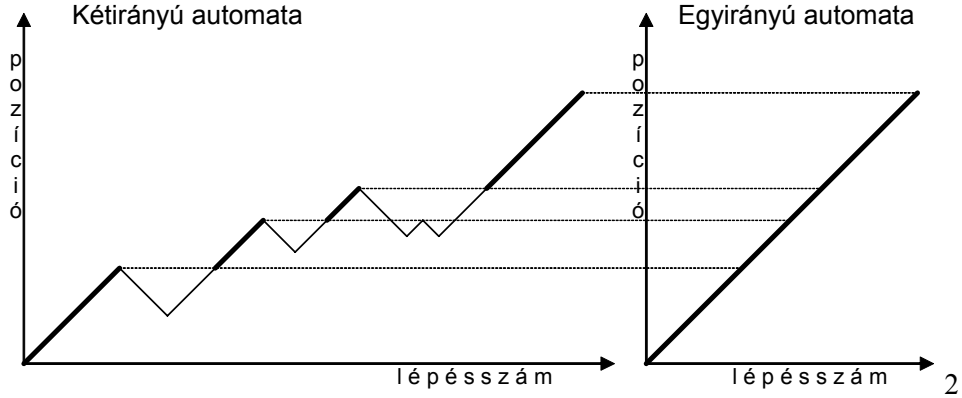
Megtehetjük, hogy a helyben járáshoz tartozó közbenső állapotokat kihagyva rögtön a tényleges helyváltoztatással járó mozgás eredményét adjuk meg.

Amennyiben elmozdulás nélkül lépve olyan szituáció adódik, amelyre nincs mozgási szabály, vagy a helyben járás végtelen ciklusban folytatódik, akkor már eleve a kiinduló állapot–karakter párhoz sem rendelünk mozgási szabályt, hiszen a fiaskó előre látható.

Így minden három mozgási lehetőséget figyelembe vevő automatához könnyen szerkeszthetünk olyan, csak két irányban mozgó automatát, amely vele egyenértékű, vagyis elegendő a csupán jobbra-balra mozgó automatákat vizsgálnunk.

Helyes, ha tisztázzuk az elfogadás feltételeit. Az automatánknak a szöveget végig kell olvasnia, mégpedig olymódon, hogy azon túlhalad. Az automatának tehát az utolsó olvasott karakter után, attól jobbra kell megállnia. Az elfogadás másik feltétele itt is az elfogadó állapot.

Ha megrajzoljuk a szalag mozgását a lépések függvényében, akkor az egyszerű automata esetében ez egy egyenes lesz, kétirányú automatánál viszont egy cikkcakkban haladó törtvonal. Itt ugyanis előrelépéseket visszalépések követhetnek, és ez a játék ismétlődhet.



.10. ábra

Az általunk szerkesztendő egyenértékű egyszerű automata csak akkor halad együtt a kétirányú automatával, amikor az új, még soha nem olvasott karaktereket tartalmazó szakaszon halad. A visszafelé megtett kerülőket, tévutakat az egyszerű automata nem követi.

A 2.10. ábrán a mozgási görbén azokat a szakaszokat, amelyeket a két automata együtt tesz meg, vastagítva rajzoltuk.

Mit csinál az egyszerű automata azalatt, amíg a kétirányú automata visszalépés után a már egyszer olvasott szakaszon bolyong? Várakozik. Vár arra, hogy a kétirányú automata újra feltűnjön a járt út végén, és ismét eddig nem olvasott karaktereket elemezzen.

Ez a helyzet nem áll elő teljes bizonyossággal. Elképzelhető, hogy a kétirányú automata megáll, vagy végérvényesen ott bolyong a már elolvasott szövegen. Ha azonban újra elérkezik arra a pontra, ahol vargabetűjét elkezdte, akkor fontos tudnunk, milyen állapotot vesz fel a kétirányú automata. Ennek alapján tudjuk ugyanis megítélni, hogy az automata átlépi-e a Rubicont, elolvassa-e a következő karaktert, vagy újabb vargabetűbe kezd.

Ennek megállapítására minden adat rendelkezésre áll. Ismerjük az automata mozgási szabályait, de ismerjük az elemzett karaktereket is, hiszen olyan szövegről van szó, amelyet az automata legalább egyszer már elolvasott. Akár azt is megtehetjük, hogy az olvasott szöveget memorizáljuk, és a kétirányú automata működését gondolatban lejátsszuk.

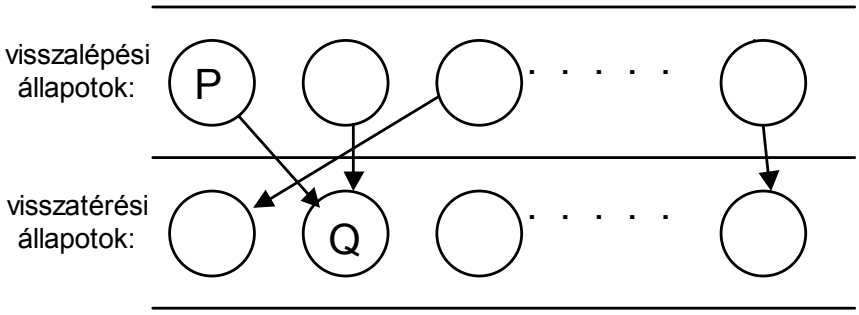
Az olvasott szöveg memorizálása csak elvi lehetőség. Az ilyen jelsorozatok hossza ugyanis nem korlátos, és így véges automatával nem memorizálható.

Valójában nincs is szükség ennyi, tehát nem korlátos számú információ tárolására, hiszen a potenciálisan végtelen sok jelsorozat csak véges sok különböző viselkedést tanúsíthat.

Helyettesítsük ugyanis az éppen olvasandó, az olvasófej alatti karaktert megelőző szöveget és a kétirányú automata ennek hatására történő működését egy fallal, nevezzük ezt visszatérési falnak.

Ezen a falon az automata állapotainak megfelelő nyílások vannak, mégpedig egy nyíláshalmaz a visszalépő, egy pedig a visszatérő állapotoknak. Úgy képzelhetjük el, hogy a nyílásokat a rexasztalhoz hasonlóan csatornák kötik össze, olymódon, hogy ezek a csatornák a visszalépő állapotokból indulnak ki, és a visszatérő állapotokba vezetnek. Például ha az automata a P állapotba lép be az olvasott szövegbe, és ilyen esetben a Q állapotban tér vissza, akkor a visszalépési állapotok P nyílását kell a visszatérési állapotok Q nyílásával összekötnünk, azaz felül a P állapotba bedobott labdát alul a Q állapotban kapjuk vissza.

Ezt a helyzetet igyekszik szemléltetni a 2.11. ábra. Természetesen, ha egy adott állapotba visszalépve az automata sohasem tér vissza, akkor ehhez a visszalépési állapothoz egy vak csatornát képzeljünk, az ide bedobott labda sohasem tér vissza, hanem ott marad.



2.11. ábra

A visszatérési fal „bekötése” függ a már elolvasott szövegtől, és jelsorozatról jelsorozatra változhat. A visszatérési fal konfigurációinak száma azonban véges és korlátos, ellentétben az olvasott szöveg változatainak számától.

A visszatérési fal ugyanakkor teljesen jellemzi a kétirányú automata működését az adott olvasott szöveg mellett. Éppen ezért az egyenértékű egyszerű automata állapottere a kétirányú automata állapotterének és a visszatérési falak halmazának direkt szorzata lesz. Minthogy mindkét halmaz korlátos, direkt szorzatuk is az, tehát megfelel egy véges automata állapotterének.

Amennyiben a kétirányú automata egy új karakter olvasásakor előrelép, ugyanezt teszi az egyenértékű egyszerű automata is, követve a kétirányú automata állapotát, ugyanakkor áttérve az új szöveg esetében érvényes visszatérési falra.

Ezt az áttérést, az új érvényes visszatérési fal megállapítását a korábbi visszatérési fal és a beolvasott új karakter ismeretében elemi úton tehetjük meg.

Amennyiben a kétirányú automata az új karakter olvasásakor visszalép, akkor az érvényes visszatérési fal alapján megnézzük, milyen állapotban tér majd vissza, ha egyáltalában visszatér, és olvassa el újból ugyanazt a karaktert.

Elképzelhető, hogy a kétirányú automata újból visszalép. Ilyenkor újból elvégezzük ezt a vizsgálatot, és így járunk el mindaddig, amíg valamilyen változást nem tapasztalunk. Ilyen jelenség lehet, hogy a kétirányú automata nem tér vissza, akár mert bent a szövegben fennakadt, akár mert bent végtelen ciklusba került. Előfordulhat, hogy a visszalépések alkotnak végtelen ciklust. Végül említsük meg azt az egyetlen kedvező esetet, amikor a kétirányú automata továbblépve előrehalad, és egy új, eddig még nem olvasott karaktert elemez.

Az első két változatnál az egyenértékű egyszerű automata nem lép, hanem megáll, illetve, ha teljesen specifikált automatát készítünk, akkor a csapdaállapotba megy át. A harmadik változatban az egyenértékű egyszerű automata az utolsó előre történő lépést a kétirányú automatával együtt teszi meg.

Mint említettem, az új visszatérési falat az előző falból és az olvasott karakterből lehet megszerkeszteni. Valahonnan azonban el kell indulnunk, a legelső visszatérési falat közvetlenül kell meghatározni.

Az első karakter olvasása előtti fal nagyon egyszerű és áttekinthető. Minthogy az első karakter előtt nincsen karakter, esetleges visszalépés esetében nincs mit olvasni, és a kétirányú automata kénytelen megállni. Itt minden visszatérési állapot vakon végződik, az automata sohasem tér vissza.

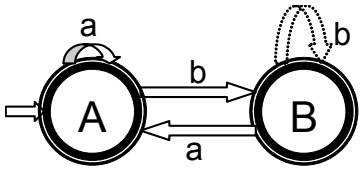
Lássunk most egy példát kétirányú automatára és a vele egyenértékű egyszerű automata megszerkesztésére.

Legyenek a kétirányú automata mozgási szabályai az alábbiak:

$$\delta(A, a) = (A, \boldsymbol{j}) \qquad \delta(A, b) = (B, \boldsymbol{j})$$
$$\delta(B, a) = (A, \boldsymbol{j}) \qquad \delta(B, b) = (B, \boldsymbol{b})$$

Legyen továbbá a kezdőállapot *A,* és legyen mind az *A,* mind a *B* elfogadó állapot. Adott esetben ez persze nem jelenti azt, hogy az automata minden jelsorozatot elfogad, hiszen előfordulhat, hogy a visszalépések miatt bizonyos jelsorozatokat az automata nem tud végigolvasni.

Ha most az előrelépéseket folytonos, a visszalépéseket pedig szaggatott nyilakkal jelöljük, a két irányban mozgó automata rajza a 2.12. ábrán látható.



2.12. ábra

Az egyenértékű egyszerű automata állapotait jelölje q és egy index, a visszatérési falakat pedig Δ, és itt is indexszel különböztessük meg az egyes falakat. Az induló fal jele így Δ_0.

Az egyszerű automata kezdőállapota a kétirányú automata *A* kezdőállapotából, és az induló Δ_0 visszatérési falból áll:

$$q_0 = (A, \Delta_0)$$

Vizsgáljuk meg, mi történik akkor, ha a q_0 állapotban *a* karaktert olvasunk be.

A kétirányú automata előrelép, és az *A* állapotba kerül. Milyen visszatérési fal tartozik majd az új állapothoz? Ehhez meg kell vizsgálnunk, hogyan viselkedik az automata, ha a Δ_0 falat egy *a* karakter követi. Jelölje az új falat Δ_1. Ilyenkor szimbolikusan a helyzetet a következőképpen jelölhetjük

$$\Delta_0 \mid a \mid \Delta_1$$

Akár az *A* akár a *B* állapotban lépünk is be a Δ_1 falba, onnan az *A* állapotban térünk vissza. Ebből következik, hogy a Δ_1 fal esetében mind az *A,* mind a *B* visszalépési állapotot az *A* visszatérési állapottal kell összekötni.

Az is érzékelhető, hogy az *a* karaktert mindig a Δ_1 visszatérési fal követi, hiszen az automata az *a* karaktert megelőző falba sohasem lép vissza, így az nem befolyásolhatja az *a* karaktert követő falat, annak konfigurációja érdektelen.

Ezek szerint a q_0 állapotban az *a* karakter beolvasására a q_1 állapot következik, amelyet az alábbi pár alkot:

$$q_1 = (A, \Delta_1)$$

Ha a q_0 állapotban *b* karaktert olvasunk be, akkor a két automata ismét lép, az egyenértékű automata állapotának egyik komponense a *B* állapot lesz, a másik pedig a Δ_2 fal, amelyet a következő szituációból határozhatunk meg:

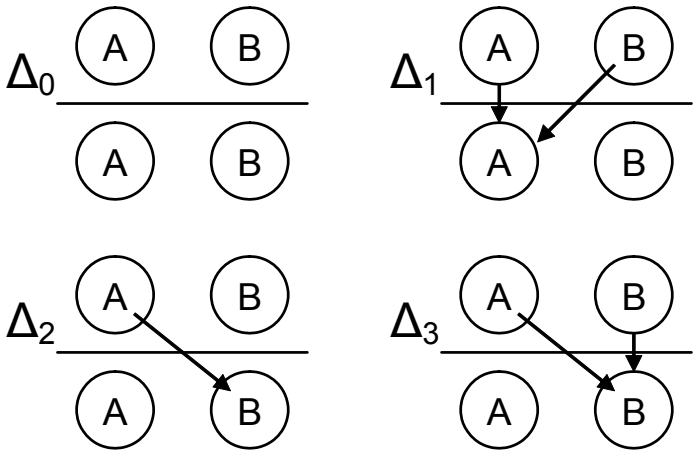
$$\Delta_0 \mid b \mid \Delta_2$$

Amikor az *A* állapotban lépünk be a Δ_2 falba, akkor onnan a *B* állapotba térünk vissza. Ha a visszalépési állapot *B*, akkor tovább kell visszalépnünk, ezúttal a Δ_0 falba, ahonnan viszont nincs visszatérés.

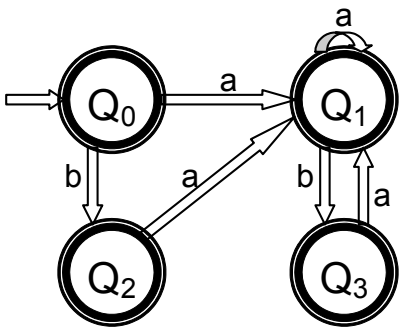
A Δ_2 falnál tehát csak az *A* visszalépési állapotot kell a *B* visszatérési állapottal összekötni, a *B* visszalépési állapotból csak vakvágány vezet.

A példa további szerkesztései, amelyek ugyanerre a kaptafára mennek, talán nem igényelnek részletesebb magyarázatot. Az alábbiakban megadjuk az egyenértékű automata állapotait és mozgási szabályait. A 2.13. ábrán feltüntettük az egyes visszatérési falak „kapcsolását”, és az egyenértékű automata gráfját is megadtuk.

A falak:



2.13.a. ábra



2.13.b. ábra

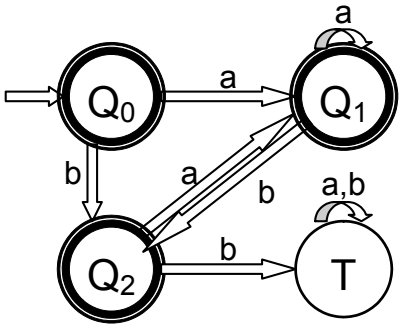
Az állapotok: $q_0 = (A, \Delta_0)$ $q_1 = (A, \Delta_1)$
$q_2 = (B, \Delta_2)$ $q_3 = (B, \Delta_3)$

a mozgási szabályok: $\delta(q_0, a) = q_1$ $\delta(q_0, b) = q_2$
$\delta(q_1, a) = q_1$ $\delta(q_1, b) = q_3$
$\delta(q_2, a) = q_1$ $\delta(q_3, a) = q_1$

A kapott automata determinisztikus, de nem teljesen specifikált, hiszen a q_2 és q_3 állapotokban a *b* karakter olvasásakor nincsen mozgási szabály.

Az automata nyelve azon *a* és *b* karakterekből álló jelsorozatokat tartalmazza, amelyekben nem áll két *b* karakter egymás mellett.

A csapdaállapottal kiegészített, tehát teljesen specifikált egyszerű automata minimálautomatáját tünteti fel a 2.14. ábra.



2.14. ábra

Fejtegetéseinkben hallgatólagosan feltételeztük, hogy a kétirányú automata determinisztikus. Ez nem előfeltétel. Az algoritmus kismértékű módosítással alkalmazható nemdeterminisztikus két irányban mozgó véges automaták esetében is. Ennek átgondolását az olvasóra bízom.

Minthogy ezek szerint minden kétirányú mozgást végző véges automatához szerkeszthető vele egyenértékű, vagyis ugyanazt a nyelvet elfogadó egyszerű automata, igazoltuk, hogy a két irányban mozgó automaták ereje,

feladatmegoldó képessége nem nagyobb az egyszerű véges automatákénál. Mindkét automataosztály ugyanazt a nyelvosztályt fogadja el.

2.5. Műveletek nyelvekkel

A nyelvek, mint tisztáztuk, matematikai objektumok, nevezetesen jelsorozatok halmazai. Mint az a matematikában általános a matematikai objektumokra műveleteket szokás értelmezni. Ez alól a nyelvek sem kivételek.

A nyelvekre nagyon sok művelet értelmezhető. Mi egyelőre megelégszünk az öt legfontosabb nyelvi művelet tárgyalásával.

A nyelvek jelsorozatok halmazai. Így vannak olyan műveletek, amelyek forrása a halmazelmélet, és vannak olyanok, amelyeket a jelsorozat jelleg határoz meg.

Ezek a műveletek a komplemensképzés, az unió és a metszet művelete, továbbá a konkatenáció és a tranzitív lezárás. Az első három a halmazelméletből van átvéve, a további kettőben a jelsorozat jelleg érvényesül.

Egy nyelv komplemensébe azok a jelsorozatok tartoznak, amelyek nem elemei a nyelvnek. Amikor komplemensről beszélünk, mindig tudnunk kell milyen halmazra vonatkozó komplemenst keresünk. Ez az alaphalmaz az univerzum. Tudnunk kell tehát, mit tekintsünk adott esetben univerzumnak.

Amennyiben Σ alfabeta felett értelmezett nyelvről van szó, akkor a Σ elemeiből alkotott összes lehetséges jelsorozat adja az univerzumot. Ez nem más, mint a Σ^* halmaz.

Egy $\boldsymbol{L}$ nyelv $\overline{\boldsymbol{L}}$ komplemensét tehát azok a Σ^* halmazbeli elemek alkotják, amelyek nem mondatai a nyelvnek. Ez a definíció teljes megegyezésben van a komplemens halmazelméleti értelmezésével.

Két $\boldsymbol{L}_1$ és $\boldsymbol{L}_2$ nyelv $\boldsymbol{L}_1 \cup \boldsymbol{L}_2$ uniója azon jelsorozatok összessége, amelyek vagy az $\boldsymbol{L}_1$ vagy az $\boldsymbol{L}_2$ nyelv mondatai. A *vagy* itt természetesen nem kizáró értelmű. Ez a definíció is pontosan követi a halmazelméleti értelmezést.

Két $\boldsymbol{L}_1$ és $\boldsymbol{L}_2$ nyelv $\boldsymbol{L}_1 \cap \boldsymbol{L}_2$ metszete azokat a jelsorozatokat tartalmazza, amelyek mind az $\boldsymbol{L}_1$ mind az $\boldsymbol{L}_2$ nyelv mondatai. Ez is egybevág a halmazelméleti értelmezéssel.

Az $\boldsymbol{L}_1$ és $\boldsymbol{L}_2$ nyelv ebben a sorrendben vett $\boldsymbol{L}_1 \bullet \boldsymbol{L}_2$ konkatenáltján azon jelsorozatok halmazát értjük, amelyeket ketté lehet vágni, fel lehet osztani olymódon, hogy a sorozat eleje az $\boldsymbol{L}_1$, a sorozat második fele az $\boldsymbol{L}_2$ nyelv mondata. Nincs olyan követelmény, hogy a szétvágás unikális legyen. Megjegyzem, hogy míg az unió és a metszet művelete kommutatív, addig a konkatenáció nem az, de könnyen belátható, hogy asszociatív. A konkatenáció jelölésénél gyakran elhagyjuk a $\bullet$ operátort, és az egymás mellé írt nyelvekre a konkatenáció műveletét értjük. Ez megegyezik az aritmetikában a szorzás műveletének jelölésével, ahol szintén elhagyhatjuk az operátort.

A konkatenáció bevezetésével értelmezhetjük egy nyelv önmagával vett konkatenáltját. Ezt az $\boldsymbol{L \bullet L}$ konkatenáltat a szorzási analógia miatt a nyelv négyzetének nevezzük, és az $\boldsymbol{L^2}$ jelölést is használhatjuk. Természetesen nem csak egy nyelv négyzetét értelmezhetjük ily módon, hanem akárhanyadik hatványát is. Így $\boldsymbol{L^i}$ a nyelv **i**-szeres konkatenáltja, amely azokat a jelsorozatokat tartalmazza, amelyeket fel lehet vágni **i** számú darabra olymódon, hogy mindegyik darab mondata az $\boldsymbol{L}$ nyelvnek.

Az aritmetikában, mint ismeretes, bármely szám nulladik hatványát az egységnek értelmezzük. Ez biztosítja ugyanis, hogy azonos alapú hatványok multiplikációját a kitevők addíciójával oldjuk meg. Hasonló módon bármely nyelv nulladik hatványának azt a nyelvet tekintjük, amelynek egyetlen eleme az üres jelsorozat, ε. Ugyanúgy ahogy bármely számnak az egységgel való szorzata a számot szolgáltatja, bármely nyelvnek ezzel az egységnyelvvel való konkatenációja a kiindulási nyelvet adja vissza.

A nyelvek hatványozásának ismeretében definiálhatjuk egy $\boldsymbol{L}$ nyelv $\boldsymbol{L^*}$ tranzitív lezártját. A tranzitív lezárt a nyelv valamennyi hatványának uniója, vagyis azokat a jelsorozatokat tartalmazza, amelyeket fel lehet úgy darabolni, hogy minden darab a nyelv mondata legyen. A darabok számára vonatkozóan nincsen megkötés. Formálisan ezt az alábbi összefüggés fejezi ki:

$$\boldsymbol{L^*} = \bigcup_{i=0}^{\infty} \boldsymbol{L^i} \tag{2.25.}$$

Felhívom a figyelmet arra, hogy az unióképzés a nulladik hatványnál kezdődik, tehát egy nyelv tranzitív lezártjának mindig eleme ε, az üres jelsorozat. Fogjuk fel a Σ alfabetát nyelvnek, amelynek elemei az alfabeta karakterei. Régi ismerősünk a Σ^* tehát nem más, mint a Σ nyelv tranzitív lezártja. Vegyük észre, hogy a csillag kitevő alkalmazása a (2.25.) képletben megfelel az eddigi értelmezéseknek.

A nyelveken értelmezett műveletek bevezetésekor rögtön felmerül a kérdés, mennyiben zártak az egyes nyelvosztályok a különböző műveletekre?

Akkor mondjuk, hogy egy halmaz egy műveletre zárt, vagy más szavakkal a művelet nem vezet ki a halmazból, ha a műveletet a halmaz tetszőleges elemén vagy elemein elvégezve, az eredmény ismét csak a halmaz eleme lesz.

A reguláris nyelvek halmaza a fenti műveletekkel szemben igen rokonszenvesen viselkedik, amennyiben valamennyi műveletre nézve zárt.

Mielőtt mind az öt műveletre külön-külön bebizonyítanánk a reguláris nyelvek zártságát, néhány megjegyzést tennék.

Minthogy a reguláris nyelvtanok által generált, és a véges automaták által elfogadott nyelvek halmazának azonosságát már igazoltuk, a zártság feltételeit elegendő csupán az egyik területen, akár a nyelvtanok, akár az automaták oldaláról belátnunk. Így élünk azzal a könnyebbséggel, hogy az igazolást ott végezzük el, ahol az érzékletesebb.

Vegyük ezután sorra a már említett műveleteket.

A komplemens képzésénél induljunk ki az automatából. Itt az alapgondolat nyilván az lehet, hogy cseréljük fel az állapotok jellegét. Ami eddig elfogadó állapot volt legyen visszautasító, és fordítva, a visszautasító állapotokból legyenek elfogadó állapotok.

Ez az ötlet valóban célhoz vezet, csak óvatosan kell alkalmazni. Ha ugyanis az automata a szöveget nem tudja végigolvasni, akkor a definíció szerint ez visszautasítást jelent, az automata a jelsorozatot nem fogadta el. Ezen a körülményen az automata állapotok jellegének megcserélése természetesen nem változtat, így mindkét esetben visszautasítást kapnánk.

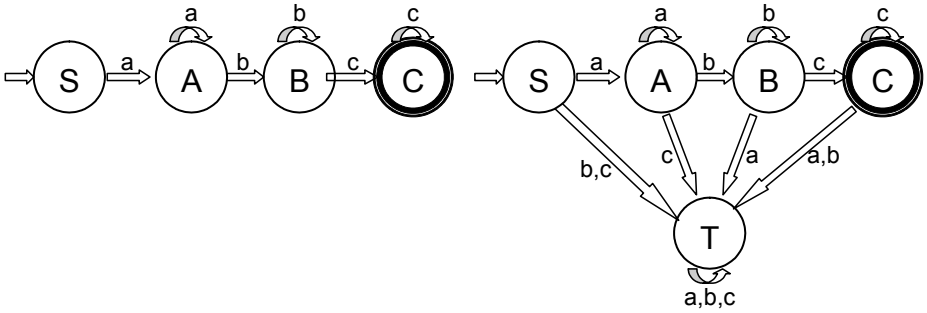
Ennek elkerülésére az automatát teljesen specifikálni kell, így az a szöveget mindig végigolvassa.

Problémát okozhat az automata esetleges nemdeterminisztikus volta. Ilyenkor adódhat egy-egy olyan mozgássorozat, amely közül az egyik elfogadó állapotba, a másik pedig visszautasító állapotba vezet. Definíció szerint ilyenkor az elfogadó állapotba vezető mozgássorozatot kell alapul venni, és ennek megfelelően a jelsorozatot el kell fogadni.

Egy ilyen nemdeterminisztikus automatánál pusztán az állapotok jellegének felcserélésével szintén nem érünk célt. A csere után ismét lesz egy elfogadó állapotba és egy visszautasító állapotba vezető mozgássorozat, azzal a megjegyzéssel, hogy a két mozgás szerepe most felcserélődött. Az automata a jelsorozatot mindkét esetben elfogadná. Ennek elkerülésére alakítsuk át az automatát determinisztikussá. Egy teljesen specifikált determinisztikus automata állapotainak jellegét felcserélve valóban a komplemens nyelvét elfogadó automatát nyerünk.

Mind a teljes specifikáció, mind a determinisztikus automatára való áttérés az előbbiek szerint megoldott, így tetszőleges véges automatához szerkeszthető olyan véges automata, amely az eredeti automata nyelvének komplemensét fogadja el. Így a komplemens is reguláris nyelv, a zártságot igazoltuk.

Végül mutassuk be ennek egy alkalmazását.



2.15. ábra

A 2.15. ábrán ugyanazt a nyelvet elfogadó két véges automatát tüntettünk fel. Közülük az első nem teljesen specifikált. Ha a teljesen specifikált automata

állapotainak jellegét cseréljük fel, valóban a komplemenst elfogadó automatát kapunk. Amennyiben a nem teljesen specifikált automatánál tesszük ezt, olyan automatához jutunk, amely a komplemens nyelvnél szűkebb halmazt fogad el, lesznek ugyanis olyan jelsorozatok, amelyek sem az eredeti, sem az átalakított automata nem fogad el.

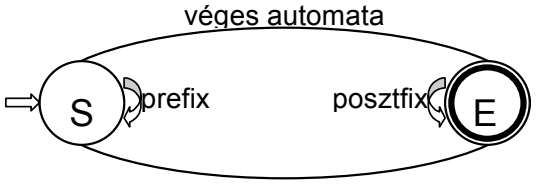
Az unió képzésénél elindulhatunk akár a nyelvtanból, akár az automatából. Itt is konstruktív bizonyítást adunk, vagyis megszerkesztjük az unióval előálló nyelv nyelvtanát, illetve az azt elfogadó automatát.

Lássuk először az automatákat érintő megoldást.

Az alapgondolat nyilván az lehet, hogy egyesítsük a két automata kezdőállapotait egyetlen közös kezdőállapotba, és az első mozgás döntse el, melyik automata fogadja el a jelsorozatot. Ha az első automatába lépünk be azzal az első nyelv mondatait fogjuk elfogadni, ha a másodikba, akkor a másodikét. Azonban itt is óvatossággal kell eljárnunk.

Vannak olyan véges automaták, ahol a kezdőállapotba nem vezet nyíl, így az első lépést követően sohasem leszünk újra a kezdőállapotban. Más automatáknál nem ez a helyzet.

A 2.16. ábra mutatja egy ilyen automata vázlatát.



2.16. ábra

Az ábrán sematikusan rajzoltunk egy olyan mozgási sorozatot, egy hurkot, amely a kezdőállapotból kiindulva visszatér a kezdőállapotba. Ugyanez az automata egy másik kellemetlenné válható tulajdonsággal is rendelkezik, nevezetesen van olyan elfogadó állapota, amelyből vissza lehet térni az automatába. Ezt is sematikusan egy az (egyik) elfogadó állapotból kiinduló és oda visszatérő kör jelzi. Az előbbi köröket prefix, illetve posztfix névvel jelöltük, hiszen ezek vagy a mondat elején, vagy a mondat végén fordulnak elő.

Amennyiben az egyesítendő két automatának soha vissza nem térő kezdőállapotai vannak, akkor a két kezdőállapot egyesítése valóan azt az automatát szolgáltatja, amelyik a két nyelv unióját fogadja el.

Ha azonban akár csak az egyik automatánál is vissza tudunk térni a kezdőállapotba, megtehetjük, hogy ebben az automatában futunk egy kört, és a kezdőállapotba visszatérve a másik automatában folytatjuk utunkat. Így olyan jelsorozatokat is elfogadhatunk, amelyek sem az egyik, sem a másik nyelvnek nem mondatai.

Ennek a hibának elkerülésére az automatákat, ha visszatérő kezdőállapotuk van, át kell alakítanunk olymódon, hogy a kezdőállapotba ne lehessen visszatérni.

Ezt könnyen megtehetjük. Fosszuk meg az eredeti kezdőállapotot ettől a címétől, és vezessünk be egy új kezdőállapotot. Az eredeti kezdőállapotból kiinduló nyilakat duplikálnunk kell, a másolatot az új kezdőállapotból indítsuk. Minthogy a bemenő nyilakat nem duplikáljuk, az új kezdőállapotnak csak kimenő nyilai lesznek, oda visszatérni nem lehet. A módosított automata természetesen ugyanazt a nyelvet fogadja el, hiszen az első mozgást követően a két automata ugyanabba az állapotba kerül, és a továbbiakban már csak az automatának az eredeti automatával megegyező részének van szerepe. Az ily módon átalakított, a kezdőállapotba vissza nem lépő automaták kezdőállapotait most már egyesíthetjük, hiszen az első lépés itt valóban eldönti, melyik automatában folytatjuk le az elemzést, és nincs lehetőség a két automata közötti kommunikációra.

Ezzel a problémával kissé módosított formában tulajdonképpen már találkoztunk. Amikor a balreguláris nyelvtanokhoz készítettünk automatát, akkor egynél több állapot tarthatott igényt a kezdőállapot titulusára. A problémát akkor lényegében ugyanezzel a módszerrel oldottuk meg.

A két nyelv uniójának elfogadására szolgáló automatát úgy is elképzelhetjük, hogy a két automatát egyetlen automatának tekintjük, amelynek azonban sajnálatos módon két kezdőállapota van. Ennek a „betegségnek” a kikúrálására pontosan a fent leírt műveleteket kell elvégeznünk.

Induljunk ki most a nyelvtanokból. Legyen tehát a két $\boldsymbol{L_1}$ és $\boldsymbol{L_2}$ nyelv nyelvtanával adott. Annak érdekében, hogy a két nyelvtan szabályait meg tudjuk különböztetni, gondoskodnunk kell arról, hogy a két nyelvtan nemterminális szimbólumainak halmaza diszjunkt legyen. Ha eredetileg nem ez volna a helyzet, akkor ezen átnevezéssel segíthetünk.

Amennyiben a mondatszimbólumok sehol sem szerepelnek levezetési szabály jobboldalán, ez annyit jelent, hogy az első levezetési szabály alkalmazása után a mondatszimbólum nem szerepel többet a mondatszerű formákban. Ilyenkor a két nyelv mondatszimbólumait közösíthetjük, vagyis egyetlen közös mondatszimbólumot alkalmazva máris előállt a két nyelv unióját generáló nyelvtan.

Persze előfordulhat, hogy a mondatszimbólum szerepel a levezetési szabályok jobboldalán is. Ez esetben a mondatszimbólum a mondatszerű formában újból felléphet. Ha ilyenkor térünk át a másik nyelvtanra, olyan jelsorozatot generálhatunk, amelyik egyik nyelvnek sem mondata.

Vegyük észre, hogy a mondatszimbólum feltűnése a levezetési szabály jobboldalán nem más, mint az automaták visszatérő kezdőállapotának nyelvtani interpretációja.

A baj kezelése is analóg az automatáknál alkalmazottal. Át kell térni olyan nyelvtanra, ahol a mondatszimbólum nem szerepel levezetési szabály jobboldalán. Új mondatszimbólumot kell bevezetni, a régi mondatszimbólumot meg kell fosztani ettől a rangjától, végül duplikálni kell azokat a levezetési

szabályokat, amelynek baloldalán az eredeti mondatszimbólum áll, olymódon, hogy annak helyébe az új mondatszimbólumot írjuk.

Annak magyarázatára, hogy az átalakítás a generált nyelvet nem változtatja meg, és ezután a két nyelvtan egyesítése valóban a két nyelv unióját generálja, nem térek ki, hiszen ezt már az automaták ismertetésekor megtettem.

Ezek szerint bármilyen legyen is a két automata, illetve a két nyelvtan, a két nyelv uniója véges automatával elfogadható, illetve reguláris nyelvtannal generálható. Ezzel igazoltuk, hogy a reguláris nyelvek zártak az unióképzés műveletére.

Lássunk egy példát az unióképzésre. Legyen a két nyelvtan:

$\boldsymbol{L}_1$ $S \to aA$ $S \to bS$ $A \to aS$ $A \to bA$ $S \to \varepsilon$

$\boldsymbol{L}_2$ $S \to aS$ $S \to bB$ $B \to aB$ $B \to bS$ $S \to \varepsilon$

Átnevezéssel érjük el, hogy ne legyen két azonos nevű nemterminális a két nyelvtanban.

$\boldsymbol{L}_1$ $P \to aA$ $P \to bP$ $A \to aP$ $A \to bA$ $P \to \varepsilon$

$\boldsymbol{L}_2$ $Q \to aQ$ $Q \to bB$ $B \to aB$ $B \to bQ$ $Q \to \varepsilon$

Sajnos a mondatszimbólumok szerepelnek a produkciós szabályok jobboldalán is. Vezessünk be tehát új mondatszimbólumokat, és két lépést egyesítve rögtön közösítsük is az új mondatszimbólumot.

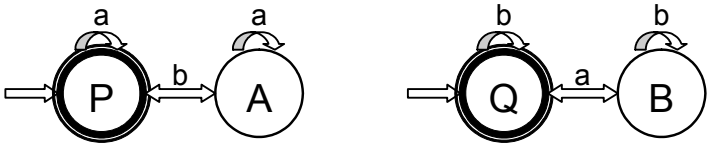
$S \to aA$ $S \to aQ$ $S \to bP$ $S \to bB$ $S \to \varepsilon$

$P \to aA$ $P \to bP$ $A \to aP$ $A \to bA$ $P \to \varepsilon$

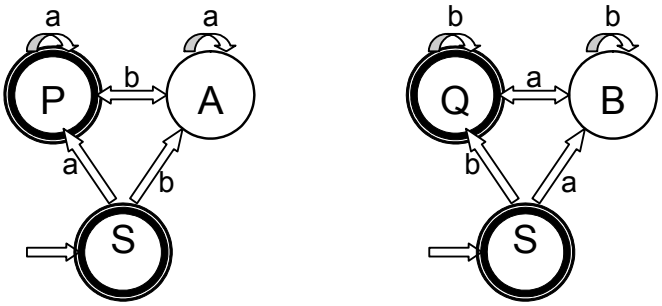
$Q \to aQ$ $Q \to bB$ $B \to aB$ $B \to bQ$ $Q \to \varepsilon$

Ezzel megkaptuk a két nyelvtan által generált nyelv unióját generáló nyelvtant.

A szerkesztés menetét az automatákon is nyomon követhetjük. A 2.17. ábrán először a két nyelv automatája van feltüntetve. Mind a nyelvtanból, mind az automatákból könnyen megállapítható, hogy a két nyelv *a* és *b* karakterekből álló jelsorozatokat tartalmaz, mégpedig az első nyelv esetében az *a*, a másodikban a *b* karakterek száma páros.

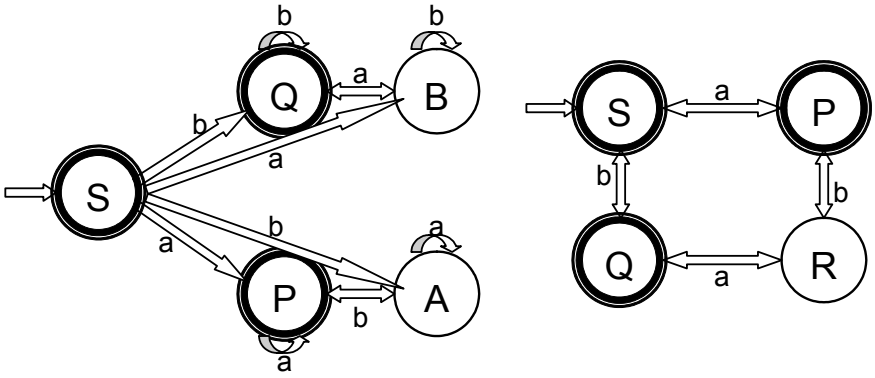


2.17.a. ábra



2.17.b. ábra

Az automatát olymódon kell átalakítani, hogy a kezdőállapotba ne vezessen nyíl, ne térhessünk vissza a kezdőállapotba. Ez a feltétel a megfelelője annak a követelménynek, hogy a mondatszimbólum ne szerepeljen produkciós szabály jobboldalán.



2.18. ábra

A 2.18. ábra baloldala az így módosított, majd a kezdőállapotok egyesítésével kiadódó automatát mutatja. Amennyiben előzetes átalakítás nélkül egyesítenénk a két automata kezdőállapotát, úgy megtörténhetne, hogy az egyik nyelv mondatát a másik nyelv prefixuma helytelenül megelőzné.

Az egyesített automata nemdeterminisztikus. Amennyiben ezt az automatát az ismert módon determinisztikussá alakítjuk, akkor a 2.18. ábra jobboldalán látható automatát kapjuk. Ez az automata csak azokat a jelsorozatokat utasítja vissza, amelyekben mind az *a*, mind a *b* karakterek száma páratlan.

A metszet műveletéről szólva tulajdonképpen ezek után annak zártságát nem kellene külön igazolnunk, hiszen a halmazelméletből jól ismert *de Morgan* szabályok szerint az unió- és metszetképzés művelete kifejezhető a másik művelettel, és a komplemens képzéssel. Minthogy a komplemens és unióképzésre már beláttuk a zártságot, a metszet műveletére vonatkozó zártság ebből már következik.

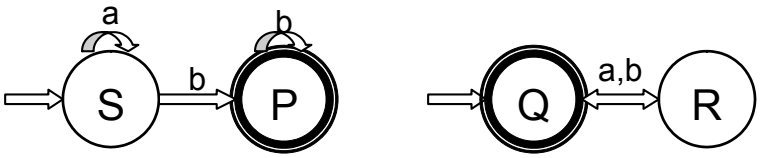
Mégis adunk egy konstruktív bizonyítást, mert a vázolt szerkesztés sokkal célratörőbb, és sok érdekes szempontra hívja fel a figyelmet.

Legyen adott két L_1 és L_2 nyelv véges automatájukkal. A metszet elfogadására olyan automatát kell szerkesztenünk, amely egyidejűen figyelemmel kíséri, mi történik a két eredeti automatában, és a jelsorozatot csak akkor fogadja el, ha mindkét eredeti automata elfogadja azt.

Készítsünk egy olyan automatát, amelynek állapothalmaza a két eredeti automata állapotterének direkt szorzata, pontosabban ennek alkalmas részhalmaza. Így az új automatán egyidejűen követni tudjuk a két eredeti automata működését. Az új automata mozgási szabályai ugyanis két eredeti automata mozgási szabályain alapulnak, amennyiben egy induló állapotpárból egy karakter beolvasásának hatására az eredményül kapott állapotpárt az eredeti automaták mozgási szabályai alapján határozzuk meg.

A metszet automatájának kezdőállapota a két kezdőállapot szorzata lesz, míg az elfogadó állapotok azok az állapotpárok lesznek, amelyek két elfogadó állapotból származnak.

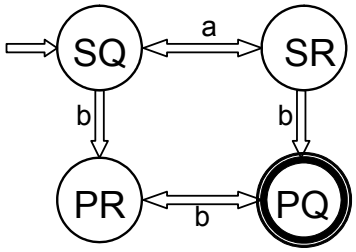
Példánkban a két nyelvet rögtön automatájukkal adtuk meg.



2.19. ábra

A 2.19. ábra első automatája azokat a jelsorozatokat fogadja el, ahol az *a* karakterek (esetleg üres) sorozatát a *b* karakterek nem üres sorozata követi. A második automata az olyan *a* és *b* karakterekből álló jelsorozatokat fogadja el, ahol a karakterek száma páros.

A 2.20. ábrán a két nyelv metszetét elfogadó automata van feltüntetve.



2.20. ábra

Amennyiben a két eredeti automata determinisztikus volt a szerkesztés menetéből következően az eredményül kapott automata is determinisztikus lesz. Megjegyzem, hogy a metszetképzésnél sem az automaták nemdeterminisztikus volta, sem a máskor „illegálisnak” minősített hurkok jelenléte nem zavaró.

A konkatenáció esetében az igazolás egyszerű, akár az automatákból, akár a nyelvtanokból indulunk is ki. Az $\boldsymbol{L_1}$ és $\boldsymbol{L_2}$ nyelv ebben a sorrendben vett $\boldsymbol{L_1L_2}$ konkatenáltját, pontosabban annak nyelvtanát az eredeti nyelvtanokból a következőképpen kapjuk.

Szokás szerint itt is el kell érnünk, hogy a két nyelvtan nemterminális szimbólumaiból alkotott két halmaz diszjunkt legyen.

Ezután vesszük az első nyelvtan második típusú szabályait, vagyis azokat, ahol a jobboldalon nincsen nemterminális szimbólum. Ezeket átírjuk olymódon, hogy a terminális szimbólum után odaírjuk a második nyelvtan mondatszimbólumát. Példaképpen egy ilyen lehetséges átalakítás formája:

$$A \to a \qquad \Rightarrow \qquad A \to aS_2$$

ahol az első szabály az első nyelvtan levezetési szabálya, az S_2 pedig a második nyelvtan mondatszimbóluma. Az ily módon megszerkesztett nyelvtan mondatszimbóluma nyilván az első nyelvtan mondatszimbóluma lesz.

Magától értetődő, hogy az előbbiek szerint megszerkesztett nyelvtan a konkatenált nyelvet generálja. Az első nyelv mondatainak generálásakor az utolsónak alkalmazott szabály szükségszerűen egy második típusú szabály volt. A mondatszerű formában ugyanis ilyen szabály „távolítja” el a nemterminális szimbólumot, és teszi ezzel a mondatszerű formát mondattá. A kapott nyelvtan azonban az első nyelv mondatai után odabiggyeszti a második nyelv mondatszimbólumát, amelyből aztán egy a második nyelvhez tartozó mondat keletkezik.

Amennyiben az első nyelvtan tartalmaz $A \to \varepsilon$ alakú ε-szabályokat, akkor a fent vázolt átalakítással ezekből $A \to S_2$ formájú úgynevezett egyszeres szabályok keletkeznének. A feltételes mód azért indokolt, mert a *Chomsky*-féle előírás ilyen szabályokat nem ismer. Ezen úgy segíthetünk, hogy az ilyen szabályok helyett a magányos jobboldal, adott esetben a második nyelv mondatszimbólumának helyébe beírjuk azon szabályok jobboldalát, ahol a baloldalon ez a magányos szimbólum szerepel. Ez a művelet annyira egyszerű, hogy végrehajtását az olvasóra bízhatom annál is inkább, mert később még lesz szó róla.

Lássunk ismét egy példát:

$\boldsymbol{L_1}$	$S \to aA$	$S \to bS$	$A \to aS$	$A \to bA$	$S \to \varepsilon$
$\boldsymbol{L_2}$	$S \to aA$	$S \to cS$	$A \to aS$	$A \to cA$	$A \to \varepsilon$

Az első nyelv az *a* és *b*, a második az *a* és *c* karakterekből áll. Ezen kívül az első nyelvben az *a* karakterek száma páros, a másodikban viszont páratlan.

Tegyük a nemterminális szimbólumok halmazait diszjunkttá:

$\boldsymbol{L_1}$	$S \to aA$	$S \to bS$	$A \to aS$	$A \to bA$	$S \to \varepsilon$
$\boldsymbol{L_2}$	$P \to aQ$	$P \to cP$	$Q \to aP$	$Q \to cQ$	$Q \to \varepsilon$

Ezek után el kell érnünk, hogy vagy az első nyelv elenyésző *S* szimbóluma ne szerepeljen a baloldalon, vagy a második nyelv P mondatszimbóluma ne legyen a szabályok jobboldalán. A *vagy* itt nem kizáró vagy.

Itt az utóbbi megoldást választottuk.

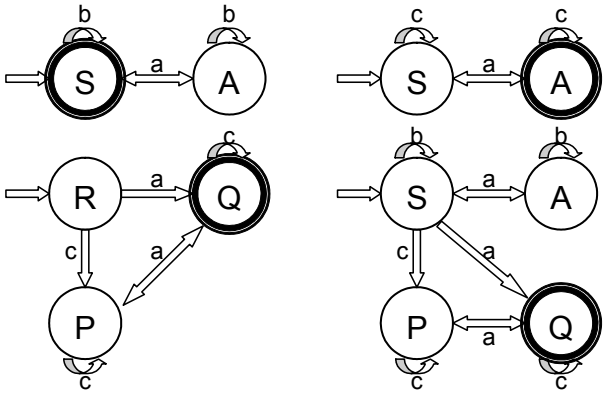
$\boldsymbol{L}_1$	$S \to aA$	$S \to bS$	$A \to aS$	$A \to bA$	$S \to \varepsilon$
$\boldsymbol{L}_2$	$R \to aQ$	$R \to cP$	$P \to aQ$	$P \to cP$	
	$Q \to aP$	$Q \to cQ$	$Q \to \varepsilon$		

Ennek alapján a konkatenált nyelv nyelvtana a következő:

$\boldsymbol{L_1L_2}$	$S \to aA$	$S \to bS$	$A \to aS$	$A \to bA$
	$S \to aQ$	$S \to cP$	$P \to aQ$	$P \to cP$
	$Q \to aP$	$Q \to cQ$	$Q \to \varepsilon$	

Az eredményt itt már végleges formában adtuk meg, vagyis a szalonképtelen egyszeres szabályokat már kiküszöböltük.

A teljesség kedvéért a fenti példát automatákkal is megoldjuk. Itt a nyelvtanok egymásba fűzésének az a művelet felel meg, hogy az első automata elfogadó állapotát „összeszegecseljük” a második automata kezdőállapotával. Amennyiben az első automatának egynél több elfogadó állapota lenne, akkor vagy minden elfogadó állapothoz hozzászegecselünk egy második automatát, vagy egyetlen elfogadó állapotot hozunk létre. Itt arra kell ügyelnünk, hogy az első automata elfogadó állapotából ne lehessen visszatérni az első automata állapotaiba, és ezzel együtt a második automata kezdőállapotába se lehessen lépni a második automata állapotaiból.



2.21. ábra

Amennyiben mindkét körülmény egyszerre fennáll, akkor a két nyelv fragmensei – helytelenül – összekeveredhetnek. Előfordulhat ugyanis, hogy elérve az első automata elfogadó állapotát futunk egy kört a második automatában, majd visszatérve a közösített állapothoz visszamegyünk az első automatába, és ott is végzünk egy körüljárást.

Ennek elkerülésére vagy az első automatánál kell megakadályoznunk, hogy az elfogadó állapotból visszatérjünk, vagy a második automatánál kell megtiltanunk a kezdőállapot újbóli meglátogatását. A *vagy* itt természetesen nem

kizáró vagy, és pusztán azt érzékelteti, hogy a két intézkedés közül egyik is elegendő a helytelen keveredés kiküszöbölésére.

Visszatérve példánkra a megoldás menete a 2.21. ábrán látható. Az első két automata az $\boldsymbol{L_1}$ és $\boldsymbol{L_2}$ nyelveket fogadja el.

Annak érdekében, hogy a két nyelv fragmensei ne keveredjenek, el kell érnünk, hogy a második automata kezdőállapotába ne vezessen nyíl. A második automata módosított változatát mutatja az ábra harmadik része, végül a konkatenált nyelvet elfogadó automata látható, amelyet úgy kaptunk, hogy az első automata elfogadó állapotát egyesítettük a második kezdőállapotával.

Esetünkben a konkatenált nyelv automatája nemdeterminisztikus.

Minthogy ez az algoritmus bármilyen nyelvtanok illetve bármilyen automaták esetében alkalmazható, ezzel igazoltuk a reguláris nyelvek zártságát a konkatenáció műveletére nézve.

A tranzitív lezárt nyelvtanának és automatájának megszerkesztésekor eddigi eredményeinket – bizonyos óvatossággal – alkalmazhatjuk.

Nyelvtanok esetében az alapgondolat az lesz, hogy a második típusú, tehát nemterminálist nem tartalmazó szabályok esetében hozzáfűzzük a mondatszimbólumot. Ezzel elérhetjük, hogy az eredeti nyelv egy mondatának generálása után hozzákezdhetünk a következő mondat generálásához.

Ezt a folyamatot azonban le kell zárnunk, ezért a nyelvtanhoz hozzávesszük az $S \rightarrow \varepsilon$ helyettesítési szabályt is, annál is inkább, mert az üres jelsorozat, az ε, megbeszélésünk szerint mindig eleme a tranzitív lezártnak.

Ez a szerkesztés általában azt a nyelvtant szolgáltatja, amelyik a tranzitív lezárt nyelvet generálja. Óvatosságra azonban itt is szükség van. Ha ugyanis az üres jelsorozat a nyelvnek nem eleme, és ugyanakkor a mondatszimbólum szerepel helyettesítési szabály jobboldalán, akkor a fenti átalakítást kritika nélkül alkalmazva az új nyelvtan – helytelenül – képes olyan jelsorozatok generálására, amelyek az eredeti nyelvnek prefixumai. Az $S \rightarrow \varepsilon$ helyettesítési szabály bevezetésével a mondatszimbólum megjelenése lezárhatja a generálást, ami korábban lehetetlen volt.

Ezen veszély elkerülésére a szerkesztést megelőzően a nyelvtant az ismert módon át kell alakítani, hogy a mondatszimbólum ne szerepeljen helyettesítési szabály jobboldalán. Amennyiben az eredeti nyelvtanban lettek volna ε-szabályok, akkor minden olyan szabályban, ahol a jobb oldalon az elenyésző nemterminálisok szerepelnek, ezeket a szabályokat duplikálni kell olymódon, hogy az elenyésző nemterminálisok helyett a mondatszimbólumot írjuk. Ezután az eredeti ε-szabályokat akár el is hagyhatjuk, hiszen a mondatszimbólumnak hivatalból van ε helyettesítése.

Példának válasszuk azt a nyelvet, amely olyan *a* és *b* szimbólumokból álló jelsorozatokat tartalmaz, ahol vagy az *a*, vagy a *b* szimbólumok száma páratlan. A *vagy* itt kizáró értelemben szerepel. Ez persze azt jelenti, hogy

mondhattuk volna egyszerűbben is, nevezetesen, hogy a nyelv a páratlan hosszúságú, az *a* és *b* szimbólumokból álló jelsorozatokat tartalmazza. A választott fogalmazásnak és tárgyalásmódnak didaktikai okai vannak.

Ezzel a nyelvtan:

$\boldsymbol{L}$ $D \to aA \mid bB$ $A \to aD \mid bC \mid \varepsilon$

$B \to aC \mid bD \mid \varepsilon$ $C \to aB \mid bA$

Itt sajnos a mondatszimbólum előfordul a helyettesítési szabályok jobboldalán is, vezessünk be hát egy új mondatszimbólumot. Ennek tartottuk fenn az *S* jelölést. Ezzel a nyelvtan:

$\boldsymbol{L}$ $S \to aA \mid bB$ $D \to aA \mid bB$

$A \to aD \mid bC \mid \varepsilon$

$B \to aC \mid bD \mid \varepsilon$ $C \to aB \mid bA$

Végül a tranzitív lezárt nyelvtana:

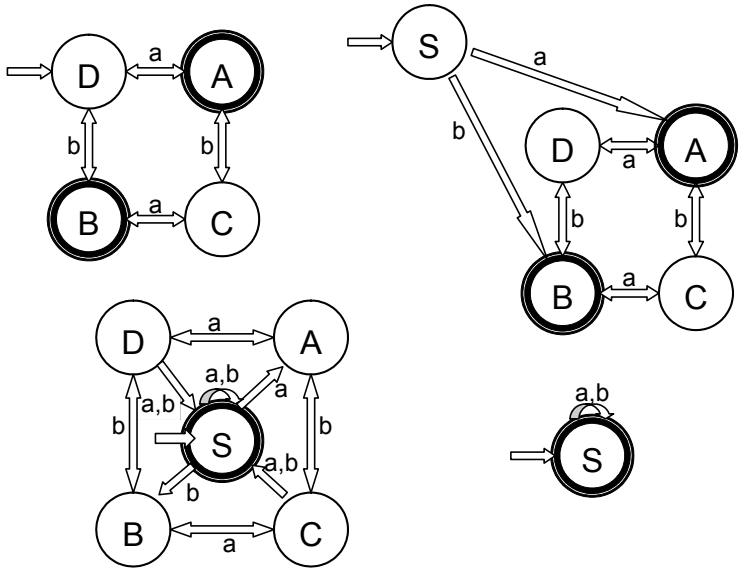
$\boldsymbol{L}^*$ $S \to aA \mid bB \mid \varepsilon$

$D \to aA \mid bB \mid aS \mid bS$ $C \to aB \mid bA \mid aS \mid bS$

$A \to aD \mid bC$ $B \to aC \mid bD$

A szalonképtelen ε-szabályokat itt is kiküszöböltük.

A 2.22. ábrán megadtuk a szerkesztés lépéseit arra az esetre, amikor az automatából kiindulva határozzuk meg a tranzitív lezárt automatáját.



2.22. ábra

A négy részábra közül az első az eredeti nyelv automatáját mutatja.

Első lépésként az automatát át kell alakítani, hogy a kezdőállapotba ne mutasson nyíl, ugyanakkor a kezdőállapot elfogadó állapot legyen. Ennek eredménye a következő ábra.

A következő művelettel érjük el, hogy csupán egyetlen elfogadó állapotok legyen. Ez egyébként azért is üdvös, mert az sem engedhető meg, hogy az elfogadó állapotból nyíl induljon ki. Ilyenkor ugyanis az eredeti nyelv bizonyos posztfixumait is elfogadná az automata, így tehát mind a kezdőállapotot, mind az elfogadó állapotot meg kell „tisztítanunk". Ezt mutatja a harmadik ábra.

A következő részábra a szerkesztésnek megfelelő, a tranzitív lezártat elfogadó automatát tünteti fel. Jóllehet az eredeti automata determinisztikus volt, a „tisztogatási" művelet során az automata elvesztette determinisztikus jellegét. Így az eredményül kapott automata sem determinisztikus.

Vegyük észre, az elfogadó *S* állapotból akár *a* akár *b* karaktert olvasunk, visszajutunk az *S* állapotba. Ezek szerint ez az automata minden jelsorozatot elfogad. Ennek automatáját adja az utolsó részábra.

Első pillanatra talán meglepő, de ez az automata csupán egyetlen állapottal bír, és minden jelsorozatot elfogad. A tranzitív lezárt nyelv ugyanis a Σ^*.

Ha egy kicsit utánagondolunk ez teljesen természetes. Az üres jelsorozat, mint minden nyelv nulladik hatványa ugyanis „hivatalból" eleme minden tranzitív lezártnak. Ami a nem üres jelsorozatokat illeti, vegyük észre, hogy a magányos *a* illetve *b* szimbólum is eleme az eredeti nyelvnek. Így legrosszabb esetben az a partíció, amikor minden egyes szimbólum külön jelsorozatot alkot, mindig beválik.

Ezt tulajdonképpen kis gondolkodással rögtön az elején is észrevehettük volna, most azonban kivételesen nem a frappáns megoldás, hanem az algoritmus bemutatása volt a cél.

Ezzel igazoltuk, hogy a reguláris nyelvek halmaza valamennyi általunk említett műveletre nézve zárt.

A reguláris nyelveken értelmezett műveletekhez kapcsolódva szeretném bemutatni a reguláris nyelvek egyik fontos tulajdonságát, a pumpálást.

Egy reguláris nyelv minden kellő hosszúságú *w* mondata felbontható három jelsorozat *xyz* egymásutánjára olymódon, hogy a középső részsorozatot tetszőlegesen sokszor – esetleg egyszer sem – megismételve megint csak a nyelv mondatát kapjuk.

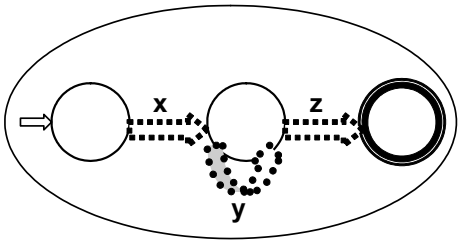
Így ha

$$xyz \in \boldsymbol{L} \qquad \text{akkor} \quad xy^{\mathbf{i}}z \in \boldsymbol{L} \qquad \mathbf{i} \geq 0 \qquad (2.26.)$$

Egy mondat elfogadása során az automata pontosan annyiszor mozog állapotról állapotra, ahány karakterből áll a mondat. Tételezzük fel, hogy a szóban forgó nyelvet egy **n** állapotú véges automata fogadja el. Az előbbiek szerit egy **n** karakterből álló jelsorozat feldolgozásakor tehát **n** lépést tesz, és közben **n+1** állapotot érint. Az automata ugyanis már az első karakter beolvasása előtt is valamilyen állapotban van, tehát ezt az állapotot is „érinti".

Ez annyit jelent, hogy kell lennie legalább egy olyan állapotnak, amelyet az automata legalább kétszer érint. Az automatán belül megtett úton tehát egy hurok van.

Bontsuk fel tehát a mondatot olymódon, hogy a hurok bejárása közben olvasott karakterek alkossanak egy külön részsorozatot. Ez lesz nyilván a középső *y* részsorozat. Ebből viszont már következik a (2.26.) állítás igazsága, hiszen ha ezt a hurkot újra és újra bejárjuk, akkor az *y* részsorozatot tetszőlegesen sokszor megismételjük.



2.23. ábra

Megjegyzem, hogy akár a hurok előtti *x,* vagy akár az utáni *z*, esetleg mindkettő üres is lehet. A fentiekből már nyilvánvaló mit jelent az adott esetben a kellő hosszúság. Sőt a (2.26.) állítás még tovább élesíthető. Azt is mondhatjuk, hogy a nyelv egy mondatában bárhol kijelölhetünk egy olyan kellő hosszúságú részsorozatot, amelyre a (2.26.) állítás igaz. Így az **n** hosszúságú részsorozat kijelölése a mi dolgunk, ennek particionálása azonban hurok előtti, a hurkot alkotó, és a hurok utáni darabokra már a nyelvtan illetve az automata dolga.

A pumpálás tulajdonsága alapján lehet belátni, hogy az

$$a^{\mathbf{i}}b^{\mathbf{i}} \qquad \mathbf{i} > 0$$

alakú nyelv nem lehet reguláris nyelv. Tételezzük fel ugyanis, hogy ezt a nyelvet egy **n** állapotú véges automata fogadja el. Válasszuk most az **i** kitevőt ennél nagyobbra, ekkor az *a* karakterek sorozatán belül jelölhetjük ki a pumpáló részsorozatot, amivel az *a* karakterek számát anélkül növelhetjük meg, hogy a *b* karakterek száma változna. Így a nyelvnek olyan mondata is lenne, amely nem egyforma számú *a* és *b* karaktert tartalmaz.

Ugyancsak a pumpálás alapján igazolhatjuk, hogy egy nem üres nyelvnek van az automata állapotok számánál rövidebb mondata.

Tételezzük ugyanis fel ennek ellenkezőjét. Ezek szerint a nyelv minden mondata legalább **n** hosszúságú. Az ilyen mondatok elemzésekor azonban szükségszerűen van hurok a mozgásban. A hurokhoz tartozó részsorozatot elhagyva ismét a nyelv egy mondatát kapjuk. Amennyiben az így lerövidített mondat még mindig túl hosszú, akkor ezt a processzust megismételjük. Nyilvánvaló, hogy előbb-utóbb az állapotok számánál rövidebb mondatot kell kapnunk.

Azt is könnyű belátni, hogy amennyiben a nyelvnek van az automata állapotainak számánál legalább egy hosszabb mondata, akkor végtelen sok mondata van. Abban a bizonyos kellő hosszúságú mondatban kell lennie huroknak. A mondatot a (2.26.) szerint particionálva, a hurkot alkotó részt ismételgetve tetszőleges számú mondatot generálhatunk.

2.6. Reguláris halmazok

A reguláris nyelvek leírására, definiálására eddig két módszert ismertünk meg. Egy reguláris nyelvet megadhatunk az azt generáló reguláris nyelvtannal, vagy az azt elfogadó véges automatával. Alábbiakban a reguláris nyelveknek, a jelsorozatokat tartalmazó reguláris halmazok útján egy harmadik meghatározási lehetőségével ismerkedünk meg.

Legyen tehát Σ egy alfabeta, amely a jelsorozatok karakterkészletét tartalmazza, és legyen a ennek egy eleme.

Lássuk ezután mi tekinthető reguláris halmaznak, pontosabban a Σ felett értelmezett reguláris halmaznak.

Reguláris az üres halmaz $\varnothing$, egyedül az üres jelsorozatot tartalmazó halmaz $\{\varepsilon\}$, és egy Σ halmazbeli karaktert, mint egyetlen jelsorozatot tartalmazó halmaz $\{a\}$.

Legyen most $\boldsymbol{P}$ és $\boldsymbol{Q}$ két reguláris halmaz. Néhány halmazművelet definíciószerűen nem vezet ki a reguláris halmazok világából. Ilyen az unióképzés, amelyet itt a + operátorral jelölünk, a konkatenáció, amelyet egyszerűen egymásután írással adunk meg, végül a tranzitív lezárás, amit szokás szerint a * jelöl.

Összefoglalva, adott Σ feletti reguláris halmaznak nevezzük azokat és csak azokat a nyelveket (sztringhalmazokat) melyek a fenti módszerrel előállíthatóak. Foglaljuk össze, milyen egyszerű halmazokat tekintünk reguláris halmazoknak, és milyen műveletek eredményeznek ismét reguláris halmazt:

$$\begin{array}{ll} \varnothing & \\ \{\varepsilon\} & \\ \{a\} & a \in \Sigma \\ \boldsymbol{P+Q} & \boldsymbol{P,Q} \text{ reguláris} \\ \boldsymbol{PQ} & \boldsymbol{P,Q} \text{ reguláris} \\ \boldsymbol{P}^* & \boldsymbol{P} \text{ reguláris} \end{array} \qquad (2.27.)$$

Az összeadás és a szorzás jelének az unióképzés illetve a konkatenáció műveletére való alkalmazását az indokolja, hogy az aritmetikában érvényes néhány szabály ebben a kontextusban is igaz marad. Ilyen például az operátorok precedencia szabálya. Előbb kell elvégezni a „hatványozás” értsd tranzitív lezárás műveletét mint a szorzással jelölt konkatenációt, végül a + operátorral jelölt unióképzést.

Persze itt is bevezethetjük a zárójelezést, mint a precedencia megtörésének eszközét. A zárójelek felbontásának szabályai is emlékeztetnek az aritmetikából ismertekre. Így például

$$(\boldsymbol{A}+\boldsymbol{B})(\boldsymbol{C}+\boldsymbol{D}) = \boldsymbol{AB}+\boldsymbol{AD}+\boldsymbol{BC}+\boldsymbol{BD}$$

Ezzel az analógiával azonban óvatosan kell bánnunk. Ha két olyan halmaznak képezzük az unióját, ahol az egyik a másik részhalmaza, akkor az „összeadás” már eltér az aritmetikai művelettől:

$$\text{ha} \quad \boldsymbol{A} \subset \boldsymbol{B} \quad \text{akkor} \quad \boldsymbol{A}+\boldsymbol{B} = \boldsymbol{B}$$

Vegyük észre, hogy az üres halmaz a zérus, az üres jelsorozatot tartalmazó halmaz pedig az egységelem szerepét játssza. Valóban

$$\boldsymbol{A}+\varnothing = \varnothing+\boldsymbol{A} = \boldsymbol{A} \quad \text{és} \quad \boldsymbol{A}\varepsilon = \varepsilon\boldsymbol{A} = \boldsymbol{A} \tag{2.28.}$$

hiszen az üres halmazzal vett unió nem bővíti az eredeti halmazt, ugyanakkor az $\boldsymbol{A}$ halmaz bármely eleme elé vagy mögé az üres jelsorozatot írva nem gyarapítottuk a halmaz elemeinek számát.

A tranzitív lezárt, mint ismeretes a következőképpen fejezhető ki:

$$\boldsymbol{A}^* = \bigcup_{i=0}^{\infty} \boldsymbol{A}^i \tag{2.29.}$$

Vezessük be ennek mintájára a kitevőként alkalmazott $^+$ jelölést, az alábbi értelmezéssel:

$$\boldsymbol{A}^+ = \bigcup_{i=1}^{\infty} \boldsymbol{A}^i \tag{2.30.}$$

Ebből következik, hogy

$$\boldsymbol{A}^* = \boldsymbol{A}^+ + \varepsilon \quad \text{és} \quad \boldsymbol{A}^*\boldsymbol{A} = \boldsymbol{A}^+ \tag{2.31.}$$

illetve ha az üres jelsorozat eleme a reguláris halmaznak

$$\varepsilon \in \boldsymbol{A} \quad \Rightarrow \quad \boldsymbol{A}^+ = \boldsymbol{A}^* \tag{2.32.}$$

hiszen az üres jelsorozat ε mindkét halmaznak eleme. A jobboldalon egy tranzitív lezárt van, amely definíció szerint tartalmazza az üres jelsorozatot, míg a baloldalon egy olyan $\boldsymbol{A}$ halmaz áll, amelynek feltételünk szerint eleme az üres jelsorozat.

Reguláris halmazokra lényegében ugyanúgy lehet a bevezetett jelölések alkalmazásával összefüggéseket, halmazok közötti kapcsolatokat definiálni, mint például az aritmetikai kifejezések segítségével a számok világában.

Felírhatunk halmazegyenleteket, halmazok közötti összefüggéseket, és megkereshetjük azokat a halmazokat, amelyek mellett ezek az összefüggések igazak lesznek, vagyis amely halmazok a halmazegyenlet megoldásai.

Vegyük például a következő egyszerű összefüggést:

$$\boldsymbol{X} = \alpha\boldsymbol{X}+\beta \tag{2.33.}$$

– ahol α és β adott reguláris halmazok,
– $\boldsymbol{X}$ pedig ismeretlen halmaz.

Határozzuk meg az $\boldsymbol{X}$ halmazt olymódon, hogy a (2.33.) összefüggés igaz legyen. Az egyenlet megoldása:

$$\boldsymbol{X} = \alpha^*\beta \qquad (2.34.)$$

Állításunk helyességéről könnyen meggyőződhetünk, ha a megoldást behelyettesítjük a (2.33.) egyenletbe. Valóban:

$$\alpha^*\beta = \alpha(\alpha^*\beta)+\beta = \alpha^+\beta+\beta = (\alpha^+ +\varepsilon)\beta = \alpha^*\beta$$

Megjegyezzük, hogy a (2.34.) megoldás csak abban az esetben unikális, ha az α halmaznak ε nem eleme. Ellenkező esetben ugyanis minden

$$\boldsymbol{X} = \alpha^*(\beta+\gamma)$$

alakú kifejezés megoldás, ahol γ tetszőleges reguláris halmaz.

Erről a szorgalmas olvasó egyszerű behelyettesítéssel meggyőződhet.

A fentiek alapján egyetlen „lineáris” összefüggéssel megadott reguláris halmazt könnyen meghatározhatunk. De mi történik azonban, ha több reguláris halmazt egy „lineáris egyenletrendszer” útján adunk meg.

Ilyenkor a (2.34.) megoldási képletet, és a behelyettesítést felváltva alkalmazva egy a *Gauss*-eliminációhoz hasonló eljárással kaphatjuk meg az ismeretlen halmazok megoldását. Hogy a megoldás során a részegyenleteknek valóban egyetlen megoldása van, az annak a következménye, hogy a véges automata nem tartalmaz ε-szabályokat.

Lássunk erre egy példát:

$$\boldsymbol{X}_1 = \boldsymbol{b}\boldsymbol{X}_1+\boldsymbol{a}\boldsymbol{X}_2+\varepsilon$$
$$\boldsymbol{X}_2 = \boldsymbol{a}\boldsymbol{X}_1+\boldsymbol{b}\boldsymbol{X}_2$$

Amennyiben a második kifejezést a (2.34.) mintájára megoldjuk, és ezt az első egyenletbe helyettesítjük, akkor a következőket kapjuk:

$$\boldsymbol{X}_2 = \boldsymbol{b}^*\boldsymbol{a}\boldsymbol{X}_1$$
$$\boldsymbol{X}_1 = \boldsymbol{b}\boldsymbol{X}_1+\boldsymbol{a}(\boldsymbol{b}^*\boldsymbol{a}\boldsymbol{X}_1)+\varepsilon = (\boldsymbol{b}+\boldsymbol{a}\boldsymbol{b}^*\boldsymbol{a})\boldsymbol{X}_1+\varepsilon$$

Ismét a (2.34.) megoldó képletet alkalmazva ezúttal az $\boldsymbol{X}_1$ halmazra, majd a kapott eredményt visszahelyettesítve az egyenletrendszert megoldhatjuk.

$$\boldsymbol{X}_1 = (\boldsymbol{b}+\boldsymbol{a}\boldsymbol{b}^*\boldsymbol{a})^*$$
$$\boldsymbol{X}_2 = \boldsymbol{b}^*\boldsymbol{a}(\boldsymbol{b}+\boldsymbol{a}\boldsymbol{b}^*\boldsymbol{a})^*$$

Mint látjuk, mindkét halmazt az $\boldsymbol{a}$ és $\boldsymbol{b}$ karakterekből álló jelsorozatok alkotják, ahol a $\boldsymbol{b}$ karakterek száma és elhelyezkedése tetszőleges, az $\boldsymbol{a}$ karakterek száma az első halmaz esetében páros a másodikében páratlan.

Könnyű belátni, hogy ez a módszer bár kissé hosszadalmas, az egyenletek illetve ismeretlenek számától függetlenül alkalmazható.

Szeretnénk érzékeltetni, hogy a reguláris nyelvtanok éppen a reguláris halmazokat generálják.

Ehhez először azt lássuk be, hogy a halmazokra felírt olyan „lineáris” egyenletrendszer megoldásai, amelynek „együtthatói” reguláris halmazok, szintén reguláris halmazok lesznek.

Ez valóban így van, hiszen a megoldás során a (2.34.) összefüggés felhasználásakor és a behelyettesítéskor csupa olyan műveletet – unióképzés,

konkatenáció, tranzitív lezárás – végzünk a reguláris halmazokon, amely nem vezet ki a reguláris halmazok világából.

Ugyanakkor minden reguláris halmaz felírható mint egy halmazegyenlet megoldása, ha másképpen nem megy egyenlőség formájában. Ezek szerint az említett egyenletrendszerek megoldásaiból alkotott halmaz, és a reguláris halmazok alkotta halmaz azonos.

Egy reguláris nyelvtan által generált nyelv megadható, mint egy egyenletrendszer megoldása. Ez azonnal kiadódik, ha a nyelvtan nemterminális szimbólumainak nyelvi értelmezést adunk.

Minden nemterminális szimbólum jelentse azon jelsorozatok halmazát, amelyeket a szóban forgó nemterminális szimbólumból a nyelvtani szabályok segítségével levezethetünk. Itt most visszaköszön az a felfogás, amikor a véges automaták állapotaihoz egy nyelvet rendeltünk, nevezetesen azt a nyelvet, amelyet az automata az adott állapot mint kezdőállapot mellett elfogad.

Ebben a szellemben a helyettesítési szabályoknak új értelmezést adhatunk. Az $S \to aA$ átírási szabály azt mondja, hogy minden olyan jelsorozat beletartozik az S nemterminális szimbólumból levezethető jelsorozatok halmazába, amely az a terminális szimbólummal kezdődik, és azt egy az A nemterminális szimbólumból levezethető jelsorozat követ.

Ez az interpretáció teljesen korrekt, és megegyezésben van eddigi értelmezésünkkel. Világos, hogy ebben a felfogásban a nyelvet a mondat-szimbólum reprezentálja.

Amennyiben egy nemterminális szimbólum több szabály baloldalán szerepel, akkor a szóban forgó nemterminálisból származtatott jelsorozatok bármely szabály alapján generálhatóak. Nyilvánvaló, hogy a teljes nyelvet az egyes szabályok alapján generált jelsorozatok uniója szolgáltatja.

Példaként vegyük a következő nyelvtant:

$$S \to bS \quad S \to aA \quad S \to \varepsilon \quad A \to aS \quad A \to bA$$

Jelöljük az unió műveletét itt is a + operátorral, akkor az S és A nemterminálisok által reprezentált nyelvek az alábbiak szerint írhatók fel:

$$S = bS+aA+\varepsilon$$
$$A = aS+bA$$

Nem kell különösebb éleslátás ahhoz, hogy felismerjük ezekben az összefüggésekben a korábban már tárgyalt egyenleteket. Az eltérés csupán annyi, hogy ott az ismeretlenek jelölése $\boldsymbol{X_1}$ és $\boldsymbol{X_2}$ volt.

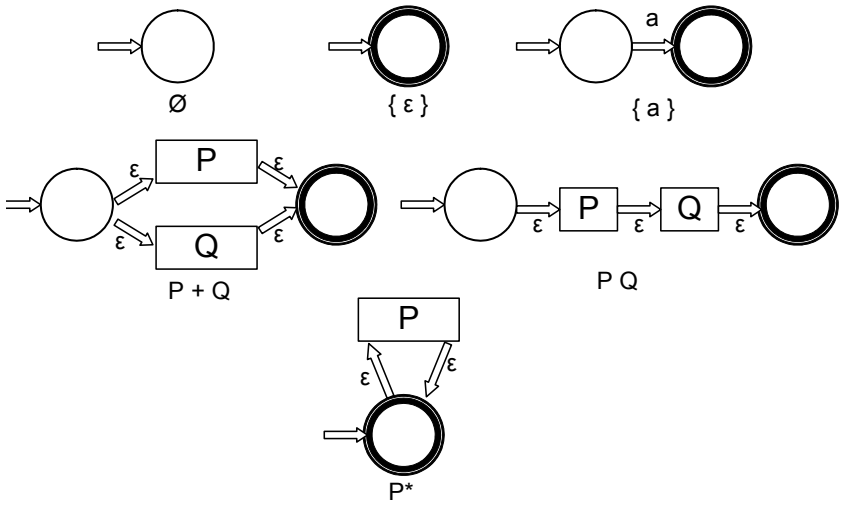
Azt sem nehéz felismerni, hogy ez a nyelvtan a páros számú a karaktert tartalmazó jelsorozatokat generálja.

A módszer, amely a nemterminális szimbólumok új értelmezéséből következik, közvetlenül szolgáltatja azt az algoritmust, amellyel bármely reguláris nyelvtanból reguláris halmazokra felírt egyenletrendszer származtatható. Ebből következik, hogy minden reguláris nyelvtan által generált nyelv reguláris halmaz.

Amennyiben a reguláris halmazt meghatározó egyenletrendszer együtthatói puszta karakterek, akkor triviális, hogy ezek a reguláris halmazok egyúttal reguláris nyelvek.

Amennyiben ezek az együtthatók bonyolultabb kifejezések más utat követhetünk.

A (2.27.) meghatározta, milyen halmazokat tekintsünk reguláris halmazoknak. Tételezzük fel, hogy ***P*** és ***Q*** reguláris nyelvek, ekkor ***P***-hez is, ***Q***-hoz is tartozik olyan véges automata, amely ezeket a halmazokat elfogadja. A 2.24. ábra tünteti fel a (2.27.) összefüggésekben definiált reguláris halmazokhoz tartozó automatákat.



2.24. ábra

Az első három automata értelmezése triviális. A továbbiakban a ***P*** és ***Q*** betűkkel jelöltük szimbolikusan azokat az automatákat, amelyek a ***P*** illetve ***Q*** reguláris halmazokat elfogadják.

Minthogy a ***P*** és ***Q*** automaták szintén reguláris halmazokat elfogadó automaták, ezek tovább felbonthatóak, és ezt az eljárást rekurzíve folytatva végül a három első automatához kell jutnunk. Ha tehát sikerül megmutatnunk, hogy ezek az automaták megvalósíthatóak, akkor igazoltuk, hogy minden reguláris halmaz véges automatával elfogadható, és a reguláris nyelvek halmaza azonos a reguláris halmazok halmazával.

Az ábrákban az ε feliratú nyilak ezekben az automatákban úgynevezett ε-mozgásokat reprezentálnak.

Az unióképzésnél a kezdő- és elfogadó állapotokat csak akkor lehet közvetlenül egyesíteni, ha a kezdőállapotokba nem térhetünk vissza, és az elfogadó állapotokból nem vezet ki nyíl. Ellenkező esetben ugyanis, mint azt már megbeszéltük, az egyik automatában egy kört futva fordulhatunk át a másik

automatába, és így egyik nyelvhez sem tartozó jelsorozatokat is elfogadhatunk. Hasonló probléma léphet fel a konkatenáció és a tranzitív lezárás esetében is.

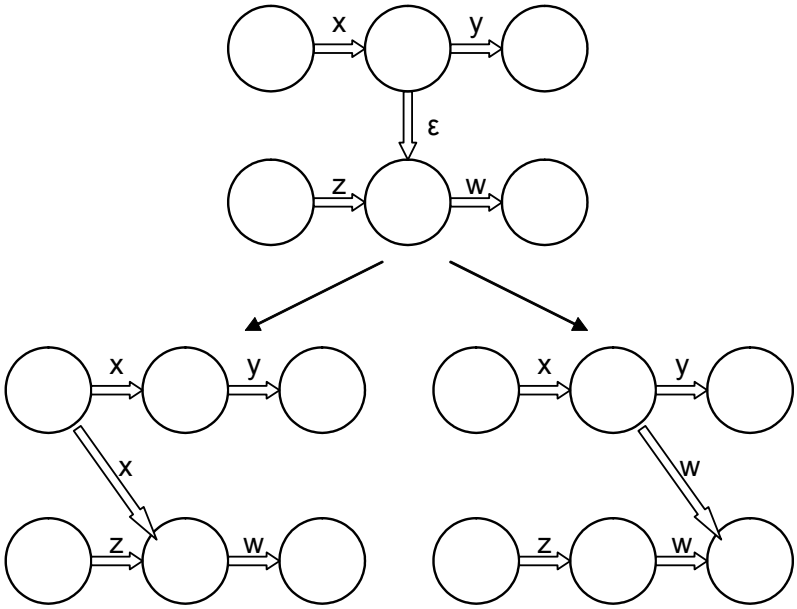
Könnyű belátni, hogy az ε-mozgások illetve az ε-nyilak alkalmazása segít ezen a bajon. Az ilyen ε-mozgások, illetve ε-nyilak használata az automataelméletben teljesen legális. A matematikai nyelvészet apostola *Chomsky* azonban nem engedélyezte az ilyen mozgásokat a **3**-as nyelvosztályban.

Itt rögtön felmerül az a kérdés, hogy az ε-szabályok száműzése nem csökkenti-e az automaták erejét. A válasz most is negatív.

Minden olyan ε-szabályokat is tartalmazó automatához szerkeszthető olyan, ilyen eretnekséget nem mutató automata, amely ugyanazt a nyelvet fogadja el.

A bizonyítás ismét konstruktív lesz, vagyis megszerkesztjük az eredeti automatával egyenértékű, de ε-szabályt nem tartalmazó automatát.

A 2.25. ábrán egy automata olyan részletét tüntettük fel, amely ε-nyilat is tartalmaz.



2.25. ábra

Az ábrán ügyeltünk arra, hogy mind az ε-nyíl kiinduló mind célállapotánál legyenek befutó és kimenő nyilak is. Az ábra két lehetőséget is mutat az ε-nyíl elhagyására.

Mindkét ábrán kiemeltük az ε-nyilat. Az első részábrán azokat a nyilakat multiplikáltuk, amelyek az ε-nyíl kiinduló állapotába vezettek. A multiplikált nyilak a célállapotba mutatnak.

Világos, hogy minden valódi, nem ε-mozgást itt is végre tudunk hajtani. Ezen túlmenően egyetlen mozgással tudjuk megoldani azoknak a mozgáspároknak modellezését, ahol egy valódi mozgást egy ε-mozgás követ.

A második megoldásnál azokat a nyilakat multiplikáltuk, amelyek az ε-mozgás célállapotából indulnak ki. Minden ilyen nyílhoz tartozik egy olyan nyíl, amely az ε-mozgás induló állapotából ered. A valódi mozgások itt is megmaradnak, de egyetlen mozgással modellezzük azokat a mozgáspárokat is, amelyeknek első eleme az ε-mozgás.

Nyilvánvaló, hogy az eredeti és a módosított automata ugyanazt a nyelvet fogadja el. Ha az automatában több ε-nyíl van, akkor ezeket külön-külön ezzel a módszerrel kiemelhetjük az automatából.

A két lehetőség közötti választás általában rajtunk áll. Ha kedvünk úgy tartja azokat a mozgáspárokat modellezzük egyetlen mozgással, amelyeknek első eleme az ε-mozgás, de dönthetünk másképpen is. Általános esetben.

Ha azonban az ε-nyíl az automata kezdőállapotából indul ki, akkor természetesen csakis olyan modellezést választhatunk, ahol az első mozgás az ε-mozgás. Ugyanígy ha az ε-mozgás egy elfogadó állapotba vezet, akkor éppen ellenkezőleg azokat a párokat kell egyetlen valódi mozgással modelleznünk, ahol a második mozgás az ε-mozgás.

Ilyenkor tehát nem rajtunk, hanem az automatán áll a választás.

Most már úgyszólván minden esetet megvizsgáltunk. Egy eset azonban még hátra van. Mit tegyünk akkor, ha az ε-nyíl kezdőállapotból elfogadó állapotba vezet.

Sajnos ilyenkor előbb el kell érnünk, hogy a kezdőállapotba ne vezessen, és az elfogadó állapotból ne induljon ki nyíl. Erre már ismeretes a módszer, amely itt is, tehát ε-nyilakat is tartalmazó automatáknál is alkalmazható. Itt persze előfordulhat az a kellemetlen eset, amikor az átalakítás során mi hozunk létre új ε-nyilakat.

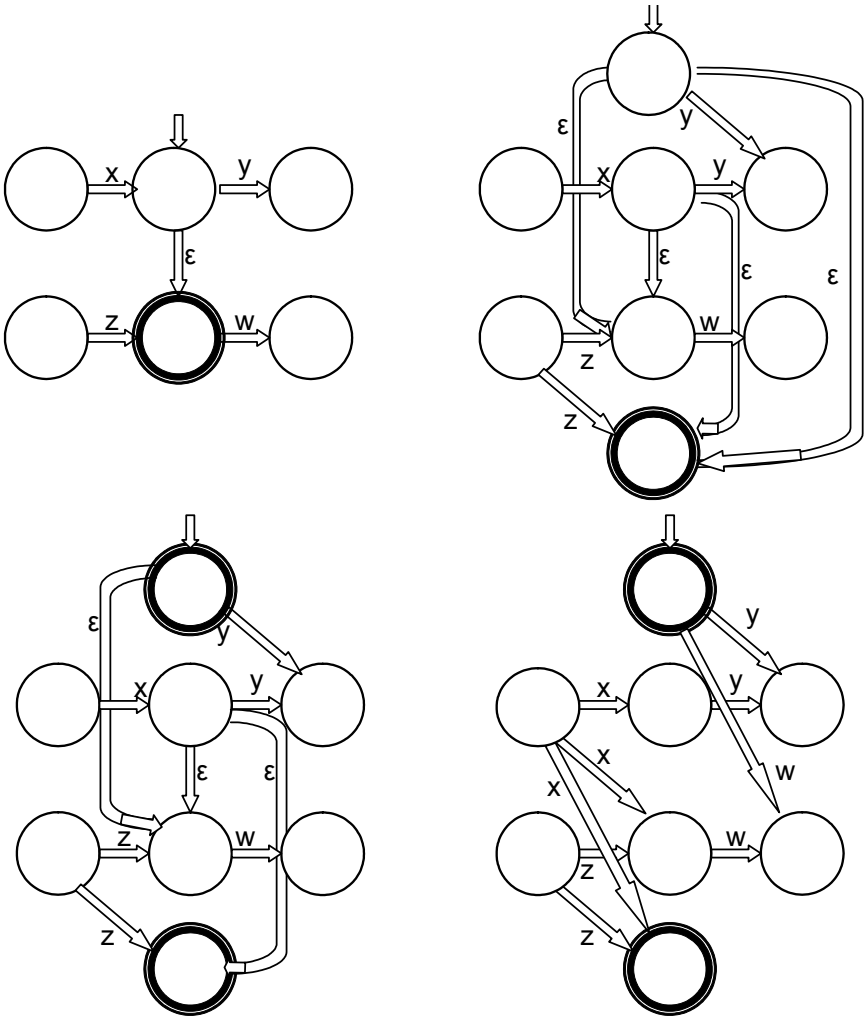
Feltételezve, hogy ezt a munkát elvégeztük, van egy automatánk, amelynek kezdőállapotából csak kiindulnak, elfogadó állapotába viszont csak befutnak nyilak. Ezen nyilak között van egy ε-nyíl, amely a kezdőállapotból az elfogadó állapotba vezet.

Vizsgáljuk meg, mikor kerülhet sor ennek a nyílnak felhasználására. Minthogy a kezdőállapotba visszatérni nem lehet, egyetlen lehetőség, hogy evvel a nyíllal kezdjük a mozgássorozatot. Amint azonban megérkezünk az elfogadó állapotba, a sorozat befejeződik, hiszen az elfogadó állapotnak csak befutó nyilai vannak. Ez a mozgássorozat tehát csak egyetlen mozgásból áll, és azt dokumentálja, hogy az üres jelsorozat eleme a nyelvnek.

Ezt az ε-nyíl elvétele után úgy modellezhetjük, hogy a kezdőállapotot is elfogadó állapottá nyilvánítjuk.

Miután a kényes, a kezdőállapotot elfogadó állapottal összekötő ε nyilat kiküszöböltük, a megmaradt illetve az átalakítások során keletkezett ε nyilakat az ismert módon szüntethetjük meg.

Ezt a folyamatot szemlélteti a 2.26. ábra.



2.26. ábra

Így bármilyen legyen is az ε-nyilak helyzete, mindig tudunk egy olyan véges automatát szerkeszteni, amely ugyanazt a nyelvet fogadja el, és nem tartalmaz ε-nyilat.

Ezzel igazoltuk azt az állításunkat, hogy az ε-mozgások megtiltása nem csökkenti az automata erejét.

3. Környezetfüggetlen nyelvek

3.1 A levezetési fa

Mint már említettük azokat a nyelveket nevezzük környezetfüggetlen nyelveknek, amelyeket a **2**-es nyelvosztályba tartozó nyelvtanok generálnak. Az ilyen nyelvtanoknak csak

$$A \to \alpha \qquad (3.1.)$$

alakú levezetési szabályai lehetnek. Adott esetben a *környezetfüggetlen* jelző arra utal, hogy egy mondatszerű formában a (3.1.) mintájú helyettesítési szabály az *A* nemterminális környezetétől függetlenül, bárhol alkalmazható. A helyettesítési szabályok formáját tekintve a környezetfüggetlen nyelveket generáló nyelvtanok nagyobb szabadságot élveznek, mint a reguláris nyelveket generálóak. Ez egy sereg új, eddig elő sem adódó problémát vet fel, amelyeket megpróbálunk az alábbiakban bemutatni, és ha lehet megválaszolni.

Lássunk ezért egy példát környezetfüggetlen nyelvre:

$$E \to E{+}T \mid T \qquad T \to T{*}F \mid F \qquad F \to (E) \mid a \qquad (3.2.)$$

Engedtessék meg, hogy ehhez a nyelvtanhoz, amellyel még sokszor lesz dolgunk, néhány megjegyzést fűzzek. Ez annál is inkább indokolt, mert a fenti nyelvtan jelentősége messze túlmutat a példán.

Ami a leírás formalizmusát illeti, konvencióinkat csak részben tartottuk be. A nemterminális szimbólumokat előírásszerűen latin nagybetűkkel jelöltük, kis jóindulattal még azt is mondhatjuk, hogy az ábécé elejéről. A terminális szimbólumok közé azonban az *a* szimbólum mellett konvenciónkkal ellentétben a +, ∗, (és) szimbólum is bekerült. Vigyázat! Itt tehát a zárójel nem a nyelvtant leíró metanyelv szimbóluma, hanem a generált nyelv egy, pontosabban egy-egy karaktere.

Annak oka, hogy jelen esetben feladtuk a latin kisbetűk alkalmazására vonatkozó megállapodásunkat az, hogy ezeknek a szimbólumoknak ebben a nyelvben konvencionális jelentésük van, nevezetesen az összeadás illetve szorzás operátora, továbbá a kezdő- és végzárójel.

A mondatszimbólum jele itt nem *S,* mint azt megszoktuk. A nemterminális szimbólumok jele ugyanis szintén jelentést hordoz, adott esetben a számítástechnikában széltében-hosszában használt angol nyelven (*E – expression, T – term, F - factor*).

A nyelvtan felírásánál éltünk azzal az egyszerűsítő jelölésmóddal, hogy az azonos baloldalakat csak egyszer írtuk le, a jobboldalakat pedig a *vagy* olvasatú | jellel választottuk el egymástól.

Tételezzük fel, hogy az *a* terminális szimbólum egy *azonosító,* vagy egy szám helyett áll. Ekkor világossá válik a nyelv „értelme", a nyelvtan egy additív + és egy multiplikatív * operátort tartalmazó, és a zárójelezést megengedő aritmetikai kifejezéseket generálja.

Megjegyzem, hogy pontosan a fenti mintára be lehetne vezetni a –, / és ↑ operátorokat a kivonás, osztás és hatványozás műveletére, de nem okozna számottevő nehézséget a + és – monadikus operátorok, előjelek bevezetése sem.

Természetesen egy igazi számítástechnikai nyelv aritmetikai kifejezéseket generáló CF nyelvtana tartalmazza mindezeket. Mondanivalónk demonstrálására azonban elegendő, ha a nyelv összesen két különböző precedenciájú operátort tartalmaz, és lehetőség van a precedencia zárójelezés útján való megtörésére. Éppen ezért részben az áttekinthetőség növelése érdekében, részben papírtakarékossági okokból megelégszünk a (3.2.) nyelvtannal.

A nyelvnek nyilvánvalóan eleme az

$$a+a*a$$

jelsorozat. Ezt az alábbi levezetés igazolja:

$$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow a+T \Rightarrow a+T*F \Rightarrow a+F*F \Rightarrow a+a*F \Rightarrow a+a*a$$

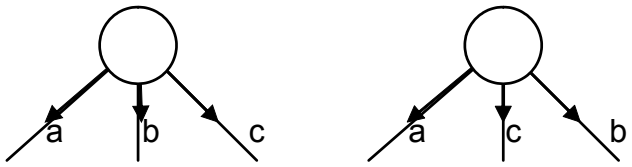
Ez tehát a mondat levezetése. Pontosabban egy levezetése, hiszen a mondatot másképpen is le lehet vezetni. Demonstráljuk ezt ennek a mondatnak egy másik levezetésével.

$$E \Rightarrow E+T \Rightarrow E+T*F \Rightarrow E+T*a \Rightarrow E+F*a \Rightarrow E+a*a \Rightarrow T+a*a \Rightarrow F+a*a \Rightarrow a+a*a$$

Amennyiben egy mondatszerű formában egynél több nemterminális szimbólum van, akkor szükségszerűen egynél több levezetés létezik, hiszen a következő mondatszerű formát különböző nemterminálisokat helyettesítve kaphatjuk meg.

Rögtön felmerül a kérdés, okoz-e zavart, és ha igen mikor az a tény, hogy egy mondatnak egynél több levezetése van. Erre a választ a levezetési fák bevezetése adja meg.

Ábrázoljuk a levezetést egy rendezett, irányított fával. A rendezettség itt annyit jelent, hogy az egy csomópontból kiinduló élek sorrendje kötött, és nem változtatható meg. Így például a 3.1. ábrán feltüntetett két gráf nem egyenértékű.



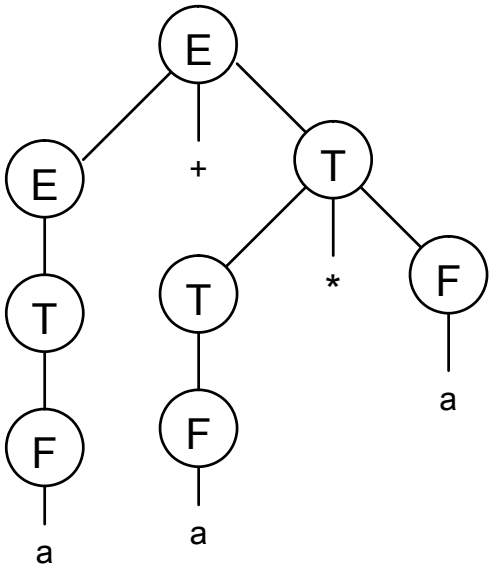
3.1. ábra

Mint említettük a levezetési gráf fa lesz, innen a levezetési fa elnevezés. A fa csomópontjainak a nemterminális, leveleinek pedig a terminális szimbólumok felelnek meg. A fa gyökere, – irányított gráfról van szó, – a mondatszimbólum.

Minden csomópontból annyi él és abban a sorrendben fut ki, amennyi a szabály jobboldalán található szimbólumok száma és a kifutó élek végén sorban a jobboldal megfelelő szimbólumai helyezkednek el.

Minden levezetésnek egy és csakis egy levezetési fa felel meg, ugyanakkor egy levezetési fához egynél több levezetés is tartozhat.

Vegyük például nyelvtanunkat és mondatunkat. A 3.2. ábra tünteti fel a levezetési fát, amely mindkét levezetésre azonos.



3.2. ábra

A levezetési fában az egyes csomópontokhoz tartozó részfákat szintaktikai egységeknek nevezzük. Minden levél, minden terminális szimbólum önmagában szintaktikai egységet alkot.

Azokat a levezetéseket, amelyek levezetési fája azonos, tehát ahol a jelsorozat felbontása szintaktikai egységekre megegyező, nem tekintjük lényegesen különböző levezetéseknek.

Vegyük észre, hogy a fenti nyelvtan ismeri az operátorok precedenciáját. Előbb kell ugyanis a szorzást elvégezni, az alkot egy szintaktikai egységet, és a szorzatot kell az első terminális szimbólumhoz hozzáadni, pontosan úgy, ahogy azt az iskolai tanulmányainkban tanultuk.

Azonos precedenciájú operátorok esetében a nyelvtan az úgynevezett balról-jobbra szabályt alkalmazza. Ez annyit jelent, hogy két azonos precedenciájú, tehát adott esetben két + vagy két * operátor esetében először mindig a baloldali operátorral meghatározott műveletet „hajtja végre”.

Fentiekből következik, hogy amennyiben az aritmetikai kifejezéseket a (3.2.) nyelvtannal illetve annak kibővített változatával generáljuk, akkor a

szintaktikai egységek minden külön beavatkozás nélkül követik a precedenciákat, és a balról-jobbra szabályt.

Az olyan nyelvtanokat, ahol minden mondathoz egy és csakis egy levezetési fa tartozik, egyértelmű nyelvtanoknak mondjuk. Természetesen ez nem zárja ki, hogy egy mondatnak több levezetése legyen, ezek azonban nem lehetnek lényegesen különbözőek.

Egyértelmű nyelvtanok esetében, minthogy a levezetési fa unikális, a szintaktikai felbontás egyértelmű.

A (3.2.) nyelvtan egyértelmű nyelvtan. Mint említettük, és mint a példa is mutatja, egy mondatnak egyértelmű nyelvtanok esetében is lehet több levezetése.

A levezetések között vannak nevezetesek. Amikor a levezetés során mindig a mondatszerű forma legbaloldalibb nemterminális szimbólumát helyettesítjük, baloldali levezetésről beszélünk. Ugyanígy a jobboldali levezetés esetében mindig a mondatszerű forma legjobboldalibb nemterminális szimbólumát bontjuk fel a következő lépésben.

A példánkban éppen a baloldali és jobboldali levezetéseket demonstráltuk.

Egyszerűsítsük most kissé nyelvtanunkat:

$$E \to E{+}E \mid E{*}E \mid (E) \mid a \qquad (3.3.)$$

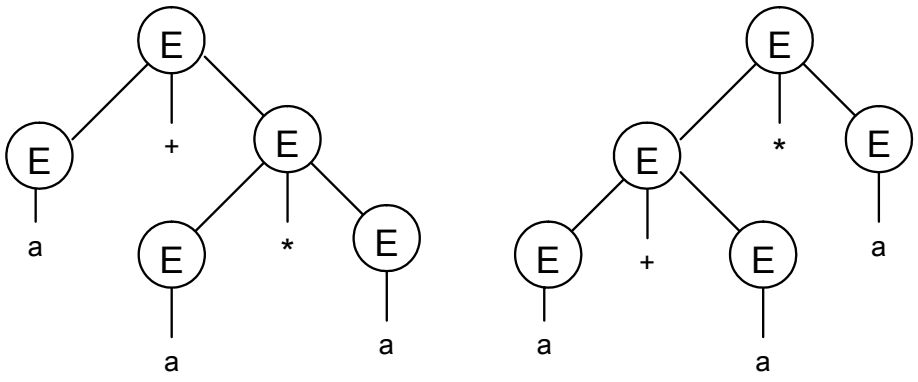
Ez a nyelvtan ugyanazt a nyelvet generálja, mint a (3.2.) szabályokkal megadott nyelvtan. Így a példaként megadott mondatot ezzel a nyelvtannal is lehet generálni:

$$E \Rightarrow E{+}E \Rightarrow E{+}E{*}E \Rightarrow a{+}E{*}E \Rightarrow a{+}a{*}E \Rightarrow a{+}a{*}a$$

Próbálkozzunk most, mint előbb, egy másik levezetéssel:

$$E \Rightarrow E{*}E \Rightarrow E{*}a \Rightarrow E{+}E{*}a \Rightarrow E{+}a{*}a \Rightarrow a{+}a{*}a$$

A 3.3. ábrán megrajzoltuk a két levezetés levezetési fáit. Mind a két fát, ugyanis itt a fák nem azonosak, a két levezetés lényegesen különbözik.



3.3. ábra

Amennyiben csak a tartalmazás problémájára szorítkozunk, vagyis csak azt vizsgáljuk, generálható-e valamely jelsorozat egy adott nyelvtannal, akkor a (3.2.) és (3.3.) nyelvtanok egyenértékűek.

Lényegében ez a matematikai nyelvészet megközelítése. A számítástechnikai nyelvészeté szükségképpen más.

A számítástechnikai nyelvészet szempontjából messzemenően nem közömbös, milyen szintaktikai egységeken keresztül jutottunk el a levezetés során a mondathoz. Példánkra gondolva létkérdés tudnunk, melyik műveletet kell előbb elvégeznünk, az összeadást vagy a szorzást.

Éppen ezért a számítástechnika szempontjából a nem egyértelmű nyelvtanok használhatatlanok. Az eddigi tapasztalatokat összegezve egy sereg a számítástechnika szempontjából nézve alapvetőnek tekinthető kérdés vetődik fel.

Az egy- illetve többértelműséget nyelvtanhoz és nem nyelvhez kapcsoltuk. Példánkban ez nyilván helyes volt, hiszen ugyanannak a nyelvnek egy egyértelmű és egy többértelmű nyelvtanáról volt szó.

Vajon mindig ez a helyzet, vagy lehet-e a többértelműség nyelvi tulajdonság is? Ki lehet-e mutatni, hogy egy nyelvtan egyértelmű-e, és ha nem, meg lehet-e állapítani, hogy létezik-e a nyelvnek egyértelmű nyelvtana.

Az első kérdésre a válasz megadható.

Léteznek olyan nyelvek, amelyről bizonyítható, hogy nem lehet egyértelmű nyelvtannal generálni. Ez esetben tehát a többértelműség nyelvi tulajdonság. Más szóval nincsen minden környezetfüggetlen nyelvnek egyértelmű nyelvtana.

Ugyanakkor példánkra gondolva, ahol a többértelműség nyelvtanhoz volt kötve, rögzíthető, hogy a többértelműség egyszer nyelvhez másszor nyelvtanhoz kapcsolható tulajdonság.

Lássunk példát nem egyértelmű nyelvre.

$$S \to aAbX \mid YbCc \qquad A \to aAb \mid ab\,C \to bCc \mid bc$$
$$X \to cX \mid c \qquad Y \to aY \mid a$$

Ez a nyelvtan az

$$\boldsymbol{L} = a^i b^i c^j \cup a^j b^i c^i$$

nyelvet generálja.

Az triviális, hogy az adott nyelvtan az $a^i b^i c^i$ alakú mondatot két lényegesen különböző levezetéssel állíthatja elő. Ezen túlmenően igazolható, hogy nincsen olyan egyértelmű környezetfüggetlen nyelvtan, amely a szóban forgó nyelvet generálná. Itt tehát a többértelműség nyelvhez kapcsolható tulajdonság.

A második kérdéscsoportra a válasz negatív. Sajnos nincs olyan módszer, amelynek segítségével általánosan, minden esetben alkalmazhatóan meg lehetne állapítani egy nyelv vagy egy nyelvtan egyértelmű voltát. Ez is az algoritmikusan eldönthetetlen problémák sorába tartozik.

Mint említettük, az a tény, hogy egy kérdés algoritmikusan eldönthetetlen, vagyis a probléma megoldására nincs általános módszer, nem jelenti azt, hogy egyes konkrét, ebbe a feladatosztályba tartozó feladatra ne lehetne adott esetben választ adni. Persze nem minden egyedi feladatra, hiszen akkor a probléma algoritmikusan eldönthető lenne.

Példaképpen igazoljuk, hogy a (3.2.) nyelvtan egyértelmű.

A mondatot alkotó jelsorozatot szintaktikai egységekre osztjuk, majd ezeket újabb, kisebb egységekre mindaddig, amíg a szintaktikai egységek egyetlen terminális szimbólumból nem állanak. Be kell látnunk, hogy ez a felosztás egyértelmű.

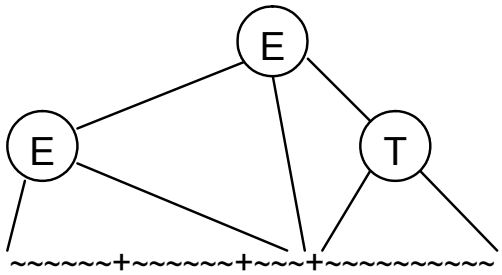
Induláskor a teljes jelsorozatot tekintjük szintaktikai egységnek, amelyet az *E* mondatszimbólumnak le kell generálnia. Két lehetőségünk van, vagy az

$$E \to E{+}T \qquad \text{vagy az} \qquad E \to T$$

szabályt alkalmazhatjuk. Amennyiben a jelsorozat tartalmaz zárójelen kívüli additív operátort, + szimbólumot nyilván az első szabályt kell alkalmaznunk. Ellenkező esetben ugyanis a *T* nemterminális szimbólumot kapnánk, amely nem tud + operátort tartalmazó jelsorozatot generálni.

Ha a jelsorozat nem tartalmaz + operátort akkor viszont a második szabály a helyes, hiszen az elsőt alkalmazva nem tudnánk mit kezdeni a generált + terminálissal.

Tételezzük fel, hogy az első szabály kellett felhasználnunk. Ilyenkor a szabály + terminális szimbólumát megfeleltetjük a jelsorozat egyik + operátorának, az operátor előtti jelsorozat alkot egy szintaktikus egységet, amelyet a szabály *E* nemterminálisa generál majd, az operátort követő rész másik szintaktikus egység lesz, amelyet viszont a szabály *T* nemterminális szimbóluma állít elő. Ezt vázolja a 3.4. ábra.



3.4. ábra

Amennyiben jelsorozatban több zárójelen kívüli + operátor található szükségszerű, hogy a szabály + szimbólumát az utolsó + operátorral azonosítsuk, hiszen ellenkező esetben a megfeleltetett + operátortól jobbra is maradna még + operátor, amit viszont a *T* nemterminális nem képes generálni. Így tehát csak egyetlen megfeleltetési lehetőségünk van a jelsorozat levezetésében.

A baloldali szintaktikus egység felbontásánál, amelyet az *E* nemterminális generál, és amely tartalmazhat + operátorokat, az előbbi gondolatmenet rekurzív alkalmazásával érünk célt. Végül sikerül a jelsorozatot olyan szintaktikus egységekre bontani, amelyek már nem tartalmaznak zárójelen kívüli additív operátort, és amelyeket a *T* nemterminális szimbólumok generálnak.

Itt a multiplikatív operátorok játszzák ugyanazt a szerepet, amelyet az előbb az additívak játszottak. Ha van zárójelen kívüli * operátor, akkor a $T \rightarrow T*F$ szabályt kell alkalmaznunk, mégpedig úgy, hogy a szabály * terminálisát a szintaktikus egység utolsó * operátorának feleltetjük meg. Ellenkező esetben a $T \rightarrow F$ szabály kerül sorra.

Ezután csak olyan szintaktikus egységek maradnak, amelyek vagy zárójeles kifejezések, vagy egyedül az *a* terminális szimbólumból állanak. Az előbbi esetben az $F \rightarrow (E)$ szabály után a zárójelen belüli kifejezésre alkalmazzuk a fenti algoritmust, az utóbbiban az $F \rightarrow a$ szabállyal befejezzük az elemzést.

A fentiekből kitűnik, hogy a (3.2.) nyelvtan alapján minden mondatot egyértelműen lehet csak felbontani, így csak egy levezetési fa tartozik hozzá, a nyelvtan egyértelmű.

A másik (3.3.) nyelvtan esetében találtunk olyan mondatot, és a mondathoz két különböző levezetést, amelyhez két különböző levezetési fa tartozik. Ezzel a nyelvtan többértelműségét demonstráltuk.

Sajnos a gyakorlatban nem mindig sikerül vagy a nyelvtan egyedi tulajdonságait, vagy egy szerencsés ötletet felhasználva a nyelvtan egy- vagy többértelműségét bizonyítanunk. Ilyenkor a kérdés megválaszolatlan marad.

Tekintsünk most egy pillanatra vissza, és vizsgáljuk meg, vajon felmerül-e egyáltalában az egyértelműség többértelműség kérdése reguláris nyelveknél!

A levezetési fát természetesen reguláris nyelveknél is megrajzolhatjuk. Itt ez a fa nagyon egyszerű lesz. Csupán egy szárból áll, amelynek jobb- vagy baloldalán vannak levelei attól függően, hogy bal- vagy jobbreguláris nyelvtanról van szó.

Egy levezetési fához itt csak egy levezetés tartozik, ami abból nyilvánvaló, hogy minden mondatszerű formában csak egyetlen nemterminális szimbólum szerepel.

Nemdeterminisztikus nyelvtanok esetében előfordulhat, hogy egy mondatnak több, szükségszerűen lényegesen eltérő, különböző levezetési fája van. Korábban igazoltuk, hogy minden reguláris nemdeterminisztikus nyelvtanhoz található egy vele egyenértékű, determinisztikus nyelvtan. Algoritmust is adtunk, amely a nemdeterminisztikus véges automata alapján megszerkeszti ezt a determinisztikus automatát.

Ennek alapján rögzíthetjük, hogy minden reguláris nyelv egyértelmű, és szerkeszthető hozzá egyértelmű nyelvtan.

3.2. Nyelvtanok átalakítása

A környezetfüggetlen nyelvtanokkal kapcsolatosan eddig egyetlen megkötést említettünk. A nyelvtan levezetési szabályainak alakja a (3.1.) szerinti kell legyen.

Célszerűségi és esztétikai szempontból azonban további igényeket is támaszthatunk a nyelvtan külalakjával, a helyettesítési szabályok formájával kapcsolatosan.

Az alábbiakban néhány ilyen többlet követelményt ismertetünk, és megadjuk azokat az algoritmusokat, amelyek segítségével az ezen követelményeket nem kielégítő nyelvtanból egy, az eredetivel egyenértékű és az új követelményeket is teljesítő nyelvtan származtatható.

Természetesen mindkét nyelvtan, az eredeti, az igényeket nem kielégítő, és az ebből kialakított szebb és célszerűbb nyelvtan is környezetfüggetlen. Ha szabad klasszikusokra hivatkozni azt mondhatjuk, hogy mindkét nyelvtan környezetfüggetlen, de az utóbbi még környezetfüggetlenebb.

Szóljunk először a felesleges szimbólumokról!

Egy terminális szimbólum akkor felesleges, ha nincs a nyelvnek olyan mondata, amelyben a szóban forgó szimbólum szerepelne.

Egy nemterminális szimbólum akkor felesleges, ha nincs a nyelvnek olyan mondata, amelynek levezetésében található mondatszerű formák legalább egyikben az illető nemterminális előfordulna.

Legyenek valamely nyelvtanban érvényesek a következő levezetések:

$$S \overset{*}{\Rightarrow} w = xay \tag{3.4.}$$

$$S \overset{*}{\Rightarrow} \alpha A \beta \overset{*}{\Rightarrow} w \tag{3.5.}$$

Amennyiben a (3.4.) és (3.5.) levezetések lehetségesek, akkor sem az a terminális, sem az A nemterminális szimbólum nem felesleges.

A felesleges szimbólumok kiszűrésére szolgáló algoritmus lényegében a szimbólumok nem felesleges voltát állapítja meg, és ezeket a nem felesleges szimbólumokat összegyűjti. Ami marad, az felesleges.

Legyen adott egy **G** grammatika. Képezzünk egy halmazsorozatot az alábbi szabályszerűség szerint.

$$\boldsymbol{B}_{i+1} = \boldsymbol{B}_i \cup \{ A \mid A \to \alpha, \alpha \in \boldsymbol{B}_i^* \} \tag{3.6.}$$

Ezek szerint a következő halmazt úgy kapjuk, hogy az előző halmazhoz hozzávesszük azokat a nemterminális szimbólumokat, amelyeknek létezik olyan levezetési szabálya, ahol a jobboldalon minden szimbólum az előző halmaz eleme.

A halmazsorozat számossága egy határig monotonon növekszik. Nyilvánvaló, hogy amennyiben egy lépés alkalmával nem találunk új, a (3.6.) összefüggés értelmében az eredeti halmazhoz hozzáfűzhető elemet, akkor a továbbiakban a halmaz számossága nem növekszik, a sorozat képzését abbahagyhatjuk.

Jelölje $\boldsymbol{B}$ az ily módon kapott legnagyobb számosságú halmazt. Amennyiben kiindulásul a terminális szimbólumok halmazát választjuk,

$$\boldsymbol{B}_0 = \Sigma$$

akkor a $\boldsymbol{B}$ halmaz a terminális szimbólumokon kívül azokat és csakis azokat a nemterminális szimbólumokat fogja tartalmazni, amelyekből kiindulva a helyettesítési szabályok alkalmazásával egy pusztán terminális szimbólumokból álló jelsorozat állítható elő.

Ezzel mintegy alulról felfelé *bottom-up* – innen adódik a $\boldsymbol{B}$ jelölés – fésüljük végig a grammatika szabályait, és csakis azokat a szimbólumokat vesszük be a halmazunkba, amelyekre nézve van remény, hogy nem feleslegesek.

Vegyünk ismét egy halmazsorozatot, de válasszunk most más képzési szabályt.

$$\boldsymbol{T}_{i+1} = \boldsymbol{T}_i \cup \{ X \mid A \to \alpha X \beta, A \in \boldsymbol{T}_i \} \qquad (3.7.)$$

Új halmaz képzésekor tehát az eredeti halmazhoz hozzá kell venni azon helyettesítési szabályok jobboldalán található terminális és nemterminális szimbólumokat, amely szabályok baloldala az eredeti halmazhoz tartozó nemterminális szimbólum.

A halmazsorozat számossága itt is monotonon növekszik, és itt is akkor áll meg a növekedés, ha egy lépésben nem tudunk az eredeti halmazhoz egy szimbólumot sem hozzáfűzni.

Jelölje itt a legnagyobb elemszámú halmazt $\boldsymbol{T}$. Legyen a kiindulásként választott halmaz egyedül a mondatszimbólumot tartalmazó halmaz:

$$\boldsymbol{T}_0 = \{ S \}$$

akkor a $\boldsymbol{T}$ halmaz azokat és csakis azokat a terminális és nemterminális szimbólumokat tartalmazza majd, amelyek egy, a mondatszimbólumból kiinduló levezetés mondatszerű formáiban szerepelhetnek.

Ezzel az algoritmussal felülről lefelé *top down* – innen a $\boldsymbol{T}$ jelölés – fésüljük meg a grammatika szabályait.

Lássunk egy példát. Legyen a grammatika:

$$S \to a \quad S \to B \quad B \to BC \quad C \to b$$

A nyelvtant alulról felfelé megfésülve a következő halmazsorozatot kapjuk:

$$\boldsymbol{B}_0 = \{ a,b \} \qquad \boldsymbol{B}_1 = \boldsymbol{B} = \{ a,b,S,C \}$$

A halmaz számossága tovább nem növelhető. Mint látható a B nemterminális szimbólum nem került bele a halmazba, így feleslegesnek bizonyult. Ezt a szimbólumot tehát, és minden olyan helyettesítési szabályt, amelyben ez a szimbólum szerepel, kihagyhatunk a nyelvtanból. Ez most így megkurtítva mindössze egy tartalmaz:

$$S \to a \quad C \to b$$

Ezt a nyelvtant felülről lefelé fésülve a következő halmazsorozatot kapjuk:

$$\boldsymbol{T}_0 = \{ S \} \qquad \boldsymbol{T}_1 = \boldsymbol{T} = \{ S,a \}$$

Végeredményben tehát a nyelvtan egyetlen helyettesítési szabályt tartalmaz, $S \to a$ és egyetlen mondatot generál.

Amennyiben a fésülést fordított sorrendben végeztük volna el, vagyis először alkalmaztuk volna a *top down* és utána a *bottom-up* vizsgálatot, akkor az előbbitől eltérő eredményt kaptunk volna. A nyelvtannak helytelenül két szabálya, nevezetesen $S \to a$ és $C \to b$ is megmaradt volna. Így tehát a kígyó a fejétől a farkáig nem ugyanolyan hosszú, mint a farkától a fejéig.

Ennek a látszólagos ellentmondásnak a forrását azok a nemterminális szimbólumok képezik, amelyekből nem vezethető le terminális jelsorozat. Ilyen példánkban a B nemterminális. Ha egyszer ilyen nemterminálisok bekerülnek egy mondatszerű formába, akkor azokat onnan hosszabb távon sem lehet kiebrudalni, az ilyen nemterminálisokat nem lehet agyonütni.

Az ilyen agyonüthetetlen nemterminálisok természetesen felesleges szimbólumok, hiszen belőlük nem vezethető le csupa terminálisból álló jelsorozat. Ezeket a szimbólumokat az alulról felfelé történő fésüléskor felismerjük, de nem detektáljuk felesleges voltukat, ha lefelé fésülünk.

Vizsgáljuk meg, miért nem lehet ezeket a nemterminálisokat, ha egyszer belekerültek a mondatszerű formába, onnan kiirtani.

Ha valamely szimbólumhoz nem tartozik olyan levezetési szabály, amelynek éppen a szimbólum lenne a baloldala, akkor triviális, hogy ez a szimbólum onnan el nem távolítható, örökre benne marad a mondatszerű formában.

A B szimbólum esetében nem ez a helyzet, mégsem tudunk megszabadulni tőle, nem kaphatunk belőle terminális jelsorozatot. Ennek jobb megértésére célszerű megismerkednünk a rekurzivitás fogalmával.

Egy nemterminális szimbólumot rekurzívnak mondunk, ha származtatható belőle olyan jelsorozat, amely az eredeti szimbólumot tartalmazza.

$$A \overset{*}{\Rightarrow} \alpha A \beta \tag{3.8.}$$

Egy nyelvtan akkor rekurzív, ha tartalmaz rekurzív nemterminálist.

Nem rekurzív nyelvtanok csak véges mondatot generálnak. Minden véges nyelv reguláris nyelvtannal generálható. Ezek igazolását az olvasóra bízom.

A fentiekből következik, hogy minden végtelen sok mondatot generáló nyelvtan rekurzív.

Természetesen a rekurzív nyelvtanoknál mindig kell találnunk egy egérutat, egy olyan levezetési lehetőséget, amikor a rekurzivitást előidéző nemterminális szimbólum többet nem fordul elő. Pontosan ugyanúgy, ahogy egy rekurzív eljáráshívást alkalmazó program esetében, itt is kötelező egy olyan alternatíva, amikor az eljárást nem hívjuk meg rekurzív módon.

Így például az alábbi formálisan helyes, a szintakszis szabályait kielégítő ALGOL 60 nyelven írt programfragmens nem rekurzív eljárást tartalmaz, hanem egyszerűen badarság.

```
integer procedure FACT (N);
value N, integer N;
FACT := N * FACT(N-1);
```

Nyilvánvaló, ha egyszer meghívtuk ezt az eljárást, akkor az – elvben – soha az életben nem hagyja abba a számolást.

Nevezzük ezt az ostobaságot álrekurzivitásnak.

Mielőtt továbbmennénk, mutassuk meg az előbbi, a faktoriális kiszámítására hivatott program helyes változatát:

```
integer procedure FACT (N);
value N, integer N;
FACT := if N=0 then 1 else
N * FACT(N-1);
```

Mint látható az N=0 esetre van egérutunk.

Itt persze feltételeztük, hogy az eljárást csak nem negatív számokkal mint aktuális paraméterekkel hívjuk meg. Ha ez a feltétel nem biztosítható, akkor valamilyen ellenőrzést, védelmet kell a programba beépítenünk.

Az álrekurzió ugyanúgy fenyeget minket nyelvtanok esetében. Itt is előfordulhat, hogy egy alkalmas nemterminális szimbólum, ha egyszer bekerült a mondatszerű formába, ott is ragad, nem tudunk tőle szabadulni. Pontosan ez a helyzet a *B* szimbólummal. Minthogy ez álrekurziót okoz nincsen olyan mondat, amelynek levezetésében a *B* szerepelhetne, így természetesen felesleges szimbólum.

Példánkban a rekurzivitás közvetlen volt, a levezetési szabály baloldalán található szimbólum szerepelt ugyanazon szabály jobboldalán is. Ez persze nem szükségszerű. Lehetséges, hogy a szimbólum csak több levezetési lépés után tér vissza. Ebben az esetben álrekurzióról akkor beszélünk, ha ebből az ördögi körből nem tudunk kilépni, az egyszer már előfordult nemterminális újabb felbukkanása elkerülhetetlen.

Vegyük észre, hogy az álrekurzióval szemben a két fésülés, a felülről lefelé és az alulról felfelé történő, másképpen viselkedik. Alulról fésülve az ilyen szimbólumok átcsúsznak a fésű fogain, és feleslegesnek minősülnek. Felülről lefelé azonban az álrekurzió ténye nem detektálható, az ilyen szimbólumok benne maradnak a nyelvtanban.

Amennyiben a fésülést az álrekurziót kiszűrő módon alulról kezdjük, akkor két fésüléssel valóban minden felesleges szimbólumtól megszabadulunk.

Ezt beláthatjuk, ha meggondoljuk, hogy a *bottom-up* fésülés eltünteti mindazokat a nemterminálisokat, amelyekből nem vezethető le terminális jelsorozat. Így a (3.5.) összefüggésben az *A* nemterminális nyilván nem ilyen szimbólum, ha egyszer fennakadt a fésűn.

A második *top down* fésülésnél viszont csak azok a szimbólumok maradnak meg, amelyek elérhetőek az *S* mondatszimbólumból. Amennyiben az *A* nemterminális túlélte mindkét fésülést, akkor az

$$S \overset{*}{\Rightarrow} \alpha A \beta \overset{*}{\Rightarrow} w$$

alakú (3.5.) összefüggésben mind az első, mind a második $\overset{*}{\Rightarrow}$ operátorral jelzett levezetése helytálló, ugyanis a fenti gondolatmenet nem csak az *A* nemterminálisra, hanem az α és β jelsorozatokban esetleg található nemterminálisokra is alkalmazható.

A következőkben az $A \rightarrow \varepsilon$ alakú úgynevezett ε-szabályok kiküszöböléséről szólunk.

Már említettük, hogy a környezetfüggő, vagyis **1**-es osztályú nyelveket nem csökkentő nyelvtanok generálják. Az ε-szabályok, ahol a baloldal hossza 1, a jobboldalé viszont 0 természetesen csökkentő szabályok. Minthogy a reguláris és környezetfüggetlen nyelvek a környezetfüggő nyelvek részhalmazai, hogyan engedhető meg ε-szabályok alkalmazása ezekben a nyelvosztályokban?

Az a helyzet, hogy majd finomítanunk kell a nyelvosztályok definícióján, de előbb vizsgáljuk meg az ε-szabályok kiküszöbölési lehetőségeit.

Fésüljük most meg ismét alulról az eredeti nyelvtant, válasszuk azonban most induló halmaznak az üres jelsorozatot!

$$\boldsymbol{B}_0 = \{ \varepsilon \}$$

Nyilvánvaló, hogy most a záróhalmazban azok a nemterminális szimbólumok találhatóak, amelyek elenyészhetnek, vagyis amelyek az ε-szabályok alkalmazása révén közvetlenül vagy közvetve utód és nyom nélkül eltűnhetnek a mondatszerű formából.

Itt különbséget kell tennünk két eset között, eleme-e ennek a halmaznak a mondatszimbólum vagy sem? Amennyiben a mondatszimbólum benne van ebben a halmazban, akkor a mondatszimbólum is elenyészhet, belőle kiindulva le tudjuk vezetni az üres jelsorozatot, az ε eleme a nyelvnek.

Ellenkező esetben, tehát amikor a mondatszimbólum nem eleme a halmaznak, a mondatszimbólum nem enyészhet el, az üres jelsorozat nem eleme a nyelvnek.

Az első esetben, amikor a nyelv tartalmazza az üres jelsorozatot, állítsuk elő a nyelvet két nyelv uniójaként. Az első komponensnek egyetlen eleme lesz, az üres jelsorozat, míg a másodiknak mondatai az üres jelsorozat kivételével megegyeznek az eredeti nyelv mondataival.

Jelölje, mint eddig, $\boldsymbol{L}_\varepsilon$ azt a nyelvet, amelynek egyetlen mondata az üres jelsorozat. Az $\boldsymbol{L}$ nyelvet tehát, amely feltételezésünk szerint tartalmazza az üres jelsorozatot, a következőképpen állítjuk elő

$$\boldsymbol{L} = (\boldsymbol{L} - \boldsymbol{L}_\varepsilon) \cup \boldsymbol{L}_\varepsilon \tag{3.9.}$$

Az $\boldsymbol{L} - \boldsymbol{L}_\varepsilon$ formalizmus az üres jelsorozattól megfosztott eredeti nyelvet jelenti. Az $\boldsymbol{L}_\varepsilon$ nyelvtana csak egyetlen szabályt tartalmaz:

$$Q \to \varepsilon$$

ahol itt azért jelöltük a mondatszimbólumot Q betűvel, mert feltételezésünk szerint S már le van foglalva az eredeti nyelv mondatszimbólumának. A (3.9.) nyelv mondatszimbólumát jelöljük P betűvel, akkor az eredeti nyelv szabályait a következő szabályokkal kell kiegészítenünk:

$$P \to Q \qquad Q \to \varepsilon \qquad P \to S$$

Ha most újra elvégezzük az eredeti nyelvtan vizsgálatát, akkor ugyanazt az eredményt másképpen kell interpretálnunk. A régi mondatszimbólum, amely bekerült az elenyésző nemterminálisok közé, mondatszimbólum rangját elvesztette, így ennek az új nyelvnek nem eleme az üres jelsorozat, ugyanakkor valamennyi ettől eltérő mondat legenerálható. Ez megfelel a fentebb vázolt koncepciónknak.

Hogyan lehet egy nyelvtanból, amely nem generálja az üres jelsorozatot, az ε-szabályokat eltávolítani?

Hagyjuk el az összes ε-szabályt, de ugyanakkor az olyan levezetési szabályoknál, ahol a jobboldalon szerepelnek elenyésző nemterminálisok, az összes lehetséges konfigurációban hagyjuk el ezeket a nemterminálisokat. Az elhagyottakról feltételezzük, hogy elenyésznek, a megmaradt többiről éppen ellenkezőleg azt tesszük fel, hogy belőlük nem üres jelsorozatok származnak. Azzal modellezzük tehát az ε-szabályokat, hogy amennyiben feltételezésünk szerint egy nemterminális elenyészik, akkor azt be sem írjuk a levezetési szabályba.

Persze ez nagyon megnövelheti a nyelvtant, így ha egy levezetési szabály jobboldalán **k** elenyésző szimbólum van, akkor ebből az egyetlen helyettesítési szabályból 2^k új szabály származhat.

Visszatérve a nem csökkentő nyelvtanok problémájára definiáljuk újra az első nyelvosztály nyelveit. Minden olyan nyelvet **1**-es osztálybelinek tekintünk, amely, ha nem tartalmazza az üres jelsorozatot, akkor előállítható nem csökkentő nyelvtan segítségével, ha tartalmazza, akkor viszont előállítható két nyelv uniójaként, ahol az első nyelvnek egyetlen mondata van, az üres jelsorozat, a második pedig az eredeti nyelv valamennyi az üres jelsorozattól eltérő mondatát tartamazza, és nem csökkentő nyelvtannal generálható.

Ez az új felfogás feloldja az ortodox értelmezés miatt jelentkező ellentmondásokat.

Ezek szerint, ha van egy olyan környezetfüggetlen nyelvtanunk, amelyben vannak ε-szabályok, akkor a fentiek szellemében át kell a nyelvtant alakítanunk, hogy a nyelv a környezetfüggő, tehát nem csökkentő nyelvek részhalmazába tartozzék? Szó sincs róla. Az osztályba sorolás nyelveket és nem nyelvtanokat

említ. Ha meggyőződtünk róla, hogy a nyelvnek van a csökkentés tekintetében enyhített kritériumokat kielégítő nyelvtana, akkor ebben megnyugodhatunk, és rajtunk áll, hogy eltávolítjuk-e az ε-szabályokat vagy sem.

Lássunk egy példát az ε-szabályok kiküszöbölésére. Legyen a nyelvtan:

$$S \to SaSb \mid \varepsilon$$

A nyelvtan, mint arról könnyen meggyőződhetünk, olyan azonos számú *a* és *b* karakterekből álló jelsorozatokat generál, ahol a mondatok bármely prefixumában legalább annyi *a* karakter van, mint *b* karakter. Elemezzük ezt a nyelvtant.

A fésülés során keletkezett halmazsorozat:

$$\boldsymbol{B}_0 = \{ \varepsilon \} \qquad \boldsymbol{B}_1 = \boldsymbol{B} = \{ \varepsilon, S \}$$

A mondatszimbólum benne van a záróhalmazban, minthogy a nyelvnek ε eleme. Az ismertetett módon új mondatszimbólumot, legyen ez $\hat{S}$, és új levezetési szabályokat kell bevezetnünk

$$\hat{S} \to \varepsilon \qquad \hat{S} \to S$$

Az eredeti nyelvtanból az ε-szabályt elhagyva mindössze egy szabályunk marad. Ennek jobboldala viszont két elenyésző szimbólumot tartalmaz. Így helyette $2^2 = 4$ új szabályt kell bevezetnünk:

$S \to SaSb \quad S \to aSb \quad S \to Sab \quad S \to ab$

A teljes nyelvtan:

$\hat{S} \to \varepsilon \quad \hat{S} \to S \quad S \to SaSb \quad S \to aSb \quad S \to Sab \quad S \to ab$

Minthogy az eredeti nyelv tartalmazta az üres jelsorozatot, az átalakítás eredményeképpen két nyelv unióját kaptuk. Az első nyelvnek csak egyetlen eleme van, az üres jelsorozat, a másik nyelv nyelvtana nem tartalmaz ε-szabályt, és generálja az eredeti nyelv minden mondatát az ε kivételével.

Míg az eredeti nyelvtan alapján nem túl nagy fáradtsággal a nyelv felismerhető volt, a módosított nyelvtanról már ez kevésbé állítható. Ezzel az átalakítással az áttekinthetőséget feláldoztuk az ortodoxia oltárán.

Általában gyakori, hogy egy nyelvtan világosabb és áttekinthetőbb, mint ε-szabályokat nem tartalmazó egyenértékese. Ez is egyik oka annak, hogy az irodalom meglehetősen nagyvonalú ezen a területen. Meghagyja a régi nyelvtant és megelégszik azzal a megnyugtató tudattal, hogy ki lehetne küszöbölni az ε-szabályokat.

A nyelvtanok csinosságának növelésére szolgáló következő lépés az úgynevezett egyszeres szabályok kiküszöbölése lesz.

Mint már említettük egyszereseknek nevezzük az

$$A \to B \tag{3.10.}$$

alakú szabályokat. Rögtön felmerül a kérdés, miért nem szeretjük ezeket a szabályokat, miért zavarnak minket?

Valójában az egyszeres szabályok léte csak akkor zavaró, ha ciklicitás van bennük, mint például az

$$A \to B \quad B \to C \quad C \to A$$

szabályok esetében.

Általában egy mondat levezetésére korlátos számú véges lépés szükséges. Amennyiben azonban ciklicitás van az egyszeres szabályok között, akkor levezetési lépésekkel vissza tudunk térni ugyanahhoz a mondatszerű formához, más szóval lesznek olyan mondatok, amelyeket nem korlátos számú lépésben is le lehet vezetni. Ez valóban nem szimpatikus jelenség.

Persze egyszeres szabályok jelenléte még nem jelent szükségszerűen ciklicitást, ennek veszélyét viszont legegyszerűbben és legdrasztikusabban az egyszeres szabályok kiirtása útján lehet elhárítani. Ha nincs egyszeres szabály, akkor bizonyosan nincs ciklicitás.

Kiküszöbölésük algoritmusa a következő lehet.

Emeljük ki a nyelvtanból az egyszeres szabályokat, és a továbbiakban azt a nyelvtant vizsgáljuk, amely csak ezekből az egyszeres szabályokból áll.

Megnézzük, hogy ebben a nyelvtanban, tehát az egyszeres szabályokban hány nemterminális szimbólum van. Ez után alkalmazzuk a felülről lefelé való fésülést olymódon, hogy kiinduló halmazként rendre egy-egy ilyen nemterminálist választunk.

Az egyes fésülések eredményeként kapott záróhalmaz nyilván azokat a nemterminálisokat tartalmazza, amelyek a kiindulásul választott nemterminális szimbólumból az egyszeres szabályok útján elérhetőek.

Az egyszeres szabályok elhagyásakor ennek megfelelően a nyelvtant új szabályokkal kell kiegészíteni. Ezen szabályok baloldala a kiindulásul választott nemterminális szimbólum lesz, jobboldala pedig megegyezik azon szabályok jobboldalával, amelyeknek baloldala a záróhalmazban található valamelyik nemterminális. Így állíthatjuk elő egyetlen lépésben azt a mondatszerű formát, amelyet az egyszeres szabályok esetleges alkalmazásának befejezése után kapunk.

Állításaink nyilvánvalóak, így további magyarázatra nincsen szükség.

Lássunk erre is egy példát. Legyen a nyelvtan a jól ismert aritmetikai kifejezéseket generáló nyelvtan:

$$E \to E{+}T \quad E \to T \quad T \to T{*}F \quad T \to F \quad F \to (E) \quad F \to a$$

Az egyszeres szabályok létét kiemelendő itt most nem éltünk az azonos baloldalak nyújtotta egyszerűsítő jelöléssel.

Nyelvtanunk két egyszeres szabályt tartalmaz:

$$E \to T \qquad T \to F$$

Az egyszeres szabályok három nemterminális szimbólumot tartalmaznak. Ennek megfelelően a felülről való fésülést három ízben, az *E, T* és *F* szimbólumokból kiindulva kell elvégezni. Megállapodás szerint minden szimbólumból saját maga mindig elérhető, ha másképp nem, az egyszeres szabályok nullaszor való alkalmazásával.

A három vizsgálat záróhalmazait úgy különböztetjük meg, hogy indexként a kiinduló nemterminálisokat használjuk.

$$\boldsymbol{T}_E = \{ E, T, F \} \qquad \boldsymbol{T}_T = \{ T, F \} \qquad \boldsymbol{T}_F = \{ F \}$$

Ennek megfelelően az egyszeres szabályok elhagyásával egyidejűen a következő új szabályokat kell bevezetni:

$$E \to T{*}F \qquad E \to (E) \qquad E \to a \qquad T \to (E) \qquad T \to a$$

Így az új, egyszeres szabályokat most már nem tartalmazó nyelvtan:

$$E \to E{+}T \mid T{*}F \mid (E) \mid a$$
$$T \to T{+}F \mid (E) \mid a$$
$$F \to (E) \mid a$$

Ezzel megkaptuk az ismert nyelvtanunkkal egyenértékű, de egyszeres szabályt nem tartalmazó nyelvtant. Őszintén szólva, az egyszeres szabályok kiirtása nem emelte az áttekinthetőséget. Ráadásul ez a nyelvtan „ártatlan" volt, amennyiben itt az egyszeres szabályokban nem volt ciklicitás. Éppen ezért nem kell feltétlenül ragaszkodnunk az egyszeres szabályok eltávolításához.

Az angol irodalom az olyan nyelvtant, amely mentes a felesleges szimbólumoktól, nem, vagy csak az ismert korlátozással tartalmaz ε-szabályt, végül nincsen benne egyszeres szabály, *proper grammar* névvel illeti.

Ennek a szóhasználatnak a magyar nyelven nincsen pontos egyenértékese, talán igazándi vagy tisztességes nyelvtan elnevezés volna megfelelő. Engedtessék meg, hogy javaslatot tegyek a *jólfésült nyelvtan* megnevezés bevezetésére.

Az eddigiekből következik, hogy minden környezetfüggetlen nyelvtannak, bármilyen összegubancolódott legyen is, mindig van jólfésült egyenértékese.

3.3. Nyelvtanok normálalakjai

Az előbbiekben láttuk, hogyan lehet egy nyelvtanból az anomáliás, vagy legalábbis annak tartott struktúrákat kiiktatni. A továbbiakban megkötéseket teszünk az alkalmazható helyettesítési szabályok formája tekintetében. Azokat a nyelvtanokat, amelyek csak ezeket, a bizonyos szempontból egységes formájú szabályokat tartalmazzák a nyelv normálalakjainak, vagy normálformáinak nevezzük.

Az első ilyen normálalakot még *Chomsky* javasolta. Az irodalomban használatos erre a **CNF** (*Chomsky Normal Form*) jelölés.

Ebben a normálalakban csak kétféle alakú helyettesítési szabály engedélyezett:

$$A \to BC \qquad A \to a \tag{3.11.}$$

Az első formájú szabálynak az a lényege, hogy a jobboldalon két nemterminális áll. Természetesen a szabály három nemterminálisa közül bármelyik kettő, sőt akár mindhárom is azonos lehet. Az alábbiakban igazoljuk, hogy minden jólfésült nyelvtannak van **CNF** egyenértékese.

Bizonyításunk szokás szerint konstruktív lesz, vagyis addig alakítgatjuk az eredeti nyelvtant, persze ügyelve arra, hogy az átalakítások során a generált nyelv ne változzék, amíg végül egy *Chomsky* normálalakot nem kapunk. Az eredeti nyelvtannak persze lehetnek olyan szabályai is, amelyek eleve eleget tesznek a (3.11.) követelményeknek. Tekintsünk ezeket a szabályokat véglegesnek.

Vezessünk be a többi, tehát a követelményeket nem teljesítő szabályokban a terminális szimbólumok helyett nemterminálisokat, álnemterminálisokat. Amennyiben például az *a* terminálist akarjuk kiváltani, akkor vezessünk be helyébe egy $\hat{A}$ nemterminálist, és ezt írjuk minden előfordulási helyére.

Ugyanakkor egészítsük ki szabályainkat egy

$$\hat{A} \to a$$

alakú szabállyal, amely szerencsés, de egyáltalán nem véletlen módon eleget tesz a *Chomsky*-féle követelményeknek, tehát véglegesíthető.

Nyilvánvalóan ez az átalakítás nem érinti a generált nyelvet, hiszen az így bevezetett nemterminálist végül – más szabály nem lévén – az eredeti terminálissá kell átírni.

Ezzel a *Chomsky* megkötéseknek eleget nem tevő helyettesítési szabályok alakja csakis a következő lehet

$$C \to C_1 C_2 \ldots C_n \tag{3.12.}$$

Itt **n** szükségképpen nagyobb kettőnél, hiszen az eredeti nyelvtan jólfésült volt, tehát nem tartalmazott egyszeres szabályokat, ha viszont a jobboldalon két nemterminális állt, akkor eleget tett a követelményeknek, és így „véglegesítésre” került.

Bontsuk fel most egy új nemterminális szimbólum bevezetésével a (3.12.) szabályt például a következőképpen:

$$C \to C_1\hat{C}_1 \qquad \hat{C}_1 \to C_2C_3 \ldots C_n \tag{3.13.}$$

A (3.12.) szabályt elhagyva, és helyette a fenti két új levezetési szabályt beiktatva látható, hogy az első rögtön kielégíti a követelményeket, míg a második szabály jobboldalának hossza eggyel rövidebb, mint elődjéé volt.

Az a tény, hogy a módosítás nem érinti a generált nyelvet, nem igényel magyarázatot.

A fenti vagy ezzel egyenértékű felbontással tehát csökkenteni lehet a jobboldalak hosszát. Ezt kellő kitartással folytatva a leghosszabb jobboldalt is kettőre tudjuk redukálni. Ezzel viszont a kijelölt célt teljesítettük is, létrehoztunk egy az eredeti nyelvtannal egyenértékű *Chomsky* normálalakot.

Megjegyzem, hogy az $S \to \varepsilon$ alakú szabály, ahol *S* a mondatszimbólum, mint egyetlen olyan ε-szabály, amely jólfésült nyelvtanokban előfordulhat, szintén *Chomsky* szabálynak minősül. Nélküle ugyanis nem lehetne olyan nyelv nyelvtanát *Chomsky* alakra hozni, amely az üres jelsorozatot tartalmazza.

A fenti konstruktív bizonyítás alapján rögzíthető, hogy minden környezetfüggetlen nyelvtannak van egyenértékű *Chomsky* normálalakja. Ez a normálforma nem unikális, hiszen az átalakításnál leválaszthattuk volna, például az utolsó két nemterminálist, vagyis a szalámit nem az elején, hanem a végén kezdtük volna szeletelni. De ez csak két lehetőség a sok közül.

Példaképpen hozzuk *Chomsky* normálalakra az alábbi nyelvtant:

$$S \to aSb \qquad S \to ab$$

A nyelvtan nyilvánvalóan az

$$a^i b^i \qquad \mathbf{i} > 0$$

nyelvet generálja.

Írjuk át a terminális szimbólumokat nemterminálisokká. Szerencsére itt nem kell jelölési konfliktustól tartani, így a vesszőzés elmaradhat.

$$S \to ASB \qquad S \to AB \qquad A \to a \qquad B \to b$$

Szerencsére a négy új szabály közül három rögtön megfelel a követelményeknek. Egyedül az első helyen álló szabályt kell tovább bontanunk:

$$S \to AC \qquad C \to SB$$

Ezzel a nyelv teljes *Chomsky* nyelvtana:

$$S \to AC \quad C \to SB \qquad S \to AB \quad A \to a \quad B \to b$$

Mielőtt az itt tárgyalt másik normálalak, a *Greibach* normálalak részleteiben elmerülnénk, szólnunk kell néhány szót a balrekurzióról, illetve annak megszüntetéséről.

Már korábban tisztáztuk, mikor mondjuk egy nyelvtanról, hogy rekurzív. Az is kiderült, hogy a rekurzivitás a nyelv és nem a nyelvtan sajátja. Ebből következik, hogy ha egy nyelvet generáló nyelvtan rekurzív, akkor minden, ezt a nyelvet generáló nyelvtan is szükségképpen rekurzív lesz. A rekurzivitást tehát nem lehet megszüntetni, egy rekurzív nyelvtannal csakis egy rekurzív nyelvtan lehet egyenértékű.

Balrekurzívnak mondunk egy nyelvtant akkor, ha van olyan a rekurzivitásért felelős nemterminális szimbólum, amelyből kiinduló levezetés során a szóban forgó nemterminális a legbaloldalibb pozícióban jelenik meg újból.

A baloldali rekurzióra tehát a következő összefüggés írható fel:

$$A \overset{*}{\Rightarrow} A\alpha \tag{3.14.}$$

A balrekurzivitást alkalmas átalakításokkal meg lehet szüntetni. Természetesen a nyelvtan továbbra is rekurzív marad, csak ez a tulajdonság nem a baloldali rekurzió formájában jelentkezik.

Itt rögtön felmerül az a kérdés, miért van erre szükség? Ha egyszer a rekurzió nem küszöbölhető ki, akkor nem mindegy, hogy ez milyen formában, hol jelentkezik?

Messzemenően nem mindegy! Mint látni fogjuk, bizonyos egyszerű szintaktikus elemző módszerek nem alkalmazhatóak, ha a nyelvtan balrekurzív. Így ennek megszüntetése nagyon is lényeges.

Elöljáróban az úgynevezett közvetlen balrekurzió megszüntetését ismertetem. Közvetlen balrekurzióról beszélünk akkor, ha van a nyelvtanban

$$A \to A\alpha \tag{3.15.}$$

alakú szabály, vagyis a balrekurzió már egyetlen helyettesítési lépés után jelentkezik. A (3.15.) alakú szabályok léte szembeszökővé teszi a balrekurzió tényét, hiányuk azonban nem biztosíték arra, hogy nincsen balrekurzió.

Legyen A egy közvetlen balrekurziót okozó nemterminális. Soroljuk az A levezetési szabályait két halmazba aszerint, hogy közvetlen balrekurziót okoznak-e vagy sem.

$$A \to A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_n \tag{3.16.}$$

$$A \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m \tag{3.17.}$$

Itt persze a β jelsorozatok nem kezdődhetnek az A nemterminálissal. Sem **n** sem **m** nem lehet zérus, hiszen akkor vagy az A szimbólum nem okozna közvetlen balrekurziót, vagy az álrekurziónak nevezett badarsággal volna dolgunk.

Vezessük be a reguláris halmazoknál már alkalmazott jelöléseket, és legyen

$$\alpha = \alpha_1 + \alpha_2 + \ldots + \alpha_n \tag{3.18.}$$

$$\beta = \beta_1 + \beta_2 + \ldots + \beta_m \tag{3.19.}$$

Ezzel a (3.16.) és (3.17.) egyenlet a következő alakot ölti:

$$A = A\alpha + \beta \tag{3.20.}$$

Könnyen belátható, hogy ebből a kifejezésből levezethető

$$A \overset{*}{\Rightarrow} \beta\alpha^{*} \tag{3.21.}$$

Az A nemterminális ugyanis csak akkor tűnik el a jobboldal első helyéről, ha az $A \to \beta$ szabályt alkalmazzuk. Előtte viszont tetszőlegesen sokszor

beleértve a nullaszoros lehetőséget is, felhasználhattuk az $A \to A\alpha$ szabályt. Ezt fejezi ki az α^* faktor.

Alakítsuk át nyelvtanunkat a következők szerint:

$$A = \beta + \beta \hat{A} \qquad \hat{A} = \alpha + \alpha\hat{A} \tag{3.22.}$$

A második egyenletből $\hat{A} = \alpha^+$ és ebből már következik, hogy a (3.22.) egyenletek ugyanazt az eredményt adják, mint a (3.20.) egyenlet. A két egyenlet tehát egyenértékű, és mivel a (3.22.) kifejezés nem tartalmaz közvetlen balrekurziót, feladatunkat megoldottuk, a közvetlen balrekurziót sikerült kiküszöbölnünk.

Térjünk most át a környezetfüggetlen nyelvtanok másik nevezetes normálalakjára, a *Greibach*-féle normálalakra. Az irodalom **GNF** néven is emlegeti (*Greibach Normal Form)*. A *Greibach* normálalakban a helyettesítési szabályok engedélyezett alakja:

$$A \to aW \tag{3.23.}$$

ahol W tetszőleges, csak nemterminálisokból álló sorozat. A tetszőleges számban természetesen a zérus is beleértendő, hiszen ellenkező esetben egyetlen mondatot sem lehetne levezetni, mindig maradna a mondatszerű formában nemterminális.

Az $S \to \varepsilon$ alakú ε-szabály ugyanúgy, ahogy a *Chomsky* normálalaknál, itt is engedélyezett *Greibach* szabály.

Vegyük észre, hogy a *Greibach* normálalakban felírt nyelvtanok nem balrekurzívak. Ez magától értetődő, ha meggondoljuk, hogy minden helyettesítési szabály jobboldalának első szimbóluma terminális.

A következőkben a *Greibach* normálalak előállítását megoldva, elvégezzük a balrekurzió kiküszöbölését is. Azt kell elérnünk, hogy csak olyan levezetési szabályok legyenek a nyelvtanban, amelyek terminálissal kezdődnek. Ha ezt megvalósítottuk, akkor már nyert ügyünk van, hiszen a levezetési szabályok belsejében található terminális szimbólumok eltávolításában már a *Chomsky* normálalak szerkesztésekor bizonyos gyakorlatra tettünk szert.

Legyen adott egy jólfésült nyelvtan. Definiáljunk a nyelvtan nemterminális szimbólumai között egy szigorú, de egyébként teljesen tetszőleges rendezést. Ennek szemléltetésére valamennyi nemterminális szimbólumot egyetlen betűvel és egy indexszel jelöljük, ahol az index sorszáma határozza meg a rendezést.

A rendezés tetszőleges abban az értelemben, hogy semmiféle feltételezést nem teszünk majd a sorrend mikéntjére, így például azt sem kötjük ki, hogy a mondatszimbólum hányadik – első vagy utolsó – legyen ebben a sorban.

A sorrend tehát elvben tetszőleges, a gyakorlatban persze lehet, hogy egy ügyetlenül megválasztott sorrend az átalakítási munkát megnöveli, ez azonban a megvalósíthatóságot nem érinti.

Első lépésben azt kívánjuk elérni, hogy az A_i nemterminálishoz tartozó valamennyi levezetési szabály vagy terminálissal kezdődjék, vagy első karaktere magasabb sorszámú nemterminális legyen.

Formálisan:

$$A_i \to aX \quad \text{vagy} \quad A_i \to A_j Y \qquad \mathbf{j > i} \tag{3.24.}$$

ahol X és Y tetszőleges szimbólumokat tartalmazhat.

Ezt a műveletet a legalacsonyabb indexű nemterminális szimbólumnál kezdjük el, és innen haladunk felfelé a nagyobb indexek felé.

Az A_1 nemterminális esetében, minthogy ennél alacsonyabb sorszámú nemterminális nincsen, csak az okozhat problémát, ha a jobboldal első szimbóluma szintén A_1. Ez viszont a közvetlen balrekurzió esete, amelynek kiküszöbölésére már van algoritmusunk. Ezt alkalmazva elérhetjük, hogy a (3.24.) feltételei erre a nemterminálisra teljesüljenek.

Ha most áttérünk az A_2 nemterminális szimbólumra, akkor először azokat a levezetési szabályokat keressük meg, amelyek jobboldala az A_1 nemterminálissal kezdődik. Ha vannak ilyen szabályok, akkor ezekben az élen álló A_1 szimbólumot levezetési szabályainak jobboldalával kell helyettesíteni, mégpedig minden lehetséges kombinációban.

Így ha az A_2 levezetési szabályai között **k** olyan van, amelynek jobboldala az A_1 szimbólummal kezdődik, és az A_1 szimbólumhoz tartozó levezetési szabályok száma **j,** akkor az eredeti **k** szabály helyébe **k*j** szabályt kell írni.

Ezzel garantáljuk, hogy A_2 szabályai között nem lesz olyan, amelyiknek a jobboldala az A_1 szimbólummal, vagyis a baloldalinál alacsonyabb sorszámú nemterminálissal kezdődne. Ennek következtében itt most csakis azok a szabályok okozhatnak problémát, amelyeknek első jobboldali szimbóluma A_2, függetlenül attól, hogy az ilyen szabályok eredetileg is benne voltak a nyelvtanban, vagy az átalakítások során kerültek bele. Ez viszont megint a közvetlen balrekurzió esete, így az annak megszüntetésére szolgáló algoritmusunkat kell mozgósítanunk. Így elérjük, hogy csak vagy terminálissal, vagy nagyobb sorszámú nemterminálissal kezdődő szabályaink lesznek, a (3.24.) követelmények teljesülnek.

Általánosságban, ha fenti célunkat az A_i nemterminális szimbólumnál kívánjuk elérni, akkor feltételezhetjük, hogy ezt az algoritmust az **i–1** indexig bezáróan már sikerrel alkalmaztuk.

Először itt is az A_1 kezdetű levezetési szabályokat – ha vannak ilyenek – szüntetjük meg helyettesítéssel. Ezzel a minimális kezdeti sorszámot eggyel feljebb vittük. Ekkor jön a következő sorszámú nemterminális, ismét csak helyettesítés, és ezt alkalmazzuk mindaddig, amíg az **i** indexhez nem érünk. Itt, ha szükséges, ismét a közvetlen balrekurzió elhárítására bevezetett módszert vetjük be.

Már a leírásból is látszik, hogy ez az algoritmus, bár biztosan célhoz ér, meglehetősen hosszadalmas.

Vegyük észre, ha átrágtuk magunkat az összes indexen, akkor az utolsó nemterminális szimbólum esetében a *Greibach* normálalak előírásainak megfelelő levezetési szabályokat kapunk, amennyiben valamennyi szabály jobboldala terminális szimbólummal kezdődik. Ez abból következik, hogy az utolsó nemterminális esetében, az utolsónál nem lévén magasabb sorszámú nemterminális, csak az a lehetőség marad, hogy a levezetési szabályok terminálissal kezdődnek. Itt persze feltételeztük, hogy az esetleges belső terminális szimbólumokat a szokott módon „nemterminálosítottuk".

Ezekkel az átalakításokkal elértük, hogy a nyelvtan nem lehet balrekurzív. Minden szabály ugyanis vagy terminálissal, vagy magasabb sorszámú nemterminálissal kezdődik, így nincs lehetőség arra, hogy egy nemterminális a jobboldal legelső pozíciójába visszatérjen, hiszen nem lehet önmagánál magasabb indexe.

Most viszont visszafordulhatunk, és az átalakítás második lépéseként fáradságos utunkat nem kevésbé fáradságos módon lefelé is végigjárjuk.

Az utolsó előtti nemterminálisról ugyanis tudjuk, hogy ha vannak is nemterminálissal kezdődő levezetési szabályai, akkor azok csakis az utolsó nemterminálissal kezdődhetnek. Ezen szabályoknál az utolsó nemterminális helyébe levezetési szabályainak jobboldalát helyettesítve elérhetjük, hogy az utolsó előtti nemterminális valamennyi szabályának jobboldala terminálissal kezdődjék. Ezek szerint ez a nemterminális is ki fogja elégíteni a *Greibach*-féle szabályokat.

Ezt az algoritmust a sorszámok csökkenő sorrendjében folytatva, mire az első nemterminálist is túlhaladtuk, az egész nyelvtanban nem lesz olyan levezetési szabály, amely ne terminális szimbólummal kezdődne.

Minthogy a fent ismertetett algoritmus bármely jólfésült nyelvtanra alkalmazható, igazoltuk, hogy minden környezetfüggetlen nyelvnek létezik *Greibach* normálalakja.

Lássunk erre is egy példát. Legyen a nyelvtan:

$$A \to BC \quad B \to CA \mid b \qquad C \to AB \mid a$$

Válasszuk a három nemterminális sorrendjének az $A - B - C$ sorrendet. Először azt az állapotot kell előállítanunk, amikor az egyes nemterminálisokhoz tartozó levezetési szabályok csak vagy terminálissal, vagy sorrendben későbbi nemterminálissal kezdődnek.

Szerencsére az A és B nemterminális szimbólumokra ez eleve teljesül. A C levezetési szabályainál a kezdő A szimbólumot helyettesítéssel el kell távolítani. Sajnos erre az új szabály jobboldalának elején megjelenik a B nemterminális, így újabb helyettesítésre van szükség.

$$C \to AB \mid a \quad \Rightarrow \quad BCB \mid a \quad \Rightarrow \quad CACB \mid bCB \mid a$$

Ezzel sajnos nem vagyunk készen, hiszen a C nemterminális levezetési szabályaiban önmaga sem szerepelhet a jobboldal első helyén. Most a közvetlen balrekurzivitást megszüntető algoritmust kell alkalmaznunk.

$$C \to bCB \mid a \mid bCB\hat{C} \mid a\hat{C}$$
$$\hat{C} \to ABC \mid ACB\hat{C}$$

Ezzel megkaptuk azt az egyenértékes nyelvtant, amelynek levezetési szabályai vagy terminálissal, vagy magasabb sorszámú nemterminálissal kezdődnek:

$$A \to BC$$
$$B \to CA \mid b$$
$$C \to bCB \mid a \mid bCB\hat{C} \mid a\hat{C}$$
$$\hat{C} \to ACB \mid ACB\hat{C}$$

Ami a kalapos szimbólumokat illeti, azokat úgy kezelhetjük, mintha sorrendben megelőznék az összes eredeti szimbólumot. Példánkban csak egy ilyen szimbólum van $\hat{C}$. Ha több ilyen volna egymás közötti sorrendjük közömbös.

Most visszafelé kell az utat bejárnunk. A sorrendben legutolsó nemterminális esetünkben C volt. Ennek levezetési szabályai már teljesítik a *Greibach* feltételeket, amennyiben terminális szimbólummal kezdődnek. Azokba a levezetési szabályokba, ahol C a kezdőszimbólum a jobboldalakat behelyettesítjük. Ezt kell tennünk a B nemterminális egyik levezetési szabályánál. Ezzel a B szimbólum valamennyi levezetési szabálya is terminálissal fog kezdődni.

Ezután a B szimbólumot kell az A egyetlen szabályában átírnunk, végül az A szimbólumra kapott eredményt a $\hat{C}$ szabályaiba beírnunk.

Mindezek után a nyelvtan a következő lesz. A szabályokat célszerűen a származtatás sorrendjében adom meg.

$$C \to bCB \mid a \mid bCB\hat{C} \mid a\hat{C}$$
$$B \to bCBA \mid aA \mid bCB\hat{C}A \mid a\hat{C}A \mid b$$
$$A \to bCBAC \mid aAC \mid bCB\hat{C}AC \mid a\hat{C}AC \mid bC$$
$$\hat{C} \to bCBACCB \mid aACCB \mid bCB\hat{C}ACCB \mid a\hat{C}ACCB \mid bCCB \mid$$
$$bCBACCB\hat{C} \mid aACCB\hat{C} \mid bCB\hat{C}ACCB\hat{C} \mid a\hat{C}ACCB\hat{C} \mid bCCB\hat{C}$$

Itt említem meg, hogy más rendezés például a C–A–B sorrend esetén talán még szövevényesebb, de mindenesetre más nyelvtant kaptunk volna. Természetesen bárhogyan is származtatjuk a *Greibach* normálalakot a kapott nyelvtanok egyenértékűek.

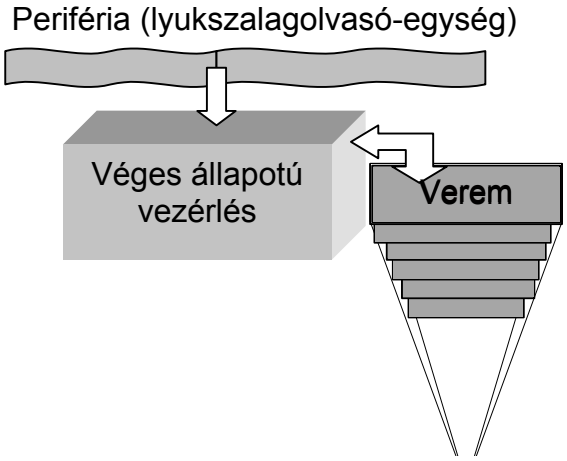
Mint látható az eredeti egyszerűnek tűnő nyelvtanból készített *Greibach* normálalak ijesztő bonyolultságú. Felmerülhet ezzel kapcsolatban a kérdés, van-e a normálalakoknak az esetleges elvi jelentőségen túlmenően gyakorlati fontossága.

Gyakran az áttekinthető nyelvtan alapján nehéz elemzőt szerkeszteni. A normálalakok erre sokkal jobb kiindulást adhatnak. Minthogy az elemzést a fordítóprogramot futtató számítógép végzi, ez esetben az elemezhetőség szempontja nagyobb prioritású, mint az olvashatóságé.

3.4. Veremautomaták

Mint említettük, minden nyelvosztályhoz tartozik egy automataosztály, amely alkalmas a nyelvosztálynak megfelelő nyelvtan által generált nyelvek felismerésére. A környezetfüggetlen nyelvek esetében ez az automataosztály a veremautomaták osztálya.

A veremautomaták „szerkezeti” felépítését a 3.5. ábra tünteti fel. Ha a már jól ismert véges automatát egy **LIFO** – *Last In First Out* – diszciplínájú memóriával, veremmel kiegészítjük, a veremautomatát kapjuk.



3.5. ábra

Mint ismeretes a verem memória **LIFO** szervezésű, vagyis mindig csak az utoljára betett objektum hozzáférhető. Ez pontosan megfelel a verem mindennapi szóhasználatának, amennyiben például egy krumplisveremből is csak az utoljára bevermelt krumplit lehet kivenni.

A veremautomata veremmemóriája pontosan ezt a diszciplínát követi. Betenni vagy írni – ezek ez esetben szinonimák – csakis a verem tetejére lehet, és ugyanígy csak a verem legfelső elemét lehet elolvasni, kivenni. Definíciószerűen az olvasás egyben a veremből való kivételt is jelenti.

Itt sajnos néhány szót kell szólni egy elég szerencsétlen terminológiai balfogásról. Azt az automatát ugyanis, amiről beszélünk, az angol nyelvű szakirodalom *push-down automaton* néven illeti. A *push-down „*nyomd le” kitétel arra utal, hogy az új információt mindig a régi tetejére írjuk, és ezzel a memória régi tartalmát mintegy lenyomjuk.

A **LIFO** szervezésű, a magyar irodalomban veremmemóriának nevezett memória angol neve ugyanakkor *stack memory*. A magyar számítástechnikai tolvajnyelvben is gyakran említünk a veremmemória helyett stack-memóriát.

Az a tény, hogy az angol nyelvű szakirodalom a veremmemóriát használó automata elnevezésére nem a verem szó angol megfelelőjét használja, bocsánatos bűn volna. A problémát az okozza, hogy ezzel egyidejűen egy nem verem, vagyis nem *stack* szervezésű memóriát tartalmazó automatának a *stack automaton* nevet adták. Az automataelmélet szakemberei, akik nem ismerve vagy tudomásul sem véve a számítástechnika szóhasználatát ezzel komoly konfúziót okoztak.

A *stack automaton* nem szerepel tanulnivalóink között, így itt csak tájékoztatásul jegyzem meg, hogy ennél az automatánál ugyanúgy csak legfelülre lehet beírni, és csak a legfelső elemet lehet kivenni, mint a veremmemóriánál – ennyiben valóban stack szervezésű a memória – de lehet kivétel nélkül is olvasni belőle, sőt ezt a műveletet nem csak a verem tetején, hanem bárhol bent a memóriában meg lehet tenni. Így olvasásra a memória egésze hozzáférhető. Ezek szerint a *push-down automaton* a *stack automaton* azon különleges esete, amikor a kivétel nélküli olvasás nem megengedett.

A magyar irodalomban az elnevezés tekintetében bizonyos habozás tapasztalható. Vannak, akik az automata funkcióját tekintik mértékadónak, és a veremmemóriával ellátott automatát veremautomatának hívják. Mások az angol eredetiből kiindulva a veremautomata elnevezést az angolban *stack automaton* elnevezésű automatának tartják fenn.

Jómagam szintén bizonyos hezitálás után a memória szervezését tekintettem perdöntőnek. Tettem ezt már csak azért is, mert az egyébként használatos *push-down automata* elnevezést nem tartottam elég magyarosnak, másrészt az ily módon kisemmizett stack automata jelentősége sokkal kisebb. A ma használatos terminológiát szemlélve úgy tűnik nem volt rossz döntés.

A terminológiáról szóló ezen hosszabb eszmefuttatást nem csak a magam mentségére írtam, hanem tájékoztatásul arra az esetre, ha valaki más terminológiát használó magyar szakirodalmat olvas.

A veremautomata formális leírására egy hetes szolgál.

$$\mathbf{P}\ (\ Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ Z_0,\ F\ ) \qquad (3.25.)$$

ahol
- Q az automata állapotainak véges halmaza
- Σ az elemzendő nyelv alfabetája
- Γ a verem alfabetája, vagyis azon szimbólumok véges halmaza, amelyek a veremben szerepelhetnek
- δ a mozgási szabályok véges halmaza, amelyről még részletesen szólunk
- q_0 az automata kezdőállapota
- Z_0 a memória kezdeti tartalma
- F az elfogadó állapotok halmaza

A **P** jelölés nem véletlen, ezt az automata a *push-down* elnevezéstől örökölte.

Valamennyi említett halmaz véges, hiszen egyébként nem lehetne véges módon a nyelvet leírni.

A definícióból következik, hogy :

$$q_0 \in Q \qquad Z_0 \in \Gamma \qquad F \subset Q \tag{3.26.}$$

A veremautomata csak akkor mozgásképes, ha verme nem üres. Ezért szükséges már induláskor a vermet egy karakterrel feltölteni, ez Z_0.

A veremautomata mozgását az automata állapota, az olvasott karakter és a verem tetején található szimbólum határozza meg. A mozgás hatására az automata új, esetleg az eredetivel megegyező állapotot vesz fel, és a verem tetejére egy véges jelsorozatot ír be.

Úgy tekintjük, hogy mozgás következtében az olvasó egy szimbólummal tovább lép, tehát az olvasófej alatt már a következő szimbólum található, ugyanakkor a verem tetején lévő, és a mozgás során figyelembe vett szimbólum kivételre kerül, megsemmisül. Amennyiben az a szándékunk, hogy a verem tetején lévő szimbólum a veremben maradjon, akkor ezt azzal a fogással érjük el, hogy a szimbólumot visszaírjuk a verembe.

A mozgási szabályok száma véges, így a mozgás során a verembe beírt jelsorozatok hossza korlátos. Egyes mozgásoknál a verembe beírt jelsorozat hossza zérus is lehet, ilyenkor azt mondjuk, hogy az ε karaktert írtuk a verembe. Ebben az esetben a mozgás következtében a verem tartalma csökken.

A veremautomatának az előbb ismertetett úgynevezett valódi mozgásain kívül van egy másik mozgási lehetősége is. Ennél az ε-mozgásnak nevezett mozgásnál a mozgást csak az automata állapota és a veremtető határozza meg.

Az olvasófej alatti karakter itt nem játszik szerepet, el sem olvassuk, és így természetesen a szalag sem továbbítódik. Sőt, ε-mozgás akkor is lehetséges, ha az olvasófej alatt már nincs is karakter.

Az ε-mozgást jelképesen az ε karakter beolvasásával szokás jelölni.

Ezzel a mozgási szabályok a következő leképezést definiálják:

$$Q \times (\Sigma \cup \varepsilon) \times \Gamma \Rightarrow Q \times \Gamma^k \tag{3.27.}$$

ahol – **k** a verembe beírt jelsorozatok hosszának korlátja. Úgy képzelhetjük el, hogy az ε üres jelsorozat eleme a Γ halmaznak, és így a Γ^k jelölés a legfeljebb **k** hosszúságú jelsorozatok halmazát jelöli.

A $(\Sigma \cup \varepsilon)$ jelölés arra utal, hogy a valódi mozgások esetében a Σ alfabeta egy elemét olvassuk, ε-mozgások esetében az „olvasott” karakter ε, tehát valójában nem történik olvasás.

Még a részletek előtt néhány általános megjegyzés.

A jelsorozat elfogadásának feltételei hasonlóak a véges automaták esetében alkalmazotthoz. Egy jelsorozatot akkor fogadtunk el, ha az automata a jelsorozatot végigolvasta, és ezt követően elfogadó állapotba került. A véges automata az utolsó karakter elolvasása után megállt, és aszerint, hogy elfogadó vagy visszautasító állapotban állt meg, fogadtuk el, vagy utasítottuk el a jelsorozatot.

A veremautomatáknál a helyzet nem ilyen egyszerű. Az elemzendő szöveg elolvasása után ugyanis egyáltalában nem szükségszerű, hogy az automata megálljon. Persze csakis ε-mozgás jöhet szóba, hiszen a valódi mozgáshoz már nincsen karakter az olvasófej alatt.

Felhívnám a figyelmet az elfogadás kritériumának óvatos megfogalmazására. Akkor fogad el, ha az elolvasás után elfogadó állapotba kerül. Teljességgel megengedett tehát, hogy az automata az elolvasás után ε-mozgásokkal kísérelje meg az elfogadó állapot elérését.

Az elfogadásnak nem feltétele a mozgásképtelenség. Olyan eset is elképzelhető, amikor az ε-mozgásokban ciklus van, és így a mozgás tetszőlegesen hosszú ideig folytatódhat. Ennek ellenére, ha az elfogadó állapotot kereső ε-mozgások ciklusában van elfogadó állapot, akkor az automata érvényesen elfogadta a jelsorozatot.

Egy fontos észrevétel az automata memóriájával kapcsolatosan. A memória véges, de nem korlátos hosszúságú. Ezzel a fogalommal nem először találkozunk.

Ebben a kontextusban ez a következőket jelenti. Egy adott jelsorozat elemzésekor az automata memóriájának legnagyobb kiterjedése valamilyen véges, az elemzett jelsorozat és az automata által meghatározott érték. Általános korlátot azonban nem adhatunk meg. Bármekkora lenne is ez a korlát, mindig találhatnánk olyan jelsorozatot, amelynek elemzésére a korlátozott memóriaméret nem elegendő.

Ezt az állítást intuitíve igen könnyen beláthatjuk. Amennyiben volna a memória hosszára ilyen korlát, akkor ez azt jelentené, hogy csak véges sok különböző memóriatartalom lenne lehetséges. Vegyük figyelembe ugyanakkor, hogy az automata állapotainak száma is véges.

Ebből viszont az következik, hogy az automatát teljesen leíró, az automata állapotának és memória tartalmának direkt szorzatából álló halmaz is véges. Ha viszont az automata teljes jellemzésére egy véges halmaz elegendő, akkor ez az automata egyenértékű egy véges automatával.

Ugyanakkor, mint látni fogjuk, a veremautomata ereje nagyobb a véges automatáénál, így a memória korlátosságára vonatkozó feltevésünk szükségképpen helytelen.

Az irodalom a memóriának ezt a tulajdonságát, véges de nem korlátos voltát azzal a szóhasználattal is jellemzi, hogy a memória potenciálisan végtelen hosszúságú.

Mielőtt egy példa kapcsán közelebbi ismeretséget kötnénk a veremautomatával, állapodjunk meg néhány konvencióban.

Az automata leírására elegendő a mozgási szabályok és az elfogadó állapotok halmazának megadása. Itt azzal a megállapodással élünk, hogy legelsőnek olyan szabályt adunk meg, amelyben az állapot a kezdőállapot, és a verem tetején az induló veremszimbólum van. Ilyen szabálynak kell lennie, hiszen különben az automata el sem tudja kezdeni az elemzést.

A mozgási szabályok leírásánál a következő sorrendet használjuk: automata állapot, olvasott szimbólum és a verem teteje, illetve új állapot és a verembe beírt jelsorozat. Amennyiben ε-mozgásról van szó az olvasott karakter helyén ε áll. Amennyiben egynél több szimbólumot írunk be a verembe, akkor a sorozat legbaloldalibb eleme kerül a verem legeslegtetejére.

Lássunk ezek után egy veremautomata-leírást:

$\delta(q_0, a, Z_0) = (q_1, aZ_0)$ $\quad$ $\delta(q_0, b, Z_0) = (q_1, bZ_0)$

$\delta(q_1, a, a) = (q_1, aa)$ $\quad$ $\delta(q_1, b, a) = (q_1, ba)$

$\delta(q_1, a, b) = (q_1, ab)$ $\quad$ $\delta(q_1, b, b) = (q_1, bb)$

$\delta(q_1, c, a) = (q_2, a)$ $\quad$ $\delta(q_1, c, b) = (q_2, b)$

$\delta(q_2, a, a) = (q_2, \varepsilon)$ $\quad$ $\delta(q_2, b, b) = (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, Z_0) = (q_3, Z_0)$

$F = \{ q_3 \}$

Ez a veremautomata a wcw^{-1} jelsorozatokat fogadja el, ahol w tetszőleges az a és b karakterekből álló nem üres jelsorozat. A **-1** hatványkitevő a sorozat inverzét jelenti, ami az eredeti jelsorozat fordított sorrendben. Az angol irodalomban szokásos még az inverzre az ^{R} kitevő (*reverse*) használata is.

Az elemzés alapgondolata a következő. Az automata veremmemóriájában elraktározza a sorozat első felét, majd a mondat közepétől kezdve, amit a c karakter beolvasása jelez, ellenőrzi, hogy a mondat második fele megegyezik-e az első fél inverzével. A verem **LIFO** diszciplínája miatt „szerencsére” a mondat első felének szimbólumai fordított sorrendben kerülnek a verem tetejére.

Sajnos a veremautomatának nincs olyan szép és egyszerű ábrázolási módja, mint volt a véges automatának, így az automata működésének nyomon követése, áttekintése nem olyan egyszerű.

Az automata működésének pillanatnyi állapotát, mint említettük, az automata konfigurációja jellemzi. A veremautomata esetében az automata további viselkedésének megállapításához a következő információk szükségesek:

a jelsorozat még be nem olvasott része,
az automata állapota,
a verem tartalma.

$$\boldsymbol{K} = (w, q, \chi) \tag{3.28.}$$

ahol
– w a még el nem olvasott jelsorozat,
– q az automata állapota
– χ a verem tartalma

Itt tehát az automata konfigurációja ez a hármas, amelyeket a fent leírt sorrendben adunk meg, azzal a konvencióval, hogy a még elolvasásra váró jelsorozat legbaloldalibb eleme van az olvasófej alatt, és a verem tartalmának legbaloldalibb eleme a verem teteje.

A konfiguráció segítségével formálisan is felírhatjuk, mi lesz az automata nyelve:

$$\boldsymbol{L}\,(\mathbf{P}) = \{\, w \mid (w, q_0, Z_0) \overset{*}{\mapsto} (\varepsilon, p, \chi) \wedge p \in F \,\} \qquad (3.29.)$$

Az összefüggés szerint azok a w jelsorozatok lesznek az automata nyelvének mondatai, amelyek a kezdőállapottal és a verem kezdő tartalmával olyan konfigurációt alkotnak, amelyből tetszőleges számú lépésben elérhető egy olyan konfiguráció, ahol a még elolvasásra váró jelsorozat az üres jelsorozat, és az állapot az elfogadó állapotok halmazának eleme. A verem tartalma érdektelen.

A veremautomata működését tehát konfigurációk sorozatán keresztül követhetjük. Tegyük most ezt az előbbi automatával, amikor az

aabacabaa

mondatot elemzi.

Az induló konfiguráció: (*aabacabaa*, q_0, Z_0)

A továbbiakban kommentár nélkül adjuk meg a konfigurációk sorozatát.

$\{ aabacabaa, q_0, Z_0 \} \mapsto \{ abacabaa, q_1, aZ_0 \} \mapsto \{ bacabaa, q_1, aaZ_0 \} \mapsto$
$\mapsto \{ acabaa, q_1, baaZ_0 \} \mapsto \{ cabaa, q_1, abaaZ_0 \} \mapsto \{ abaa, q_2, abaaZ_0 \} \mapsto$
$\mapsto \{ baa, q_2, baaZ_0 \} \mapsto \{ aa, q_2, aaZ_0 \} \mapsto \{ a, q_2, aZ_0 \} \mapsto \{ \varepsilon, q_2, Z_0 \} \mapsto$
$\mapsto \{ \varepsilon, q_3, Z_0 \}$

Az automata a jelsorozatot elfogadta, hiszen a bemenet valamennyi szimbólumát feldolgozta, és azt követően elfogadó állapotba került. Így a fenti jelsorozat az automata nyelvének egy mondata.

Az elemzés adott esetben igen egyszerűnek mutatkozik, hiszen az automatának minden szituációjában csak egy mozgási lehetősége kínálkozik, az automata determinisztikus.

Ez nem mindig van így. Lássunk most egy példát a nemdeterminisztikus veremautomatákra.

Legyenek az automata mozgási szabályai a következők:

$\delta(q_0, a, Z_0) = (q_1, aZ_0)$ $\qquad$ $\delta(q_0, b, Z_0) = (q_1, bZ_0)$

$\delta(q_1, a, a) = (q_1, aa) \mid (q_2, \varepsilon)$ $\qquad$ $\delta(q_1, b, a) = (q_1, ba)$

$\delta(q_1, a, b) = (q_1, ab)$ $\qquad$ $\delta(q_1, b, b) = (q_1, bb) \mid (q_2, \varepsilon)$

$\delta(q_2, a, a) = (q_2, \varepsilon)$ $\qquad$ $\delta(q_2, b, b) = (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, Z_0) = (q_3, Z_0)$

$$F = \{ q_3 \}$$

Ennek az automatának két olyan szabálya van, ahol egyetlen baloldalhoz két jobboldal tartozik. Itt is éltünk azzal az egyszerűsítő jelölési konvencióval, hogy az azonos baloldalhoz tartozó jobboldalakat a *vagy* jelentésű $\mid$ szimbólummal

választottuk el. Ez annyit jelent, hogy ebben a szituációban két mozgásra van lehetőség, és közülük az automata bármelyiket választhatja. Az automata nemdeterminisztikus.

Egy veremautomata hasonlóan a véges automatákhoz, akkor deterministikus, ha minden szituációhoz legfeljebb egy mozgás tartozik. Más szavakkal ez annyit jelent, hogy minden automata állapot, olvasott szimbólum és veremtető hármashoz legfeljebb egy mozgás rendelhető. Ezen túlmenően, gondolva az ε-mozgásokra is, kikötjük, hogy ha valamely automata állapot és veremtető pár esetében lehetséges ε-mozgás, akkor csakis ez a mozgás legyen lehetséges, és a fenti párt bármilyen olvasott szimbólummal egészítsük is ki, valódi mozgás nem engedhető meg.

Így az egyértelműség nem csak a valódi és ε-mozgásokon belül igaz, hanem a két mozgásfajta között is fennáll.

A két példának választott automata, a deterministikus és nemdeterminisztikus, nagyon hasonlít egymásra. Valóban ez utóbbi nyelve is emlékeztet a korábban tárgyalt deterministikus automata nyelvére. Ez ugyanis:

$$ww^{-1}$$

ahol *w* az *a* és *b* szimbólumokból álló tetszőleges jelsorozat.

A két nyelv között „mindössze” az a különbség, hogy míg az első nyelv szájbarágósan közli, hol van a mondat közepe, addig a második nyelv esetében úgy kell fáradságos munkával megtalálnunk, hol végződik a mondat első fele, és kezdődik annak inverze. Valahányszor két azonos szimbólum követi egymást, mindig felmerül a gyanú, nem éppen a mondat közepén állunk-e.

Ennek a dilemmának – közép vagy nem közép – felel meg a kétféle mozgási lehetőség.

Elemezzük most az

$$abbaabba$$

mondatot, vagyis írjuk le a konfigurációk sorozatát:

$$\{\, abbaabba,\, q_0,\, Z_0 \,\} \mapsto \{\, bbaabba,\, q_1,\, aZ_0 \,\} \mapsto \{\, baabba,\, q_1,\, baZ_0 \,\} \mapsto ?$$

Itt most egy válaszúthoz érkeztünk. Legyünk optimisták, és válasszuk azt az alternatívát, hogy elértük már a mondat közepét.

$$\{\, aabba,\, q_2,\, aZ_0 \,\} \mapsto \{\, abba,\, q_2,\, Z_0 \,\} \mapsto ?$$

Ez a levezetés befuccsolt, sajnos ilyen módon nem sikerült a mondatot levezetnünk. Ennek két oka lehet. Vagy a jelsorozat valójában nem mondata a nyelvnek, vagy a válaszúton rossz irányban indultunk el. Térjünk tehát vissza arra a helyre, ahol a dilemma felmerült, és most azzal feltételezéssel éljünk, hogy még nem értük el a mondat közepét. Folytassuk ennek megfelelően az elemzést:

$$\mapsto \{\, aabba,\, q_1,\, bbaZ_0 \,\} \mapsto \{\, abba,\, q_1,\, abbaZ_0 \,\} \mapsto ?$$

Ismét válaszút előtt állunk. Legyünk konzekvensek és ismét reménykedjünk abban, hogy már elértük a mondat közepét. Az ennek megfelelő folytatás:

$\mapsto \{bba, q_2, bbaZ_0\} \mapsto \{ ba, q_2, baZ_0 \} \mapsto \{ a, q_2, aZ_0 \} \mapsto$

$\mapsto \{ \varepsilon, q_2, Z_0 \} \mapsto \{ \varepsilon, q_3, Z_0 \}$

Az automata most elfogadta a jelsorozatot, tehát az az automata nyelvének egy mondata. Emlékeztetek arra, hogy veremautomaták esetében is akkor tekintjük a jelsorozatot elfogadottnak, ha létezik olyan mozgássorozat, amely mellett az automata a szóban forgó jelsorozatot elfogadja. Így nemdeterminisztikus automaták esetében az összes lehetséges mozgássorozatot végig kell tanulmányoznunk.

Az a korábbi állításunk, hogy **CF** nyelvet nem lehet korlátos memóriával elemezni, adott esetben triviális, éspedig mindkét automatát tekintve az. Bármilyen nagyra választjuk is a memóriát, ha olyan mondatot kívánunk elemezni, amelynek fele hosszabb, mint a memória hossza, akkor memória mérete nem teszi lehetővé az elemzést.

Persze felmerül a kérdés, az adott nyelvet nem lehet-e valamilyen más, de korlátos automatával elfogadni. Ilyen automatát nem lehet szerkeszteni. Ahhoz, hogy a mondat két felét összevessük, nyilván memorizálni kell a mondat első felét. Márpedig egy korlátos memóriában nem lehet tetszőleges mennyiségű információt tárolni.

Mint azt a véges automatáknál említettük, egy automataosztály definiálására több hasonló „szerkezetű” automata közül bármelyiket választhatjuk.

A veremautomatákra fentebb megadott specifikáció sem az egyetlen lehetőség az ilyen automaták meghatározására. A veremautomata elfogadási feltételét köthetjük ugyanis a verem üres voltához. Az ilyen üres veremmel elfogadó automata állapotai között nem disztingválunk, hiszen az elfogadás nem az automata állapotától, hanem a verem üres voltától függ. Persze az üres veremmel elfogadó automatának is végig kell olvasnia a jelsorozatot.

Ennek megfelelően az üres veremmel elfogadó veremautomatát egy hatos írja le, a korábbi automata leírásból most ugyanis kiesik az elfogadó állapotok halmaza.

$$\mathbf{P}_e (Q, \Sigma, \Gamma, \delta, q_0, Z_0) \qquad (3.30.)$$

Az $_e$ index itt az angol *empty* – üres – szó rövidítése.

Az elfogadóállapotok hiánya még azzal az előnnyel is jár, hogy a már ismert konvenciókat alkalmazva, pusztán a mozgási szabályok megadása teljesen specifikálja az automatát.

A két automataosztály, az állapottal illetve üres veremmel elfogadó automaták osztálya egyenértékű. Bármely egyik osztályba tartozó automatához szerkeszthető ugyanis egy másik osztálybeli, vele ekvivalens, vagyis ugyanazt a nyelvet elfogadó automata.

Legyen adott egy állapottal elfogadó veremautomata. Vezessünk be egy új q_e veremürítő állapotot, és egészítsük ki az eredeti automata szabályait két szabályhalmazzal:

$$\delta(q_i, \varepsilon, X) = (q_e, X) \qquad (3.31.)$$
$$\forall\, q_i \in F \text{ és } X \in \Gamma$$

amely mozgási szabályt valamennyi elfogadó állapotra és valamennyi veremszimbólumra felírunk, míg az alábbi szabály:

$$\delta(q_e, \varepsilon, X) = (q_e, \varepsilon) \qquad (3.32.)$$

valamennyi veremszimbólumra érvényes lesz.

Nyilvánvaló, hogy az eredeti automata bármely elfogadó állapotában át tudunk térni a (3.31.) szabállyal a veremürítő állapotra. Ezután a (3.32.) szabályok segítségével szép sorjában ki tudjuk üríteni a vermet.

Itt felmerülhet az a veszély, hogy az eredeti automata kiüríti a vermét. Ez az új automata esetében elfogadást jelent. Ha ez az eredeti automata visszautasító állapotában következik be, akkor az új automata olyan jelsorozatot is elfogad, amelyet az eredeti visszautasít.

Ezen úgy segíthetünk, hogy egy új q_0' kezdőállapotot és egy új $\check{Z}_0$ veremszimbólumot vezetünk be. A q_0' állapot egyetlen mozgási szabályával

$$\delta(q_0', \varepsilon, Z_0) = (q_0, Z_0 \check{Z}_0) \qquad (3.33.)$$

kibéleljük a vermet a $\check{Z}_0$ szimbólummal.

Ez a szimbólum akkor is a veremben marad, ha az eredeti automata a vermet kiüríti. Ha ez elfogadó állapotban következik be, akkor a (3.31.) szabállyal áttérhetünk a q_e veremürítő állapotba, és a $\check{Z}_0$ szimbólumot eltávolítjuk. Ha viszont visszautasító állapotban vagyunk, további lépésekre nincs lehetőség, a $\check{Z}_0$ szimbólum a veremben marad, így a jelsorozatot az új automata sem fogadja el.

Könnyű belátni, hogy az új, üres veremmel elfogadó automata azokat és csakis azokat a jelsorozatokat fogadja el, amelyeket az eredeti automata is elfogadott.

Az üres veremmel elfogadó automatából állapottal elfogadó automatát a következőképpen készíthetünk.

Vezessünk be most is, egy új $\check{Z}_0$ verembélelő szimbólumot és egy q_0' új kezdőállapotot, valamint egy q_a egyetlen elfogadó állapotot.

Az elemzés itt is a (3.33.) szabállyal kezdődik, amely az eredeti Z_0 induló veremszimbólum alá dugja a $\check{Z}_0$' újonnan bevezetett szimbólumot. Ezután az eredeti, üres veremmel elfogadó automata mozgási szabályai érvényesülnek. Amennyiben az eredeti automata kiüríti a vermet, akkor a módosított automata vermében csak az aládugott $\check{Z}_0$ szimbólum marad. Ekkor térünk át egy új szabály segítségével az egyetlen q_a elfogadó, akceptáló állapotra. Elfogadó állapotba így csak akkor kerülhetünk, ha az eredeti automata kiürítette a vermét.

$$\delta(q_i\, \varepsilon, \check{Z}_0) = (q_a, \varepsilon) \qquad \forall\, q_i \in Q \qquad (3.34.)$$

Az, hogy a két automata, az eredeti és a módosított ugyanazt a nyelvet fogadja el, nem igényel magyarázatot.

Kíséreljük meg az automata erejének növelését azzal, hogy a veremből nyerhető információt ne korlátozzuk csupán a verem legfelső szimbólumára. Engedjük meg, hogy a mozgás eldöntésénél az automata bizonyos mélységig figyelembe vegye a verem tartalmát.

A belelátási mélység mozgási szabályról mozgási szabályra változhat, és a mozgás során figyelembe vett valamennyi szimbólum eltűnik a veremből. Ez tulajdonképpen megfelel a korábbi konvenciónak, csak ott mindig egyetlen szimbólumot veszünk figyelembe, és ezért csak egyetlen szimbólumot emeltünk le a veremből.

Minthogy a mozgási szabályok száma itt is véges, így a belelátási mélység korlátos. A véges számú szabályok között ugyanis nyilván van olyan, amelyik legmélyebben lát bele a verembe, és ez a mélység korlátot szolgáltat az automata belelátási mélységére.

Legyen a legmélyebbre látó szabály belelátási mélysége **i**. Ezzel formálisan leírhatjuk a mozgási szabályok által definiált leképezést:

$$Q \times (\Sigma \cup \varepsilon) \times \Gamma^{i} \Rightarrow Q \times \Gamma^{k} \qquad (3.35.)$$

ahol – Γ^{i} és Γ^{k} itt is, mint előbb, a Γ alfabetából alkotott és legfeljebb **i,** illetve **k** hosszúságú jelsorozatokat jelöli.

A mélybelátó automata ereje nem lehet kisebb az eredeti, mondjuk klasszikus automatáénál. Ugyanis, ha **i** értéke egy, akkor a klasszikus automatát kapjuk.

Az alábbiakban konstruktív bizonyítást adunk arra, hogy az **i** mélységbe belelátó automata ereje nem nagyobb a klasszikus, csupán egyetlen szimbólumot figyelembe vevő automatánál. Konstruálunk ugyanis egy, a belelátóval ekvivalens klasszikus veremautomatát.

Az algoritmus azon a meglátáson alapszik, hogy a belelátás mélysége korlátos, feltevésünk szerint **i**. A legfelső **i** veremszimbólum ezért korlátos információt hordoz. Tároljuk ennek alapján a legfelső **i** szimbólum által képviselt információt az új automata állapotaiban, vagyis legyen az új automata állapottere az eredeti automata állapotterének, és az **i** veremszimbólumnak direkt szorzata:

$$Q' = Q \times \Gamma^{i} \qquad (3.36.)$$

Az eredeti automata mozgásához szükséges valamennyi információt az automata állapota, az olvasott karakter, és a verem első, legfeljebb **i** számú szimbóluma hordozta. Ezeket az információkat mind megtalálhatjuk az új automata állapotában és az olvasott szimbólumban.

Két megjegyzés.

Egy veremszimbólumot a klasszikus veremautomata is figyelembe tud venni. Így tulajdonképpen elegendő lenne az eredeti verem első **i-1** szimbólumát az új automata állapotában tárolni. Az egységesebb tárgyalás miatt suvasztottuk bele mind az **i** szimbólumot az új állapotba.

Az új automata verme így az eredeti veremnek csak a felső **i** szimbólumnál mélyebben elhelyezkedő szimbólumait tárolja. Előfordulhat, hogy az eredeti automata vermében kevesebb, mint **i** szimbólum van. Ilyen esetekben egyrészt az új automata állapotában hiányozni fognak egyes szimbólumok, másrészt az új automata vermébe már nem jut szimbólum.

Ezen úgy segíthetünk, hogy az eredeti, belelátó automata veremtartalma alá **i** számú, eddig nem szerepelt szimbólumot, az irodalomban szokásos választással mondjuk a $ szimbólumot tesszük. Persze a Γ halmazt ezzel a szimbólummal ki kell bővítenünk.

Most már az új klasszikus automata verméből csak akkor fogyhatnak el a szimbólumok, ha az eredeti, belelátó automata verme kiürült. Ilyenkor az új automata állapotterében tárolt **i** szimbólum mindegyike a most bevezetett $ szimbólum lesz. Nagyon lényeges, hogy az új és régi automata vermei egyszerre ürülnek ki, hiszen ez előfeltétele annak, hogy egy üres veremmel elfogadó belelátó automatát egy üres veremmel elfogadó klasszikus automatával modellezzünk.

Nyilvánvaló, hogy az eredeti belelátó és új klasszikus automatáknak az automata állapotával és a verem tartalmával meghatározott szituációi között egy-egy megfeleltetés van. Az egyik automata állapotával és vermének tartalmával jellemzett szituációhoz kölcsönösen egyértelműen hozzárendelhető a másik automata egy adott szituációja, vagyis állapota és veremtartalma.

Az új automata mozgási szabályainak megállapításánál a fenti kapcsolatot kell szem előtt tartani. Amennyiben az eredeti automatát egy mozgás az $A \Rightarrow B$ szituáció átmenetet hozza létre, akkor az ennek az új automata esetében egy olyan mozgás vagy mozgássorozat feleljen meg, amely a hozzárendelt szituációk közötti átmenetet biztosítja.

Amennyiben az eredeti, belelátó automata olyan mozgási szabályait tekintjük, ahol a verembe beírt jelsorozat hossza nem rövidebb a mozgás során figyelembe vett, és ezért a veremből kitörölt jelsorozat hosszánál, akkor a két automata mozgásai között az egy-egy megfeleltetés könnyen megvalósítható.

Legyen ugyanis az eredeti, belelátó automata egy ilyen mozgási szabálya:

$$\delta(q, a, \alpha) = (p, \beta) \qquad |\alpha| \le |\beta| \qquad (3.37.)$$

A fenti szabálynak megfelelő új szabály aszerint, hogy a verembe beírt jelsorozat hossza nagyobb-e vagy sem a maximális betekintési mélységnél, a következő lesz:

$$\delta(q - \alpha\lambda, a, X) = (p - \beta\lambda_1, \lambda_2 X) \quad \text{ha} \quad |\beta| \le \mathbf{i} \qquad (3.38.)$$

$$\delta(q - \alpha\lambda, a, X) = (p - \beta_1, \beta_2 \lambda X) \quad \text{ha} \quad |\beta| > \mathbf{i} \qquad (3.39.)$$

A kötőjellel az új automata direkt szorzatként előálló állapotainak két összetevőjét kapcsoltuk össze.

A fenti képletekben

$$\lambda = \lambda_1 \lambda_2 \quad \text{illetve} \quad \beta = \beta_1 \beta_2 \quad \text{továbbá}$$
$$|\alpha| + |\lambda| = |\beta| + |\lambda_1| = \mathbf{i} \text{ illetve } |\beta_1| = \mathbf{i} \qquad (3.40.)$$

Határesetben, ha $|\beta| = \mathbf{i}$ akkor λ_1, illetve ha $|\alpha| = |\beta|$ akkor λ_2 az üres jelsorozat lesz. Minthogy valamennyi releváns szimbólumot belevettünk az új állapotba, a verem teteje itt nem játszik szerepet, ezért X tetszőleges veremszimbólum.

Az a tény, hogy a két megfeleltetett mozgási szabály valóban teljesíti a kikötött feltételeket, nem igényel magyarázatot.

Kissé bonyolultabb a helyzet, ha a verembe beírt jelsorozat rövidebb a kitöröltnél. Tételezzük most fel, hogy

$$|\alpha| > |\beta|$$

A nehézséget most az okozza, hogy az eredeti automata vermének első **i** szimbóluma a mozgás után nem áll rendelkezésre, hiszen többet töröltünk ki a veremből, mint amennyit beírtunk. Átmeneti megoldásként feleltessünk meg az eredeti automata (3.37.) szerint, tehát

$$\delta(q, \alpha\lambda, a, X) = (p, \beta)$$

alakú mozgási szabályának a

$$\delta(q, \alpha\lambda, a, X) = (p, \beta\lambda\sigma\sigma \ldots \sigma, \varepsilon) \qquad (3.41.)$$

mozgási szabályt. Természetesen most is

$$|\alpha| + |\lambda| = \mathbf{i}$$

A (3.41.) összefüggésben a Γ halmazhoz most hozzácsatolt, egyébként sohasem volt σ szimbólumból helykitöltés céljából annyit suvasztunk be az állapotban tárolt első szimbólumok közé, hogy a szükséges mennyiség, **i** darab kiteljék.

Ezen kívül az új automata mozgási szabályaihoz hozzáveszünk még egy ε-mozgásokból álló halmazt, a következők szerint:

$$\delta(p, \gamma\sigma\sigma \ldots \sigma, \varepsilon, X) = (p, \gamma X\sigma \ldots \sigma, \varepsilon) \qquad (3.42.)$$

Mint látható, a (3.42.) mozgások a klasszikus veremautomata vermének tetejét hozzáveszik az állapotban tárolt szimbólumokhoz, és a helykitöltő σ szimbólumokból egyet elhagynak. A fenti szabályt kellő türelemmel alkalmazva elérhető, hogy valamennyi helykitöltő szimbólumot igazándival helyettesítsünk. Ezzel a következő mozgás megállapításához szükséges valamennyi információ rendelkezésre áll.

Abban az esetben tehát, amikor a beírt jelsorozat hossza rövidebb a kitöröltnél, csak egy kis vargabetűvel érünk célt. Ilyenkor a (3.42.) típusú szabályok ismételt alkalmazásával biztosítjuk, hogy az új klasszikus automata végül is abba a szituációba kerüljön, amely az eredeti automata mozgás utáni szituációjához van rendelve.

Itt feltételeztük, hogy az eredeti automata valódi mozgást végez. ε-mozgás esetén ugyanez a módszer ugyanígy célra vezet.

Végeredményben, ha egy jelsorozat elemzésekor figyelemmel kísérjük mind az eredeti belelátó, mind az új klasszikus automata sorsát, akkor azt tapasztaljuk, hogy azok egymáshoz rendelt szituációpárokon át haladnak. Ebből viszont következik, hogy a két automata ugyanazt a nyelvet fogadja el.

3.5. A környezetfüggetlen nyelvek és veremautomaták ekvivalenciája

Említettük már, hogy a környezetfüggetlen nyelvtanok éppen azokat a nyelveket generálják, amelyeket a veremautomaták elfogadnak. A két halmaz, a környezetfüggetlen nyelveké, és a veremautomaták által elfogadottaké azonos. Ennek igazolásával még adósak vagyunk.

Bizonyításunk ismét konstruktív lesz.

Először egy környezetfüggetlen nyelvtanhoz készítünk olyan veremautomatát, amely pontosan a nyelvtan által generált nyelvet fogadja el. Ebből következik, hogy a veremautomaták által elfogadott nyelvek halmaza nem lehet szűkebb, mint a környezetfüggetlen nyelvtanok által generáltaké. Ezután csináljuk meg a fenti szerkesztés visszáját, vagyis egy veremautomatához egy olyan környezetfüggetlen nyelvtant konstruálunk, amely éppen az automata nyelvét generálja. Ebből viszont már következik az ekvivalencia.

Nézzük először a nyelvtan $\Rightarrow$ automata átalakítást.

Legyen adott egy környezetfüggetlen nyelvtan levezetési szabályaival. Definiáljunk egy üres veremmel elfogadó automatát a következőképpen. A verem alfabetája a nyelvtan terminális és nemterminális szimbólumainak uniója lesz. A verem kiinduló tartalma legyen a mondatszimbólum. Az egyszerűség kedvéért az automata állapothalmaza egyetlen elemet tartalmaz, jelölje ezt *q,* így abban nem tárolunk információt.

Az automata mozgási szabályai kétfélék. Az egyik fajta mozgási szabályhalmaz független a nyelvtan szabályaitól, a másik természetesen függ tőle.

Az első csoportba tartozó szabályok alakja a következő:

$$\delta(q, a, a) = (q, \varepsilon) \qquad (3.43.)$$

Ezek szerint, ha a verem tetején terminális elem van, és az megegyezik az éppen beolvasottal, akkor az a verem tetejéről leemelhető. Arra az esetre, amikor a verem tetején lévő terminális nem azonos a beolvasott karakterrel nincs kádencia, az elemzés – legalábbis ezen az úton – zsákutcába jutott.

A szabályok másik fajtája, amelyek a nyelvtant és így a nyelvet tükrözik, ε-mozgásokat határoznak meg.

Alakjuk a következő:

$$\delta(q, \varepsilon, A) = (q, \alpha) \tag{3.44.}$$

ahol – A a nyelvtan egy nemterminális szimbóluma, és létezik egy

$$A \to \alpha \tag{3.45.}$$

alakú levezetési szabály. Minden levezetési szabálynak megfelel egy ilyen szabály.

Szavakban ez annyit jelent, hogy valahányszor egy nemterminális kerül a verem tetejére, azt egy olyan szabály jobboldalával helyettesítjük, amelynek baloldala éppen a szóban forgó nemterminális.

Könnyű belátni, hogy ez az automata éppen a baloldali, a *top-down* vagyis felülről lefelé történő levezetést illetve elemzést hajtja végre. Vegyük észre, ha a már elolvasott jelsorozathoz a verem tartalmát hozzáfűzzük, éppen a baloldali levezetés mondatszerű formáit kapjuk.

Induláskor, amikor még egyetlen szimbólumot sem olvastunk be, a verem tartalma a mondatszimbólum. Ez tehát levezetésünk első mondatszerű formája. Az elemzés során a (3.44.) típusú szabályok alkalmazásakor a baloldali levezetés egy újabb mondatszerű formáját állítjuk elő. A (3.43.) mintájú mozgási szabályok nem váltanak mondatszerű formát, csupán a következő nemterminálist keresik. Minthogy a verem tetejére mindig a legbaloldalibb nemterminális kerül, következésképpen az automata a baloldali levezetést hajtja végre.

A gondolatmenetből következik, hogy minden sikeres baloldali levezetéshez tartozik egy olyan automata mozgássorozat, amely a levezetett mondat elfogadásához vezet, és fordítva, minden az automata által elfogadott mondatnak van a nyelvtanban baloldali levezetése.

Természetesen a fenti algoritmussal származtatott automata nem unikális, nem az egyetlen, amely a nyelvtani szabályok alapján generált mondatokat elfogadja. Mint ahogy a baloldali levezetés sem az egyetlen üdvözítő módszer. Ennek dokumentálására környezetfüggetlen nyelvtanunkhoz most egy olyan veremautomatát szerkesztünk, amely a jobboldali *bottom-up* elemzést végzi.

A jobboldali elemzés a jobboldali levezetést követi, csak éppen fordított sorrendben. Így a generált mondatból indul ki, és a levezetési szabályok jobboldalát megkeresve, azokat a baloldallal helyettesítve halad mondatszerű formáról mondatszerű formára visszafelé, mindaddig, amíg a végén az eredeti mondatot a mondatszimbólumra nem redukálta.

A jobboldali elemzést modellező veremautomata üres veremmel elfogadó, belelátó automata lesz. Kényelmi okokból a belelátó automatánál a verem tartalmát nem a megszokott módon balról jobbra, hanem fordítva írjuk. Itt tehát a verem teteje nem a legbaloldalibb, hanem a legjobboldalibb szimbólum. Hogy ez mennyiben jelent kényelmet az a későbbiekből derül ki. Az automata állapothalmaza itt is egyetlen elemet tartalmaz.

Ebben az esetben is a mozgási szabályoknak két csoportját definiálunk. Az egyik csoport itt is független a nyelvtani szabályoktól, és csak a másik csoport tükrözi a nyelvtan sajátosságait.

A nyelvtantól független szabályok alakja a következő :

$$\delta(q, a, X) = (q, Xa) \; \forall \; a \in \Sigma \qquad \text{és} \qquad \forall \; X \in \Gamma \qquad (3.46.)$$

ahol – X tetszőleges verembeli szimbólum.

Ezen szabály szerint az olvasott szimbólumot minden esetben beírhatjuk a verem tetejére – vigyázat, a verem teteje most a jobboldalon van – anélkül, hogy a verem korábbi tartalmát törölnénk.

A nyelvtantól függő szabályok alakja, amelyek ez esetben is ε-szabályok lesznek, a következő:

$$\delta(q, \varepsilon, \alpha) = (q, A) \qquad (3.47.)$$

ha létezik egy $A \rightarrow \alpha$ alakú levezetési szabály.

Ez annyit jelent, hogy ha a verem tetején kialakul egy levezetési szabály jobboldala – fordított sorrendben, – akkor helyette beírhatjuk a megfelelő baloldalt.

Amikor ugyanis a (3.46.) szabályok segítségével szép sorjában az olvasott szimbólumokat a verembe beírjuk, akkor azok ott fordított sorrendben jelennek meg. Amikor a szokástól eltérően a verem tetejét a jobboldalra írjuk, akkor ebben az a kényelem, hogy ilyenkor a levezetési szabályok fordított sorrendben elhelyezkedő jobboldalai a megszokott betűképpel olvasmányosan láthatóak.

A fenti két megadott típusú mozgási szabályokon kívül még egy mozgási szabályra van szükségünk :

$$\delta(q, \varepsilon, Z_0 S) = (q, \varepsilon) \qquad (3.48.)$$

ahol

- Z_0 a verem kiindulási szimbóluma, amely nem azonos egyik nemterminális szimbólummal sem,
- S a nyelvtan mondatszimbóluma.

Amennyiben a veremben a kiindulási szimbólumon kívül csak a mondatszimbólum marad, akkor a vermet kiürítve az elemzett jelsorozatot elfogadjuk.

Az elemzés gondolatmenete megegyezésben korábbi megjegyzésünkkel a következő: A kész mondatból indulunk ki és keressük azokat a fragmenseket, amelyek egy nemterminális helyettesítése útján kerülhettek be a mondatba vagy mondatszerű formába. Ezek nyilván azonosak a levezetési szabályok jobboldalával. Ezeket a jobboldalakat nyeleknek nevezzük. Az ilyen nyelek helyébe írhatjuk be azt a nemterminálist, amelyből származott. Ezt a műveletet a szakmai zsargon a nyelek letörésének nevezi.

Minthogy balról jobbra olvasunk, elsőként mindig a legbaloldalibb nyelet fogjuk letörni. Ennek alapján nem nehéz belátni, hogy ez az elemzés fordított sorrendben végzi el a jobboldali levezetést, tulajdonképpen jobboldali «felvezetést» hajt végre.

A teljes mondatból indul ki, és a nyelek letörésével a jobboldali levezetés mondatszerű formáinak sorozatán visszafelé haladva siker esetén egészen a mondatszimbólumig jut.

Ügyelnünk kell arra, hogy nem minden fragmens, amely megegyezik egy szabály jobboldalával, bizonyul nyélnek. Lehet, hogy nem igazándi nyél, hanem csak nyélnek látszó nyelvi tünemény. Ha azután egy ilyen álnyelet törünk le, akkor az elemzés előbb utóbb kátyúba kerül.

Az a probléma, hogy egy jobboldallal megegyező fragmens valódi nyél-e vagy csak álnyél, a formális nyelvek elméletének egyik igen fontos kérdése. A későbbiekben ezzel még mélyrehatóbban foglalkozunk.

Szerencsére pillanatnyilag nincsen problémánk, hiszen automatánk nemdeterminisztikus. Ha a verem tetején nyelet, pontosabban annak fordítottját találjuk, akkor mind a nyél letörését jelentő (3.47.) típusú szabályt, mind annak negligálásával egyenértékű (3.46.) szabályt alkalmazhatjuk.

A nemdeterminisztikus automaták esetében tudjuk, hogy valamennyi lehetséges lépést figyelemmel kell kísérnünk. Így tehát a nyél letörését is, meg negligálását is ki kell próbálnunk. Ha a jelsorozat eleme a nyelvnek, akkor valamelyik út szükségképpen eredményre vezet.

Abból, hogy az automata a jobboldali levezetést állítja elő, – az, hogy fordított sorrendben, az ebből a szempontból közömbös – következik, hogy a nyelvtan által generált nyelv azonos az automata nyelvével. Ha ugyanis a mondat generálható, akkor a jobboldali levezetés szerint az automata el tudja fogadni. Fordítva, ha az automata a jelsorozatot elfogadta, akkor létezik annak az adott nyelvtan szerinti jobboldali levezetése, vagyis a jelsorozat generálható, tehát mondat.

Mielőtt a bizonyítás második lépését megtennénk, vagyis egy veremautomatához készítenénk környezetfüggetlen nyelvtant, lássunk példát az eddigiekre:

Legyen a környezetfüggetlen nyelvtan:

$$S \rightarrow aSa \mid bSb \mid aa \mid bb$$

Ez a nyelvtan, mint már tudjuk, a

$$ww^{-1}$$

nyelvet generálja. Itt w tetszőleges a és b karakterekből álló jelsorozat.

Írjuk fel a baloldali levezetést előállító veremautomata mozgási szabályait. Előbb a nyelvtantól független mozgási szabályok:

$$\delta(q, a, a) = (q, \varepsilon) \qquad \delta(q, b, b) = (q, \varepsilon)$$

A nyelvtan levezetési szabályaitól függő mozgási szabályok :

$$\delta(q, \varepsilon, S) = (q, aSa) \mid (q, bSb) \mid (q, aa) \mid (bb)$$

Itt az egyszerűsítés kedvéért ismét éltünk a vagy jelentésű $\mid$ jel használatával. Az automata nemdeterminisztikus, hiszen az egyetlen nemterminális szimbólum négyféleképpen írható át.

Elemezzük most az *abbaabba* mondatot. A mondat természetesen az adott nyelvtan segítségével generálható, de ennek demonstrálását a szorgos olvasóra bízom.

Az egyszerűség és papírtakarékosság okából a nemdeterminisztikus automatának – bölcs előrelátással – mindig csak azokat a lépéseit vesszük, amelyek célra vezetnek. Az olvasó persze ellenőrizheti, hogy a negligált lépések valóban zsákutcát eredményeznek.

Kövessük konfigurációról konfigurációra az elemzés menetét:

$$\{\, abbaabba, q, S \,\} \mapsto \{\, abbaabba, q, aSa \,\} \mapsto \{\, bbaabba, q, Sa \,\} \mapsto$$
$$\mapsto \{\, bbaabba, q, bSba \,\} \mapsto \{\, baabba, q, Sba \,\} \mapsto \{\, baabba, q, bSbba \,\} \mapsto$$
$$\mapsto \{\, aabba, q, Sbba \,\} \mapsto \{\, aabba, q, aabba \,\} \mapsto \{\, abba, q, abba \,\} \mapsto$$
$$\mapsto \{\, bba, q, bba \,\} \mapsto \{\, ba, q, ba \,\} \mapsto \{\, a, q, a \,\} \mapsto \{\, \varepsilon, q, \varepsilon \,\}$$

Az automata a jelsorozatot elfogadta, hiszen azt végigolvasta, és verme kiürült.

A másik esetre, amikor belelátó automata segítségével végzünk jobboldali elemzést, válasszuk példának az aritmetikai kifejezéseket generáló nyelvtant.

$$E \to E{+}T \mid T \qquad T \to T{*}F \mid F \qquad F \to (E) \mid a$$

Talán most engedjük meg magunknak azt a lazaságot, hogy mellőzzük a nyelvtantól nem függő mozgási szabályok leírását. A nyelvtantól függő szabályok alakja:

$$\delta(\, q, \varepsilon, E{+}T \,) = (\, q, E \,) \qquad \delta(\, q, \varepsilon, T \,) = (\, q, E \,)$$
$$\delta(\, q, \varepsilon, T{*}F \,) = (\, q, T \,) \qquad \delta(\, q, \varepsilon, F \,) = (\, q, T \,)$$
$$\delta(\, q, \varepsilon, (E) \,) = (\, q, F \,) \qquad \delta(\, q, \varepsilon, a \,) = (\, q, F \,)$$

Remélem, hogy a két különböző funkciójú zárójel jelenléte nem zavaró. Mindenesetre a mozgási szabályokat leíró metanyelv zárójeleit egyszerűen, a generált nyelvét pedig vastagítva írtuk.

A záróaktusként használt mozgási szabály most:

$$\delta(\, q, \varepsilon, Z_0 E \,) = (\, q, \varepsilon \,)$$

hiszen esetünkben a mondatszimbólum E.

Elemezzük az alábbi mondatot:

$$a{*}(a{+}a)$$

Ez alkalommal készítsük el a jobboldali levezetést is, egyrészt, mert ilyet még úgy sem csináltunk, másrészt, hogy összevessük a jobboldali „felvezetéssel”, pontosabban a felvezetés során kapott mondatszerű formákkal.

Ezeket a mondatszerű formákat majd úgy állíthatjuk össze, hogy a verem tartalmához hozzáfűzzük a még el nem olvasott jelsorozatot.

A jobboldali levezetés:

$$E \Rightarrow T \Rightarrow T{*}F \Rightarrow T{*}(E) \Rightarrow T{*}(E{+}T) \Rightarrow T{*}(E{+}F) \Rightarrow T{*}(E{+}a) \Rightarrow$$
$$\Rightarrow T{*}(T{+}a) \Rightarrow T{*}(F{+}a) \Rightarrow T{*}(a{+}a) \Rightarrow F{*}(a{+}a) \Rightarrow a{*}(a{+}a)$$

Az elemzés során itt is csak a valódi nyeleket törjük le. Elvben minden nyélnek látszó fragmenst le kellene törnünk, és csak fiaskó esetén térni át a még reményt keltő ágakra.

Minden olyan helyen, ahol a nyélnek látszó fragmenst nem törtük le, a $\mapsto$ származtatási jelet követő felkiáltó jellel hívjuk fel erre a körülményre az olvasó figyelmét. Az elemzés:

$$\{ a*(a+a), q, Z_0 \} \mapsto \{ *(a+a), q, Z_0a \} \mapsto \{ * (a+a), q, Z_0F \} \mapsto$$
$$\mapsto \{ *(a+a), q, Z_0T \} \mapsto \mathbf{!}\ \{ (a+a), q, Z_0T* \} \mapsto \{ a+a), q, Z_0T*(\} \mapsto$$
$$\mapsto \{ +a), q, Z_0T*(a \} \mapsto \{ +a), q, Z_0T*(F \} \mapsto \{ +a), q, Z_0T*(T \} \mapsto$$
$$\mapsto \{ +a), q, Z_0T*(E \} \mapsto \{ a), q, Z_0T*(E+ \} \mapsto \{), q, Z_0T*(E+a \} \mapsto$$
$$\mapsto \{), q, Z_0T*(E+F \} \mapsto \{), q, Z_0T*(E+T \} \mapsto \mathbf{!}\{), q, Z_0T*(E \} \mapsto$$
$$\mapsto \{ \varepsilon, q, Z_0T*(E) \} \mapsto \{ \varepsilon, q, Z_0T*F \} \mapsto \{ \varepsilon, q, Z_0T \} \mapsto$$
$$\mapsto \{ \varepsilon, q, Z_0E \} \mapsto \{ \varepsilon, q, \varepsilon \}$$

A felkiáltójellel azokat a helyeket jelöltük meg, ahol a *T* nemterminális letörése helyett a következő karaktert csúsztattuk be a verembe. A jelzett helyen kívül volt olyan eset is, amikor „nyéltúltengésben" szenvedtünk. Például amikor a verem felső három szimbóluma *E+T* volt, persze fordított sorrendben. Ekkor a *T,* de az *E+T* is potenciális nyél volt. Természetesen itt is csak az eredményre vezető változatot vettük figyelembe.

Nyelvtanból tudunk már veremautomatát készíteni, de hogyan szerkesszünk egy veremautomatához környezetfüggetlen nyelvtant?

Előre kell bocsátanunk, hogy a tárgyalás itt kissé terjedelmesebb lesz. Eddigi eredményeinket ugyanis nem tudjuk közvetlenül megfordítani, felhasználni.

Ha a fentiek szerint egy nyelvtanhoz akár a baloldali levezetést, akár a jobboldali „felvezetést" modellező automatát készítünk is, az automatánknak csak egy állapota lesz. Ez annyit jelent, hogy a kidolgozott, a nyelvtanból automatát szerkesztő algoritmus megfordítása csak akkor használható, ha az automatának csak egy állapota van. Szándékunk szerint viszont olyan módszerre, olyan algoritmusra van szükségünk, amely bármely veremautomatához, így a több állapottal bíró automatához is rendel egy környezetfüggetlen nyelvtant.

Kiindulásul tételezzük fel, hogy automatánk üres veremmel fogad el. Ez nem jelent korlátot, hiszen mint láttuk, minden veremautomatához szerkeszthető egy vele egyenértékű üres veremmel elfogadó automata.

Az az alapötlet, hogy az elemzés automata szituációihoz a nyelvtani levezetés mondatszerű formái tartozzanak, itt is gyümölcsöző lesz.

Vizsgáljunk meg egy – az elemzés során előálló – szituációt. Tételezzük fel, hogy az automata a jelsorozatot végül el fogja fogadni. A már olvasott szimbólumokat az automata már feldolgozta. Ennek a feldolgozásnak az eredménye a verem tartalmában tükröződik. A siker feltétele az, hogy a verem tartalma lehetővé tegye a még el nem olvasott szimbólumok feldolgozását,

mégpedig olymódon, hogy mire az utolsó szimbólumot is elolvassuk, a verem éppen kiürüljön.

Amennyiben egy ilyen szituációhoz megpróbáljuk a nyelvtani levezetés egy mondatszerű formáját hozzárendelni, akkor a következőképpen járhatunk el.

Az elemzés során elolvasott szimbólumok kiléte már kiderült. A mondatszerű forma célszerűen ezekkel a már elolvasott szimbólumokkal kezdődjék. Ezt követi a verem tartalmának megfelelő nemterminálisok sorozata. A megfeleltetés nyilván akkor helytálló, ha ezekből a nemterminálisokból akkor és csak akkor generálható a még el nem olvasott jelsorozat, ha azt az automata elfogadja, a teljes jelsorozat az automata nyelvének mondata.

Vizsgáljuk most meg, hogyan lehet az automata egy mozgási szabályának a nyelvtan egy levezetési szabályát megfeleltetni. Rendeljünk a verem minden szimbólumához a mondatszerű forma egy nemterminálisát.

Az automata mozgása során megállapodásunk szerint töröljük a verem legfelső szimbólumát, és helyébe egy – esetlegesen üres – jelsorozatot írunk. Ezen túlmenően, ha valódi mozgásról van szó, akkor még egy karaktert is elolvasunk.

Az automata mozgási szabályához rendelt levezetési szabály baloldala a verem tetején lévő, a mozgás során kitörölt szimbólumnak megfelelő nemterminális lesz. Valódi mozgás esetén a jobboldal az elolvasott terminális szimbólummal kezdődik. Ez biztosítja, hogy a mondatszerű forma mindig a már elolvasott jelsorozattal kezdődjék.

A levezetési szabály jobboldalának többi eleme a verem tetejére most beírt jelsorozat szimbólumainak feleljen meg. Ha például semmit sem írunk a verembe, akkor a jobboldal nem tartalmaz nemterminális szimbólumot.

Első látásra kézenfekvőnek tűnik az a megoldás, hogy amikor a veremszimbólumokhoz olymódon rendelünk nemterminálisokat, hogy különböző szimbólumhoz különböző, azonos szimbólumhoz azonos nemterminálist feleltetünk meg. Ezzel a mondatszerű formában azt az információt tároltuk, amit az automata verme hordoz.

Ezzel tulajdonképpen célt is érnénk, ha az automatának csupán egy állapota lenne. Ebben az esetben ugyanis az automatában tárolt információt egyedül a veremtartalom szolgáltatja. Amennyiben az automatának több állapota van, ezt az információt is érvényre kell juttatnunk a mondatszerű formában. Ilyenkor az elolvasott jelsorozat nyújtotta információ nem csak a verem tartalmában testesül meg, hanem az automata állapotában is. Éppen ezért az egyes nyelvtani szabályok alkalmazhatóságának függenie kell az automata állapotától. A megoldást az adja, hogy az automata állapotait is beépítjük a nyelvtan nemterminális szimbólumaiba.

A verem minden szimbólumához két, az adott szimbólum életében fontos szerepet játszó állapotot társítunk.

Az első állapot az, amikor a szimbólum a verem tetejére kerül, és befolyásolja az automata következő mozgását, amelynek során azután kitörlődik.

A másik nevezetes állapot az az állapot, amikor a szóban forgó verembeli szimbólum és összes utódai kihalnak. Amikor a verem tetejéről a szimbólumot kitöröljük, szimbólumok sorozatát írjuk be a helyére. Ezeket kis fantáziával a szimbólum gyerekeinek tekinthetjük. A verem tetején a legidősebb gyermek, a sorozat utolsó eleme a legfiatalabb gyermek. Itt szigorú hierarchia van. Először a legidősebb gyermek hal ki, majd az azt követő, legvégül a legfiatalabb gyermek kihalásával szűnik meg a család. Persze ezt a kihalást rekurzíve kell értelmezni, hiszen a gyermekeknek is lehetnek gyermekei, és így tovább. Kérdés, valóban kihal-e mindegyik veremszimbólum családostul. Igen, mondatok elemzésekor ez szükségszerű, hiszen az automata üres veremmel fogad el, így végül egy szimbólum sem marad a veremben.

Ezt másképpen úgy is fogalmazhattuk volna, hogy a második állapot az az állapot, amikor a szóban forgó alatti veremszimbólum kerül a verem tetejére. Természetesen a két megfogalmazás egyenértékű.

A nyelvtan nemterminális szimbólumainak tehát tartalmazniuk kell ezt a két állapotot is, így tulajdonképpen egy hármast alkotnak. Jelölésük:

$$[pAr] \tag{3.49.}$$

A félreértések elkerülése végett szeretném ismét hangsúlyozni, hogy a (3.49.) alakú fragmensek a nyelvtan *egy* nemterminális szimbólumát jelölik. A leírt nemterminális megfelel az automatában egy olyan A verembeli szimbólumnak, amely a p állapotban kerül a verem tetejére, és az r állapotban hal ki, közvetlenül vagy utódaiban.

Így természetesen a verem alfabetájának egy szimbólumához a nyelvtan több nemterminális szimbóluma tartozhat, hiszen annak különböző példányai különböző nevezetes állapotokat vehetnek fel. Persze ez azzal jár, hogy a nyelvtan nemterminálisainak száma jelentősen nagyobb lehet a verem alfabetájának számosságánál.

Tekintsük most az automata lehetséges mozgási szabályait, és feleltessünk meg ezeknek nyelvtani levezetési szabályokat.

Aszerint, hogy az automata valódi vagy ε-mozgást végez, illetve, hogy történik-e beírás a verembe vagy sem, négy szabályváltozatot fogunk megkülönböztetni.

$$\delta(p, a, A) = (r, \alpha) \tag{3.50.}$$

$$\delta(p, \varepsilon, A) = (r, \alpha) \tag{3.51.}$$

ahol – α a veremszimbólumok nem üres sorozata:

$$\alpha = X_1X_2 \ldots X_k \qquad \mathbf{k} \geq 1 \tag{3.52.}$$

A másik két szabályváltozat arra az esetre érvényes, amikor nem történik beírás a verembe:

$$\delta(\, p,\, a,\, A\,) = (\, r,\, \varepsilon\,) \tag{3.53.}$$

$$\delta(\, p,\, \varepsilon,\, A\,) = (\, r,\, \varepsilon\,) \tag{3.54.}$$

Most rendre megadjuk az egyes változatoknak megfelelő nyelvtani szabályokat:

$$[pAs_k] \rightarrow a\,[rX_1s_1]\,[s_1X_2\,s_2]\,.\,.\,.\,.[s_{k-1}X_ks_k] \tag{3.55.}$$

$$[pAs_k] \rightarrow\ [rX_1s_1]\,[s_1X_2s_2].\,.\,.\,.\,[s_{k-1}X_ks_k] \tag{3.56.}$$

Itt természetesen a (3.55.) levezetési szabály a (3.50.) mozgásnak, míg a (3.56.) szabály a (3.51.) mozgásnak felel meg.

Az összefüggések bizonyos magyarázatot igényelnek.

Az $s_1,s_2,\ .\ .s_k$ szimbólumok nem szükségszerűen különböző automata állapotokat jelölnek.

A szabályok megértéséhez próbáljuk meg összefoglalni, mi mindent tudunk az egyes verembeli szimbólumok megjelenési és kihalási állapotáról.

A (3.50.) és (3.51.) mozgási szabályból következik, hogy az A szimbólum a p állapotban, míg az α jelsorozat első eleme az X_1 szimbólum az r állapotban jelenik meg a verem tetején. Az, hogy a többi szimbólum milyen állapotban tűnik fel és milyen állapotban hal ki, arra az automata mozgási szabályai nem adnak felvilágosítást.

Pusztán azt tudjuk, hogy amikor valamely szimbólum akár önmagában, akár utódaiban kihal, akkor jelenik meg az őt sorrendben követő szimbólum. Így egy szimbólum kihalási állapota meg kell egyezzen a következő szimbólum megjelenési állapotával.

Az automata mozgási szabályaiból következik, hogy az A veremszimbólum utódaiban hal ki. Pontosan akkor, amikor annak legkisebb gyermeke, a most beírt sorozat utolsó eleme X_k kihal. Ezek szerint az A és X_k szimbólum kihalása ugyanaz az esemény, így természetesen ugyanaz az állapot tartozik hozzájuk.

Ezeket a megfontolásokat a (3.55.) és (3.56.) kifejezések felírásánál már érvényre juttattuk. A két szabály között csak annyi a különbség, hogy az egyik valódi, a másik ε-mozgást feltételez, így az első szabály jobboldala az elolvasott terminálissal kezdődik.

Az összes rendelkezésünkre álló információ tehát be van építve a levezetési szabályokba. Mit kezdjünk azonban azokkal, az egyelőre szabadon hagyott állapotokkal, amelyekről közelebbi információnk nincsen. A válasz: írjuk be az összes lehetséges változatot. További ismeret hiányában akkor járunk el biztonságosan, ha valamennyi lehetséges helyzetre felkészülünk.

Ha – mint esetünkben – egy mozgás során **k** számú szimbólumot írunk be a verembe, és az automata állapotainak száma **i**, akkor az automata egyetlen

ilyen mozgási szabályához **i k** osztályú ismétléses variációjának megfelelő számú, vagyis $\mathbf{i}^{\mathbf{k}}$ levezetési szabály tartozik.

Sajnos ez kétségbe ejtően megnöveli a levezetési szabályok számát, és felduzzasztja a nyelvtant. Szerencsére az automata működése során nem minden állapotpár realizálódhat, ami annyit jelent, hogy a nyelvtanban sok lesz a felesleges szimbólum. Egy alapos fésülés tehát jelentősen megritkíthatja a nyelvtant.

Adósok vagyunk még a (3.53.) és (3.54.) mozgási szabályoknak megfelelő levezetési szabályokkal.

$$[pAr] \rightarrow a \qquad (3.57.)$$

$$[pAr] \rightarrow \varepsilon \qquad (3.58.)$$

Itt egy olyan pillanatnak vagyunk szemtanúi, amikor az A veremszimbólum kihal. A korábban elmondottak alapján talán felesleges a további magyarázat. Szerencsére itt minden mozgási szabályhoz csak egy levezetési szabály tartozik.

Amikor egy környezetfüggetlen nyelvtanhoz szerkesztettünk veremautomatát, akkor a nyelvtan levezetési szabályai, és az automata mozgási szabályai között egy-egy értelmű kapcsolat volt. Itt sajnos nem ez a helyzet.

Ennek alapvető oka az, hogy míg a mondatszerű formák, illetve a levezetési szabályok messzemenően predeterminálják az automata állapotváltozásait, hiszen a nemterminálisokban egy állapotsorozat van rögzítve, addig a mozgási szabályok csak a következő automataállapotot írják elő.

Ezek szerint, míg egy levezetési szabály alkalmazásakor a nyelvtannak el kell döntenie, hogy az automata a jövőben milyen állapotokat vegyen fel, addig az automata mozgási szabályai még nagymértékben szabad kezet biztosítanak az automatának.

Bárhogyan „dönt” is azonban a későbbiekben az automata az állapotok sorozatát illetően, a mozgási szabályból származtatható levezetési szabályok között biztos lesz olyan, amely pontosan ezt az állapotsorozatot írja elő. Vagyis a mozgást követő valamennyi állapot-egymásutánhoz tartozik levezetési szabály.

Előbbi megállapításunk triviális, ha meggondoljuk, hogy a levezetési szabályok származtatásánál az állapotok sorrendjének összes lehetséges kombinációját figyelembe vettük. Bármi legyen is tehát az állapotok sorrendje, van rá levezetési szabály. Sőt a veremautomata általában nem is tud minden elképzelhető állapot sorrendet létrehozni, így a kombinációk között előfordulhatnak lehetetlenek is.

Természetesen, ha az így kiadódó környezetfüggetlen nyelvtanból kiirtjuk a felesleges szimbólumokat, akkor a lehetetlen állapotkombinációknak megfelelő levezetési szabályok kiesnek.

Amennyiben a nyelvtan képes valamely mondat generálására, akkor ez az állítás egyenértékű azzal, hogy csakis lehetséges állapotsorozatnak megfelelő levezetési szabályt használtunk fel. Ez annyit jelent, hogy az automata mindig

képes működését olyan értelemben folytatni, hogy a levezetési szabály által megszabott állapotsorozat korrekt legyen.

Így, ha van egy szituáció sorozatunk, amely egy jelsorozat elfogadásához vezet, akkor mindig található a mondatszerű formáknak olyan egymásutánja, amely az automata által elfogadott mondat levezetését szolgáltatja.

Fordítva, ha a nyelvtan képes valamely mondat generálására, akkor az automata a mozgási szabályok alkalmazásával el fogja fogadni a kérdéses jelsorozatot.

Egy dologgal még adósak vagyunk. Ha már van egy mondatszerű formánk, és vele együtt egy kompatibilis automata szituációnk, akkor a továbbiakban már tudjuk, mit kell tennünk. Ezt az állapotot azonban elő kell állítanunk, mert különben nem indulhatunk el a levezetéssel.

Ennek érdekében egy új levezetési szabálycsoportot hozunk létre. Legyen az automata kezdőállapota q_0, és az induló veremtartalom Z. A veremtartalomnál az eddig szokásos indexet a későbbiekre tekintettel elhagytuk. Jelölje itt is S a nyelvtan mondatszimbólumát. Ekkor a következő levezetési szabályokat kell bevezetnünk:

$$S \to [q_0 Z s_i] \tag{3.59.}$$

ahol – s_i végigfut az összes lehetséges automata állapotokon.

Valóban. Induláskor még egyetlen szimbólumot sem olvastunk el, így a mondatszerű forma elején nem lehetnek terminális szimbólumok. A verem tartalma csak egyetlen szimbólum, így a mondatszerű forma is csak egyetlen nemterminálisból állhat. Tudjuk, a Z verembeli szimbólum megjelenésekor az automata a q_0 állapotban van, arról azonban nincs információnk, milyen állapot lesz érvényes akkor, amikor a Z szimbólum kihal. Minthogy a Z szimbólum alatt nincs semmi, ez a verem kiürülését, vagyis a mondat elfogadását jelenti. Fogalmazhattunk volna tehát úgy is, hogy nincs tudomásunk arról, milyen állapotban lesz az automata akkor, amikor elfogadja a jelsorozatot. A jól bevált módszer szerint az összes lehetséges állapotra írunk egy-egy szabályt, valamelyik majd csak eltalálja.

Lássunk erre is egy példát. Legyenek a veremautomata mozgási szabályai:

$\delta(q_0, a, Z) = (q_1, A)$ $\quad$ $\delta(q_0, b, Z) = (q_1, B)$

$\delta(q_1, a, A) = (q_1, AA) \mid (q_2, \varepsilon)$ $\quad$ $\delta(q_1, b, A) = (q_1, BA)$

$\delta(q_1, a, B) = (q_1, AB)$ $\quad$ $\delta(q_1, b, B) = (q_1, BB) \mid (q_2, \varepsilon)$

$\delta(q_2, a, A) = (q_2, \varepsilon)$ $\quad$ $\delta(q_2, b, B) = (q_2, \varepsilon)$

Ez az automata régi ismerősünk, bár kicsit át lett alakítva annak érdekében, hogy üres veremmel fogadjon el. Nyelve a nem szájbarágós palindrom, azaz

$$ww^{-1}$$

ahol w az a és b szimbólumokból alkotott tetszőleges nem üres jelsorozat.

Kezdjük először a (3.59.) szerinti szabályokkal:

$$S \to [q_0 Z q_0] \mid [q_0 Z q_1] \mid [q_0 Z q_2]$$

Egy megjegyzés és egy megállapodás.

Megbeszélésünk szerint felkészültünk arra, hogy a verem bármilyen állapotban kiürülhet. Ha azonban jobban megvizsgáljuk a nyelvtant, kiderül, hogy a jelsorozat elfogadása, a verem kiürülése csakis a q_2 állapotban következhet be. Ebből következően a fent leírt három szabály közül csak a harmadik, az utolsó vezethet eredményre. Pontosan az, ahol az az állapot szerepel a verem legalsó szimbólumának halála, magyarán a verem kiürítése esetére, amelyben a verem a valóságban kiürül. A másik két szabályban található hármas nemterminálisok nem fordulhatnak elő, így feleslegesek. Erről azonban ne vegyünk tudomást, ennek a fésülésnél ki kell derülnie.

Az áttekinthetőség növelésére, egyszerűsítés céljából a nemterminális szimbólumokat alkotó hármast jelöljük egy kettős indexszel ellátott nagybetűvel. Így például az A_{12} lesz a $[q_1Aq_2]$ nemterminális egyszerűsített jelölése.

A kapott eredmények átírása az új jelölésre nyilván nem okozhat gondot.

Az automata tíz mozgási szabálya közül négy szabály két-két szimbólumot, két szabály egy-egy szimbólumot végül további négy semmit sem ír be a verembe. Minthogy az állapotok száma három, ezekhez a szabályokhoz rendre $3^2 = 9$, $3^1 = 3$ és $3^0 = 1$ levezetési szabály tartozik. Ha még ezekhez hozzávesszük az induláshoz szükséges három szabályt, akkor kiadódik a levezetési szabályok száma, ami 49. Kétségbe ejtően nagy szám, különösen, ha meggondoljuk, hogy ezt a nyelvet egyszer már sikerült leírnunk egy négy szabályból álló nyelvtannal. Csak reménykedni lehet, hogy a fésülés majd jelentősen redukálja a szabályok számát.

Ehhez azonban le kell írnunk ezeket a szabályokat. Már csak papírtakarékossági okokból is alkalmazzunk egy jelöléstechnikai huszárvágást. Ott, ahol fel kell venni minden lehetséges állapotot, vezessünk be indexváltozókat azzal, hogy ezeknek végig kell futni a {0,1,2} indexhalmazon.

Írjuk le ezzel a konvencióval a levezetési szabályokat, a kiinduló szabályokkal kezdve, egyébként a mozgási szabályok leírásának sorrendjét követve.

$$
\begin{array}{lll}
S \to Z_{0\mathrm{i}} & Z_{0\mathrm{i}} \to aA_{1\mathrm{i}} & Z_{0\mathrm{i}} \to bB_{1\mathrm{i}} \\
A_{1\mathrm{i}} \to aA_{1\mathrm{k}}A_{\mathrm{ki}} & A_{12} \to a & A_{1\mathrm{i}} \to bB_{1\mathrm{k}}A_{\mathrm{ki}} \\
B_{1\mathrm{i}} \to aA_{1\mathrm{k}}B_{\mathrm{ki}} & B_{1\mathrm{i}} \to bB_{1\mathrm{k}}B_{\mathrm{ki}} & B_{12} \to b \\
A_{22} \to a & B_{22} \to b &
\end{array}
$$

Azok a szabályok, ahol két futó index szerepel, 9 levezetési szabályt reprezentálnak, ahol csak egy futó index van, azok hármat. Kezdjük el a nyelvtan fésülését alulról:

$$\mathbf{B}_0 = \{ a, b \} \qquad \mathbf{B}_1 = \{ a, b, A_{22}, B_{22}, A_{12}, B_{12} \}$$
$$\mathbf{B}_2 = \{ a, b, A_{22}, B_{22}, A_{12}, B_{12}, Z_{02} \}$$
$$\mathbf{B}_3 = \mathbf{B} = \{a, b, A_{22}, B_{22}, A_{12}, B_{12}, Z_{02}, S \}$$

A fésülés hatására valóban sok felesleges szimbólum kihullott. Az is kiadódott, hogy a verem a q_2 állapotban ürül ki. A nyelvtan leírásának további

egyszerűsítése érdekében vegyük észre, hogy az A_{22} és B_{22} szimbólumoknak csak egyetlen levezetési szabálya van, így azzal egyszerűsíthetünk, hogy ennek jobboldalát írjuk be mindenüvé, ahol ezek a nemterminálisok levezetési szabályok jobboldalán szerepelnek. Persze ezzel a két érintett szabály el is hagyható.

Amint azonban a fenti két nemterminálist kiküszöböltük, valamennyi indexes nemterminálisnak csak egyetlen állapotpárja, indexkettőse van. Ezért az indexek nélkül is azonosítani tudjuk ezeket a nemterminálisokat.

Végül az $S \rightarrow Z_{02}$ egyszeres szabályt is elhagyhatjuk, ha a jobboldalon szereplő nemterminális helyébe ezen nemterminális levezetési szabályainak jobboldalát írjuk. Mindezeket a rövidítéseket és egyszerűsítéseket végrehajtva nyelvtanunk a következő lesz:

$$S \rightarrow aA \mid bB$$
$$A \rightarrow a \mid aAa \mid bBa$$
$$B \rightarrow b \mid aAb \mid bBb$$

Ami azt illeti, a nyelvtan valóban megkarcsúsodott.

3.6. Determinisztikus veremautomata

Műveletek környezetfüggetlen nyelvekkel

Mielőtt a környezetfüggetlen nyelvekre a nyelvi műveleteket sorra vennénk, ismerkedjünk meg a környezetfüggetlen nyelvek egy fontos tulajdonságával, amelyre egyébként a továbbiakban szükségünk is lesz. Ez a kettős pumpálás, amely remélem, minthogy a reguláris nyelvek pumpálásához hasonló jelenség, mint elnevezés polgárjogot fog nyerni. A szakirodalom nem túl nagy találékonysággal **vwxyz** tételnek nevezi.

A tétel állítása a következő:

Legyen adott egy környezetfüggetlen nyelvtan. Amennyiben sikerül ezen nyelvtan segítségével egy kellő hosszúságú q mondatot generálni, akkor ez a mondat felbontható öt részsorozatra

$$q = vwxyz \tag{3.60.}$$

olymódon, hogy az alábbi jelsorozat

$$vw^{i}xy^{i}z \tag{3.61.}$$

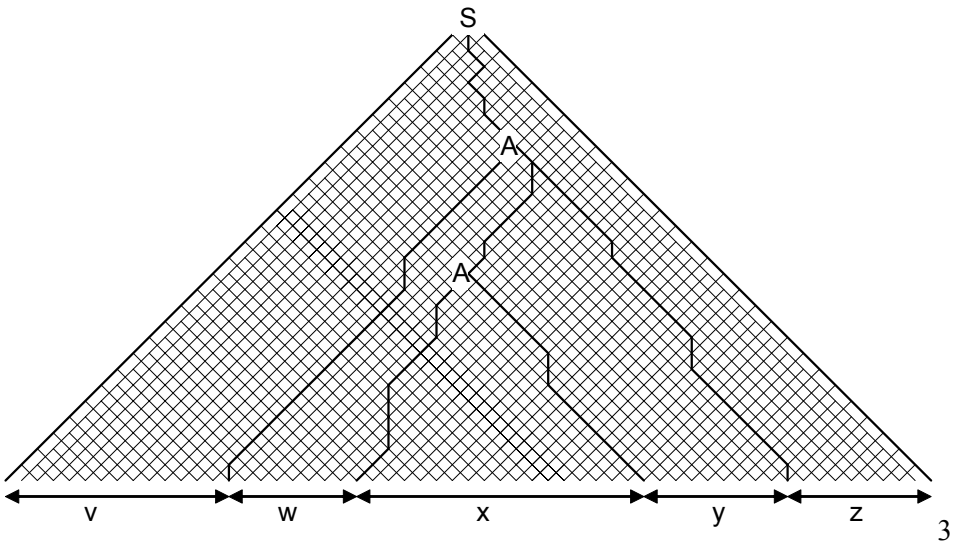
is mondata a nyelvnek.

Ennek igazolására tegyük fel, hogy a nyelvtan nemterminális szimbólumainak száma **n.**

Vegyünk egy olyan mondatot, amelynek levezetési fájában van olyan út, amely a gyökértől a levélig több mint **n** csomópontot tartalmaz. Ez a feltétel adja a kellő hosszúság ismérvét.

Emlékeztetve arra, hogy a levezetési fában a csomópontok a nemterminális szimbólumoknak felelnek meg, kell hogy legyen olyan nemterminális, amely ezen az úton legalább kétszer előfordul

A 3.6. ábrában feltüntettük egy ilyen kellő hosszúságú mondat levezetési fáját. Bejelöltük azt az *A* nemterminális szimbólumot is, amely a fa egyik útján egynél többször előfordul. Megadtuk ezen kívül az egész mondat felosztásának módját.



3

.6. ábra

Természetesen semmi akadálya sincsen annak, hogy az alsóbb szinten helyet foglaló *A* nemterminális ne csak az *x* jelsorozatot generálja, hanem ugyanazt, mint magasabb emeleten helyet foglaló ikertestvére, vagyis a *wxy* jelsorozatot. Más szavakkal megtehetjük, hogy az alsó *A* nemterminálisra ugyanazt a részfát akasztjuk, amilyen pillanatnyilag a felső, hasonló szimbólumon függ.

Ezt a műveletet kellő türelemmel folytatva elérhetjük, hogy a *w* és *y* jelsorozat annyiszor ismétlődjék, amennyiszer csak jónak látjuk.

Az igazolás feltételezte, hogy a nyelvtanból az egyszeres szabályokat kiküszöböltük. Az olvasó ezt – gondolom – minden magyarázat nélkül érzékeli. Ez persze nem korlátozza a tétel érvényességét, hiszen korábban láttuk, hogy minden nyelvtanból készíthetünk egy vele egyenértékű, de egyszeres szabályt nem tartalmazó nyelvtant.

A két *w* és *y* részsorozat közül bármelyik, de egyidejűen nem mindkettő az üres jelsorozat is lehet.

Legyen a nyelvtan levezetési szabályainak leghosszabb jobboldala **s** hosszúságú. Ahhoz, hogy a levezetési fában szükségszerűen találjunk olyan utat, amelyben egy nemterminális legalább kétszer előfordul, **n** különböző szinten kell helyettesítési szabályt alkalmaznunk.

Amennyiben csak **n-1** emeletünk van, akkor nem biztos, hogy van a fában olyan út, amely valamelyik nemterminálist kétszer tartalmazza. Ezek szerint **n-1** emeletes fával még nem generáltunk kellő hosszúságú mondatot. Ha most feltételezzük, hogy minden alkalmazott szabály jobboldala **s** hosszúságú volt, akkor ily módon egy

$$\mathbf{s}^{n-1} \tag{3.63.}$$

mondatot kapunk. Az ennél hosszabb mondatok levezetési fája már biztosan tartalmaz olyan utat, amelyben valamely nemterminális legalább kétszer előfordul, tehát kellő hosszúságú.

Emlékeztetnék arra, hogy az

$$a^{i}b^{i}$$

nyelvről éppen a pumpálási képesség hiánya miatt ismertük fel, hogy az nem reguláris nyelv. A nyelv ugyanakkor – ahogy az egy környezetfüggetlen nyelvhez illik – teljesíti a kettős pumpálás feltételeit, kielégíti a **vwxyz** tételt.

A legelső nyelv, amellyel találkoztunk, az

$$a^{i}b^{i}c^{i} \qquad \mathbf{i} > 0$$

nyelv volt. Az erre a nyelvre megadott nyelvtan, mint az később kiderült, nem csökkentő, azaz első nyelvosztályba tartozó volt. Mostani ismereteink tükrében ez teljesen korrekt lépés volt, hiszen ez a nyelv nem képes kettős pumpálásra, tehát nem lehet környezetfüggetlen nyelv. Ennek a megállapításnak a későbbiekben hasznát vesszük.

Térjünk most át a környezetfüggetlen nyelveken végzett műveletekre, és azok zártságára.

Most is, mint a véges automatáknál kezdjük el a komplemensképzéssel. Az univerzum persze itt is a Σ^* halmaz lesz.

A véges automaták esetében alkalmazott alapgondolat, az állapotok minősítésének felcserélése itt is jó ötletnek látszik. Ennek kapcsán persze olyan veremautomatát kell szem előtt tartanunk, amely állapottal fogad el. Ez a nyelvre nem jelent korlátozást, hiszen minden nyelvhez szerkeszthető állapottal elfogadó veremautomata.

A véges automatáknál végzett vizsgálatokból kitűnt, hogy nemdeterminisztikus automaták esetében problémák merülhetnek fel. Ezért aztán, ha nemdeterminisztikus automatával volt dolgunk, akkor munkánkat azzal kezdtük, hogy egy, az eredeti automatával egyenértékű determinisztikus automatát készítettünk. Tehettük, mert volt erre algoritmusunk.

A veremautomatáknál, mint kitűnik nem ez a helyzet. Mint később igazoljuk, nem készíthető olyan algoritmus, amely tetszőleges nemdeterminisztikus veremautomatából determinisztikusat szerkeszt.

Ugyanis, mint arról már említést tettünk, a veremautomaták világában is definiálhatunk determinisztikus automatát. Emlékeztetőül álljon itt ismét ennek meghatározása.

Egy veremautomata akkor determinisztikus, éspedig mind a valódi, mind az ε-mozgásokra nézve, ha minden esethez, tehát állapot, beolvasott karakter, veremtető hármashoz, illetve ε-mozgások esetében állapot–veremtető párhoz legfeljebb egy mozgási szabály tartozik. Ezen túlmenően a valódi és ε-mozgások között sem lehet átfedés, így ha valamely állapot–veremtető párosra ε-mozgás van definiálva, akkor ez a páros semmilyen beolvasott karakterrel sem alkothat olyan hármast, amelynél valódi mozgás lehetséges.

Ez annyit jelent, hogy nem csak a valódi mozgásokon belül, illetve ε-mozgások között nem jöhet létre konfliktushelyzet, de a valódi és ε-mozgások között sem.

Sajnos a fentiek értelmében vannak olyan nemdeterminisztikus veremautomaták, amelyek nem alakíthatóak át. Az ilyen automatáknál a véges automatáknál bevált módszer nem alkalmazható a komplemens nyelv automatájának megszerkesztésére.

Ennek a problémának a megkerülésére tételezzük fel, hogy veremautomatánk eleve determinisztikus volt. Persze itt tudomásul kell vennünk, hogy ez egy lényeges korlátozást jelent, a továbbiakban csak determinisztikus környezetfüggetlen nyelveket vizsgálunk, vagyis olyanokat, amelyek determinisztikus veremautomatával elfogadhatóak.

A veremautomatáknál gondot okozott az is, ha az automata nem volt teljesen specifikálva. Ezt a betegséget itt is tudjuk kezelni. Vezessünk be itt is egy nem elfogadó csapdaállapotot, és ha valamelyik szituációra nincsen mozgási szabály, akkor erre az esetre hozzunk létre egy olyan mozgási szabályt, amely a csapdaállapotba viszi át az automatát. Ha egyszer beleestünk a csapdába, akkor onnan nem lehet kimászni, hiszen az ismert módszer szerint a csapdában mindent el tudunk olvasni, ezért lesz az automata teljesen specifikált, de ugyanakkor benne maradunk a csapdaállapotban.

A véges automatáknál ennyi előkészület után már megcserélhettük az állapotok minősítését, és ezzel megkaphattuk a komplemens nyelvet elfogadó véges automatát. Sajnos a veremautomaták esetében a helyzet bonyolultabb.

Itt kétféle veszéllyel is szembe kell néznünk.

Az egyik a végtelen ε-ciklus. A teljes specifikáció ugyanis nem zárja ki, hogy az automata ε-mozgások végtelen ciklusába kerüljön. Ilyenkor persze több valódi mozgást, olvasást nem végez, és a jelsorozatot nem tudja végigolvasni, jóllehet az automata állandóan mozgásban van. Az átminősítés ezen nem

változtat, a jelsorozatot sem az eredeti, sem az átminősített veremautomata nem fogadja el.

A másik zavart az okozhatja, hogy a jelsorozat végigolvasása után az automata ε-mozgásokkal bolyongásba kezd, és ezalatt mind elfogadó, mind visszautasító állapotokat érint. Ez a helyzet az átminősítéssel sem változik meg, csupán az elfogadó és visszautasító állapotok cserélnek szerepet. A jelsorozatot mind az eredeti, mind az átminősített automata elfogadja.

Az alábbiakban ezeken a nehézségeken igyekszünk majd eredménnyel úrrá lenni. Az első esetben a végtelen ciklus tényét, bekövetkezését próbáljuk detektálni. Ha ez sikerül, akkor nyert ügyünk van. Ilyenkor ugyanis a ciklusból kilépve csapdaállapotba vihetjük át az automatát, ahol a jelsorozatot végig-olvassuk, és visszautasítjuk.

A második esetben, az utólagos ε-bolyongások alkalmával külön kell választanunk azokat a mozgássorozatokat, amikor az automata érintett, illetve nem érintett elfogadó állapotot.

Kezdjük a végtelen ciklussal. Amennyiben a veremautomata végtelen ε-ciklusba kerül, akkor a verem tartalma vagy minden határon túl növekszik, vagy egy minimum és maximum között lélegzik. A két szindróma különböző gyógykezelést igényel.

Tekintsük először azt az esetet, amikor a verem tartalma minden határon túl növekszik. Bár ebben az esetben is akadhat olyan ε-mozgás, amikor a verem tartalma csökken, magyarán szólva a leemelt szimbólum helyébe nem írunk semmit, összességében a verem növekedik.

Lesznek tehát szükségképpen olyan ε-mozgások, amelyet követően a verem a jelenlegi mélységénél mindig nagyobb lesz, az automata sohasem tér vissza a verem ezen pontjára. Ez annyit jelent, hogy ilyen mozgásoknál a veremnek a veremtető alatti része örök időkre el van temetve, így annak tartalma az automata további működését nem befolyásolhatja. Az ilyen mozgások esetén tehát az egész folyamat további menetét csakis a mozgást kiváltó automata állapot és veremtető alkotta pár határozza meg.

A végtelen mozgássorozat egy ciklusát két ilyen, azonos állapotveremtető párossal kiváltott, soha vissza nem térő veremtartalmú ε-mozgás által határoltnak tekintjük. Az a feltételezés, hogy a verem tartalma minden határon túl növekszik annyit jelent, hogy egy ilyen ciklus alatt a verem tartalma valamelyest nő.

Felmerül a kérdés, hány ε-mozgás lehetséges, és mekkora lehet a verem tartalmának növekedése egy ilyen ciklusban.

Legyen az automata állapotok száma **n**, és a verem alfabetájának számossága pedig **m**. Nyilvánvaló, hogy a ciklust alkotó ε-mozgások számának felső korlátja **n** x **m.**

Ez természetesen az összes lehetséges mozgások száma. Persze ha az ε-mozgások kihasználnának valamennyi lehetőséget, akkor – minthogy az automata determinisztikus, – valódi mozgásokra már nem jutna szabály. Így az előbb megadott érték durva felső becslés.

Legyen az egy mozgás következtében előálló maximális veremtartalom növekedés **s**. Ez a legszószátyárabb szabály által beírt szimbólumok számánál eggyel kevesebb. Ezzel az egy ciklus során mutatkozó tartalom növekedés felső korlátja

$$\mathbf{n} \times \mathbf{m} \times \mathbf{s} \tag{3.64.}$$

Ha azt tapasztaljuk, hogy a veremtartalom növekedése ennél az értéknél nagyobb, akkor bizonyosak lehetünk abban, hogy a vermet egy végtelen ε-ciklus tölti túl.

A fenti eredmény kezünkbe adja annak kulcsát is, hogyan lehet a két szélső állás között ingadozó veremtartalom esetét detektálni.

Az ingadozó verem változó hosszára ugyanis a (3.64.) összefüggés korlátot jelent. A lélegző veremrészlet nagysága így nem lehet tetszőlegesen hosszú.

A legfeljebb (3.64.) szerinti hosszúságú verem viszont csak véges sok különböző tartalmat vehet fel. Ennek számossága az eddigi jelölésekkel:

$$\mathbf{m}^{(n \times m \times s)} \tag{3.65.}$$

Ezzel a nagyon durva becsléssel általában igen nagy, de azért véges számot kapunk. Ha most figyelembe vesszük, hogy ugyanaz a veremtartalom és ugyanaz az automataállapot határol egy ciklust, akkor megadhatjuk azt a legnagyobb lépésszámot, amelyből egy ciklus állhat. Ennek értéke:

$$\mathbf{n} \times \mathbf{m}^{(n \times m \times s)} \tag{3.66.}$$

Ezek szerint az ε-mozgások okozta végtelen ciklus, ha az túltölti a vermet, akkor a növekedés mértékével, ha a verem korlátos marad, akkor a lépésszámmal detektálható.

Éppen ezért az automata állapotába iktassunk be két indikátort, az egyik a verem hosszát, a másik a lépésszámot vigyázza. Minthogy ez az ellenőrzés csak az ε-mozgásokra terjed ki, minden valódi mozgás nullázza ezeket az indikátorokat. Ez az információ beiktatható az automata állapotterébe, hiszen a két indikátor legfeljebb a (3.64.) illetve (3.66.) összefüggésekkel megszabott értékeket veheti fel. Így az eredeti állapottér és a két indikátor direkt szorzata is véges halmazt, állapotteret alkot.

Készítsünk tehát egy olyan veremautomatát, amelynek állapothalmaza az eredeti automata állapothalmazának, és a két indikátor értékkészletének direkt szorzata. Egészítsük ki ezt még egy csapdaállapottal, és detektált végtelen ciklus esetében ebbe a csapdába essünk bele. Az eredeti és az új automata, mint látjuk,

ugyanazt a nyelvet fogadja el, azzal a lényeges különbséggel, hogy míg az előbbi bizonyos jelsorozatok esetén végtelen ciklusba kerül, és a jelsorozatot el sem tudja olvasni, addig az új a végtelen ciklusból kitör, és minden jelsorozatot végigolvas. Az előbbi kényes jelsorozatoknál persze visszautasítja az elolvasott jelsorozatot.

Ezzel az egyik betegségre már megtaláltuk a gyógyszert.

A másik problémát, az elolvasás után az elfogadó és visszautasító állapotok között kószáló automata kérdését a következőképpen oldhatjuk meg.

Készítsünk az eredeti automatából származtatott olyan új veremautomatát, ahol az eredeti automata mindegyik állapotának az új automata négy állapota felel meg. Tekintsük ezeket az eredeti állapot alállapotainak, és a **0, 1, 2** és **3** indexekkel különböztessük meg azokat.

Egy valódi mozgás során az automata vagy a **0** vagy az **1** alállapotot veszi fel, aszerint, hogy az állapot visszautasító vagy elfogadó állapot. Természetesen a „főállapot" vagyis az az állapot, amelynek alállapotairól beszélünk, minden esetben megegyezik az eredeti automata állapotával.

Ha most az automata a valódi mozgást követően ε-mozgásokba kezd, akkor a **0** indexű alállapot addig lesz érvényes, amíg az automata ε-mozgása során csakis visszautasító állapotokat érint. Ha közben egyszer is elfogadó állapotba került, azonnal átvált az **1** indexű alállapotokra, és azt többet az ε-mozgások során nem hagyja el. Ily módon az alállapot memorizálja, érintett-e az utolsó valódi mozgás óta az automata az ε-mozgások során elfogadó állapotot. Itt az elfogadó illetve visszautasító állapot az eredeti automata állapotainak minősítését jelenti.

Ha most az ε-mozgássorozat végén valódi mozgásra kerül sor, akkor ezt megelőzően az új automata még egy ε-mozgást végez, nevezetesen alállapotot vált. Ha az alállapot **0** volt, akkor a **2** alállapotra, ha **1** volt, akkor a **3** alállapotra vált át. Ha most bekövetkezik a valódi mozgás, akkor az egész folyamat a maga alállapotaival újra kezdődik.

A módszer értelme akkor tűnik ki, amikor – bár az automata felkészült a valódi mozgás végrehajtására, – erre nem kerül sor. Ez akkor következik be, amikor az automata a teljes jelsorozatot már elolvasta, és hiába van meg a szándék a valódi mozgás végrehajtására, nincs hozzá karakter az olvasófej alatt. Ilyenkor az automata mozgásképtelenné válik, és ott marad, ahol van, vagyis a **2** alállapotban, ha az utolsó valódi mozgás óta nem érintett elfogadó állapotot, vagy a **3** alállapotban, ha elfogadó állapotot érintett.

Nyilvánvaló, hogy az új automatában a **3** alállapotokat kell elfogadó állapotoknak tekinteni, míg a **2** alállapot visszautasító. A komplemens nyelvet elfogadó automatánál persze a **2** lesz az elfogadó alállapot, a **3** pedig a visszautasító. A **0** és **1** alállapotok mindkét automatánál visszautasítanak. Ez nem okoz

gondot, hiszen a jelsorozat elolvasása után az automatának mindent el kell követnie, hogy elfogadó állapotba kerüljön.

Két megjegyzés.

Azt állítottuk, hogy az eredeti automata minden állapotához négy alállapotot rendelünk. A helyes szöveg legfeljebb négy lett volna, hiszen az elfogadó főállapothoz nem tarozik **0** alállapot, és ennek következtében **2** alállapot sem.

A vázolt megoldással mellékeredményként azt is elérjük, hogy elfogadást követően az automata már nem végezhet több mozgást.

Mint láttuk bár kissé nyögvenyelősen, és nagy árat fizetve, de sikerült a determinisztikus veremautomaták minden betegségét kikúrálnunk, és a komplemens nyelvre egy determinisztikus veremautomatát szerkesztenünk.

Definíció szerint determinisztikusak azok a környezetfüggetlen nyelvek, amelyekre determinisztikus veremautomatát lehet szerkeszteni. Ennek alapján ezt az eredményt úgy is fogalmazhatjuk, hogy a determinisztikus környezetfüggetlen nyelvek zártak a komplemens képzés műveletére.

A többi nyelvi művelettel nem lesz ennyi gondunk. Nézzük két nyelv unióját.

Legyen adott két környezetfüggetlen nyelv nyelvtanával. Amennyiben a két nyelvtan nemterminális szimbólumai nem alkotnak diszjunkt halmazokat, akkor ezt a nemterminálisok átnevezésével érjük el. Legyenek S_1 és S_2 a két nyelvtan mondatszimbólumai.

A két nyelvtan valamennyi levezetési szabályából alkossunk egy új nyelvtant, fosszuk meg az eredeti mondatszimbólumokat ettől a rangjuktól, végül vezessünk be egy új S mondatszimbólumot, és két új levezetési szabályt:

$$S \to S_1 \qquad S \to S_2$$

Vegyük észre, hogy az így kiadódó nyelvtan akkor sem determinisztikus, ha az eredeti két nyelvtan determinisztikus volt.

Triviális, hogy az így konstruált új környezetfüggetlen nyelvtan a két nyelv unióját generálja. A környezetfüggetlen nyelvek tehát zártak az unióképzésre.

Vegyük most két nyelv metszetét. Definiáljunk két nyelvet. Az első mondatainak alakja:

$$a^i b^i c^k$$

a másiké

$$a^i b^k c^k$$

Minkét nyelv determinisztikus környezetfüggetlen nyelv. Erről az olvasó is meggyőződhet, ha szerkeszt hozzájuk determinisztikus veremautomatát. A feladat nem túl nehéz.

Könnyű belátni, hogy a két nyelv metszete

$$a^i b^i c^i$$

amiről nemrég láttuk be, hogy nem környezetfüggetlen nyelv, hiszen nem tud kettősen pumpálni. Ezek szerint a környezetfüggetlen és ezen belül a determinisztikus környezetfüggetlen nyelvek nem zártak a metszet műveletére.

Az a tény, hogy a környezetfüggetlen nyelvek nem zártak a metszés műveletére persze nem jelenti azt, hogy két környezetfüggetlen nyelv metszete minden esetben kilép a környezetfüggetlen nyelvek halmazából. Például, ha egy környezetfüggetlen nyelvet önmagával mint másik környezetfüggetlen nyelvvel elmetszünk, akkor az eredmény nyilván az eredeti nyelv lesz, amely definíció-szerűen környezetfüggetlen. Egy másik triviális példa. Ha két környezetfüggetlen nyelv diszjunkt, akkor metszete az üres nyelv, amely nem csak környezetfüggetlen, hanem reguláris is.

Az az állítás, hogy a környezetfüggetlen nyelvek nem zártak a metszésre, úgy értelmezendő, hogy biztosan lehet találni két olyan környezetfüggetlen nyelvet, amelyeknek a metszete nem környezetfüggetlen. Ami azt illeti mi találtunk ilyet.

Azt nehezen, de igazoltuk, hogy a determinisztikus környezetfüggetlen nyelvek zártak a komplemens képzésre. Nem lehetne ezt az állítást általánosítani, más szóval nem zártak-e a környezetfüggetlen nyelvek általában a komplemens képzésre.

Nem lehet. Azt ugyanis tudjuk, hogy a környezetfüggetlen nyelvek zártak az unióképzésre. Ha a komplemens képzésre is zártak lennének, akkor a *de Morgan* szabályok alapján

$$(A \cup B) = \overline{(\overline{A} \cap \overline{B})} \qquad (A \cap B) = \overline{(\overline{A} \cup \overline{B})}$$

zártaknak kellene lenniök a metszet képzésre is. Erről viszont tudjuk, hogy nem igaz, következésképpen az a feltételezésünk, hogy a környezetfüggetlen nyelvek zártak a komplemens képzésre hamis.

Hasonló módon lehet belátni, hogy a determinisztikus környezetfüggetlen nyelvek nem zártak az unióképzésre. Azt tudjuk, hogy ezek a nyelvek a komplemens képzésre zártak, ha zártak lennének az unióképzésre is, akkor zártak volnának a metszet műveletére is, ami viszont nem helytálló.

Ebből viszont az is következik, hogy a csak nemdeterminisztikus veremautomatával elfogadható környezetfüggetlen nyelvek halmaza nem üres, hiszen kell, hogy legyen két olyan determinisztikus környezetfüggetlen nyelv, amelynek az uniója nemdeterminisztikus. Így a determinisztikus környezetfüggetlen nyelvek halmaza a környezetfüggetlen nyelven halmazának valódi részhalmaza.

Két környezetfüggetlen nyelv konkatenáltja.

Legyen a két nyelv nyelvtanával adott. Gondoskodjunk arról, hogy a két nyelvtan nemterminálisai diszjunkt halmazt alkossanak. Fosszuk meg

mondatszimbólum címétől a két nyelv S_1 és S_2 eredeti mondatszimbólumait, vezessünk be egy új S mondatszimbólumot, egyesítsük a két nyelvtan levezetési szabályait, végül adjuk hozzá az

$$S \to S_1 S_2 \qquad (3.67.)$$

levezetési szabályt. Magyarázat nélkül is érthető, hogy ez a nyelvtan valóban a konkatenált nyelvet generálja. Ezek szerint a környezetfüggetlen nyelvek zártak a konkatenáció műveletére.

Környezetfüggetlen nyelv tranzitív lezártja.

Legyen a környezetfüggetlen nyelv nyelvtanával adott, és mondatszimbóluma legyen *S*. Egészítsük ki a nyelvtan levezetési szabályait két új szabállyal:

$$S \to \varepsilon \qquad S \to SS \qquad (3.68.)$$

Az első szabály gondoskodik arról, hogy az üres jelsorozat eleme legyen a nyelvnek, a második szabály segítségével viszont olyan hosszú csak a mondatszimbólumból álló mondatszerű formát generálhatunk, amilyen hosszút csak jólesik, Ezek mindegyikéből pedig generálható az eredeti nyelvnek egy mondata.

Azonban itt is szükség van óvatosságra. Ha az eredeti nyelvnek az üres jelsorozat nem volt eleme, vagyis nem volt levezethető az *S* mondatszimbólumból, ugyanakkor a mondatszimbólum szerepel levezetési szabályok jobboldalán, vagyis újra megjelenhet mondatszerű formákban, akkor az $S \to \varepsilon$ szabály bevezetésével olyan jelsorozatokat is elfogadhatunk, amelyek sem az eredeti nyelvnek, sem tranzitív lezártjának nem mondatai.

Ennek elkerülésére el kell érnünk, hogy a mondatszimbólum ne szerepeljen levezetési szabály jobboldalán. A módszer – új mondatszimbólum bevezetése, és a régihez tartozó szabályok multiplikálása – betűre megegyezik a reguláris nyelvtanoknál követettel.

Ebből következően a környezetfüggetlen nyelvek zártak a tranzitív lezárt képzésre.

Térjünk vissza még egyszer a metszet műveletére.

Azt tudjuk, hogy két reguláris nyelv metszete reguláris. Az is ismert, hogy két környezetfüggetlen nyelv metszete általában nem környezetfüggetlen. Kérdés mit állíthatunk egy reguláris és egy környezetfüggetlen nyelv metszetéről. A metszet nyelv környezetfüggetlen lesz.

A bizonyítás persze itt is konstruktív, vagyis készítünk egy olyan veremautomatát, amely a metszet nyelvét fogadja el.

A véges automaták esetében láttuk, hogyan lehet két automatát összekapcsolni olymódon, hogy az új automata a két automata nyelvének metszetét fogadja el. Ezt az ötletet itt is alkalmazhatjuk, hiszen a veremautomata magja egy véges automata. Bizonyos óvatossággal egyesíthetjük ezt a magot a reguláris nyelvet elfogadó automatával.

Ha a véges automata illetve veremautomata állapottere Q_M és Q_P akkor a szerkesztett veremautomata állapottere a két állapottér direkt szorzata Q_M x Q_P lesz.

Amennyiben a környezetfüggetlen nyelv veremautomatája valódi mozgást végez, vagyis olvas, az új automata mozgási szabálya a két mozgás összekapcsolásából adódik. Legyenek az eredeti mozgási szabályok:

$$\delta(p, a) = q \quad \delta(s, a, X) = (t, \alpha) \quad p, q \in Q_M \text{ és } s, t \in Q_P \qquad (3.69.)$$

Ebből a két szabályból az új automata következő mozgási szabálya származik:

$$\delta(p\text{-}s, a, X) = (q\text{-}t, \alpha) \qquad (3.70.)$$

Amikor a veremautomata ε-mozgást végez, akkor a véges automata „pihen”, nem változtatja állapotát

$$\delta(s, \varepsilon, X) = (t, \alpha)$$

így ennek a szabálynak az átírása:

$$\delta(x\text{-}s, \varepsilon, X) = (x\text{-}t, \alpha) \qquad \forall x \in Q_M \qquad (3.71.)$$

A szerkesztett automata kezdőállapota természetesen a két automata kezdőállapotainak „direkt” szorzata lesz, míg a kezdő veremszimbólum megegyezik a környezetfüggetlen nyelv veremautomatájának kezdő veremszimbólumával.

Amennyiben az eredeti veremautomata állapottal fogadott el, akkor már készen is vagyunk, a szerkesztett automata is állapottal fogadjon el, és az elfogadó állapotok halmaza a két automata elfogadó állapotainak direkt szorzatából adódik.

Amennyiben az eredeti veremautomata üres veremmel fogadott el óvatosan kell eljárnunk, hiszen ha az eredeti veremautomata kiürítette a vermét, akkor a szerkesztett is ezt teszi, függetlenül attól, hogyan viselkedik az eredeti véges automata. Ha a véges automata nem elfogadó állapotban fejezte be vándorlását, akkor hiába üres a verem, a jelsorozatot nem szabad elfogadni.

Ha az új szerkesztett automatát is üres veremmel elfogadónak képzeljük, akkor a következőképpen járhatunk el. Induláskor a vermet kibéleljük. Amennyiben az eredeti veremautomata elfogadott, vagyis kiürítette a vermét, akkor az új automata vermében már csak ez a bélés marad. Ha ezzel egyidejűen a véges automata is elfogadást mutat, akkor egy ε-mozgással kiemeljük a bélést a veremből, és teljesen kiürítjük azt. Ezzel az automata a jelsorozatot elfogadja.

Ez lényegében megegyezik azzal a módszerrel, amelyet az állapottal elfogadó veremautomatával egyenértékű üres veremmel elfogadó automata szerkesztésekor alkalmaztunk.

Ezekhez a módosításokhoz szükséges mozgási szabályok a következők:

$$\delta(q_0\text{-}s_0', \varepsilon, Z_0) = (q_0\text{-}s_0, Z_0 Z_0') \qquad (3.72.)$$

illetve

$$\delta(x\text{-}y, \varepsilon, Z_0') = (x\text{-}y, \varepsilon) \qquad \forall x \in F_M \wedge \forall y \in Q_P \qquad (3.73.)$$

ahol – F_M a véges automata elfogadó állapotainak halmaza.

Az összefüggések megértéséhez magyarázat nem szükséges.

4. Fordító automaták

4.1. Véges fordítók

Mielőtt a fordító automatákról szót ejtenék, definiálnunk kell mit értünk fordítás alatt. Mint azt korábban megszokhattuk, erre a fogalomra is egy teljesen formális meghatározást adunk, amelynek természetesen mégis van kapcsolata a fordítás köznapi fogalmával.

Legyen adott két véges alfabeta Σ és Δ. Jelölje most is Σ^* illetve Δ^* a fenti alfabetákból alkotott véges, de nem korlátos jelsorozatok halmazát. Képezzük most ezen két megszámlálhatóan végtelen számosságú halmaz direkt szorzatát:

$$\Sigma^* \times \Delta^* \tag{4.1.}$$

A fordítás ennek az egyébként szintén megszámlálhatóan végtelen számosságú halmaznak egy részhalmaza.

Ezen részhalmaz elemei jelsorozatpárok lesznek, ahol az első jelsorozat a Σ^*, a második pedig a Δ^* halmaz eleme. Ha a $\boldsymbol{T}$ részhalmaz ($\boldsymbol{T}$ mint *translation*) tartalmazza az α-η párt, vagyis

$$(\alpha, \eta) \in \boldsymbol{T} \tag{4.2.}$$

akkor az η jelsorozat az α jelsorozat fordítása.

Vegyük észre, hogy sem az egyértelműségre, sem a teljességre nem tettünk semmiféle kikötést.

Ez megegyezésben van a fordításról alkotott eddigi nézeteinkkel. Gondoljunk például egy angol – magyar fordításra. Ebben az esetben a Σ, illetve Δ alfabeta az angol illetve magyar ábécé jelkészlete.

Nyilvánvaló, hogy nem minden jelsorozat képvisel értelmes, és ezért fordítható angol mondatot, így a Σ^* nem minden elemének van fordítása.

Ugyanakkor egy angol mondatnak több magyar egyenértékese lehet, és fordítva különböző angol mondatoknak felelhet meg ugyanaz a magyar mondat.

Ez a talán túlzottan absztrakt definíció, amely valóban lefed minden elképzelhető fordítási esetet, éppúgy teljesen használhatatlan, mint a nyelv teljesen általános meghatározása volt.

Persze vannak nagyon egyszerű fordítások, amelyek formális eszközökkel kezelhetőek.

A legegyszerűbb az izomorfizmus, amikor az egyes szimbólumokat külön-külön írjuk át az egyik alfabetáról a másikra. Ez a helyzet például akkor, amikor valamilyen mondjuk cirill vagy görög betűkkel leírt szöveget írunk át latin betűs alakra. Egy görög példa

$$(\Theta\varepsilon\sigma\sigma\alpha\lambda\iota\alpha, \text{Thessalia}) \in \boldsymbol{T}$$

A fordítás egyszerűsége nem ad semmilyen felvilágosítást a fordítandó nyelv bonyolultságáról. Adott esetben ez a görög nyelv, amelynek leírásához bizonyára legalábbis **0**-ás osztályú nyelvtan lenne szükséges.

A legegyszerűbb fordítóautomata a véges fordító. Ez csak néhány részletben tér el a véges automatától.

A fordító automatáknak nem csak bemeneti, hanem kimeneti berendezésük is van. Így itt az olvasó mellett egy nyomtatót, vagy szalaglyukasztót kell elképzelnünk.

Ami a véges fordító formális leírását és ábrázolását illeti, ez nagyon hasonlít a véges automatáéhoz.

Természetesen az állapotok száma itt is véges. A bemenő alfabeta mellett azonban itt egy kimenő alfabeta is van. Az automata működése során ugyanis a bemeneti szöveg olvasásával egyidejűen kiadja a kimeneti berendezésen az olvasott szöveg fordítását.

A mozgási szabályok annyiban bővebbek, hogy a mozgás eredménye kapcsán nem csak az automata vesz fel egy új állapotot, hanem kiadhat egy korlátos hosszúságú jelsorozat fragmenst is. Ha egy véges fordító egy mozgás során kiírt jelsorozatainak maximális hossza **k**, akkor a mozgások az alábbi leképezést adják:

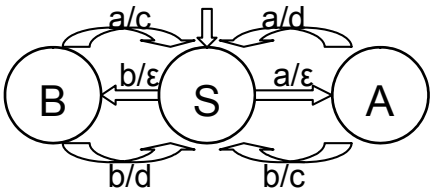
$$Q \times \Sigma \rightarrow Q \times \Delta^k \qquad (4.3.)$$

ahol – Q az állapotok halmazát,
– Σ és Δ a bemenő illetve kimenő alfabetát,
– Δ^k a Δ alfabeta feletti, legfeljebb **k** hosszúságú jelsorozatok halmazát jelenti.

A fordító automatának nincsenek megkülönböztetett elfogadó állapotai. Nem a fordító automata feladata ugyanis a fordítandó szöveg helyességének ellenőrzése.

Az előbbi, a görög ábécéről latinra átíró, és véges fordítóval magától értetődően megoldható fordítás esetében nem az automata feladata a bemeneti szöveg görög voltának megállapítása.

A 4.1. ábra egy véges fordítót tüntet fel, amely *a* és *b* szimbólumokból álló jelsorozatokat alakít át *c* és *d* szimbólumokat tartalmazó jelsorozatokká. Az eredeti *a* és *b* szimbólumokból álló jelsorozatot kettesével párosítva foglalja össze, és ha a pár azonos szimbólumokat tartalmaz, *d*, ellenkező esetben *c* szimbólumot ad ki.

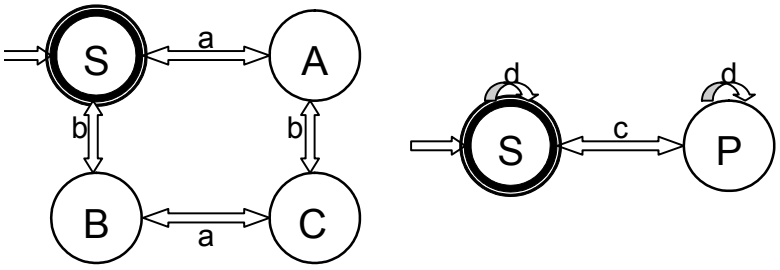


4.1. ábra

Az ábrában a mozgásnak megfelelő nyilakon most nem csak az olvasott, hanem a kiírt szimbólumot is feltüntettük. A két szimbólumot a / jel választja el. Természetesen, ahol ε szerepel akár mint olvasott, akár mint kiírt szimbólum, ott nem történik olvasás, illetve kiírás.

Ha például a bemenő nyelvnek azt a nyelvet használjuk, amelyben mind az *a* mind a *b* karakterek száma páros, akkor olyan nyelvet kapunk, ahol a *c* szimbólumok száma páros, a *d* szimbólumoké tetszőleges.

Ez esetben mind a bemenő, mind a kimenő nyelv reguláris nyelv. A 4.2. ábrán egymás mellett tüntettük fel a bemenő, illetve kimenő nyelvet elfogadó automatákat.



4.2. ábra

Az, hogy a fordítás eredménye is reguláris nyelv nem véletlen. A fordító automatákkal végzett művelet egy nyelvből egy másikat készít, a kimenet nyelvét. Ezt felfoghatjuk úgy, mint nyelvi műveletet, amely egy nyelvhez egy másikat rendel.

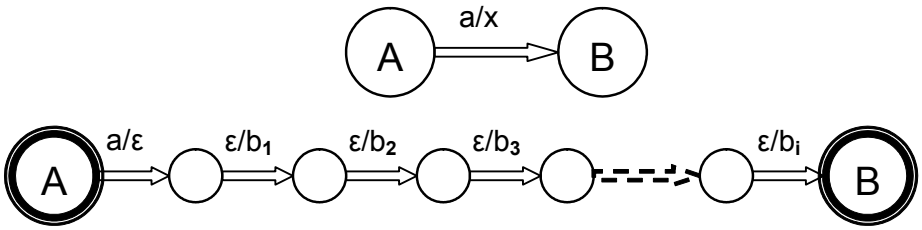
Amennyiben a fordító automata egy véges fordító, akkor az automatával végrehajtott transzformációra mind a reguláris, mind a környezetfüggetlen nyelvek zártak. Így, ha a bemenet nyelve reguláris illetve környezetfüggetlen volt, az lesz a kimenet nyelve is.

Igazoljuk ezt az állításunkat a környezetfüggetlen nyelvek esetére. A bizonyítás – mint megszokhattuk, – ismét konstruktív lesz, vagyis a bemenő nyelv automatájából és a véges fordítóból megszerkesztjük a kimenet nyelvét elfogadó automatát.

Esetünkben, minthogy környezetfüggetlen nyelvről van szó a bemenet és a kimenet nyelvét elfogadó automaták veremautomaták.

Legyen tehát adott a bemenő nyelvet elfogadó veremautomata, és a véges fordító. Alakítsuk át a fordító automatát, ha szükséges, olymódon, hogy egy mozgás során vagy csak egyetlen szimbólumot olvas, vagy csak egyetlen szimbólumot ír ki. Így a módosított fordító mozgási szabályai vagy *a*/ε vagy ε/*x* alakúak. Ez nem okoz problémát. Ha az eredeti automata egy mozgása során egy szimbólumot olvasott, és **i** szimbólumot írt ki, akkor ennek hatását **i+1** sorba kapcsolt olyan mozgással tudjuk leképezni, amelyből egy csak olvas, **i** pedig egy-egy karaktert kiír.

Tételezzük fel, hogy a fordítónak van *a*/*x* mozgása, ahol $x=b_1b_2...b_i$. Az átalakítást a 4.3. ábra szemlélteti.



4.3. ábra

Persze ez az átalakítás azzal jár, hogy a fordító automatát óvatosabban kell kezelnünk. Tételezzük fel, hogy az eredeti fordító automatában egyik él az *a*/*x* feliratot viselt, ahol *x* egy **i** hosszúságú jelsorozat volt. Ennek a mozgásnak végrehajtása során az automata beolvasta az *a* karaktert, és a teljes *x* sorozatot kiírta. Az új automata sem viselkedhet másként, annak ellenére, hogy ezt az egyetlen mozgást itt **i+1** mozgás sorozataként végezzük el. Ez annyit jelent, hogy ha az automata egyszer elindult ezen a soros mozgásláncon, akkor azt végig kell járnia, nem állhat meg útközben. A számítástechnikában az olyan műveletsorozatot, amelyet vagy nem hajtunk végre, vagy ha igen, akkor teljes egészében, mégpedig úgy, hogy közben más műveletet nem kezdünk el, primitívnek nevezik. Az *a*/*x* művelet tehát primitív.

Ezek szerint lesznek az új automatának olyan állapotai, ahol befejezheti működését, más állapotokban viszont nem állhat le. A kétféle állapotot meg kell különböztetnünk. Kézenfekvőnek látszik az a megoldás, amikor a megállásra engedélyezett állapotokat kettős körrel emeljük ki. Az automata rajza szerint tehát mégis lesznek elfogadó, és visszautasító állapotok, pontosabban itt ezt úgy kell értelmezni, hogy bizonyos állapotokban abba lehet hagyni a működést, másokban viszont végig kell menni a következő megállóhelyig.

Tételezzük fel, hogy a véges fordító automatán elvégeztük ezt a beavatkozást, tehát az a mozgások során egyetlen karaktert vagy olvas, vagy ír, és a megállás tekintetében különbséget teszünk az állapototok között.

Ami a bemenő nyelv veremautomatáját illeti, feltesszük, hogy az egyetlen állapottal bíró, és üres veremmel elfogadó automata. Ha eredetileg nem ilyen lett volna, átalakítjuk. Emlékeztetőül az ilyen automatát a nyelvtanból úgy lehet származtatni, hogy a bemeneti Σ alfabeta minden *a* szimbólumára létesítünk egy

$$\delta(q, a, a) = (q, \varepsilon) \qquad \forall\, a \in \Sigma$$

mozgási szabályt, míg minden $A \to \alpha$ alakú levezetési szabályának egy

$$\delta(q, \varepsilon, A) = (q, \alpha) \qquad \forall\, A \to \alpha \in \delta$$

alakú szabályt.

Az utóbbi szabályokkal generáljuk a mondatot, míg a nyelvtantól független szabályokkal ellenőrizzük, hogy a generált szöveg megegyezik-e a bemenettel.

Az új veremautomata állapothalmaza a véges fordító állapothalmaza lesz. Pontosabban a két automata állapotainak direkt szorzatát kellene vennünk, de minthogy szerencsénkre az eredeti veremautomatának csak egyetlen állapota van, megengedhetjük magunknak azt a lazaságot, hogy a veremautomata állapotát egyszerűen elhagyjuk. A véges fordítónak az átalakítás után lesznek olvasó és kiíróállapotai.

Legyenek a véges fordító állapotai $p_0, p_1, p_2, \ldots p_n \in P$. A veremautomata mondatgeneráló mozgási szabályait megtartjuk azzal a kiegészítéssel, hogy ezek a szabályok minden állapotban alkalmazhatóak:

$$\delta(p_i , \varepsilon, A) = (p_i , \alpha) \qquad \forall p_i \in P \tag{4.4.}$$

A koncepció szerint az új veremautomata is generálja ezekkel a szabályokkal a veremben a fordítandó szöveget. Ha a verem tetején terminális szimbólum található, akkor ez a fordítandó szöveg következő karaktere. Ezt kell tulajdonképpen elolvasnia a véges fordítónak, és ennek során állapotot váltania. Ezt úgy képezhetjük le, ha a véges fordító nem a bemenetről, hanem a verem tetejéről olvas. Figyelembe véve, hogy az új veremautomata állapotai követik a véges fordító állapotváltozásait, az eredetileg olvasó mozgások átírása a következő lesz:

$$\delta(p_i , a) = (p_k , \varepsilon) \Rightarrow \delta(p_i , \varepsilon, a) = (p_k , \varepsilon) \tag{4.5.}$$

A véges fordító másik szabályrendszere eredetileg a kimenetet, a fordítást írta le. Minthogy ilyenkor az automata nem olvas ezek tulajdonképpen ε-szabályok.

$$\delta(p_u , \varepsilon) = (p_v , b) \tag{4.6.}$$

A véges fordító eredetileg a fordított, kimeneti nyelv egy mondatát adta ki. Minthogy feladatunk a kimeneti nyelvet elfogadó automata készítése, az automatának éppen ezt a mondatot, amelyet a véges fordító kiad, kell elfogadnia. Vagyis azt kell biztosítanunk, hogy az olvasófej alatt azok a karakterek jelenjenek meg, amelyeket a véges fordító kiadna. Ebből már látható az átírás metodikája. Az új veremautomata persze nem ír ki semmit, nem is tudna, hiszen nincs is kiíró berendezése, Viszont össze kell vetni az olvasófej alatti és a véges automata által kiadni szándékozott szöveget. Ennél a műveletnél a verem tetejének értéke, amely a fordítandó szöveget tartalmazza, közömbös.

$$\delta(p_u , \varepsilon) = (p_v , b) \Rightarrow \delta(p_u , b, X) = (p_v, X) \tag{4.7.}$$

Végül is sikerült elérnünk, hogy az új veremautomata azokat a jelsorozatokat fogadja el, amelyet a véges fordító a bemeneti, fordítandó nyelv mondatainak fordításaként kiad.

Még egy veszélyt kell elhárítanunk. Az algoritmusból következik, hogy amikor a véges fordító elolvasta a bemenetet, akkor az új veremautomata verme kiürül. Ez bajt okoz. Ilyenkor ugyanis az automata már nem tud lépni, és ha a

véges fordító éppen megállni tilos állapotban van, akkor el kellene jutnia az első megállóig, különben a fordított szöveg végét nem olvassa el.

A már korábban is alkalmazott módszerrel induláskor kibéleljük a vermet, így amikor a verem egyébként kiürülne, a bélés még benne marad. Az automata tehát tovább folytatja munkáját addig, amíg egy megállóhelyhez nem ér. Ezekben az állapotokban a bélés eltávolítható, és a verem teljesen kiüríthető. Ezért van szükség erre a bélelésre:

$$\delta(p_0', \varepsilon, Z_0) = (p_0 , Z_0\check{Z}_0) \tag{4.8.}$$

A megállóhelyeken a következő mozgási szabályokkal üríthetjük ki a vermet:

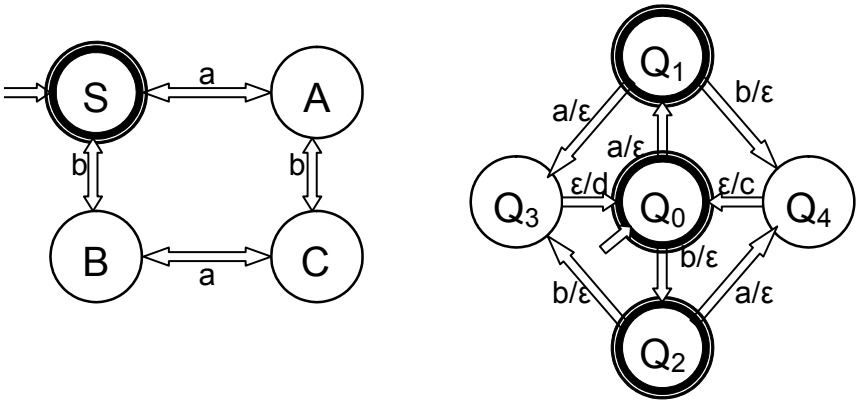
$$\delta(p_i, \varepsilon, \check{Z}_0) = (p_i, \varepsilon) \qquad \forall p_i \in P_S \tag{4.9.}$$

ahol – P_S a véges fordító megállásra engedélyezett állapotainak halmaza.

Ezzel igazoltuk, hogy környezetfüggetlen nyelv véges fordítása veremautomatával elfogadható, tehát szintén környezetfüggetlen nyelv.

Foglaljuk még egyszer össze a gondolatmenetet. A veremben most is a fordítandó szöveget generáljuk. Ezt azonban nem a bemenettel vetjük össze, hanem követjük, milyen állapot változásokon megy át ezt olvasva a véges fordító, hiszen veremautomatánk állapottere azonos a véges fordító állapotterével. Amikor a véges fordító a fordított mondat valamilyen szimbólumát készülne kiírni, akkor ezt összevetjük a bemeneten található szimbólummal. Ha a bemeneten lévő jelsorozat megegyezik a fordításként kiadni szándékolt jelsorozattal, akkor az automata elfogadja azt.

Reguláris nyelvekre csupán egy példát adunk, amelyből a teljesen hasonló gondolatmenetű bizonyítás rekonstruálható.



4.4. ábra

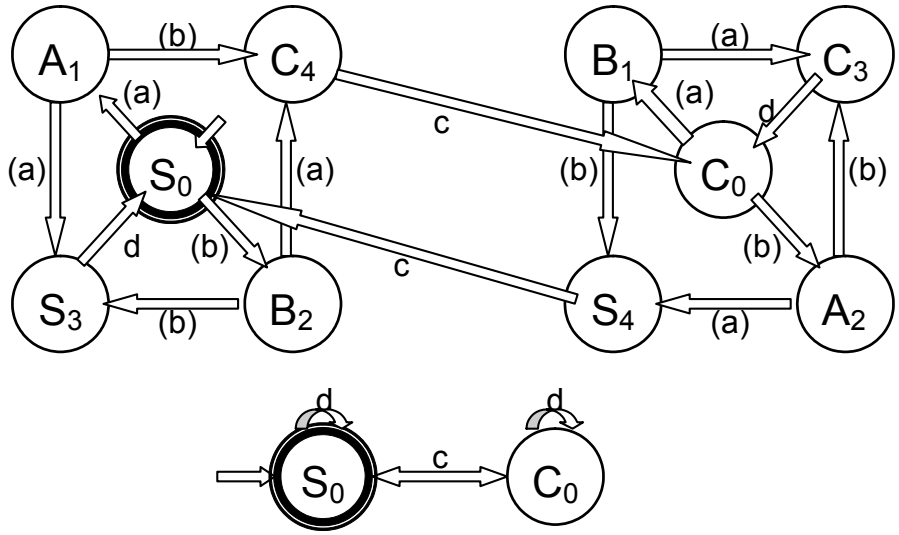
A 4.4. ábra a bemenő nyelv véges automatáját, és a véges fordítót mutatja. A nyelv az olyan *a* és *b* karakterekből álló jelsorozatokból áll, ahol mind az *a* mind a *b* karakterek száma páros.

A fordító is régi ismerősünk. A bemenő nyelv szimbólumait párosával összefogja, és ha két szimbólum különbözik *c*, ha megegyezik *d* szimbólumot ír ki. Minthogy az eredeti véges fordító néhány oszthatatlan, primitív művelete itt fel van aprózva, a megállásra engedélyezett és tiltott állapotokat meg kellett különböztetni.

Az új szerkesztett véges automata állapothalmaza a véges automata és véges fordító állapothalmazának direkt szorzata lesz. A már megismert elvnek megfelelően az eredeti véges automata szimbólumait párosítjuk össze a véges fordító bemeneti szimbólumaival, míg a fordításként kiadódó szöveg lesz az új automata bemenete.

Az így megszerkesztett, és a véges automata éleihez megtévesztésig hasonló párosított élekhez természetesen nem kapcsolódik valódi mozgás. Hasznuk csupán annyi, hogy általuk követhető a fordítandó nyelvet elfogadó, és a fordítást végző automaták együttműködése.

A 4.5. ábra ezt az egyesített automatát tünteti fel.



4.5. ábra

Az ábrában a zsúfoltság elkerülésére az állapotokat indexelt betűvel jelöltük. A betű a fordítandó nyelvet elfogadó automatára, az index pedig a fordítóra utal. Elfogadó csak az az állapot lehet, amely elfogadó illetve megállásra engedélyezett állapot direkt szorzataként jön létre. Ilyen állapot csupán az S_0.

A párosított éleken zárójelben feltüntettük a bemenő nyelvnek azt a karakterét, amelyet az azt elfogadó véges automata és a véges fordító egyidejűen elemez. Mint tisztáztuk, ezen élek mentén nem történik olvasás.

Az ilyen lényegében ε-mozgásokkal összekötött állapotokat összevonhatjuk. Ekkor viszont – mi más történhetne – megkapjuk a 4.2. ábrán már egyszer amúgy ösztönösen felrajzolt véges automatát.

4.2. Szintakszisvezérelt fordítási sémák

A fordításoknak a véges fordítónál lényegesen nagyobb osztálya jellemezhető szintakszisvezérelt fordítási sémákkal.

Itt nem a kész szöveg alapján készül el a fordítás, hanem a két mondat, fordítandó és fordított, egyidejűen generálódik. Ez annyit jelent, hogy a levezetés során a mondatszimbólumból kiindulva párhuzamosan lépünk mondatszerű formáról mondatszerű formára, pontosabban mondatszerű formapárról mondatszerű formapárra.

A szintakszisvezérelt fordítási sémák elemei a helyettesítési szabályoknak megfelelő, és tulajdonképpen két ilyen szabályból álló konstrukciók. A szabályok környezetfüggetlen jellegűek, vagyis baloldaluk egyetlen nemterminális szimbólum. Alakjuk:

$$A \rightarrow \alpha_1,\alpha_2 \qquad (4.10.)$$

Ez lényegében az $A \rightarrow \alpha_1$ és az $A \rightarrow \alpha_2$ alakú helyettesítési szabály egyesítése.

Az ilyen szabálypárokat úgy alkalmazzuk, hogy a párt alkotó mondatszerű formák első, a fordítandó szöveget szolgáltató tagjában az $A \rightarrow \alpha_1$, a másik tagban pedig, amely a fordított szöveget generálja, az $A \rightarrow \alpha_2$ helyettesítést végezzük el.

Természetesen a sémaelemeknek olyanoknak kell lenniök, hogy ez a párhuzamos helyettesítés elvégezhető legyen. Ennek az a feltétele, hogy a két mondatszerű formában ugyanazok a nemterminálisok és ugyanannyiszor szerepeljenek. Ennek a megkötésnek minden mondatszerű formapárra teljesülnie kell.

Ez csak úgy biztosítható, ha a sémák elemeihez tartozó két jobboldalban a nemterminálisok nemben és számban megegyeznek.

Egy további megkötést is figyelembe kell venni a szintakszisvezérelt fordítási sémák használatánál. A párt alkotó mondatszerű formában minden nemterminálisnak egy és csakis egy párja lehet a másik mondatszerű formában.

Ha a mondatszerű formában minden nemterminális különböző, akkor ez a megkötés további intézkedés nélkül teljesül. Amennyiben vannak azonos nemterminálisok, akkor azokat úgy kell párosítani, hogy az egy időben keletkezettek alkossanak egy párt.

Ha a szintakszisvezérelt fordítási sémának van olyan eleme, ahol valamelyik nemterminális egynél többször szerepel a jobboldalon, akkor egyértelműen definiálni kell, hogyan állnak párba az azonos nemterminálisok.

Lássunk egy példát a szintakszisvezérelt fordítási sémákra.

Ismeretes az aritmetikai kifejezések, a megszokott infixnél sok szempontból szerencsésebb prefix illetve posztfix lengyel jelölése. Itt nincsen precedencia az operátorok között, így zárójelek sincsenek, hiszen ezek a precedencia felülbírálására lennének hivatottak.

Amennyiben itt is csak az összeadás és szorzás műveletére szorítkozunk, akkor az alábbi nyelvtan az aritmetikai kifejezéseket posztfix lengyel alakban generálja.

$$E \to EE+ \quad E \to EE* \quad E \to a$$

Készítsünk olyan szintakszisvezérelt fordítási sémát, amely az aritmetikai kifejezéseket infix jelölésben generáló kedvenc nyelvtanunk mondatait posztfix lengyel jelölésbe írja át. Íme a séma:

$E \to E+T$	$ET+$
$E \to T$	T
$T \to T*F$	$TF*$
$T \to F$	F
$F \to (E)$	E
$F \to a$	a

Sajnos itt kénytelenek voltunk a posztfix lengyel jelölés nyelvtanát kissé komplikáltabbá tenni. Ugyanis – mint tisztáztuk – a fordítási sémák két jobboldalán azonos nemterminális elemeknek kell lenniök. Hasonló szabály a terminálisokra természetesen nem vonatkozik.

Minthogy az előzőek szerint a lengyel jelölésben nincsenek zárójelek, az infix jelölés zárójeleit nemes egyszerűséggel elhagytuk.

Állapítsuk meg mi az alábbi infix jelölésben megadott aritmetikai kifejezés posztfix lengyel alakja.

$$(a+a)*a$$

Felhasználva a szintakszisvezérelt fordítási sémát, állítsuk elő a fenti infix jelölésű aritmetikai kifejezést. A párhuzamos generálás következtében ezen művelet közben óhatatlanul megkapjuk a kifejezés posztfix lengyel formáját is.

Induljunk ki az { *E, E* } mondatszimbólumpárból. A példa kidolgozása során a nemterminális szimbólumokat – néhol feleslegesen is – indexszel láttuk el. Ezt kizárólag annak érdekében tesszük, hogy ettől a párosítás mindig követhető legyen.

Ugyancsak indexszel láttuk el az azonosító értelmű *a* terminális szimbólumot, hogy az aritmetikai műveletekben szereplő mennyiségek nyomon követhetőek legyenek.

$\{ E_1, E_1 \} \Rightarrow \{ T_1, T_1 \} \Rightarrow \{ T_2*F_1, T_2F_1* \} \Rightarrow \{ F_2*F_1, F_2F_1* \} \Rightarrow$
$\Rightarrow \{ (E_2)*F_1, E_2F_1* \} \Rightarrow \{ (E_3+T_3)*F_1, E_3T_3+F_1* \} \Rightarrow$
$\Rightarrow \{ (T_4+T_3)*F_1, T_4T_3+F_1* \} \Rightarrow \{ (F_3+T_3)*F_1, F_3T_3+F_1* \} \Rightarrow$
$\Rightarrow \{ (a_1+T_3)*F_1, a_1T_3+F_1*) \} \Rightarrow \{ (a_1+F_4)*F_1, a_1F_4+F_1* \} \Rightarrow$
$\Rightarrow \{ (a_1+a_2)*F_1, a_1a_2+F_1* \} \Rightarrow \{ (a_1+a_2)*a_3, a_1a_2+a_3* \}$

Az infix kifejezés fordítása, vagyis ugyanazon aritmetikai kifejezés posztfix alakja – ahogy illik – kiadódott.

A fordítási sémák végrehajthatóságának elemzése kapcsán egy sereg elég szigorú megkötést tettünk az azokban szereplő nemterminális szimbólumokra vonatkozóan. Egy szót sem szóltunk azonban a terminális szimbólumokról.

Nyilvánvaló, hogy ezekre semmiféle megkötést sem kell tennünk. Azonosságuk vagy különbözőségük, meglétük vagy hiányuk nem befolyásolja a szintakszisvezérelt fordítási sémák végrehajthatóságát.

A szintakszisvezérelt fordítási sémák között megkülönböztetett figyelmet érdemelnek azok a sémák, ahol a párt alkotó mondatszerű formákban a nemterminális szimbólumok nem csak nemben és számban egyeznek meg, hanem a két összetartozó mondatszerű formában még sorrendjük is azonos. Az ilyen tulajdonságú sémák neve egyszerű szintakszisvezérelt fordítási séma.

Visszatérve példánkra látható, hogy az infix jelölést posztfix jelöléssé átalakító fordítás egyszerű szintakszisvezérelt fordítási sémával valósítható meg.

Az egyszerű szintakszisvezérelt fordítási sémák jelentősége abban van, hogy ezek működését automatával lehet realizálni. Ezt majd a következő pontban tárgyaljuk.

Vizsgáljuk meg most azt a kérdést, hogy hogyan lehet leírni egy mondat levezetését. Számozzuk meg a helyettesítési szabályokat, és írjuk le abban a sorrendben ezeket a sorszámokat, amelyben az egyes szabályok alkalmazásra kerültek. Tekintsük ezt a számsorozatot a szintaktikus elemzés leírásának.

Az egyértelműség kedvéért itt azt is meg kell mondanunk, hogy milyen elemzésről van szó. Bár vannak más elemzési módok, itt csak kétféle elemzésre szorítkozunk.

Amikor a mondatszimbólumból kiindulva, a baloldali levezetés lépéseit követve jutunk el az elemzendő mondathoz, akkor felülről lefelé történő, *top-down* elemzésről, illetve balelemzésről beszélünk.

A másik elemzési módnál a mondatból indulunk ki, és a nyelek már bemutatott letörése útján redukáljuk a mondatot, illetve a mondatszerű formát, ha lehet a mondatszimbólumig. Ilyenkor – mint azt megmutattuk a 107.-108.

oldalon – a jobboldali levezetést követjük, csak fordított sorrendben. Ebben az esetben alulról felfelé, vagy *bottom-up* elemzésről, jobbelemzésről van szó. A jobbelemzés kifejezésben tehát a *jobb* előtag nem az olvasás irányára utal, amely egyébként mindkét elemzési módnál a nekünk megszokott irányban balról jobbra történik, hanem az így kapott mondatszerű formákból álló sorozat jellegére.

Az eljárás demonstrálására számozzuk be ismert nyelvtanunk helyettesítési szabályait.

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

Legyen a mondat ismét:

$$(a+a)*a$$

Ennek a mondatnak a baloldali levezetését, vagyis a balelemzés eredményét a következő számsorozat jellemzi:

23451246466

Ugyanígy a jobbelemzés eredménye:

64264154632

Az előbbiek szerint ez a jobboldali redukálást, vagyis fordított sorrendben a jobboldali levezetést adja.

Az első eredmény azonnal kiadódik, ha az előbbi példát áttekintjük. Ott ugyanis a baloldali levezetés alapján nyertük a fenti mondatot.

A jobboldali redukálás helyességének ellenőrzése sem okozhat túl nagy gondot, és a szorgalmas olvasónak kifejezetten kellemes feladat.

Mivel mindkét levezetésnél a levezetési fa azonos – ugyanazokat a levezetési szabályok kerültek alkalmazásra, csak különböző időpontokban – a két levezetés leírásában ugyanazok a számok szerepelnek, csak más sorrendben.

Ezeket az eredményeket mechanikusan is megkaphatjuk, a szintakszisvezérelt fordítási séma felhasználásával. Tekintsük ugyanis fordításnak azt a transzformációt, amelyik egy mondatból annak szintaktikus elemzését állítja elő, és szerkesszünk ehhez egy szintakszisvezérelt fordítási sémát.

Legyen adott egy nyelvtan, és legyenek helyettesítési szabályai számozottak. Keressünk most minden szabályhoz egy párt, amivel a szintakszisvezérelt fordítási séma elemei kiadódnak.

Legyen ez egyszerű szintakszisvezérelt fordítási séma. Ezzel eldőlt, hogy a szabályok párjaiban milyen sorrendben szerepeltessük a nemterminális szimbólumokat. Minthogy végeredményben a levezetést leíró számsorozatot kívánjuk megkapni, terminális szimbólumként ezek a sorszámok szerepeljenek, mégpedig olymódon, hogy minden helyettesítési szabály „fordítási” felében a saját sorszáma áll terminálisként.

Ezek a sorszámok alkotják terminálisként az adott sorrendű nemterminálisokkal együtt a helyettesítési szabálypárját. Azt kell csak meggondolnunk, hová írjuk ezt a sorszámot. Két önmagát kínáló lehetőségünk van, eléje írjuk, vagy utána.

Maradjunk továbbra is kedvenc nyelvtanunknál. Ez a két lehetőség az alábbi szintakszisvezérelt fordítási sémákat szolgáltatja:

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

Ha nincsen a jobboldalon nemterminális, akkor a párban csakis a sorszám szerepel.

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

Ha most ezekkel a szintakszisvezérelt fordítási sémákkal generáljuk az aritmetikai kifejezéseket, akkor az első séma esetében a baloldali, a második használatakor a jobboldali elemzést, pontosabban annak leírását kapjuk meg.

Ennek igazolása érdekes időtöltést jelenthet az olvasónak.

4.3. Veremfordító

A véges fordítóval való megismerkedés után aligha okoz nagy meglepetést az a tény, hogy a veremautomatának is van fordító változata.

Vegyük sorra ismét, miben tér el a veremfordító a jól ismert veremautomatától Természetesen a bemeneti egységen kívül itt is szükség van egy kimeneti egységre, mondjuk nyomtatóra.

A veremfordító mozgásainál, éppúgy, mint a véges fordító esetében, az automata egy korlátos jelsorozatot adhat ki ezen a kimeneten. Minthogy csak véges számú mozgás van, a kiadott jelsorozat hosszának van felső korlátja, a legbőbeszédűbb mozgás által nyomtatott szöveg hossza, legyen ennek mérete **i**.

Legyen a fordító bemenő alfabetája Σ és a kimeneté Δ, és az ismert módon tekintsük az üres jelsorozatot is a Δ alfabeta elemének, akkor a veremfordító mozgási szabályai az alábbi leképezést jelentik:

$$Q \times (\Sigma \cup \varepsilon) \times \Gamma \Rightarrow Q \times \Gamma^{k} \times \Delta^{i} \qquad (4.11.)$$

ahol a megszokott jelölésekkel Q az automata állapothalmaza, Σ, Γ és Δ pedig rendre a bemenet, a verem és a kimenet alfabetája, míg **k** illetve **i** a mozgások során a verembe beírt, illetve a kimeneten megjelenő jelsorozat legnagyobb hossza.

Ezzel egy valódi illetve ε-mozgást leíró szabály:

$$\delta(q, a, X) = (p, \alpha, \eta) \quad \delta(q, \varepsilon, X) = (p, \alpha, \eta) \qquad (4.12.)$$

ahol α a Γ^{k} az η pedig a Δ^{i} halmaz eleme.

A következőkben kimutatjuk, hogy az egyszerű szintakszisvezérelt fordítási sémák, és a veremfordítók által definiált fordítások ugyanazt a fordítási osztályt alkotják.

A bizonyítás – ez már nem meglepetés – konstruktív. Legyen adott egy egyszerű szintakszisvezérelt fordítási séma. Szerkesszünk hozzá egy vele egyenértékű, vagyis ugyanazt a fordítást realizáló veremfordítót.

Legyen az egyszerű szintakszisvezérelt fordítási séma egy eleme, szabálypárja:

$$A \rightarrow x_0B_1x_2B_2x_3 \ldots B_nx_n\,,\; y_0B_1y_1B_2y_2 \ldots B_ny_n \qquad (4.13.)$$

Minthogy a szintakszisvezérelt fordítási séma egyszerű, a $B_1,B_2 \ldots B_n$ nemterminálisok szimbólumok mindkét jobboldalban ugyanabban a sorrendben követik egymást.

Az $x_0, x_1, \ldots x_n$ illetve $y_0, y_1, \ldots y_n$ jelsorozatok a Σ bemeneti illetve Δ kimeneti alfabeták karaktereiből álló, esetleg üres jelsorozatok.

Amennyiben a bemenet és kimenet alfabetája, tehát a Σ és a Δ halmaz nem diszjunkt, akkor definiáljunk egy olyan Δ' halmazt, amely egyrészt diszjunkt a Σ halmazzal, másrészt a Δ és a Δ' halmaz elemei között kölcsönösen egyértelmű megfeleltetés áll fenn.

Ekkor a (4.13.) szerinti szabálypárhoz a következő mozgási szabályt rendeljük:

$$\delta(\, q,\, \varepsilon,\, A\,) = (\, q,\, x_0y_0{}'B_1x_1\,y_1{}'B_2x_2y_2{}'. \ldots B_nx_n\,y_n{}'\,) \qquad (4.14.)$$

ahol az $y_0{}',y_1{}' \ldots y_n{}'$ jelsorozatok a Δ' halmaz elemeiből képzett, és rendre a $y_0,y_1, \ldots y_n$ jelsorozatoknak megfelelő jelsorozatok.

A (4.14.) szabály szerint tehát, ha a verem tetején nemterminális szimbólum van, akkor a szintakszisvezérelt fordítási séma szabálypárjainak valamilyen kombinációját írjuk a verembe. Itt messzemenően felhasználtuk azt a körülményt, hogy a két jobboldalon a nemterminális szimbólumok sorrendje azonos, vagyis egyszerű szintakszisvezérelt fordítási sémánk van.

Megjegyzem ez nem az egyetlen lehetőség a mozgási szabályra, ugyanis az

$$x_iy_i{}'$$

jelsorozatpárok bármelyike felcserélhető, sőt szimbólumaik össze is keverhetőek azzal a megkötéssel, hogy az x_i és az $y_i{}'$ jelsorozatok szimbólumai külön-külön tartsák meg sorrendjüket. Persze elegendő a sok lehetőség közül egyet, például a (4.14.) szabályban leírtat alkalmazni.

Arra az esetre, amikor a verem tetejére Σ halmazbeli szimbólum kerül, a következő mozgási szabály lesz érvényes:

$$\delta(\, q,\, a,\, a\,) = (\, q,\, \varepsilon,\, \varepsilon\,) \qquad (4.15.)$$

vagyis a verem tetején található szimbólum „rekombinálódik” a beolvasottal, és sem a verembe, sem a kimenetre nem írunk sem be, sem ki semmit.

Amennyiben a verem tetejére egy Δ' halmazbeli szimbólum kerül, akkor az ennek megfelelő szabály a következő lesz:

$$\delta(\, q,\, \varepsilon,\, b' \,) = (\, q,\, \varepsilon,\, b \,) \qquad (4.16.)$$

vagyis egy ε-mozgás során a Δ' halmazbeli szimbólum Δ halmazbeli párját írjuk ki a kimenetre.

Ez a megoldás nem tartalmaz sok új gondolatot. A (4.14.) típusú szabályok mind a fordítandó, mind a fordított szöveg terminálisait beírják a verembe. Persze a megfelelő nemterminálisok elé, közé illetve mögé.

A legelső, tehát legbaloldalibb nemterminális előtt álló terminális szimbólumokat aztán a (4.15.) és (4.16.) alakú szabályok dolgozzák fel. A bemenő alfabeta szimbólumainál ellenőrzik, hogy valóban a várt jelsorozat van-e a bemeneten, a kimenő alfabeta esetében pedig a megfelelő szimbólum kinyomtatódik.

A működésmódból az is világos, hogy ez a veremfordító éppen a kiindulásul választott szintakszisvezérelt fordítási sémával meghatározott fordítást realizálja.

Gyakorlásképpen írjuk fel az infix jelölésből posztfix lengyel jelölésbe fordítást leíró egyszerű szintakszisvezérelt fordítási sémából származtatott veremfordító mozgási szabályait. Kezdjük most a szintakszisvezérelt fordítási séma elemeitől függő szabályokkal:

$$\delta(\, q,\, \varepsilon,\, E \,) = (\, q,\, E+T+',\ \varepsilon \,) \qquad \delta(\, q,\, \varepsilon,\, E \,) = (\, q,\, T,\, \varepsilon \,)$$
$$\delta(\, q,\, \varepsilon,\, T \,) = (\, q,\, T*F*',\, \varepsilon \,) \qquad \delta(\, q,\, \varepsilon,\, T \,) = (\, q,\, F,\, \varepsilon \,)$$
$$\delta(\, q,\, \varepsilon,\, F \,) = (\, q,\, \mathbf{(}E\mathbf{)} \,.\, \varepsilon \,) \qquad \delta(\, q,\, \varepsilon,\, F \,) = (\, q,\, aa',\, \varepsilon \,)$$

Következzenek azok a szabályok, amelyeket akkor alkalmazunk, ha bemenő terminális van a verem tetején:

$$\delta(\, q,\, +,\, + \,) = (\, q,\, \varepsilon,\, \varepsilon \,) \qquad \delta(\, q,\, *,\, * \,) = (\, q,\, \varepsilon,\, \varepsilon \,)$$
$$\delta(\, q,\, \mathbf{(},\, \mathbf{(} \,) = (\, q,\, \varepsilon,\, \varepsilon \,) \qquad \delta(\, q,\, \mathbf{)},\, \mathbf{)} \,) = (\, q,\, \varepsilon,\, \varepsilon \,)$$
$$\delta(\, q,\, a,\, a \,) = (\, q,\, \varepsilon,\, \varepsilon \,)$$

Végül adjuk meg azokat a szabályokat, amelyek akkor érvényesek, amikor a verem tetején Δ' halmazbeli szimbólum jelenik meg:

$$\delta(\, q,\, \varepsilon,\, +' \,) = (\, q,\, \varepsilon,\, + \,) \qquad \delta(\, q,\, \varepsilon,\, *' \,) = (\, q,\, \varepsilon,\, * \,)$$
$$\delta(\, q,\, \varepsilon,\, a' \,) = (\, q,\, \varepsilon,\, a \,)$$

Az eredmény annyira áttekinthető, hogy a példa kidolgozását bátran az olvasóra bízhatom.

A bizonyítás visszája, vagyis annak igazolása, hogy minden veremfordítóhoz szerkeszthető egy egyszerű szintakszisvezérelt fordítási séma, már korántsem ilyen könnyű. Lényegében ezzel a problémával egyszer már megszenvedtünk, amikor igazoltuk, hogy minden veremautomatához rendelhető egy környezetfüggetlen nyelvtan.

Még egyszer ne keveredjünk ilyen részletekbe, hanem csupán lényegében a korábbival azonos gondolatmenetet ismertetjük.

A nehézséget persze itt is az jelenti, hogy a veremfordítónak több állapota is lehet, így itt a nyelvtan, pontosabban az egyszerű szintakszisvezérelt fordítási séma nemterminálisaiban nem csak a veremszimbólumokra, hanem a veremfordító állapotaira vonatkozó információt is tárolnunk kell. Az elgondolás és megoldás, és sajnos a munka terjedelme ugyanaz, mint volt a veremautomatánál.

A verem tartalmát, és az abból készített nemterminálisokat most is minden állapot kombinációban elő kell állítani. Mégpedig két példányban, hiszen a szintakszisvezérelt fordítási sémának két jobboldala van. A megfeleltetett nemterminálisok nem csak azonos veremszimbólumra vonatkoznak, hanem a hozzájuk rendelt állapotpár is azonos kell legyen.

Ha valamely mozgási szabályban olvasunk, akkor a fordítandó szöveget generáló jobboldal elé kell írni a szimbólumot. Ez betűre megegyezik a veremautomatánál követett eljárással.

Amennyiben a mozgási szabály kiírásról intézkedik, akkor a fordított szöveget generáló jobboldal elé írjuk a kiírandó jelsorozatot.

Ha történetesen valamelyik szabály olvas is meg ki is ír, akkor a neki megfelelő fordítási séma szabálypárjának mindkét tagja prefixálva lesz, az első az olvasott, a második a kiírt szöveggel.

4.4. Jellemző nyelvtanok

A szintakszisvezérelt fordítási sémák kapcsán tulajdonképpen két összekapcsolt nyelvtannal határoztunk meg egy fordítást.

Felmerülhet a kérdés nem lehet-e a fordítást egyetlen nyelvtannal, egyetlen nyelvvel jellemezni? Kis segítséggel, két homomorfizmus alkalmazásával, ez a kérdés sokszor megoldható.

A homomorfizmus, mint eddigi tanulmányainkból ismeretes, egy halmaz elemeihez egy másik halmaz elemeit rendeli. A mi speciális esetünkben ez annyit jelent, hogy egy alfabeta, egy jelkészlet elemeihez egy másik alfabeta elemeit rendeli. Ez a hozzárendelés adja az első alfabeta elemeinek homomorf képét. Egy jelsorozat homomorf képét úgy kapjuk, hogy a jelsorozat karaktereinek homomorf képéből alkotunk egy jelsorozatot. Amennyiben ez a leképezés kölcsönösen egyértelmű, akkor izomorf leképezésről beszélünk, ami egyszersmind persze homomorfizmus.

A homomorfizmust két oldalán álló alfabetának nem kell diszjunktnak lennie, sőt az is megengedett, hogy a két alfabeta azonos legyen. Ha ilyen esetben a homomorfizmus minden karakternek önmagát felelteti meg, akkor ez az identikus leképezés, a homomorfizmus, pontosabban izomorfizmus speciális esete.

A görög ábécé átírása például olyan homomorfizmus, amely egyúttal izomorfizmus. Jelölje h a homomorfizmus által definiált függvényt, akkor a példánkban említett görög-latin átírás felírható:

$$h(\alpha) = a \qquad h(\beta) = b \qquad h(\gamma) = g \ldots .$$

A jellemző nyelvtan meghatározása az eddigi fogalmak felhasználásával a következő.

Legyen adott **G** grammatikájával egy ***L*** nyelv, és két homomorfizmus: h_1 és h_2. Az ***L*** akkor jellemzi a ***T*** fordítást, ha a nyelv minden w mondatának a fenti két homomorfizmussal történő leképezése éppen a fordítás egy elemét adja, ugyanakkor a fordítás valamennyi eleméhez található a nyelvnek olyan z mondata, hogy a fordítandó és fordított jelsorozat éppen a z mondatnak a két homomorfizmussal való leképezése.

$$\boldsymbol{T} = \{ h_1(w), h_2(w)) \mid w \in \boldsymbol{L} \} \qquad (4.17.)$$

Amennyiben a fenti feltételek teljesülnek, akkor a **G** grammatika, illetve az ***L*** nyelv a ***T*** fordítás jellemző nyelvtana, illetve nyelve.

Ha az ***L*** nyelv alfabetája Π, míg a fordítandó illetve fordított nyelv alfabetája Σ illetve Δ, akkor h_1 és h_2 homomorfizmus a Π alfabetát képezi le a Σ, illetve Δ alfabetára.

$$h_1 : \Pi \rightarrow \Sigma \qquad h_2 : \Pi \rightarrow \Delta \qquad (4.18.)$$

Mint látható, a nyelvtan illetve nyelv, és a két homomorfizmus nem választható el egymástól. Egy adott nyelvtan csakis az adott homomorfizmus pár alkalmazásával lesz jellemző nyelvtan.

Lássunk erre egy példát. Legyen a fordítás a következő szintakszisvezérelt fordítási sémával meghatározott:

$$S \rightarrow xS, yS \qquad S \rightarrow yS, xS \qquad S \rightarrow x, y \qquad S \rightarrow y, x$$

A fordítás ezek szerint az x és y szimbólumokból álló jelsorozatokat viszi át az ugyancsak x és y karakterekből szimbólumokat tartalmazó olyan jelsorozatokba, ahol a két szimbólum szerepe az eredetihez képest meg van cserélve.

Feltéve, hogy a két választott homomorfizmus

$$h_1(a) = x \qquad h_1(b) = y \qquad h_2(a) = y \qquad h_2(b) = x$$

akkor az alábbi nyelvtan jellemzi a fordítást:

$$S \rightarrow aS \mid bS \mid a \mid b$$

Az is megengedett továbbá, hogy a homomorfizmus valamely szimbólumot az üres jelsorozatba vigyen át. Ilyenkor a leképezés során ez a szimbólum nyom nélkül eltűnik.

Amennyiben a két homomorfizmus olyan, hogy a nyelv terminálisait vagy az egyik, vagy a másik leképezés az üres jelsorozatba viszi át, akkor az ilyen homomorfizmusok mellett érvényes jellemző nyelvtant a fordítás szigorúan jellemző nyelvtanának mondjuk.

Az előbbi megfogalmazásban a *vagy* kitételt kizáró értelemben illik felfogni, ha ugyanis egy terminálist mindkét homomorfizmus eltüntet, akkor minek szerepel a nyelvben egyáltalában.

Az előbbi példában megadott homomorfizmusok nem ilyen szigorú homomorfizmusok. Ennek következtében nyelvtanunk csak úgy közönségesen, és nem szigorúan jellemzi a fordítást.

Persze választhattunk volna másik, szigorú jellemzést biztosító homomorfizmust is. Például:

$h_1(a) = x$ $h_1(b) = y$ $h_1(c) = \varepsilon$ $h_1(d) = \varepsilon$

$h_2(a) = \varepsilon$ $h_2(b) = \varepsilon$ $h_2(c) = y$ $h_2(d) = x$

Ezek mellett a homomorfizmusok mellett az alábbi nyelvtan jellemzi, és mást már szigorúan jellemzi a fordítást:

$$S \to acS \mid bdS \mid ac \mid bd$$

Amennyiben egy fordításnak van jellemző nyelvtana, akkor van szigorúan jellemző nyelvtana is.

Ezt könnyű belátni, és igen egyszerű egy jellemző nyelvtanból és a hozzátartozó két homomorfizmusból két más, alkalmas homomorfizmust, és egy most már szigorúan jellemző nyelvtant konstruálni.

Legyen adott egy jellemző nyelvtan homomorfizmusaival. Keressük meg azokat a terminális szimbólumokat, amelyek egyik homomorfizmus esetében sem tűnnek el. Ilyen biztosan akad, hiszen ellenkező esetben már eleve szigorúan jellemző nyelvtannal lett volna dolgunk.

Ezen terminálisok helyébe írjunk két új terminálist, amelyek közül az első leképezése az első homomorfizmussal az eredeti eredményt adja, míg a második homomorfizmus ezt a terminálist megsemmisíti. A második terminálisnál pont fordítva járunk el. Itt az első homomorfizmus tünteti el a terminálist.

A jellemző nyelvtanok bevezetésével két fontos megállapítást tehetünk.

Amennyiben egy fordítást egy reguláris nyelvtan jellemez, akkor ez a fordítás véges fordítóval megvalósítható, és viszont, minden véges fordítóval megadott fordításnak van reguláris jellemző nyelvtana.

Ha egy fordítást egy környezetfüggetlen nyelvtan jellemez, akkor ez leírható egy egyszerű szintakszisvezérelt fordítási sémával, és következésképpen megvalósítható veremfordítóval. Itt is igaz az állítás megfordítása, vagyis minden veremfordítóval megadott fordításnak van környezetfüggetlen jellemző nyelvtana.

Nézzük először az első tételt.

Jellemezze a fordítást egy reguláris nyelvtan, és legyen h_1 és h_2 a fordítandó, illetve fordított szövegre leképező homomorfizmus. Szerkesszük meg a véges fordítót olymódon, hogy a nyelvtan minden nemterminálisának a véges fordító egy állapota feleljen meg. A mondatszimbólumnak a kezdőállapot a megfelelője. Minthogy a jellemző nyelvtan csak egész mondatok fordítását értelmezi,

ezek az állapotok „non stop”, vagyis lényegében visszautasító állapotok lesznek. Vezessünk be ezeken kívül egyetlen elfogadó állapotot is.

Legyen a jellemző nyelvtan egy szabálya: $A \to aB$

Ekkor kössük össze az automata A és B állapotait egy az utóbbi állapotba mutató, és

$$h_1(a) / h_2(a) \qquad (4.19.)$$

jelzésű nyíllal.

A jellemző nyelvtan $A \to a$ alakú szabályainak átírásánál ugyanezt a módszert alkalmazzuk, csak most a második állapot szerepét az egyetlen elfogadó állapot játssza.

Ennek alapján gondolom, nem okoz problémát annak belátása, hogy az ily módon kialakított véges fordító éppen a jellemző nyelvtan és a homomorfizmusok által meghatározott fordítást valósítja meg.

Szerkesszünk most jellemző nyelvtant egy véges fordítóhoz.

Előbb alakítsuk át a véges fordítót olymódon, hogy az egyes mozgások során az automata csak egyetlen szimbólumot olvasson, vagy írjon. Ennek részleteit már megbeszéltük.

Legyen Σ illetve Δ a fordítandó, illetve fordított nyelv alfabetája. Amennyiben ezek nem diszjunktak, akkor – szintén ismert módon – vezessük be a Δ’ halmazt, amely a Σ halmazra nézve diszjunkt, és amelyből a Δ halmaz izomorfizmussal származtatható.

A véges fordító olvasó illetve kiíró éleinek egy-egy helyettesítési szabály felel meg, amelynek alakja:

$$A \to aB \qquad \text{illetve} \qquad A \to b'B$$

ahol A és B az él induló illetve érkező állapota, míg a az olvasott szimbólum, és b' pedig a kiírtnak megfelelő Δ’ halmazbeli szimbólum. Rendeljünk ezen kívül még minden „stop” állapothoz egy-egy

$$A \to \varepsilon$$

alakú ε-szabályt.

Legyen a két homomorfizmus:

$$h_1(a) = a \qquad h_1(b') = \varepsilon$$
$$h_2(a) = \varepsilon \qquad h_2(b') = b$$

A jellemző nyelvtan és a homomorfizmusok által definiált, illetve a véges fordítóval megvalósított fordítás azonossága itt is nyilvánvaló.

Második állításunk, amely a környezetfüggetlen nyelvtanok és a veremfordítók közötti megfeleltetésről szól, még könnyebben belátható.

A fordító veremautomaták és az egyszerű szintakszisvezérelt fordítási sémák megfeleltetéséről már beszéltünk. Így elegendő, ha itt csak a környezetfüggetlen nyelvtanok és az egyszerű szintakszisvezérelt fordítási sémák közötti kapcsolatról szólunk.

Megegyezésben a (4.13.) és (4.14.) összefüggésekben foglaltakkal, a szintakszisvezérelt fordítási séma elemeihez tartozó levezetési szabály a következő lesz:

$$A \to x_0 B_1 x_1 B_2 \ldots B_n x_n,\ y_0 B_1 y_1 B_2 \ldots B_n y_n \Rightarrow$$
$$A \to x_0 y'_0 B_1 x_1 y'_1 B_2 \ldots B_n x_n y'_n \qquad (4.20.)$$

ahol az $y'_0 y'_1 \ldots y'_n$ jelsorozatok az $y_0, y_1 \ldots y_n$ Δ^* halmazbeli jelsorozatok Δ'^* halmazbeli párjai.

Fordítva, ha adott a jellemző nyelvtan, és vele együtt a homomorfizmusok, akkor a nyelvtan egy szabályát a következő módon kell az egyszerű szintakszisvezérelt fordítási séma elemévé átírni:

$$A \to z_0 B_1 z_1 B_2 \ldots B_n z_n$$
$$A \to h_1(z_0) B_1 h_1(z_1) B_2 \ldots B_n h_1(z_n),\ h_2(z_0) B_1 h_2(z_1) B_2 \ldots B_n h_2(z_n) \qquad (4.21.)$$

A megfeleltetés helyessége itt sem igényel bővebb magyarázatot.

Befejezésül egy megjegyzés. A fentiekből következik, hogy nem egyszerű szintakszisvezérelt fordítási sémákkal meghatározott fordítást általában nem lehet veremfordítóval megvalósítani.

5. Szintaktikus elemzők

5.1. Általános elemzők, bal- és jobbelemezhetőség

Szintaktikus elemző tágabb értelemben minden olyan automata, amelynek feladata a mondat, pontosabban minden mondatnak vélt jelsorozat levezetési fájának előállítása. Így szintaktikus elemzők azok a veremfordítók, amelyek a mondatra mint bemenetre a levezetési fa leírását adják meg, mint fordítást.

Szűkebb értelemben csak azokat az automatákat tekintjük szintaktikus elemzőknek, amelyeknek működése determinisztikus.

A szintaktikus elemzés igénye majd mindig egyértelmű nyelvtanok esetében lép fel. Legtöbbször a szintaktikus elemző szerkesztésekor feltételezzük, és megköveteljük, hogy a nyelvtan egyértelmű legyen. Vannak azonban olyan elemzők, amelyek nem támasztanak a nyelvtannal szemben semmiféle követelményt. Mielőtt az egyértelmű nyelvtanokra szorítkoznánk, két ilyen, általánosan használható elemzőt ismertetnénk.

Az egyik az *Earley* algoritmus, a felülről lefelé, a másik a *Coke-Younger-Kasami* módszer az alulról felfelé elemzés elvét követi.

Az *Earley* algoritmus alapgondolata a következő. A mondatszimbólumból indulunk ki, és valamennyi lehetséges levezetést vizsgálat tárgyává teszünk. Amennyiben a levezetés során a mondatszerű forma elejére egy terminális szimbólum generálódik, akkor ellenőrizzük, hogy ez a terminális szimbólum megegyezik-e az elemzett mondatjelölt első terminális szimbólumával. Eltérés esetén nyilvánvaló, hogy ezen az úton a szóban forgó jelsorozatot nem lehet levezetni, éppen ezért ezeket a levezetési próbálkozásokat abbahagyjuk.

Csak abban az esetben érdemes továbbvinni az elemzést, ha a generált terminális megegyezik a mondatjelölt megfelelő helyén álló terminálisával. Ekkor viszont a levezetéseket minden lehetséges módon tovább kell folytatnunk.

Újabb és újabb terminálisokat generálva egyre több szimbólumát vesszük figyelembe az elemzett jelsorozatnak, miközben egyes levezetési utakat mint reményteleneket abbahagyunk, másokat viszont több útra ágaztatunk szét. Ha sikerül eljutnunk a jelsorozat utolsó szimbólumáig, akkor találhatunk olyan levezetést, amely ezt a jelsorozatot generálja. Ez esetben a jelsorozatról bebizonyosodott, hogy mondat, és annak levezetését, esetleg levezetéseit megállapíthatjuk.

Az algoritmus gyakorlati megvalósításakor úgynevezett elemekből – angol nevük *item* – csoportokat alakítunk ki. A csoportok számozottak, és sorszámuk azt fejezi ki, hogy a mondatjelöltből már hány szimbólumot vettünk figyelembe. Amennyiben a jelsorozat **n** karakterből áll, akkor a csoportok számozása **0** és **n** között változik. A **0** sorszámú csoport azt az állapotot tükrözi, amikor még egyetlen karaktert sem vettünk figyelembe.

Egy elem alakja a következő:

$$\{ A \rightarrow \alpha_1 \bullet \alpha_2, \mathbf{i} \} \qquad (5.1.)$$

Feltételezve, hogy ez az elem a **k** sorszámú csoportban van, az elem jelentése, szemantikája a következő.

Az elem alakjából következik, hogy a nyelvtanban kell lennie egy

$$A \rightarrow \alpha_1 \alpha_2$$

alakú levezetési szabálynak. Ez a szabály felhasználható a jelsorozat levezetésekor, mégpedig olymódon, hogy a jelsorozatnak az **i** sorszámú szimbólumától a **k** sorszámúig terjedő fragmense, – nem beleértve az **i,** de beleértve a **k** sorszámút, – a levezetési szabály α_1 részéből levezethető. Az elem nem zárja ki az $A \rightarrow \alpha_1 \alpha_2$ szabálynak a levezetés során máshol és másképp történő alkalmazását.

Fontos, hogy az elemek szemantikáját pontosan megértsük, mert ennek alapján lesz világos az a mechanizmus, amellyel a már ismert csoportokból az újabb csoportok elemeit megszerkeszthetjük.

Új elemeket háromféle módon származtathatunk. Tekintsük a **k-1** sorszámú csoportot, és vizsgáljuk azokat az elemeket, amelyekben a pont után terminális szimbólum áll. Ha ezek között van olyan, ahol a pont után álló terminális megegyezik a jelsorozat következő, tehát **k** sorszámú karakterével, akkor ezt a szimbólumot a pont elé helyezve a születendő **k** sorszámú csoport egy elemét kapjuk.

Formálisan, ha a

$$\{ B \rightarrow \beta_1 \bullet b \beta_2 , \mathbf{i} \}$$

elem a **k-1** sorszámú csoport tagja, és a mondatjelölt **k** sorszámú karaktere *b,* akkor a

$$\{ B \rightarrow \beta_1 \, b \bullet \beta_2 , \mathbf{i} \} \qquad (5.2.)$$

tagja, mondhatni alapító tagja az új **k** sorszámú csoportnak.

A származtatás jogosultsága eléggé nyilvánvaló. Ha a β_1 jelsorozat valamely szimbólumtól kezdve egészen a **k-1** sorszámú karakterig generálja az elemzett mondatjelöltet, akkor ehhez hozzávéve a mondatjelölt **k** sorszámú karakterét, a kettő együtt nyilván a **k** szimbólumig generál.

A **k-1** sorszámú csoportnak azok az elemei tehát, amelyekben a pont után a mondatjelölt **k** sorszámú terminálisa áll, alkotják az előbbi kismértékű átalakítással az új csoport alapító tagjait, elemeit.

Ez adja az új elemek származtatásának első, időben is első módszerét.

Amennyiben a pont mögött „rossz” terminális áll, úgy azok a levezetések, amelyeket ezek az elemek képviselnek, fiaskónak bizonyulván, a további játékban nem vesznek részt.

Az így kapott „alapító” tagok sorsa más és más, aszerint, hogy most a pont után terminális szimbólum, nemterminális szimbólum vagy semmi sem áll.

Amennyiben az alapító elemekben a pont után terminális található, akkor pillanatnyilag ezen elemek szerepe befejeződött. Rájuk majd akkor kerül újra sor, ha a következő, **k+1** sorszámú csoport alapító tagjait keressük.

Nem bizonyos azonban, hogy az elemzett jelsorozat következő eleme ezek közül a pont után álló terminálisok közül kerül ki. Lehet, hogy ezt a karaktert egy olyan nemterminális generálja, amely egy másik elemben a pont mögött áll, és a levezetés folytatásakor egy terminálist, pontosabban egy terminálissal kezdődő részsorozatot generál. Éppen ezért mindegyik pont után álló nemterminálisból származtatható karaktert figyelembe kell vennünk.

Tételezzük most fel, hogy a **k** sorszámú csoport alapító tagjai között szerepel a következő elem:

$$\{ C \rightarrow \gamma_1 \bullet D\gamma_2, \mathbf{i} \} \tag{5.3.}$$

és a nyelvtannak van egy

$$D \rightarrow \eta$$

alakú helyettesítési szabálya. Ekkor az alábbi elemet is be kell vennünk ebbe a csoportba:

$$\{ D \rightarrow \bullet\eta, \mathbf{k} \} \tag{5.4.}$$

Ennek szemantikája a következő.

Az (5.4.) elem pont előtti jelsorozata – minthogy a pont előtt semmi sem áll, ez természetesen az üres jelsorozat – alkalmas arra, hogy a mondatjelöltnek a **k** sorszámú karakter után kezdődő, és a **k** sorszámú karakterrel bevégződő fragmensét, amely nyilvánvalóan az üres jelsorozat, generálja. Ez más szavakkal azt állítja, hogy az üres jelsorozat alkalmas az üres jelsorozat generálására. Tulajdonképpen ez nem meglepő.

Ennek ellenére a fenti megállapítás nem tautológia, és nem olyan trivialitás, mint aminek az első pillanatban látszik. Ugyanis nem minden levezetési szabály jobboldala előtt álló üres jelsorozat alkalmas arra, hogy ott, azon a helyen egy üres jelsorozatot generáljon, hanem csak azok, amelyek az adott helyen bekapcsolódhatnak a mondat levezetésébe.

Amennyiben az új csoportba így bevezetett új elemekben a pont után megint csak nemterminális áll, – magyarán a szabály jobboldalának első szimbóluma nemterminális, – akkor a fenti származtatást rekurzív módon mindaddig folytatni kell, amíg új, eddig nem szerepelt elemeket fűzhetünk hozzá a csoporthoz. Ez annak az esetnek felel meg, amikor az alapító elemben a pont után álló nemterminális nem egyetlen lépésben, hanem egy lépéssorozattal generálja a következő terminális szimbólumot.

Ez az új elemek hozzáfűzésének második lehetősége.

Vegyük végül azt az esetet, amikor a pont után már nem áll semmi, vagyis egy levezetési szabály jobboldalát teljesen felhasználtuk.

Legyen például az újonnan kapott **k** csoportbeli elem

$$\{ A \rightarrow \alpha\bullet, \mathbf{j} \} \qquad (5.5.)$$

Ekkor vissza kell mennünk ahhoz a csoporthoz, ahol a szóban forgó helyettesítési szabály bekapcsolódott a mondat levezetésébe. Ez nyilván a **j** sorszámú csoport lesz, hiszen az említett levezetési szabály a **j** sorszámú szimbólumtól kezdve generálja a mondat egy fragmensét.

Könnyű belátni, hogy a **j** sorszámú csoportban kell legyen legalább egy olyan elem, amelyik az (5.5.) összefüggésben szereplő helyettesítési szabályt inicializálta, vagyis ahol a pont után az *A* nemterminális szerepel. Legyen a **j** sorszámú csoport egy ilyen elemének alakja:

$$\{ B \rightarrow \beta_1\bullet A\beta_2, \mathbf{i} \} \qquad (5.6.)$$

Ennek alapján az új, **k** sorszámú csoportba a következő elemet kell felvennünk:

$$\{ B \rightarrow \beta_1 A\bullet\beta_2, \mathbf{i} \} \qquad (5.7.)$$

A származtatás jogosságát itt is könnyű belátni. Ha ugyanis a β_1 jelsorozat képes a mondat **i** és **j** sorszámú szimbólumai közötti fragmens generálására, ugyanakkor az *A* nemterminálisból levezethető a mondat **j** és **k** sorszámú karakterei közötti részsorozat, akkor a $\beta_1 A$ alkalmas az **i** és **k** közötti teljes jelsorozat generálására.

Ezzel az új elemek létrehozásának alapját képező mindhárom szabályrendszert ismertettük. Így ha már van valamennyi csoportunk, a következő csoport elemeit már létre tudjuk hozni. Hogyan induljunk azonban el, hogyan hozzuk létre az első csoportot?

Az első, sorszám szerint **0** csoport alapító elemeit azok a levezetési szabályok alkotják, amelynek baloldala a mondatszimbólum, hiszen minden levezetésnek ebből kell kiindulnia. Így az összes $S \rightarrow \sigma$ alakú helyettesítési szabály a

$$\{ S \rightarrow \bullet\, \sigma, \mathbf{0} \} \qquad (5.8.)$$

formában jelenik meg a legelső csoport elemei között. Az alapító elemek birtokában aztán meg kell kezdeni a csoport felépítését az ismertetett módon.

Ezzel az *Earley* algoritmus valamennyi formális szabályát megtárgyaltuk.

Meg kell még beszélnünk, mikor fogadjunk el egy jelsorozatot. Természetesen akkor, ha az a mondatszimbólumból teljes egészében generálható. Ezt úgy ismerhetjük fel, hogy az utolsó csoportban kell legyen egy

$$\{ S \rightarrow \sigma\, \bullet, \mathbf{0} \} \qquad (5.9.)$$

alakú elem. Minthogy ez az elem az utolsó csoportban van, ennek az elemnek a szemantikája a következő: az $S \rightarrow \sigma$ szabály segítségével a jelsorozat **0** sorszámú karakterétől kezdődően az utolsó karakterig generálható, más szóval a jelsorozat a mondatszimbólumból levezethető, tehát mondat.

Lássunk erre egy példát. Szándékosan balrekurzív nyelvtant választottunk, hogy a módszer erre érzéketlen voltát kihangsúlyozzuk. Generálja a nyelvtan az egyszerűsített hátsó lengyel jelölést

$$E \rightarrow EE+ \mid EE* \mid a$$

Elemezzük a következő mondatot:

$$aaa+a*+$$

Az áttekinthetőség növelése érdekében az egyes csoportok fejlécén megadtuk a mondat megfelelő szimbólumát. A **0** sorszámú csoporthoz természetesen az ε szimbólum tartozik.

0 - ε

$\sim\{ E \rightarrow \bullet EE+, \mathbf{0} \}$
$\{ E \rightarrow \bullet EE*, \mathbf{0} \}$
$\sim\{ E \rightarrow \bullet a, \mathbf{0} \}$

1 - *a*

$\sim\{ E \rightarrow a\bullet , \mathbf{0} \}$
$\sim\{ E \rightarrow E\bullet E+, \mathbf{0} \}$
$\{ E \rightarrow E\bullet E*, \mathbf{0} \}$
$\sim\{ E \rightarrow \bullet EE+, \mathbf{1} \}$
$\sim\{ E \rightarrow \bullet EE*, \mathbf{1} \}$
$\sim\{ E \rightarrow \bullet a, \mathbf{1} \}$

2 - *a*

$\sim\{ E \rightarrow a\bullet, \mathbf{1} \}$
$\{ E \rightarrow EE\bullet+, \mathbf{0} \}$
$\{ E \rightarrow EE\bullet*, \mathbf{0} \}$
$\sim\{ E \rightarrow E\bullet E+, \mathbf{1} \}$
$\{ E \rightarrow E\bullet E*, \mathbf{1} \}$
$\{ E \rightarrow \bullet EE+, \mathbf{2} \}$
$\{ E \rightarrow \bullet EE*, \mathbf{2} \}$
$\{ E \rightarrow \bullet a, \mathbf{2} \}$

3 – *a*

$\sim\{ E \rightarrow a\bullet, \mathbf{2} \}$
$\sim\{ E \rightarrow EE\bullet+, \mathbf{1} \}$
$\{ E \rightarrow EE\bullet*, \mathbf{1} \}$
$\{ E \rightarrow E\bullet E+, \mathbf{2} \}$
$\{ E \rightarrow E\bullet E*, \mathbf{2} \}$
$\{ E \rightarrow \bullet EE+, \mathbf{3} \}$
$\{ E \rightarrow \bullet EE*, \mathbf{3} \}$
$\{ E \rightarrow \bullet a, \mathbf{3} \}$

4 - +

$\sim\{ E \rightarrow EE+\bullet, \mathbf{1} \}$
$\{ E \rightarrow EE\bullet+, \mathbf{0} \}$
$\{ E \rightarrow EE\bullet*, \mathbf{0} \}$
$\{ E \rightarrow E\bullet E+, \mathbf{1} \}$
$\sim\{ E \rightarrow E\bullet E*, \mathbf{1} \}$
$\{ E \rightarrow \bullet EE+, \mathbf{4} \}$
$\{ E \rightarrow \bullet EE*, \mathbf{4} \}$
$\sim\{ E \rightarrow \bullet a, \mathbf{4} \}$

5 - *a*

$\sim\{ E \rightarrow a\bullet, \mathbf{4} \}$
$\{ E \rightarrow EE\bullet+, \mathbf{1} \}$
$\sim\{ E \rightarrow EE\bullet*, \mathbf{1} \}$
$\{ E \rightarrow E\bullet E+, \mathbf{4} \}$
$\{ E \rightarrow E\bullet E*, \mathbf{4} \}$
$\{ E \rightarrow \bullet EE+, \mathbf{5} \}$
$\{ E \rightarrow \bullet EE*, \mathbf{5} \}$
$\{ E \rightarrow \bullet a, \mathbf{5} \}$

6 - *

$\sim\{ E \rightarrow EE*\bullet, \mathbf{1} \}$
$\sim\{ E \rightarrow EE\bullet+, \mathbf{0} \}$
$\{ E \rightarrow EE\bullet*, \mathbf{0} \}$
$\{ E \rightarrow E\bullet E+, \mathbf{1} \}$
$\{ E \rightarrow E\bullet E*, \mathbf{1} \}$
$\{ E \rightarrow \bullet EE+, \mathbf{6} \}$
$\{ E \rightarrow \bullet EE*, \mathbf{6} \}$
$\{ E \rightarrow \bullet a, \mathbf{6} \}$

7 - +

$\sim\{ E \rightarrow EE+\bullet, \mathbf{0} \}$
$\{ E \rightarrow E\bullet E+, \mathbf{0} \}$
$\{ E \rightarrow E\bullet E*, \mathbf{0} \}$
$\{ E \rightarrow \bullet EE+, \mathbf{7} \}$
$\{ E \rightarrow \bullet EE*, \mathbf{7} \}$
$\{ E \rightarrow \bullet a, \mathbf{7} \}$

Az adott nyelvtan egyértelmű, így a mondatnak csak egy levezetése van. Az a tény, hogy az utolsó csoportban van (5.9.) alakú elem arra utal, hogy a vizsgált jelsorozat valóban mondat. Ezen a körülményen természetesen az sem változtat, hogy itt a mondatszimbólum jelölése nem *S,* mint a (5.9.) kifejezésben, hanem *E*.

Néhány megjegyzés, amely megkönnyítheti az algoritmus kezelését és megértését.

Az első és második, sorszám szerint a **0** és **1** sorszámú csoportban a pont mögött csak egyetlen terminális szimbólum található, az *a* terminális. Ez annyit jelent, hogy a nyelv valamennyi mondatában, ebben a pozícióban, vagyis az első két helyen csak az *a* terminális állhat. Valóban a nyelv ilyen tulajdonságú.

A **2** sorszámú csoportban már mindhárom terminális a +, ∗ és *a* szerepel a pont mögött. Ez megegyezésben van azzal, hogy ebben a pozícióban már mindhárom terminális előfordulhat. Persze egy adott mondat esetében mindig csak egyik válik be, – jelen esetben az *a* – a másik kettő elvetél.

Az algoritmus eredménye alapján a vizsgált mondat levezetése megszerkeszthető. A jelsorozat elfogadásáról a **7** csoport $\{E \rightarrow EE+\bullet, \mathbf{0}\}$ eleme informál. A különböző csoportok elemei közül azokat kell kiválasztanunk, amelyek a végeredményként kiadódó említett elem létrehozásában szerepet játszottak. Ezen elemek által reprezentált levezetési szabályok szolgáltatják a mondat levezetését.

A megértés megkönnyítésére ezek az elemek bajusszal vannak ellátva.

Az elemzés balelemzés, így a kapott eredmény a baloldali levezetést szolgáltatja

$$E \Rightarrow EE+ \Rightarrow aE+ \Rightarrow aEE*+ \Rightarrow aEE+E*+ \Rightarrow aaE+E*+ \Rightarrow aaa+E* + \Rightarrow aaa+a*+$$

A nyelvtan egyértelmű, és ez abból is látszik, hogy mindegyik csoportnak csak egy alapító tagja van. Az egyértelműség biztosítására elegendő lenne, ha csak azoknak a csoportoknak volna unikális alapító tagjuk, amelyek úgynevezett lezárt elemmel kezdődnek. Az ilyen elemeknél a pont az utolsó szimbólum után helyezkedik el. Esetünkben minden alapító elem ilyen, ami természetes, ha meggondoljuk, hogy az alapító tagok a pontnak egy terminálison való átugratásával keletkeznek. Itt pedig a levezetési szabályoknak csak az utolsó szimbóluma terminális, tehát ha ezt átugrottuk, akkor már a szabály végére értünk.

A másik általános elemző módszer, a *Coke-Younger-Kasami* eljárás megköveteli, hogy a nyelvtan *Chomsky* normálalakban legyen felírva. Ez megkötést jelent a nyelvtannal, de nem a nyelvvel szemben, hiszen minden környezetfüggetlen nyelvtanhoz rendelhető egy vele egyenértékű *Chomsky* normálformában adott nyelvtan.

A problémát az jelentheti, hogy az elemzés a levezetést természetesen nem az eredeti nyelvtan szabályai szerint adja meg, hanem az átalakított, már *Chomsky* normálalakban megadottnak megfelelően. Amennyiben minket az eredeti nyelvtan szolgáltatta levezetés érdekel – és általában ez a helyzet – akkor meg kell vizsgálnunk, hogyan lehet a normálalakban érvényes levezetésből az eredeti nyelvtan levezetését előállítani.

A *Chomsky* normálalak előállításánál úgy jártunk el, hogy az eredeti nyelvtan helyettesítési szabályainak alkalmazását egy sorozat *Chomsky* szabály

végrehajtásával modelleztük. A levezetési fa leírása céljából most is adjunk a helyettesítési szabályoknak sorszámot. Mind az eredeti, mind a *Chomsky* normálforma esetében. Ha most az eredeti nyelvtan szabályának sorszámát adjuk az őt modellező *Chomsky* sorozat egyik szabályának, a sorozat többi szabályának pedig nagyobb, az eredeti nyelvtanban nem szereplő sorszámokat adunk, akkor lehetőség van arra, hogy a normálforma szabályaival felírt levezetésből megkapjuk az eredeti nyelvtan levezetését. Ezt úgy érhetjük el, ha a „túlzottan" magas sorszámokat egyszerűen kihagyjuk a leírásból. Ha ezt a balelemzés illetve a jobbelemzés leírásánál el tudjuk érni, akkor azt mondjuk, hogy a normálalak, pontosabban a sorszámozott normálalak balról illetve jobbról lefedi az eredeti nyelvtant.

Amennyiben egy eredeti levezetési szabályt leképező *Chomsky* szabálysorozat első tagjának adjuk az eredeti nyelvtan szóban forgó szabályának sorszámát, akkor balról, ha az utolsónak, akkor jobbról fedtük le a normálalakkal az eredeti nyelvtant.

Mint említettem ez az elemzés alulról felfelé halad. Az elemzés során kapott eredmények egy alsó háromszögként rajzolt mátrix mezőit töltik ki. A mezőkbe a *Chomsky* normálforma nemterminális szimbólumai kerülnek.

A mátrix $\mathbf{t}_{jk}$ mezejébe akkor kerül egy nemterminális, ha a szóban forgó nemterminális képes a vizsgált mondat **j** számú konzekutív karakterének generálására, mégpedig éppen a **k** sorszámú szimbólummal kezdve.

A mátrix azért háromszög mátrix, mert míg egy **n** szimbólumot tartalmazó mondatban éppen **n** darab egyetlen szimbólumot tartalmazó részsorozat található, addig a kételemű részsorozatok száma már csak **n-1,** végül **n** elemű, tehát valamennyi szimbólumot magában foglaló sorozat már csak egy akad, maga a mondat.

A mátrixot alulról felfelé haladva kell kitölteni, vagyis a legalsó, tehát a fenti konvenció értelmében első sorral kell kezdenünk.

Ide az előbbiek szerint olyan nemterminálisok kerülnek, amelyek egyetlen karakterből álló jelsorozatokat képesek generálni. Itt tehát a *Chomsky* normálalakban engedélyezett két szabály, $A \rightarrow a$ és $A \rightarrow BC$ közül az előbbi típusúakat kell alkalmaznunk. Most és a későbbiekben elképzelhető, hogy egy mezőbe egynél több nemterminális kerül, sőt utóbb még az is előfordulhat, hogy egy nemterminális több okból írható be ugyanabba a mezőbe.

A második és az azt követő sorok megszerkesztésekor már a második szabálytípus szerinti levezetési szabályokat kell figyelembe venni. A második sor kitöltésekor azt vizsgáljuk, hogy van-e olyan levezetési szabály, amelynek jobboldala két, az első sorban szomszédos szimbólumból áll. Ha igen, akkor annak baloldalát kell beírnunk a második sor megfelelő mezejébe.

A generált részsorozat hosszának növekedésével az eljárás mind bonyolultabbá válik. Míg ugyanis egy két szimbólumból álló részsorozatot csak

egyféleképpen partícionálhatunk, addig egy hosszabb részsorozat már többféleképpen osztható fel.

Így például ha a $\mathbf{t}_{43}$ mezőt kívánjuk kitölteni**,** ahová definíciónk szerint azok a nemterminálisok kerülnek, amelyek a **3** pozícióval kezdődő, és a **6** pozícióval végződő sorozat négy szimbólumát generálják, akkor a $\mathbf{t}_{13}$-$\mathbf{t}_{34}$, a $\mathbf{t}_{23}$-$\mathbf{t}_{25}$ és a $\mathbf{t}_{33}$-$\mathbf{t}_{16}$ mezőpárokat kell megvizsgálnunk, ami megfelel a négy szimbólumból álló jelsorozat **1-3, 2-2** illetve **3-1** arányban történő felosztásának.

Példának válasszunk most egy nem egyértelmű nyelvtant.

$E \to E{+}E$ **1** $\quad E \to E{*}E$ **2** $\quad E \to (E)$ **3** $\quad E \to a$ **4**

Minthogy itt csak *Chomsky* normálformában megírt nyelvtan használható, végezzük el a nyelvtan átírását. A terminálisokból kialakított nemterminálisokat vesszővel különböztettük meg, és a sorszámozást úgy állapítottuk meg, hogy a normálalak balról fedje le az eredeti nyelvtant.

$E \to EA$ **1** $\quad E \to EM$ **2** $\quad E \to (\text{'}P$ **3** $\quad E \to a$ **4**
$A \to +\text{'}E$ **5** $\quad M \to *\text{'}E$ **6** $\quad P \to E\,)\text{'}$ **7**
$+\text{'} \to +$ **8** $\quad *\text{'} \to *$ **9** $\quad (\text{'} \to ($ **10** $)\text{'} \to)$ **11**

Legyen a vizsgálandó mondat, helyesebben mondatjelölt

$$a{+}a{*}a{+}a$$

Az eddigiek alapján a háromszögmátrix megszerkeszthető.

E^3						
	A					
E^2		E^2				
	A		M			
E		E		E		
	A		M		A	
E	$+\text{'}$	E	$*\text{'}$	E	$+\text{'}$	E

Az a tény, hogy a legfelső kockában szerepel a mondatszimbólum, arra utal, hogy belőle kiindulva a teljes jelsorozat levezethető, vagyis ez esetben a jelsorozat mondat.

A hatványkitevőben jeleztük, hányféleképpen kaphatjuk meg az adott nemterminálist az adott mezőben. Így például a teljes mondatot generáló E mondatszimbólum háromféle módon bontható fel:

$$E_{1\text{-}7} \to E_1A_{2\text{-}7} \mid E_{1\text{-}3}M_{4\text{-}7} \mid E_{1\text{-}5}A_{6\text{-}7}$$

Itt az index pozíciójában jeleztük, hogy az egyes nemterminális szimbólumok a mondat mely részét generálják.

Minthogy vannak további alternatívák ez a mondat öt lényegesen különböző módon vezethető le. Adjuk itt meg az egyik ilyen levezetést:

$E_{1\text{-}7} \Rightarrow E_{1\text{-}3}M_{4\text{-}7} \Rightarrow E_1A_{2\text{-}3}M_{4\text{-}7} \Rightarrow aA_{2\text{-}3}M_{4\text{-}7} \Rightarrow a{+}\text{'}E_3M_{4\text{-}7} \Rightarrow$
$a{+}E_3M_{4\text{-}7} \Rightarrow a{+}aM_{4\text{-}7} \Rightarrow a{+}a{*}\text{'}E_{5\text{-}7} \Rightarrow a{+}a{*}E_{5\text{-}7} \Rightarrow a{+}a{*}E_5A_{6\text{-}7} \Rightarrow$
$a{+}a{*}aA_{6\text{-}7} \Rightarrow a{+}a{*}a{+}\text{'}E_7 \Rightarrow a{+}a{*}a{+}E_7 \Rightarrow a{+}a{*}a{+}a$

Adjuk meg most ezt a levezetést a szabályok sorszámával:

2145846914584

Minthogy minket az eredeti nyelvtan alapján készült levezetés érdekel, hagyjuk el ebből a sorozatból a „túlságosan” nagy sorszámokat:

2144144

Ez a számsorozat valóban az eredeti nyelvtan levezetését írja le.

$E \Rightarrow E*E \Rightarrow E+E*E \Rightarrow a+E*E \Rightarrow a+a*E \Rightarrow a+a*E+E \Rightarrow a+a*a+E \Rightarrow a+a*a+a$

Ezzel azt is bemutattuk, hogyan fedi le a *Chomsky* normálalak az eredeti nyelvtant.

Az általános elemzők tárgyalása után áttérve a szűkebb értelemben vett, tehát determinisztikus elemzőkre, meg kell állapodnunk néhány terminológiai kérdésben.

Egy nyelvtant balelemezhetőnek illetve jobbelemezhetőnek mondunk, ha szerkeszthető hozzá determinisztikus, a baloldali levezetést illetve a jobboldali redukciót szolgáltató szintaktikus elemző.

Vizsgáljuk meg ebből a szempontból az alábbi két nyelvtant, amelyről könnyű belátni, hogy ugyanazt a nyelvet generálják:

1 $S \rightarrow BAb$	**1** $S \rightarrow BAb$
2 $S \rightarrow CAc$	**2** $S \rightarrow CAc$
3 $A \rightarrow BA$	**3** $A \rightarrow AB$
4 $A \rightarrow a$	**4** $A \rightarrow a$
5 $B \rightarrow a$	**5** $B \rightarrow a$
6 $C \rightarrow a$	**6** $C \rightarrow a$

A generált nyelv: $\boldsymbol{L} = a^{n+2}b \cup a^{n+2}c$

Talán említenem sem kell, hogy a példa illusztratív jellegű, hiszen ez a nyelv reguláris nyelvtannal is generálható.

Vizsgáljuk először az első, baloldalon álló nyelvtant. A nyelv egy mondatának baloldali levezetése

15 (35)^n 4 illetve **26 (35)^n 4**

aszerint, hogy az utolsó szimbólum *b* illetve *c* karakter. Erre az alábbi módon szerkeszthető determinisztikus elemző. A dolognak az a bibéje, hogy a legelső alkalmazott szabály kérdése csak az utolsó karakter beolvasásakor dől el.

A mondat elemzésekor tehát semmilyen részeredmény sem adható ki addig, amíg a teljes mondatot el nem olvastuk. Ez nem jelent leküzdhetetlen akadályt. Persze meg kell jegyeznünk a beolvasott *a* szimbólumok számát, célszerűen úgy, hogy elraktározzuk azokat az automata vermében. Az utolsó karakter beolvasása után már kiadhatjuk a beolvasott karaktertől függően a **15** vagy **26** eredményt, majd a **35** kombináció következik, a tárolt *a* szimbólumok számának megfelelően, ezt követi végül a **4** sorszám. Az a körülmény, hogy a beolvasott *a* szimbólumok száma kettővel több, mint ahány kombinációt ki kell nyomtatni, nem jelenthet problémát.

Ismét csak az első, baloldali nyelvtannál maradva, a jobboldali elemzés egészen más eredményt ad:

$$\mathbf{55^n43^n1} \qquad \text{illetve} \qquad \mathbf{65^n43^n2}$$

újra a beolvasott utolsó szimbólumtól függően.

Az világos, hogy az utolsó karakter beolvasása előtt itt sem adható ki semmilyen eredmény. A teljes mondat beolvasása után természetesen itt is kinyomtathatjuk az **5,** illetve **6** szabály sorszámát. Annak sincs akadálya, hogy az *a* karakterek számát valamilyen formában megjegyezzük. Amikor azonban az $\mathbf{5^n}$ kiadására kerül sor, akkor ezt az információt destruktívan, vagyis az információ megsemmisítésével együtt járó módon fel kell használnunk.

Ennek következtében aztán a $\mathbf{3^n}$ sorozat kiadására nincsen mód, hiszen hiányzik a kitevő értékére vonatkozó információ.

A fentiek alapján az első, baloldali nyelvtan balelemezhető, de nem jobbelemezhető.

A második, a jobboldalon álló nyelvtan esetében a helyzet pont fordított. Itt a baloldali elemzés lesz problematikus. Ennek eredménye ugyanis:

$$\mathbf{153^n45^n} \qquad \text{illetve} \qquad \mathbf{263^n45^n}$$

Itt nyilván ugyanazzal a nehézséggel kell, pontosabban kellene megbirkóznunk, mint előbb a jobboldali elemzés esetében.

Ugyanakkor a jobboldali elemzés eredménye itt is lehetőséget ad determinisztikus elemző készítésére. A szóban forgó számsorozat ugyanis

$$\mathbf{54\ (54)^n\ 1} \qquad \text{illetve} \qquad \mathbf{64\ (53)^n\ 2}$$

A már alkalmazott receptúra szerint persze itt is meg kell várnunk az utolsó karakter beolvasását, miközben memorizáljuk a beolvasott *a* szimbólumok számát. Ezzel a fogással jobboldali determinisztikus elemző szerkeszthető.

Ennek alapján rögzíthető, hogy a balelemezhető illetve jobbelemezhető nyelvtanok inkommenzurábilisak, összemérhetetlenek, vagyis egyik sem részhalmaza a másiknak.

Valójában az első nyelvtan balelemzője, illetve a második nyelvtan jobbelemzője egy szerencsés gondolatnak, mondhatni ügyes trükknek köszönheti létét.

Való igaz, hogy az *a* szimbólumokat a **35** illetve **53** levezetési szabályok kombinációja generálja. Ezt a tényt használjuk fel akkor, amikor a veremből előszedett információ alapján a kellő számú **35** illetve **53** kombinációt kiírjuk. A turpisság abban áll, hogy amikor a legelső ilyen kombinációt kiírjuk, amely szabályok szükségképpen a legelső szimbólumot generálták, akkor azt a legutoljára beolvasott szimbólumból nyert információ alapján tesszük.

Egy ilyen megoldású elemzőt egy szerencsés ötletet felhasználva megvalósíthatunk, de egy algoritmus segítségével, vagyis lényegében mechanikus úton származtatva aligha. Az ilyen elemző kialakítása nem „szabályos”.

Szabályosnak mondjuk az olyan elemzőket, ahol a szabály sorszámok kiadása azon szimbólumok vizsgálatakor történik meg, amely szabályok ezeket a szimbólumokat generálják. Példánkban, mint tisztáztuk nem ez a helyzet. A fenti két nyelvtanra készíthető balelemző illetve jobbelemző ezek szerint nem szabályos.

Ezt a disztinkciót azért lényeges megtennünk, mert amennyiben csak a szabályos elemzőkre szorítkozunk, akkor a balelemezhető nyelvtanok halmaza a jobbelemezhetőeknek valódi részhalmaza lesz. Erre az állításunkra még visszatérünk.

5.2. A balelemzés, *LL*(k) nyelvtanok

Baloldali elemzésnél mindig a mondatszimbólumból indulunk ki, és mindig a mondatszerű forma legbaloldalibb nemterminálisát bontjuk fel a következő lépésben. A mondatszerű forma elején található esetleges terminális jelsorozat természetesen meg kell egyezzen az elemzett mondat első terminálisaival. Ha nem így lenne, akkor már korábban be kellett volna dobnunk a törülközőt.

Ezt felfoghatjuk úgy is, hogy a mondatot az elemzés során addig olvastuk el, ameddig az élen álló terminálisokat generálni tudtuk. A következő lépésben a legbaloldalibb nemterminálist úgy kell felbontanunk, hogy a mondat további terminálisai kiadódjanak.

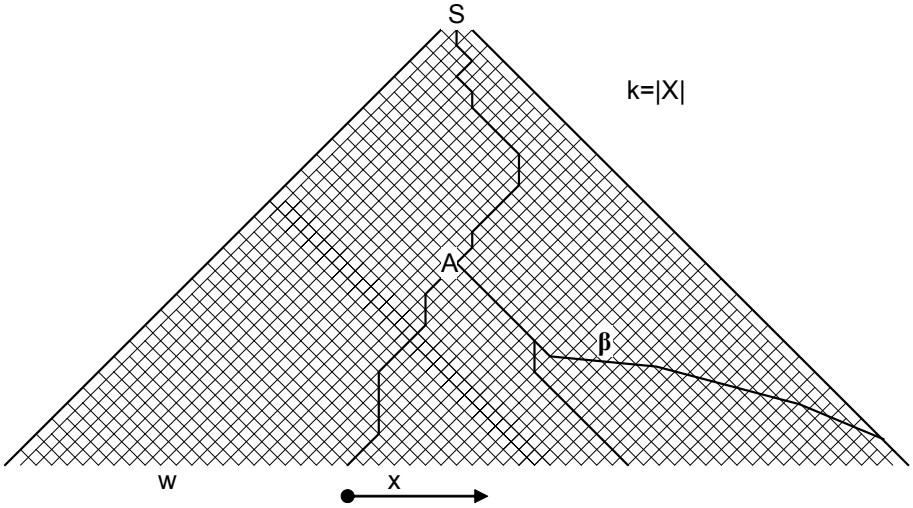
Általában az a helyzet, hogy a felbontandó nemterminális szimbólum több levezetési szabály baloldalán állhat, vagyis a felbontásnak több alternatívája van. Ezek közül kell a megfelelőt kiválasztani, hiszen egyértelmű nyelvről lévén szó, csak egyetlen felbontás vezethet eredményre.

Ennek érdekében az *LL*(**k**) elemző úgy jár el, mint a fegyelmezetlen olvasó a krimivel. Ha kíváncsi a folytatásra, akkor egy kicsit előrelapoz. Ez itt úgy történik, hogy az olvasás helyétől az elemző **k** szimbólumot előre néz, és ezekből a terminálisokból igyekszik eldönteni, melyik alternatívát kell választania, melyik a helyes folytatás.

Az elemző elnevezésében első *L* – *left* – arra utal, hogy az olvasás balról jobbra történik. Ez tehát csak a latin betűs és mondjuk héber betűs írásmód közötti különbséget jelöli.

A második *L* – *left* – az elemzés baloldali voltát jelenti. Végül a **k** azt mondja meg, hány szimbólumot nézünk előre, amikor a felbontás alternatívái között döntünk.

A gondolat magvát az 5.1. ábra szemlélteti. A már kiadódott mondatszerű forma $wA\beta$, ahol w a már megkapott terminális jelsorozat, A a legbaloldalibb nemterminális, míg β a mondatszerű forma további része. Amennyiben a w sorozatot követő **k** számú szimbólumot tartalmazó x jelsorozat elegendő információt szolgáltat arra, hogy a lehetséges alternatívák közül a helyeset kiválasszuk, dönteni tudjunk, akkor az adott nyelvtan *LL*(**k**) nyelvtan.



5.1. ábra

A szabatos leírás lehetővé tételére vezessük be az alábbi formalizmust. Legyen adott egy **G** grammatika, és legyen adott egy tetszőleges α jelsorozat. Jelölje

$$\mathrm{FIRST}^{G}_{k}(\alpha) \tag{5.10.}$$

azt a terminálisokból álló és legfeljebb **k** hosszúságú jelsorozatokat tartalmazó véges nyelvet, amelyet úgy kapunk, hogy vesszük az α jelsorozatból a **G** grammatika segítségével nyerhető összes jelsorozatot, illetve ha azok hosszabbak, mint **k**, akkor azok első **k** szimbólumát. Ha speciálisan az α terminális jelsorozat, akkor a $\mathrm{FIRST}^{G}_{k}(\alpha)$ vagy α, vagy annak első **k** szimbóluma. Ha nem okozhat félreértést, a grammatikára utaló megkülönböztető kitevőt elhagyjuk.

Legyen a felbontandó nemterminális A, és legyen $A \to \xi$ illetve $A \to \eta$ két lehetséges felbontás. A nyelvtan akkor *LL*(**k**) nyelvtan, ha minden nemterminálisra és minden alternatívapárra igaz a következő állítás. Legyen adott két levezetés:

$$S \overset{*}{\Rightarrow} wA\alpha \Rightarrow w\xi\alpha \overset{*}{\Rightarrow} wx \qquad \text{és}$$

$$S \overset{*}{\Rightarrow} wA\alpha \Rightarrow w\eta\alpha \overset{*}{\Rightarrow} wy \tag{5.11.}$$

akkor

$$\mathrm{FIRST}_k(x) = \mathrm{FIRST}_k(y) \tag{5.12.}$$

azonosságból következik a

$$\xi = \eta$$

azonosság is. Ez nem más, mint formális megfogalmazása annak, amit az előbb verbálisan már elmondtunk, nevezetesen a mondat következő **k** szimbóluma egyértelműen eldönti a helyes lebontást.

Vizsgáljuk meg, mi a feltétele annak, hogy egy nyelvtan *LL*(**k**) nyelvtan legyen.

Valamely nyelvtan egyik *A* nemterminálisának összes lehetséges levezetési szabálya legyen

$$A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$$

A nyelvtan csakis akkor lehet *LL*(**k**) nyelvtan, ha a $\mathrm{FIRST}_k(\alpha_i)$ véges nyelvek diszjunktak:

$$\mathrm{FIRST}_k(\alpha_i) \cap \mathrm{FIRST}_k(\alpha_j) = \varnothing \quad i \neq j \tag{5.13.}$$

Ellenkező esetben, vagyis ha a metszet páronként nem üres előfordulhat, hogy az elemző előretekintve éppen a metszet halmazába eső jelsorozatot olvas, és ennek alapján nem tud dönteni, melyik levezetési szabályt alkalmazza. Ez tehát szükséges feltétel.

Az (5.13.) megkötés nem csak szükséges, hanem elégséges is abban az esetben, ha az *A* nemterminális utódai mindenkor legalább **k** szimbólumot generálnak. Lássunk erre egy példát.

Nyelvként válasszuk a prefix lengyel jelölést:

1 $E \to +EE$ **2** $E \to *EE$ **3** $E \to a$

Ez *LL*(**1**) nyelvtan. Az (5.13.) feltétel teljesülése **k** = **1** mellett nyilvánvaló, ugyanakkor az *E* valamennyi leszármazottja generál legalább egy hosszúságú jelsorozatot.

Az úgynevezett elemzési táblát a nyelvtan alapján könnyen felírhatjuk. A vízszintes fejléc az előretekintés során kapható, esetünkben egy hosszúságú jelsorozatokat tünteti fel. A függőleges fejléc a verem tetejére kerülő szimbólumokat adja. A tábla mezői a szükséges intézkedésekre tartalmaznak utasításokat.

	+	*	*a*	ε
E	+*EE*, **1**	*****EE*, **2**	*a*, **3**	
+	*pop*			
*		*pop*		
a			*pop*	
ε				*acc*

A táblázat rovatainak jelentése a következő.

A *pop* az a már eddig is alkalmazott művelet, amikor a verem legfelső szimbólumát, az olvasott, és a veremben találhatóval azonos szimbólummal együtt megsemmisítjük. Gondolom, magyarázatot nem igényel, mikor kerül sor erre a műveletre.

A **k** mértékű előretekintést mindig úgy kell értelmezni, hogy a még fel nem dolgozott jelsorozat **k** hosszúságú prefixumát vesszük. Rövidebb előretekintő jelsorozatot csak akkor kaphatunk, ha a megmaradt jelsorozat hossza kisebb, mint **k**. Az ε jelsorozatot, mint egy hosszúságú jelsorozatot tehát csak akkor láthatjuk előretekintve, ha a teljes jelsorozatot már beolvastuk.

Amennyiben a teljes jelsorozatot beolvastuk, és ezzel egyidejűen a verem is kiürült, akkor a jelsorozatot, mint a nyelv mondatát elfogadjuk. Ezt fejezi ki az alkalmas mezőben az *acc* (*accept*) rövidítés.

Az üresen maradt mezők, ha a jelsorozat valóban a nyelv egy mondata, sohasem fordulhatnak elő. Ha mégis előadódnának, ez azt jelenti, hogy az elemzett jelsorozat nem mondata a nyelvnek, és így hibát kell jeleznünk.

A táblázatnak az a része, amely azokat az eseteket tünteti fel, amikor terminális van a verem tetején, vagy a verem üres, mindig azonos, és semmiféle a nyelv vagy nyelvtan specifikumára vonatkozó információt nem tartalmaz. Éppen ezért a továbbiakban a táblázatnak ezt a részét elhagyjuk. Természetesen, ha az elemzőt számítógépen kívánjuk megvalósítani, akkor az ehhez a részhez kapcsolódó tevékenységek programját ugyanúgy meg kell írnunk, mint a releváns információt hordozó részét.

Az adott esetben érdemleges információt csak a táblázat első sora hordoz. Az egyes mezők megmondják, mit kell beírnunk a nemterminális helyébe, és milyen levezetési szabály sorszámát kell kiadnunk, mint eredményt.

Nézzünk a táblázat alapján egy elemzést. A konfiguráció egy hármassal jellemezhető, az elemzendő jelsorozat még el nem olvasott részével, a verem tartalmával, és a kiadott eredménnyel.

Legyen a vizsgált mondat: $+a*aa$

$$\{ +a*aa, E, \varepsilon\} \mapsto \{ +a*aa, +EE, \mathbf{1}\} \mapsto \{ a*aa, EE, \mathbf{1}\} \mapsto$$
$$\mapsto \{ a*aa, aE, \mathbf{13}\} \mapsto \{ *aa, E, \mathbf{13}\} \mapsto \{ *aa, *EE, \mathbf{132}\} \mapsto$$
$$\mapsto \{ aa, EE, \mathbf{132}\} \mapsto \{ aa, aE, \mathbf{1323} \} \mapsto \{ a, E, \mathbf{1323}\} \mapsto$$
$$\mapsto \{ a, a, \mathbf{13233}\} \mapsto \{ \varepsilon, \varepsilon, \mathbf{13233}\} \mapsto acc$$

Visszatérve az (5.13.) kifejezés szükséges, de nem elégséges voltára baj akkor adódhat, ha az A nemterminális utódai nem minden esetben alkotnak kellő hosszúságú, vagyis az előretekintés **k** hosszánál nem rövidebb jelsorozatot.

Az elemző ugyanis nem a $FIRST_k(\alpha_i)$ jelsorozatokat vizsgálja, hanem az első **k** szimbólumot. Ha a nemterminális derivátuma nem tölti ki teljesen a kívánt hosszat, akkor az elemző a mögötte található szimbólumokból hozzámarkol annyit, hogy kiteljék a teljes méret.

Előfordulhat, hogy a rövid jelsorozat éppen valamelyik másik csoport egyik elemének prefixuma, és a hozzámarkolással történő kiegészítés szerencsétlen módon éppen egy másik csoport valamelyik elemét szolgáltatja. Ilyenkor megint csak nem lehet megkülönböztetni a két jelsorozatot, és nem lehet dönteni a két lehetséges folytatás között.

Ez az eset nem annyira valószínűtlen és hajánál fogva előráncigált, mint azt első pillanatban hisszük. Mint később látni fogjuk, gyakran célszerű a nyelvtanokat olymódon átalakítani, hogy abban ε-szabályok keletkezzenek. Így viszonylag gyakran van olyan nyelvtannal dolgunk, amelyben ε-szabályok vannak, és az epszilon bármely jelsorozat prefixumaként felfogható.

A probléma szabatos tárgyalására vezessük be a követő nyelv fogalmát. Legyen A a nyelvtan egyik nemterminális szimbóluma. Ekkor

$$\mathrm{FOLLOW}^{G}_{k}(A) \tag{5.14.}$$

jelentse azon, legfeljebb **k** hosszúságú jelsorozatok halmazát, amelyet az adott grammatika az A nemterminális utódait követve generálhat. Ezt a nyelvet nevezzük az A nemterminális követő nyelvének.

Amennyiben az A nemterminális szűkmarkúan generálja a jelsorozatot, és nem ad ki kellő számú szimbólumot, akkor az elemző azt a követő nyelvből egészíti ki.

Végül is az elemző által vizsgált jelsorozatok az alábbi összefüggéssel írhatóak le:

$$\mathrm{FIRST}_k(\ \mathrm{FIRST}_k(\alpha_i) \bullet \mathrm{FOLLOW}_k(A)) \tag{5.15.}$$

Itt α_i ismét az A nemterminális egyik lehetséges felbontása, $\bullet$ pedig a konkatenálás operátora.

Az (5.15.) kifejezés magyarázata a következő. A külső zárójelen belül az α_i jobboldalból származtatható, illetve az A nemterminálist követő, külön-külön legfeljebb **k** hosszúságú jelsorozatok konkatenáltja áll. Ezen konkatenáltnak vesszük az első **k** szimbólumát.

Amennyiben α_i egymagában generálja a megfelelő számú szimbólumot, akkor a követő nyelv jelenléte nincs hatással az eredményre. Ha viszont az α_i által generált jelsorozat rövidebb, akkor a hiányzó szimbólumokat a követő nyelv szolgáltatja.

A fentiekből nyilvánvaló, hogy az (5.15.) összefüggés éppen azokat a jelsorozatokat adja, amelyeket az elemző vizsgálat tárgyává tesz.

Amennyiben az (5.15.) szerinti kifejezések a különböző alternatívákra diszjunktak, akkor ez elegendő feltétel arra, hogy ennek alapján a helyes felbontás egyértelműen eldönthető legyen. A nyelvtan tehát *LL*(**k**) nyelvtan, mégpedig erősen *LL*(**k**) nyelvtan. Annak magyarázatára, hogy az (5.15.) feltétel miért nem szükséges, és mikor gyengén *LL*(**k**) egy nyelvtan, később kerítünk sort.

Lássunk most egy olyan példát, ahol az (5.13.) összefüggéseken túlmenően az (5.15.) típusú kifejezéseket is meg kell vizsgálnunk.

Válasszuk a már jól ismert, az aritmetikai kifejezéseket generáló nyelvtant. Sajnos ez a nyelvtan balrekurzív, és így nem balelemezhető.

Tegyünk talán egy kis kitérőt, hogy a fenti, és többször is hivatkozott állításunkat igazoljuk. Az egyszerűség kedvéért vegyünk egy közvetlen balrekurzív nyelvtant. A bizonyítás menetéből kitűnik, hogy a gondolatmenet éppígy alkalmazható nem közvetlenül, hanem csak közönségesen balrekurzív nyelvtanok esetében is.

Legyen a balrekurzivitás forrása az A nemterminális, és vonjuk össze két helyettesítési szabályba a lehetséges levezetéseket.

$$A \to A\alpha \mid \beta$$

Mint ismeretes, a felbontás eredménye abban a pillanatban, amikor az A nemterminális eltűnik a baloldalról

$$\beta\alpha^*$$

Nyilvánvaló, hogy előretekintve a β jelsorozatot, illetve az abból származtatott terminálisokat fogjuk látni. Ez lesz a helyzet annak ellenére, hogy az $A \to \beta$ levezetést megelőzte, pontosabban megelőzhette az $A \to A\alpha$ produkciós szabály tetszőleges számú alkalmazása. Az alkalmazások számát pedig semmilyen előretekintés sem mondhatja meg.

Vegyük például az alábbi, nem túl bonyolult, de balrekurzív nyelvtant:

$$S \to Sa \mid \varepsilon$$

Ez a nyelv nyilván az a szimbólumokból álló, és tetszőleges hosszúságú jelsorozatokat tartalmazza. A nyelv egy mondatát úgy generáljuk, hogy először az első szabályt alkalmazzuk annyiszor, ahány karakter van a mondatban. Az így kiadódó mondatszerű formák első szimbóluma az S nemterminális, amelyet egy sereg a karakter követ. Ezután kell alkalmazni egy ízben a második szabályt, amivel „agyonütjük" a nemterminálist. Sajnos a veremben található szimbólumok mindaddig nem távolíthatóak el *pop* művelettel, ameddig a nemterminálist el nem tüntettük. Erre szolgál a második szabály, amelyet azonban csak akkor alkalmazhatunk, ha már pontosan annyi a szimbólumot generáltunk, amennyi az elemzendő mondatban van. Ennek megállapítására azonban el kell, pontosabban el kellene látnunk a mondat végéig. Bármekkorára választanánk is azonban a betekintési mélységet, kapásból tudnánk ennél hosszabb mondatot generálni. Így tehát a fenti nyelvtan balrekurzivitása miatt nem balelemezhető.

Térjünk ez után a kis kitérő után vissza eredeti célkitűzésünkhöz, az aritmetikai kifejezéseket generáló nyelvtan balelemzéséhez. Az eredeti nyelvtan, mint ismeretes:

$$E \to E{+}T \mid T \qquad T \to T{*}F \mid F \qquad F \to (E) \mid a$$

A balrekurzivitás megszüntetésére használjuk a korábban megadottól kissé eltérő, de áttekinthetőbb eredményt adó alábbi átírást:

$$A \to A\alpha \mid \beta \qquad A \to \beta\hat{A} \qquad \hat{A} \to \alpha\hat{A} \mid \varepsilon$$

Ezzel az átalakított nyelvtan a következő alakú lesz:

1 $E \rightarrow T\hat{E}$ **2** $\hat{E} \rightarrow +T\hat{E}$ **3** $\hat{E} \rightarrow \varepsilon$
4 $T \rightarrow F\check{T}$ **5** $\check{T} \rightarrow *F\check{T}$ **6** $\check{T} \rightarrow \varepsilon$
7 $F \rightarrow (E)$ **8** $F \rightarrow a$

Az új levezetési szabályokat sorszámmal is elláttuk.

Sajnos az átalakított nyelvtan nem fedi le a kiindulási nyelvtant, így az átalakított nyelvtan szerinti levezetés alapján nehezebb előállítani az eredeti nyelvtan szerinti levezetést.

Tegyünk kísérletet egy *LL*(**1**) elemző készítésére. Minthogy két olyan nemterminálisunk is van, amely ε-szabály alkalmazása kapcsán eltűnhet, és így egyetlen szimbólumot sem generál, szükségünk lesz ennek a két nemterminálisnak a követő nyelvére.

Az $\hat{E}$ nemterminális az **1** produkciós szabály alkalmazásával kerülhet be a mondatszerű formába. Minthogy ilyenkor az E nemterminális helyébe lép, nyilván ugyanaz követheti, mint ami elődjét az E nemterminálist követte.

Az E szimbólum a mondatszimbólum. Így induláskor a mondatszerű forma egyedül ebből a szimbólumból áll. Ilyenkor semmi sem, pontosabban az ε jelsorozat követi. Az E nemterminális a **7** szabály révén kerülhet ismét a mondatszerű formába. Ekkor azt végzárójel követi. Ezzel a követő nyelv:

$$\text{FOLLOW}_1(\hat{E}) = \{ \varepsilon,) \}$$

Kissé bonyolultabb a másik eltűnő nemterminális, a $\check{T}$ követő nyelvének megállapítása. A $\check{T}$ a **4** sorszámú szabály alkalmazásával lesz eleme a mondatszerű formának. Ilyenkor a T nemterminális helyébe lép, így ugyanaz követheti, ami a T nemterminálist követte. Ez utóbbit, akár az **1**, akár a **2** sorszámú szabály kapcsán tűnt fel a mondatszerű formában, utána az $\hat{E}$ szimbólum áll. Ez a nemterminális fogja tehát generálni a T és így a $\check{T}$ követő szimbólumait.

Amennyiben az $\hat{E}$ felbontására a **2** szabályt alkalmazzuk, akkor a helyére beírt jelsorozat első karaktere a + szimbólum lesz. Előfordulhat azonban, hogy a **3** szabály nyom nélkül eltűnteti az $\hat{E}$ szimbólumot. Mi követi ilyenkor a $\check{T}$ szimbólumot? Nyilván az, ami a mögüle eltűntetett $\hat{E}$ szimbólum után állt, vagyis ami eredetileg az $\hat{E}$ szimbólumot követte. Ennek alapján a $\check{T}$ követő nyelve:

$$\text{FOLLOW}_1(\check{T}) = \{ +, \varepsilon,)\}$$

Ezzel már el tudjuk dönteni, hogy a szóban forgó nyelvtan erősen balelemezhető-e, vagyis diszjunktak-e az (5.15.) szerinti kifejezések. Ezt a vizsgálatot természetesen csak azokra a szabályokra kell elvégezni, amelyek alternatívát jelentenek, amelyek egymás konkurensei. Ilyenek a **2-3**, az **5-6** és a **7-8** szabálypárok.

Nos ezekre az (5.15.) szerinti kifejezések, amelyek mint láttuk, az előretekintést szolgáltatják a következők lesznek:

2-3	{ + }	{ ε,) }
5-6	{ * }	{ +, ε,) }
7-8	{ (}	{ *a* }

Mint látható, az alternatívaként szóba jöhető párokra nézve a halmazok mind diszjunktak, így a nyelvtan erősen *LL*(**1**) nyelvtan.

Szerkesszük meg ezek után az elemző táblát, megállapodásunk szerint elhagyva a függőleges fejlécből a terminális szimbólumokat.

	(	*a*	)	+	*	ε
E	*TÊ,***1**	*TÊ,***1**				
Ê			ε,**3**	*+TÊ,***2**		ε,**3**
T	*FŤ,***4**	*FŤ,***4**				
Ť			ε,**6**	ε,**6**	**FŤ,***5**	ε,**6**
F	(*E*),**7**	*a,***8**				

A táblázat segítségével a nyelv egy mondatának szintaktikus elemzését könnyen elvégezhetjük. Példaképpen legyen az elemzendő mondat: *a+a*a*

Az elemzés menete:

{ *a+a*a, E,* ε } ↦ { *a+a*a, TÊ,* **1**} ↦ { *a+a*a, FŤÊ,* **14** } ↦
↦ { *a+a*a, aŤÊ,* **148** } ↦ { *+aa, ŤÊ,* **148** } ↦ { *+a*a, Ê,* **1486** } ↦
↦ { *+a*a, +ŤÊ,* **14862** } ↦ { *a*a, ŤÊ,* **14862** } ↦ { *a*a, FŤÊ,* **148624 }** ↦
↦ { *a*a, aŤÊ,* **1486248** } ↦ { **a, ŤÊ,* **1486248** } ↦
↦ { **a, *FŤÊ,* **14862485** } ↦ { *a, FŤÊ,* **14862485** } ↦
↦ { *a, aŤÊ,* **148624858** } ↦ { ε, *ŤÊ,* **148624858** } ↦
↦ { ε, *Ê,* **148628586** } ↦ { ε, ε, **1486285863** } ↦ *accept*

Az a tény, hogy az (5.15.) szerinti halmazok nem diszjunktak nem jelenti szükségképpen azt, hogy a nyelvtan nem balelemezhető. Legcélszerűbb ezt először egy példán bemutatni, ennek okait majd ezt követően vizsgálhatjuk meg.

Az alábbi nyelvtan, mint később kitűnik *LL*(**2**) nyelvtan.

1 $S \to aAbaS$ **2** $S \to bAabS$ **3** $S \to \varepsilon$
4 $A \to a$ **5** $A \to b$ **6** $A \to \varepsilon$

Érzékelhetően az *A* nemterminális felbontása a kritikus.

Állapítsuk meg az *A* nemterminális követő nyelvét:

$$\text{FOLLOW}_2(A) = \{ ab, ba \}$$

Ezzel a **4, 5** és **6** szabályhoz tartozó (5.15.) szerinti kifejezések:

4 - { *aa, ab* } **5 -** { *ba, bb* } **6 -** { *ab, ba* }

A halmazok nem diszjunktak, így amennyiben az előretekintéssel kapott jelsorozat *ab* illetve *ba,* akkor az első esetben a **4** és **6,** második esetben az **5** és **6** szabály között nem tudunk választani.

Segíthetünk azonban ezen a problémán, ha a nyelvtanon egy apró, jelentéktelennek tűnő módosítást hajtunk végre.

1 $S \to aA_1baS$ **2** $S \to bA_2abS$ **3** $S \to \varepsilon$
4 $A_1 \to a$ **5** $A_1 \to b$ **6** $A_1 \to \varepsilon$
4 $A_2 \to a$ **6** $A_2 \to b$ **6** $A_2 \to \varepsilon$

Itt tulajdonképpen semmi mást nem tettünk, mint megkülönböztettük az **1** és **2** levezetési szabályban szereplő A nemterminálist. Természetesen a helyette bevezetett két új nemterminális produkciós szabályaikat az eredeti A nemterminálistól öröklik, így ezeket ugyanazzal a számozással láttuk el.

A fenti nyelvtanhoz viszont már könnyen szerkeszthetünk egy $LL(\mathbf{2})$ elemzőt. Íme a táblázat:

	aa	*ab*	*ba*	*bb*	*a*	*b*	ε
S	aA_1baS, **1**	aA_1baS, **1**	bA_2abS, **2**	bA_2abS, **2**			ε, **3**
A_1		*a*, **4**	ε, **6**	*b*, **5**			
A_2	*a*, **4**	ε, **6**	*b*, **5**				

Természetesen most két karaktert tekintünk előre, a táblázat vízszintes fejléce tehát ennek megfelelően alakul. Jóllehet van olyan szituáció, amikor a jelsorozat még el nem olvasott része már csak egy karaktert tartalmaz, ilyen esetekben mindig a *pop* művelet következik, vagyis a tennivalókat a táblázatnak azon részei szabják meg, amelyeket korábbi megállapodásunknak megfelelően elhagytunk.

Emlékeztetek arra, mi volt az $LL(\mathbf{k})$ tulajdonság szabatos megfogalmazása. Az áttekinthetőség miatt megismételjük az (5.11.) és (5.12.) alatti állításokat.

Egy nyelvtan akkor $LL(\mathbf{k})$ nyelvtan, ha minden nemterminálisának – jelölje itt A – bármely szabálypárjára – legyen ez itt $A \to \xi$ és $A \to \eta$ – igaz, hogy ha adott két levezetés

$$S \overset{*}{\Rightarrow} wA\alpha \Rightarrow w\xi\alpha \overset{*}{\Rightarrow} wx \qquad \text{és}$$

$$S \overset{*}{\Rightarrow} wA\alpha \Rightarrow w\eta\alpha \overset{*}{\Rightarrow} wy$$

azonosságból következik az alábbi azonosság is:

$$\mathrm{FIRST}_{\mathbf{k}}(x) = \mathrm{FIRST}_{\mathbf{k}}(y)$$

az alábbi azonosság is: $\xi = \eta$

A különböző jobboldalak által létrehozott terminális jelsorozatokból, pontosabban azok első **k** szimbólumából alkotott halmazok nyilvánvalóan diszjunktak kell legyenek. Amennyiben A utódai, tehát jelen esetben ξ és η mindig generálnak kellő számú karaktert, akkor ez nem csak szükséges, hanem elégséges feltétel is, és az, hogy az A szimbólum mögött álló α milyen terminális jelsorozatot hoz létre érdektelen, hiszen az elemző úgy sem veszi figyelembe.

Amennyiben a produkciós szabályok jobboldalai nem képesek minden esetben generálni a szükséges **k** karaktert, akkor figyelembe kell venni, milyen terminális jelsorozat származhat a mondatszerű formának az A nemterminálist követő részéből. Ennek egyszerű módja, ha számításba vesszük mindazokat a jelsorozatokat, amelyek az A derivátumai után állhatnak. Ezt fejeztük ki a FOLLOW függvénnyel, és ezt tükrözi a (5.15.) összefüggés.

Ez a sommás elintézési mód azonban túl nagyvonalú, mert nem veszi figyelembe azt az információt, amelyet a már elolvasott szövegből, jelöléseinkkel a *w* jelsorozatból nyerhetünk. Egy adott levezetésnél ugyanis az *A* nemterminális előtt álló szöveg egy olyan kontextust jelent, amelyik korlátozhatja az *A* nemterminális derivátumai után álló jelsorozatokat. Ha ezt figyelembe vesszük, pontosabb disztinkciót biztosíthatunk.

Szó szerint ezt tettük akkor, amikor finomítottuk az előbbi *LL*(**2**) nyelvtanunkat. Felismertük ugyanis, hogy ha megkülönböztetjük az **1** és **2** szabály generálta *A* nemterminálist, akkor csatát nyerhetünk, hiszen a két úton nyert kontextus a kettéhasítással született két nemterminálishoz két különböző úgynevezett lokális követő nyelvet, két különböző FOLLOW függvényt rendel.

Amennyiben az eredeti nyelvtan valamely nemterminális szimbólumát fel tudjuk bontani olymódon több nemterminálisra, hogy a születésükkor meghatározott követő nyelvek, a lokális FOLLOW függvények eltérőek legyenek, és ennek segítségével a

$$\mathrm{FIRST}_{\mathbf{k}}(\mathrm{FIRST}_{\mathbf{k}}(A_{\mathrm{i}})\bullet\mathrm{FOLLOW}_{\mathbf{k}}(A_{\mathrm{i}})) \cap \mathrm{FIRST}_{\mathbf{k}}(\mathrm{FIRST}_{\mathbf{k}}(A_{\mathrm{j}})\bullet\mathrm{FOLLOW}_{\mathbf{k}}(A_{\mathrm{j}})) = \varnothing$$

kifejezéseket diszjunkttá tudjuk tenni, akkor gyengén *LL*(**k**) nyelvtanról beszélünk. Itt A_{i} és A_{j} az eredeti nyelvtan *A* nemterminálisának különböző lokális FOLLOW függvényekkel bíró példányai.

Megjegyzem, hogy ezeknek az újonnan született nemterminálisoknak a $\mathrm{FIRST}_{\mathbf{k}}$ függvénye természetesen megegyezik a $\mathrm{FIRST}_{\mathbf{k}}$ (*A*) függvénnyel, hiszen az, hogy mi áll egy nemterminális derivátumai után az függhet attól, hogyan került bele a nemterminális a mondatszerű formába, de az, hogy mit vezetünk le belőle az nyilván ettől független.

Annak érdekében, hogy gyenge *LL*(**k**) nyelvtanokra, tehát olyanokra, ahol a lokális FOLLOW függvény szerinti disztinkció szükséges, készíthessünk elemzőt, célszerű, ha ezt a különbségtételt már az induláskor megtesszük. Ez annyit jelent, hogy egy nemterminális két vagy több előfordulását megkülönböztetjük, a táblázatot finomítjuk, ha a kontextus alapján különböző követő nyelv tartozik hozzájuk. Így az elemző tábla rovatai a nemterminálisoknak a hozzájuk tartozó lokális követő nyelvvel együtt felelnek meg.

Persze ez a disztinkció bonyolultabbá teszi az elemző tábla elkészítését. A nagy elemző tábla minden rovatát egy kisebb táblácska alapján szerkesztjük meg. Ennek a táblácskának három oszlopa lesz. Megadjuk az adott szituációban az előretekintéssel kapható jelsorozatokat, feljegyezzük az egyes esetekben szükséges intézkedéseket, végül, ha a verembe új nemterminálist írunk be, akkor megadjuk annak követő nyelvét.

Az eredetileg azonos, de különböző lokális követő nyelvük miatt megkülönböztetett nemterminálisokat indexszel sorszámozzuk. Amennyiben egy nemterminálishoz olyan követő nyelv kombináció adódik, amely eddig még nem fordult elő, akkor az indexet eggyel növelve, erre is elkészítjük ezt a táblácskát.

Induláskor a legelső mondatszerű formára kell gondolnunk. Ez egyedül a mondatszimbólum, amelyet semmi sem követ, Ennek következtében az első nemterminális – lokális követő nyelv párosításunk a mondatszimbólum, és egyedül az üres jelsorozatot tartalmazó követő nyelv lesz. Ezt elemezve találjuk meg a nyelvtan többi nemterminális – lokális követő nyelv párjait.

Végezzük el ezt a párosítást a már ismert, az aritmetikai kifejezéseket generáló nyelvtanra.

Emlékeztetőül álljon itt a nyelv módosított nyelvtana:

1 $E \to T\hat{E}$	**2** $\hat{E} \to +T\hat{E}$	**3** $\hat{E} \to \varepsilon$	**4** $T \to F\check{T}$
5 $\check{T} \to *F\check{T}$	**6** $\check{T} \to \varepsilon$	**7** $F \to (E)$	**8** $F \to a$

A táblácskák struktúrája olyan, hogy a harmadik oszlopban egymás után sorban szerepelnek a második oszlop nemterminálisainak követő nyelvei. Adott esetben, minthogy $LL(\mathbf{1})$ elemzőt készítünk, ezen követő nyelvek mondatai legfeljebb egy karakterből állanak.

A táblácskák kitöltése megér néhány szó magyarázatot.

A kiindulás, mint azt megbeszéltük, a mondatszimbólum és az {ε} követő nyelv. Számítva arra, hogy a mondatszimbólum más követő nyelvvel is előfordul majd, előrelátóan **1** indexszel láttuk el ezt a változatot.

Az első táblácska szerint, miután $\hat{E}$ kerül az E_1 nemterminális helyére, Mindkettőnek azonos lesz a követő nyelve. Itt is **1** indexet adtunk ennek a változatnak.

A T nemterminális követő nyelvét az utána álló $\hat{E}_1$ nemterminális generálja. Amennyiben valóban generál valamit, akkor annak első karaktere a + szimbólum lesz. Előfordulhat azonban, hogy az $\hat{E}_1$ szimbólum a **3** szabály szerint eltűnik. Ilyenkor az válik láthatóvá, ami eredetileg mögötte állt, adott esetben az ε üres jelsorozat. Így a T_1 nemterminális követő nyelve { +, ε } lesz.

Hasonló gondolatmenettel adódik ki a többi táblácskában is a lokális követő nyelv. Ha valamelyik nemterminálishoz eddig nem szerepelt követő nyelv kerül, akkor azt új indexszel mint önálló nemterminálist tüntetjük fel a nagy táblázatban. A lokális követő nyelvek figyelembe vétele, a nemterminális szimbólumok finomítása ebben a példában pontosan kétszeresére növelte az elemző tábla méretét.

	E_1	{ε}
(	$T_1\hat{E}_1$,**1**	{+,ε},{ε}
a	$T_1\hat{E}_1$,**1**	{+,ε},{ε}

	E_2	{)}
(	$T_2\hat{E}_2$,**1**	{+,)},{)}
a	$T_2\hat{E}_2$,**1**	{+,)},{)}

	$\hat{E}_1$	{ε}
+	$+T_1\hat{E}_1$,**2**	{+,ε},{ε}
ε	ε,**3**	

	$\hat{E}_2$	{)}
+	$+T_2\hat{E}_2$,**2**	{+,)},{)}
)	ε,**3**	

	T_1	{+,ε}
(	$F_1\check{T}_1$,**4**	{*,+,ε},{+, ε}
a	$F_1\check{T}_1$,**4**	{*,+,ε},{+,ε}

	T_2	{+,)}
(	$F_2\check{T}_2$,**4**	{*,+,)},{+,)}
a	$F_2\check{T}_2$,**4**	{*,+,)},{+,)}

	$\check{T}_1$	{+,ε}
*	$*F_1\check{T}_1$,**5**	{*,+,ε},{+,ε}
+	ε,**6**	
ε	ε,**6**	

	$\check{T}_2$	{+,)}
*	$*F_2\check{T}_2$,**5**	{*,+,)},{+,)}
+	ε,**6**	
)	ε,**6**	

	F_1	{*,+,ε}
(	(E_2),**7**	{)}
a	*a*,**8**	

	F_2	{*,+,)}
(	(E_2),**7**	{)}
a	*a*,**8**	

Ennek alapján az elemző tábla elkészíthető:

	(	*a*	)	+	*	ε
E_1	$T_1\hat{E}_1$,**1**	$T_1\hat{E}_1$,**1**				
E_2	$T_2\hat{E}_2$,**1**	$T_2\hat{E}_2$,**1**				
$\hat{E}_1$				$+T_1\hat{E}_1$,**2**		ε,**3**
$\hat{E}_2$			ε,**3**	$+T_2\hat{E}_2$,**2**		
T_1	$F_1\check{T}_1$,**4**	$F_1\check{T}_1$,**4**				
T_2	$F_2\check{T}_2$,**4**	$F_2\check{T}_2$,**4**				
$\check{T}_1$				ε,**6**	$*F_1\check{T}_1$,**5**	ε,**6**
$\check{T}_2$			ε,**6**	ε,**6**	$*F_2\check{T}_2$,**5**	
F_1	(E_2),**7**	*a*,**8**				
F_2	(E_2),**7**	*a*,**8**				

Látható, hogy az F nemterminális kivételével minden esetben van valamelyes különbség az **1** és **2** indexszel jelölt nemterminálisok sorai között. Ennek ellenére a két sor között nincsen konfrontáció, vagyis nem tartalmaznak ellentétes utasítást. Ez arra utal, hogy az azonos nemterminálisból származtatott új nemterminálisok sorait össze is lehetne vonni, vagyis a nyelvtan erős $LL(\mathbf{k})$ nyelvtan.

Ez természetes, hiszen mint kiderült, a nyelvtan erős $LL(\mathbf{1})$ nyelvtan, és így nincs szükség a nemterminálisok lokális követő nyelv szerinti finomítására.

Emlékeztetek arra, hogy a gyengén balelemezhető nyelvtanok esetében a kétféle kontextusban más és más tennivaló áll, a két sor között konfrontáció van. Éppen ezért az ilyen nyelvtanok esetében a nemterminálisok lokális követő nyelv alapján történő szétválasztása létkérdés. Ez a finomítás teszi ugyanis lehetővé az ellentmondásmentes elemző elkészítését.

Felmerül a kérdés, van-e értelme az erősen balelemezhető nyelvtanok esetében ennek a finomításnak. Erre a kérdésre egy olyan jelsorozat elemzése ad választ, amely nem a nyelv mondata. Ugyanis ha összevetjük a szétbontott nemterminálisokhoz tartozó sorokat, akkor konfrontációt ugyan nem találunk, olyan helyzet azonban adódik, amikor az egyik sor határozott utasítást ad, a másik sorban viszont a mezőben nem áll semmi, ami konvenciónk szerint hibára utal. Ezért lesz tanulságos egy hibás jelsorozat vizsgálata.

Legyen ez a jelsorozat: $a{+}a){*}a$

Lássuk az elemzés menetét.

$\{\ a{+}a){*}a,\ E_1,\ \varepsilon\ \} \mapsto \{\ a{+}a){*}a,\ T_1\hat{E}_1,\ \mathbf{1}\ \} \mapsto \{\ a{+}a){*}a,\ F_1\check{T}_1\hat{E}_1,\ \mathbf{14}\ \} \mapsto$

$\mapsto \{\ a{+}a){*}a,\ a\check{T}_1\hat{E}_1,\ \mathbf{148}\ \} \mapsto \{\ {+}a){*}a,\ \check{T}_1\hat{E}_1,\ \mathbf{148}\ \} \mapsto$

$\mapsto \{\ {+}a){*}a,\ \hat{E}_1,\ \mathbf{1486}\ \} \mapsto \{\ {+}a){*}a,\ {+}T_1\hat{E}_1,\ \mathbf{14862}\ \} \mapsto$

$\mapsto \{\ a){*}a,\ T_1\hat{E}_1,\ \mathbf{14862}\ \} \mapsto \{\ a){*}a,\ F_1\check{T}_1\hat{E}_1,\ \mathbf{148624}\ \} \mapsto$

$\mapsto \{\ a){*}a,\ a\check{T}_1\hat{E}_1,\ \mathbf{1486248}\ \} \mapsto \{\ ){*}a,\ \check{T}_1\hat{E}_1,\ \mathbf{1486248}\ \}$

Az elemzés eddig ugyanúgy halad akár az eredeti, akár a finomított elemző táblát használjuk. Most azonban, amikor a hibát okozó szimbólum kerül sorra, a két elemző magatartása eltér.

A finomabb felosztású elemző hibát jelez, hiszen a $\check{T}_1$ nemterminális szimbólum esetében az előretekintés nem eredményezhet) szimbólumot. Az eredeti elemző előbb alkalmazza a **3** majd a **6** szabályt, és csak azután jelez hibát. Ez magától értetődő, hiszen csak a finomabb felosztású elemző tesz különbséget a zárójelen kívüli és zárójelen belüli nemterminálisok között.

Két szintaktikus elemzőt erősen ekvivalensnek mondunk, ha bármely jelsorozatnál az esetleges hibát ugyanannál az elemzési lépésnél fedezik fel.

Két szintaktikus elemző gyengén ekvivalens, ha a jelsorozatnak ugyanannál a szimbólumánál állanak meg, de ebben a helyzetben különböző számú elemzési lépést tesznek meg.

Két szintaktikus elemző nem ekvivalens, ha van olyan – szükségképpen hibás – jelsorozat, ahol a két elemző a hibát más-más karakternél detektálja.

A mi két példánkban szereplő két elemző gyengén ekvivalens.

Természetesen a lokális követő nyelv figyelembe vételével készített elemző nagyobb terjedelmű, és kezelése is nehézkesebb. Gondoljuk meg azonban, hogy a fordítóprogram futása során hány ízben eredményez hibajelzést és hányszor ad hibátlan kódot.

Talán első hallásra kissé bizarr, de alapvetően sokatmondó az az állítás, hogy a fordítóprogram elsődleges funkciója a hibajelzés, és a fordítás csak mint mellékhatás jelentkezik. Ennek fényében pedig a jó hibadetektálási képesség megkülönböztetett fontosságú.

Nyilvánvaló, hogy az összes rendelkezésre álló információt feldolgozó, tehát a lokális követő nyelveket is tekintetbe vevő elemző adja balelemzés esetén a legpontosabb hibajelzést.

Hogyan lehet egy nyelvtanról kideríteni, hogy balelemezhető. Sajnos sehogyan. Csupán azt lehet megállapítani, hogy egy nyelvtan egy adott rögzített **k** esetében *LL*(**k**) nyelvtan-e.

Ennek az a módja, hogy elkezdjük az ismertetett táblácskák elkészítését. Amennyiben nem tapasztalunk konfliktus helyzetet, akkor ellentmondásmentes elemző tábla szerkeszthető, a nyelvtan *LL*(**k**) nyelvtan.

Felmerül a kérdés, hogy a balelemezhetőség nyelvhez, vagy nyelvtanhoz köthető tulajdonság.

A tárgyalás során találkoztunk olyan nyelvvel, amelynek egyik nyelvtana nem volt balelemezhető, de találtunk ugyanakkor egy másik balelemezhető nyelvtant. Ebben az esetben a balelemezhetőség nyilván nyelvtanhoz kötött tulajdonság. Ez általában is igaz, ezért rendszerint balelemezhető nyelvtanról és nem balelemezhető nyelvről beszélünk.

Vannak azonban nem balelemezhető nyelvek is. Az ilyen nyelveknek nincsen balelemezhető nyelvtanuk.

Balelemzés esetén nagyon kedvező, ha az előretekintés hossza, a **k** minél kisebb. Vannak módszerek a nyelvtanok olyan átalakítására, amellyel az előretekintés hosszát csökkenteni lehet. Ilyen módszer az úgynevezett faktorizálás.

Ez az eljárás akkor alkalmazható, ha valamelyik nemterminális két vagy több levezetési szabályának jobboldala azonosan kezdődik. Legyen például az A nemterminális két levezetési szabálya:

$$A \to \alpha\xi \qquad A \to \alpha\eta$$

Ilyenkor a két szabály között csak olyan előretekintéssel tehetünk különbséget, amely túlmutat az α prefixum derivátumán. Amennyiben „kiemeljük” az α prefixumot a két levezetési szabályból, akkor sokkal kedvezőbb nyelvtant kapunk:

$$A \to \alpha B \qquad B \to \xi \mid \eta$$

Ennél a megoldásnál nem kell még az α feldolgozása előtt a ξ illetve η folytatás felől döntenünk, hanem erre ráérünk akkor, amikor az már elkerülhetetlen, az α felszámolása után.

Kérdés mikor lehet a nyelvtant úgy átalakítani, hogy faktorizálással az előretekintés mértéke **k** csökkenjen? Minden környezetfüggetlen nyelvtannak megszerkeszthető a *Greibach* normálalakja. Mostani szándékaink szerint fontos tudnunk, hogy ha az eredeti nyelvtan *LL*(**k**) nyelvtan volt, akkor a művelet végrehajtható olymódon, hogy annak során **k** értéke ne növekedjék. Ennek ellenkezője persze előfordulhat, hiszen például egy nem balelemezhető balrekurzív nyelvtan *Greibach* normálalakja rögtön *LL*(**k**) elemezhetővé válik.

Ha megkaptuk a *Greibach* normálalakot, akkor vizsgáljuk meg, hogy az egyes nemterminálisokhoz tartozó produkciós szabályok milyen terminális szimbólummal kezdődnek. Ha ezek a terminális szimbólumok valamennyi nemterminális esetében mind különbözőek, akkor nyilván *LL*(**1**) nyelvtannal van dolgunk, és így az előretekintés hosszában elértük az optimumot.

Amennyiben van olyan nemterminális, ahol két levezetési szabály ugyanazzal a terminálissal kezdődik, akkor faktorizálással segíthetünk a dolgon. Ha most az új, most már a *Greibach* alaktól kissé eltérő nyelvtant újra *Greibach* normálalakra hozzuk, ezt a játékot folytatva tovább csökkenhetjük az előretekintés hosszát, a **k** paramétert. Ebben a folyamatban nincsen megállás, mindig szerkeszthetünk *LL*(**1**) nyelvtant? Nem, mint alább kiderül, van megállás.

Ugyanis, ha az átalakítások során ε-szabály kerül a nyelvtanba, akkor nem minden esetben folytatható ez az eljárás, a **k** paraméter csökkentését nem tudjuk továbbfolytatni.

Ehhez a gondolathoz kapcsolódik a következő kérdés. Azt már tudjuk, hogy vannak *LL*(**k**) nyelvtanok. Vannak-e *LL*(**k**) nyelvek? Vannak.

Egy nyelvet akkor nevezünk *LL*(**k**) nyelvnek, ha a nyelvnek van *LL*(**k**) elemezhető nyelvtana, de nincsen **k-1** előretekintéssel elemezhető grammatikája.

Vizsgáljuk a következő nyelvet:

$$a^i (b \cup b^k c)^i$$

Ennek egy lehetséges grammatikája:

1 $A \to aAB$ **2** $A \to aB$ **3** $B \to b$ **4** $B \to b^k c$

Ha ennek a nyelvnek a mondatait elemezzük, akkor nincsen semmi gondunk, amíg az *a* karakterek vannak soron. A *b* karakterek esetében már problémás a helyzet. Annak érdekében, hogy a **3** és **4** szabály között dönteni tudjunk, meg kell tekintenünk a vizsgálat helyétől számított **k+1** sorszámú karaktert. Amennyiben az *b*, akkor a **3,** ha viszont *c*, a **4** szabályt kell választanunk. Ez **k+1** hosszúságú előretekintést jelent.

Minthogy azonban a nyelvtanban nincsen ε-szabály, bizonyosak lehetünk abban, hogy az előretekintés mértéke csökkenthető. Valóban itt alkalmazható a faktorizálás, és ennek eredményeképpen megkapjuk az *LL*(**k**) nyelvtant:

1 $A \to aAB$ **2** $A \to aB$ **3** $B \to bC$ **4** $C \to b^{k-1}c$ **5** $C \to \varepsilon$

Ezzel nem csak azt demonstráltuk, hogy a nyelvek a balelemezhetőség szerint az előretekintési hossz nagysága szerint egy végtelen egyre szűkülő halmazsorozatot alkotnak, hanem arra is láttunk példát, hogyan lehet mindaddig faktorizálni, ameddig egy ε-szabály ezt meg nem akadályozza.

Még egy megjegyzés. Amikor a FIRST függvények diszjunkt voltát mint szükséges, de nem elégséges feltétel említettük, akkor kicsit erőltetettnek tűnt az a félelem, hogy egy nem kellően hosszú FIRST mondat éppen úgy folytatódik, hogy egy másik FIRST függvény elemét szolgáltatja. Ott említettük, hogy ha van ε-szabály, akkor ez nem is annyira valószínűtlen. Most viszont látjuk, hogy a **k** előretekintés csökkentése érdekében gyakran célszerű az ε-szabályok bevezetése, ez a probléma tehát igenis gyakorlati szempontból is fontos.

5.3. A jobbelemzés, *LR*(k) nyelvek

Idézzük fel pár szóval az alulról felfelé történő, vagy másképp jobbelemzést.

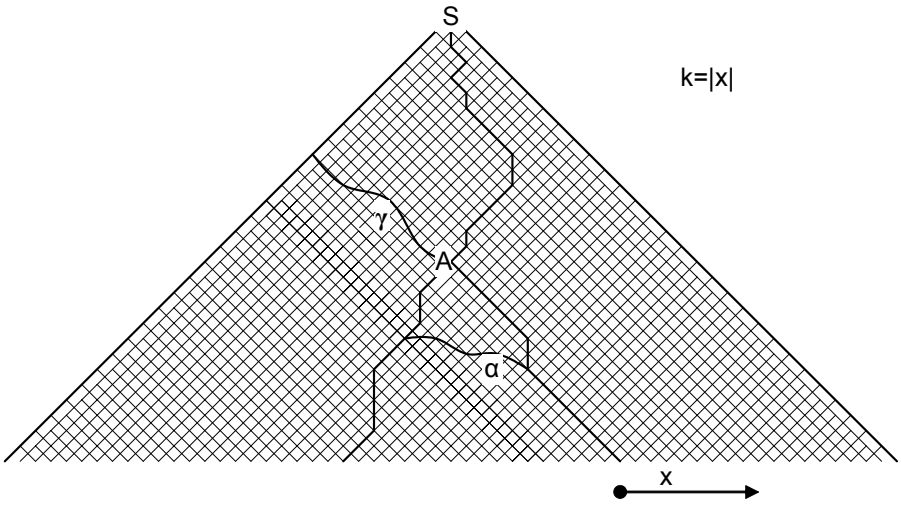
A mondatjelölt szimbólumait olvasva azokat rendre a veremben helyezzük el. Közben minden alkalommal megvizsgáljuk, nem alakult-e ki a veremben egy olyan fragmens, amely egy levezetési szabály jobboldala, és így egy potenciális nyél. Amennyiben ez a fragmens valóban nyélnek bizonyul, akkor letörjük, vagyis helyébe a szabály baloldalán álló nemterminálist írjuk.

Az alulról felfelé történő elemzésnek tulajdonképpen ez az egész tudománya. A következő szimbólumot átcsúsztatjuk a verembe, megvizsgáljuk keletkezett-e nyél, ha igen, letörjük, redukáljuk, ha nem, akkor áttérünk a következő szimbólumra. Az algoritmus angol elnevezése *shift-reduce* elég világosan és tömören fejezi ki ezt a stratégiát.

Az egész algoritmus kritikus pontja a nyél létének megállapítása. A korábbi példákból ugyanis már tudjuk, hogy egy jobboldallal való azonosság még nem jelenti feltétlenül azt, hogy nyél az illető.

Ezen kívül, ha több levezetési szabálynak azonos a jobboldala, vagy jobboldaluk azonosan végződik, akkor azt is el kell döntenünk, melyik szabály szerint végezzük el a letörést.

Ezen kérdések eldöntésében segít, ha kissé előre lapozunk, előretekintünk, és ezt az információt is felhasználjuk. Az *LR*(**k**) nyelvtanok esetében **k** szimbólummal előretekintve döntünk a nyéljelölt ügyében. A gondolatot az 5.2. ábra szemlélteti



5. 2. ábra

Az ábrán a *wx* mondat jobbodali levezetésének egy állapota van feltüntetve.

$$S \overset{*}{\Rightarrow} \gamma Ax \Rightarrow \gamma\alpha x \overset{*}{\Rightarrow} wx$$

Mint ismeretes, a jobboldali elemzés fordított irányban halad végig a jobboldali levezetésen. Adott esetben azt, hogy az $A \rightarrow \alpha$ levezetési szabály jobboldalával azonos α fragmenst nyélnek tekintsük-e vagy sem, és azt az A nemterminálissá törjük-e le, a $\mathrm{FIRST}_{\mathbf{k}}(x)$ alapján döntjük el.

Egy nyelvtan akkor *LR*(**k**) nyelvtan, ha a vélelmezett nyél után álló **k** terminális szimbólum alapján egyértelműen eldönthető, hogy valódi nyélről van-e szó, és ha igen, melyik szabályt alkalmazva kell azt letörni.

Az elnevezésben az első betű *L* most is a *left* rövidítése, hiszen ebben az esetben is balról jobbra olvasva elemzünk. Az *R – right* természetesen a jobb-elemzésre utal.

Legyen egy nyelvtan két levezetési szabálya

$$A \rightarrow \alpha \qquad \text{és} \qquad B \rightarrow \beta$$

legyen továbbá két lehetséges jobboldali levezetés,

$$S \overset{*}{\Rightarrow} \gamma Ax \Rightarrow \gamma\alpha x$$

$$S \overset{*}{\Rightarrow} \delta By \Rightarrow \delta\beta y \qquad (5.19.)$$

melyekben az utolsó lépést megelőzően azonos helyettesítéseket tettünk, tehát

$$\gamma\alpha = \delta\beta$$

akkor *LR*(**k**) nyelvtan esetében a

$$\mathrm{FIRST}_{\mathbf{k}}(x) = \mathrm{FIRST}_{\mathbf{k}}(y)$$

azonosságból következnek az

$$\alpha = \beta, \quad \gamma = \delta \quad \text{és az} \quad A = B \qquad (5.20.)$$

azonosságok is.

A vázlatos ábrából még az is kitűnik, hogy a jobboldali levezetés mondatszerű formái a verem tartalmából és a jelsorozat még el nem olvasott részéből rakhatóak össze. A verem tartalma tehát ezen mondatszerű formák prefixuma. Ebben a prefixumban valódi nyél csakis a jobboldali szélen lehet. Az elemzés során ugyanis nem megyünk el szótlanul a valódi nyelek mellett, hanem azonnal letörjük azokat.

Ez persze nem jelenti azt, hogy a prefixum belsejében ne lehetne olyan fragmens, amely megegyezik egy jobboldallal. Erről a fragmensről azonban annak idején az előretekintés figyelembe vételével lefolytatott részletes vizsgálatnak ki kellett mutatnia, hogy az nem valódi nyél.

Az elemzés mindaddig sikerrel kecsegtet, amíg szerkeszthető a prefixumhoz egy olyan folytatás, amely azt egy jobboldali levezetés mondatszerű formájává egészíti ki. Az ilyen prefixumokat életképes prefixumoknak nevezzük.

Az elemzés során egy adott szituációban a követendő eljárás a prefixumtól és az előretekintéssel nyert jelsorozattól függ. Ennek megfelelően az elemző tábla vízszintes fejlécében – éppúgy, mint a balelemző táblában – az előretekintéssel kapott jelsorozatok állanak. Az egyes rovatok, vagyis a függőleges fejléc bejegyzései az életképes prefixumoknak felelnek meg.

Az óvatos fogalmazás mindenképpen helyénvaló, mert vannak olyan nyelvtanok, ahol az életképes prefixumok száma nem korlátos. Ilyen például az

$$S \to aS \mid a$$

nyelvtan, ahol az

$$aaa \ldots a \qquad \text{és az} \qquad aaa \ldots aS$$

alakú jelsorozatok mind életképes prefixumok.

Persze „szerencsére” lehetnek olyan különböző prefixumok, amelyek esetében az elemzés során követendő eljárás azonos. Tekintsük ezeket az életképes prefixumokat ekvivalenseknek. Könnyű belátni, hogy ez a reláció valóban ekvivalencia reláció. Ennek megfelelően az életképes prefixumokat ekvivalenciaosztályokba sorolhatjuk, és ekvivalenciaosztályonként csak egy rovatot alkalmazunk.

Később látni fogjuk, hogy az ekvivalenciaosztályok száma mindig véges, így elemző táblánknak is mindig véges rovata lesz.

Minden életképes prefixumhoz – pontosabban ekvivalenciaosztályhoz – elemeket fogunk rendelni. Ezek az elemek mind szintakszisukat, mind szemantikájukat tekintve emlékeztetnek az *Earley* algoritmusnál alkalmazott elemekre, jóllehet az ottani meg az itteni elemek nem azonosak.

Valamely életképes prefixumhoz, illetve ekvivalenciaosztályhoz tartozó elem alakja:

$$\{ A \to \alpha_1 \bullet \alpha_2, L \} \tag{5.21.}$$

ahol az $A \to \alpha_1\alpha_2$ a nyelvtan egy produkciós szabálya, L pedig egy legfeljebb **k** hosszúságú jelsorozatokból álló véges nyelv.

Az elem értelmezése, szemantikája a következő.

Az adott prefixum néhány utolsó, közelebbről meg nem határozott számú szimbóluma az α_1 jelsorozatból generálódik, és a mondatban az $A \rightarrow \alpha_1\alpha_2$ szabály által generált fragmenst követő első **k** terminális az L nyelv valamelyik szava.

Tételezzük most fel, hogy egy adott életképes prefixumhoz valamilyen módon sikerült megszerkesztenünk az összes hozzátartozó elemet.

Két kérdést kell megvizsgálnunk. Miképpen állapíthatjuk, meg milyen életképes prefixumokat kell még figyelembe vennünk, és hogyan következtethetünk az elemekből az elemzés során követendő lépésekre.

Ami az első kérdést illeti, a prefixum csakis olyan szimbólummal folytatódhat, lett légyen az terminális vagy nemterminális, amely a prefixumhoz tartozó elemekben közvetlenül a pont után áll. Ez megadja, milyen új prefixumot kell számításba vennünk.

Az új prefixumokhoz rendelt csoportoknak éppen ezek az elemek lesznek alapító tagjai, természetesen azzal az eltéréssel, hogy itt a pont eggyel tovább kerül, vagyis átugorja a pontot követő szimbólumot. Ez a lépés önmagáért beszél, és külön magyarázatot nem igényel.

Amennyiben az így származtatott elemekben a pont után nemterminális áll, – és ebben a követendő eljárás megegyezik az *Earley* algoritmusban követettel, – akkor ezen nemterminálisok valamennyi levezetési szabályából új elemet kell képeznünk, a pontot a jobboldal elé helyezve, és megállapítva a követő nyelvet.

Egy prefixumot követően azokat a terminálisokat csúsztathatjuk – számítástechnikai argóban siftelhetjük, – amely a prefixumhoz tartozó csoport elemeiben a pont után áll. Ez annyit jelent, hogy csak akkor csúsztathatunk, ha az ily módon kialakuló prefixum továbbra is életképes marad.

Amennyiben a pontot sikerült áttuszkolnunk a teljes jobboldalon, és így az most már a levezetési szabály után áll, akkor ez annyit jelent, hogy a prefixum nyélben végződik, ami letörhető, feltéve, hogy az előretekintéssel nyert jelsorozat eleme a L követő nyelvnek.

Ennek alapján a leírt módon haladhatunk prefixumról prefixumra. Az eljárás garantálja, hogy a prefixumok mindig életképesek lesznek. Amennyiben két különböző prefixum csoportja megegyezik, akkor ez annak a jele, hogy a két prefixum azonos ekvivalenciaosztályba tartozik, és nem kell az elemző táblában új rovatot nyitnunk.

A csoport elemei azt is megmondják, milyen terminálisokat van jogunk a verembe átcsúsztatni, és mikor és milyen szabály alapján lehet az adott nyelet letörni.

A gondot itt is az indulás jelenti. Szerencsére van egy olyan prefixum, az üres jelsorozat, ε, amely minden nyelvtan minden mondatszerű formájának prefixumaként felfogható. Nos, mindenkor ezzel az életképes prefixummal kezdjük a vizsgálatot.

Gondot jelenthet még az elfogadás, az akceptálás felismerése. Ennek érdekében vezessünk be egy új mondatszimbólumot, és egy új **0** sorszámú produkciós szabályt:

$$\mathbf{0} \qquad \hat{S} \to S \tag{5.22.}$$

ahol S az eredeti, míg $\hat{S}$ az új mondatszimbólum. Ezzel természetesen a generált nyelv nem változik, pusztán annyi történik, hogy minden levezetés a fenti (5.22.) szabállyal kezdődik, de egyébként változatlan.

Minthogy ez a szabály áll minden levezetés elején, ha a jobboldali felvezetés során eljutunk ehhez a szabályhoz és letörjük, akkor ez annyit jelent, hogy a jelsorozatot el kell fogadnunk. Ezen szabály szerinti letörés lesz tehát az akceptálás jele.

Most akár el is kezdhetjük a különböző életképes prefixumhoz tartozó csoportok kialakítását. Az első, vagyis a **0** sorszámhoz tartozó, csoport első, alapító eleme az új mondatszimbólum egyetlen produkciós szabálya lesz, végül az L követő nyelv nyilván szintén az üres jelsorozat, ε, hiszen induláskor semmi sem áll a mondatszimbólum után.

A világ kezdete tehát minden nyelvtannál:

$$\{ \hat{S} \to \bullet S, \varepsilon \} \tag{5.23.}$$

A jelölés egyszerűsítése érdekében engedjük meg azt a lazaságot, hogy a követő nyelvet ne tegyük az ismert „halmazzárójelek” közé, és a nyelv mondatait a *vagy* jelentésű | jellel választuk el.

Úgy hiszem, most már kellő ismerettel rendelkezünk ahhoz, hogy egy $LR(\mathbf{k})$ elemző táblát megszerkesszünk. Próbálkozzunk $LR(\mathbf{1})$ elemzővel, és válasszuk most a posztfix lengyel jelölést generáló nyelvtant. Ez azt is demonstrálja, hogy balrekurzív nyelvtanok is jobbelemezhetőek.

$$E \to EE+ \mid EE* \mid a$$

Az életképes prefixumokat az irodalomban – ki tudja miért – a $\boldsymbol{T}$ betűvel, és megkülönböztető indexszel jelölik. Ezek szerint első prefixumunk

$$\varepsilon \Rightarrow \boldsymbol{T}_0$$

A többi prefixum, pontosabban prefixum osztály esetében azt a jelölést alkalmazzuk, hogy a származtató prefixumhoz hozzáfűzzük az új szimbólumot.

A $\boldsymbol{T}_0$ prefixumhoz tartozó csoport alapító eleme:

$$\{ \hat{E} \to \bullet E, \varepsilon \}$$

Minthogy az E nemterminális előtt pont áll, ehhez hozzá kell fűznünk az alábbi elemeket:

$$\{ E \to \bullet EE+, \varepsilon \} \qquad \{ E \to \bullet EE*, \varepsilon \} \qquad \{ E \to \bullet a, \varepsilon \}$$

Minthogy ezek az elemek az első alapító elem utolsó szimbóluma, az E nemterminális helyére kerülnek, így követő nyelveik megegyeznek.

Ezekben az új elemekben azonban az E nemterminális előtt ismét pontot találunk, így további elemeket fűzhetünk hozzá a csoporthoz. Persze ezekben az új elemekben is ugyanazok a levezetési szabályok szerepelnek majd, más lesz azonban a követő nyelv. Itt ugyanis a felbontandó nemterminális, vagyis az első E után egy második E áll, amely végső soron az a terminális szimbólumot generálja.

$\{ E \to \bullet EE+, a \}$ $\{ E \to \bullet EE*, a \}$ $\{ E \to \bullet a, a \}$

Ezekben az elemekben megint találhatunk a pont után nemterminálist. Természetesen rekurzívan ismét új elemeket vezethetünk, illetve csak vezethetnénk be, hiszen kiderül, hogy a bevezetendő elemek már benne vannak a csoportban.

Tovább egyszerűsíthetjük a jelölést, ha azokat az elemeket, amelyek ugyanabból a levezetési szabályból származnak, és ráadásul a pont helyzete is azonos, összevonjuk. Az összevont elemben természetesen a követő nyelv a két nyelv uniója lesz.

Ennek szellemében megszerkeszthetjük a prefixum osztályokhoz tartozó csoportokat, és ennek alapján az elemző táblát.

$\varepsilon \Rightarrow \boldsymbol{T}_0$

$\{ \hat{E} \to \bullet E, \varepsilon \}$
$\{ E \to \bullet EE+, \varepsilon \mid a \}$
$\{ E \to \bullet EE*, \varepsilon \mid a \}$
$\{ E \to \bullet a, \varepsilon \mid a \}$

$\boldsymbol{T}_0 E \Rightarrow \boldsymbol{T}_1$

$\{ \hat{E} \to E\bullet, \varepsilon \}$
$\{ E \to E\bullet E+, \varepsilon \mid a \}$
$\{ E \to E\bullet E*, \varepsilon \mid a \}$
$\{ E \to \bullet EE+, + \mid * \mid a \}$
$\{ E \to \bullet EE*, + \mid * \mid a \}$
$\{ E \to \bullet a, + \mid * \mid a \}$

$\boldsymbol{T}_1 E \Rightarrow \boldsymbol{T}_3$

$\{ E \to EE\bullet+, \varepsilon \mid a \}$
$\{ E \to EE\bullet*, \varepsilon \mid a \}$
$\{ E \to E\bullet E+, + \mid * \mid a \}$
$\{ E \to E\bullet E*, + \mid * \mid a \}$
$\{ E \to \bullet EE+, + \mid * \mid a \}$
$\{ E \to \bullet EE*, + \mid * \mid a \}$
$\{ E \to \bullet a, + \mid * \mid a \}$

$\boldsymbol{T}_0 a \Rightarrow \boldsymbol{T}_2$

$\{ E \to a\bullet, \varepsilon \mid a \}$

$\boldsymbol{T}_1 a \Rightarrow \boldsymbol{T}_4$

$\{ E \to a\bullet, + \mid * \mid a \}$

$\boldsymbol{T}_3 a \Rightarrow \boldsymbol{T}_4$

$\boldsymbol{T}_3 E \Rightarrow \boldsymbol{T}_5$

$\{ E \to EE\bullet+, + \mid * \mid a \}$
$\{ E \to EE\bullet*, + \mid * \mid a \}$
$\{ E \to E\bullet E+, + \mid * \mid a \}$
$\{ E \to E\bullet E*, + \mid * \mid a \}$
$\{ E \to \bullet EE+, + \mid * \mid a \}$
$\{ E \to \bullet EE*, + \mid * \mid a \}$
$\{ E \to \bullet a, + \mid * \mid a \}$

$\boldsymbol{T}_3 + \Rightarrow \boldsymbol{T}_6$

$\{ E \to EE+\bullet, \varepsilon \mid a \}$

$\boldsymbol{T}_3 * \Rightarrow \boldsymbol{T}_7$

$\{ E \to EE*\bullet, \varepsilon \mid a \}$

$\boldsymbol{T}_5 a \Rightarrow \boldsymbol{T}_4$

$\boldsymbol{T}_5 E \Rightarrow \boldsymbol{T}_5$

$\boldsymbol{T}_5 + \Rightarrow \boldsymbol{T}_8$

$\{ E \to EE+\bullet, + \mid * \mid a \}$

$\boldsymbol{T}_5 * \Rightarrow \boldsymbol{T}_9$

$\{ E \to EE*\bullet, + \mid * \mid a \}$

Minthogy a $\boldsymbol{T}_3$ és a $\boldsymbol{T}_5$ prefixummal az a terminális olyan csoportot eredményez, amely egy már előfordult prefixuméval azonos, ezt újólag nem írtuk le. Hasonló a helyzet abban az esetben, amikor a $\boldsymbol{T}_5$ prefixumot az E nemterminális követi.

Valójában az így kialakuló prefixumok viselkedésükben megegyeznek más, már korábban szerepelt prefixumokkal. Ez az első két prefixum esetében a $\boldsymbol{T}_4$, a harmadikában pedig a $\boldsymbol{T}_5$ életképes prefixum. Mint említettük, az életképes prefixumok ekvivalenciaosztályokat alkothatnak, és az említett prefixumok egy-egy életképes prefixum osztály reprezentánsai.

Nyelvtanunknak – könnyen beláthatóan – végtelen sok életképes prefixuma lehet. Minthogy a prefixumokhoz tartozó csoportokban véges számú produkciós szabály hozhat létre elemeket, ezekben a pont véges számú helyzetet vehet fel, végül a lehetséges követő nyelvek száma is véges, csak véges számú különböző csoport képzelhető el. Ennek következtében a prefixum ekvivalenciaosztályok száma véges.

Az életképes prefixumokhoz tartozó elemek alapján az elemző tábla megszerkeszthető. Ez két részből áll. Az első, az úgynevezett tevékenységi tábla – *action table* – mondja meg, hogy egy adott szituációban, adott életképes prefixum és adott előretekintés mellett mi a teendő.

Négyféle tevékenység lehetséges, csúsztatás, letörés, akceptálás és hibajelzés. A csúsztatást és az elfogadást az *S* és *A* betűk – *shift, accept* – a letörést a levezetési szabály sorszáma, míg a hibajelzést az üres mező jelöli.

A tábla második fele az úgynevezett ugró tábla – *go to table.* Ez megmondja, hogy az érvényes életképes prefixumhoz egy újabb szimbólumot fűzve milyen sorszámú prefixumot kapunk. Az elemzés következő lépését aztán ezen prefixum sora alapján kell megállapítanunk.

Mindezek előrebocsátása után az adott nyelvtan jobboldali elemző táblája a következő:

	Tevékenységi tábla				Ugró tábla			
	a	+	*	ε	*E*	*a*	+	*
$\boldsymbol{T}_0$	*S*				$\boldsymbol{T}_1$	$\boldsymbol{T}_2$		
$\boldsymbol{T}_1$	*S*			*A*	$\boldsymbol{T}_3$	$\boldsymbol{T}_4$		
$\boldsymbol{T}_2$	**3**			**3**				
$\boldsymbol{T}_3$	*S*	*S*	*S*		$\boldsymbol{T}_5$	$\boldsymbol{T}_4$	$\boldsymbol{T}_6$	$\boldsymbol{T}_7$
$\boldsymbol{T}_4$	**3**	**3**	**3**					
$\boldsymbol{T}_5$	*S*	*S*	*S*		$\boldsymbol{T}_5$	$\boldsymbol{T}_4$	$\boldsymbol{T}_8$	$\boldsymbol{T}_9$
$\boldsymbol{T}_6$	**1**			**1**				
$\boldsymbol{T}_7$	**2**			**2**				
$\boldsymbol{T}_8$	**1**	**1**	**1**					
$\boldsymbol{T}_9$	**2**	**2**	**2**					

A korábbiak alapján remélem, nem okoz gondot az elemzési tábla származtatása.

Használatát mutassuk be egy példán. Legyen az elemzendő mondat

aaa+∗

Az elemzés során beírjuk az eredménybe az életképes prefixumokat is, hiszen ezek ismeretére szükség van a tevékenység megállapítására. Természetesen ezek a szimbólumok csak tájékoztatásra szolgálnak, és valójában nem részei a verem tartalmának.

Az induló prefixum, mint tisztáztuk, az üres jelsorozat, vagyis $\boldsymbol{T}_0$

$\{ aaa{+}{*}, \boldsymbol{T}_0, \varepsilon \} \mapsto \{ aa{+}{*}, \boldsymbol{T}_0 a \boldsymbol{T}_2, \varepsilon \} \mapsto \{ aa{+}{*}, \boldsymbol{T}_0 E \boldsymbol{T}_1, \mathbf{3} \} \mapsto$

$\mapsto \{ a{+}{*}, \boldsymbol{T}_0 E \boldsymbol{T}_1 a \boldsymbol{T}_4, \mathbf{3} \} \mapsto \{ a{+}{*}, \boldsymbol{T}_0 E \boldsymbol{T}_1 E \boldsymbol{T}_3, \mathbf{33} \} \mapsto$

$\mapsto \{ {+}{*}, \boldsymbol{T}_0 E \boldsymbol{T}_1 E \boldsymbol{T}_3 a \boldsymbol{T}_4, \mathbf{33} \} \mapsto \{ {+}{*}, \boldsymbol{T}_0 E \boldsymbol{T}_1 E \boldsymbol{T}_3 E \boldsymbol{T}_5, \mathbf{333} \} \mapsto$

$\mapsto \{ {*}, \boldsymbol{T}_0 E \boldsymbol{T}_1 E \boldsymbol{T}_3 E \boldsymbol{T}_5 {+} \boldsymbol{T}_8, \mathbf{333} \} \mapsto \{ {*}, \boldsymbol{T}_0 E \boldsymbol{T}_1 E \boldsymbol{T}_3, \mathbf{3331} \} \mapsto$

$\mapsto \{ \varepsilon, \boldsymbol{T}_0 E \boldsymbol{T}_1 E \boldsymbol{T}_3 {*} \boldsymbol{T}_7, \mathbf{3331} \} \mapsto \{ \varepsilon, \boldsymbol{T}_0 E \boldsymbol{T}_1, \mathbf{33312} \} \mapsto$ *accept*

Annak megállapítása, hogy létezik-e olyan **k**, amivel egy nyelvtan *LR*(**k**) nyelvtan, algoritmikusan eldönthetetlen feladat. Rögzített **k** esetében viszont kaphatunk választ. Az ismert módon elkezdjük az életképes prefixumokhoz tartozó elemek felírását, és ha ezek alapján az elemző tábla megszerkeszthető, más szóval nincsen benne ellentmondás, akkor a nyelvtan *LR*(**k**) nyelvtan.

5.4. Egyszerűsített jobbelemzés

Az *LR*(**k**) elemzők nagyon hatásosak abból a szempontból, hogy az esetleges hibát azonnal észlelik. Amennyiben ugyanis egy „rossz” szimbólum következik, ez azonnal életképtelenné teszi a prefixumot, és így a hiba napfényre kerül.

Az ilyen elemzők kedvezőtlen velejárója viszont, hogy eléggé terjedelmesek. Még az olyan egyszerű nyelvtannak is, mint amilyen a posztfix lengyel jelölést generáló, tíz soros táblája kerekedett. Ennek megfelelően az ilyen elemzők memória igénye és futási időtartama viszonylag nagy. Ennek a hely- és időigénynek a csökkentésére egy sereg egyszerűsítő eljárás született, ami bizonyos esetekben előnyösen alkalmazható.

Gyors és kis helyigényű módszer a precedencia elemzés. Lényege abban áll, hogy a csúsztatás vagy letörés hamleti kérdésére nem a teljes életképes prefixum alapján kíván válaszolni, hanem megelégszik annak néhány utolsó karakterével. Legegyszerűbb ez az elemző akkor, ha csak egyetlen karaktert veszünk a prefixumból figyelembe. Ha ugyanakkor az előretekintés is egyetlen karakter, akkor egyszerű precedencia elemzőről beszélünk.

Tételezzük fel, hogy egyszerű precedencia elemzőt kívánunk szerkeszteni. Vizsgáljuk meg a prefixum utolsó és az el nem olvasott szöveg első karakteréből álló szimbólumpárokat. Állapítsuk meg, hogy a csúsztatás vagy letörés tekintetében milyen kapcsolatban, milyen *Wirth–Weber* precedencia relációban vannak egymással.

Négy ilyen reláció képzelhető el. Az első szimbólummal záródik egy nyél, a két szimbólum egy nyél két szomszédos eleme, a második szimbólum egy nyél kezdő karaktere, míg végül nem keletkezhet olyan mondatszerű forma a jobboldali elemzés során, amelyben a két szimbólum egymás mellett állna.

Adott nyelvtan esetében, jobboldali elemzést feltételezve, ezeket a relációkat kis gyakorlattal igen gyorsan és könnyen megállapíthatjuk.

Ezeket a precedencia relációkat egy mátrixban szokás ábrázolni. A mátrix sorai és oszlopai a nyelvtan szimbólumainak felelnek meg. Sor-oszlop sorrendet feltételezve a mátrix megfelelő mezejébe írjuk az érvényes precedencia jelét. A nyél eleje, nyél közepe és nyél vége relációkat rendre a kissé módosított, <• , $\underline{\bullet}$ és •> reláció jelek jelölik, míg inkompatibilitás esetén a mező üresen marad.

Lássunk egy példát. Legyen a nyelvtan

$$S \rightarrow aSc \mid ac \mid bSd \mid bd$$

Ennek a nyelvtannak a következő precedencia mátrixa van:

	S	*a*	*b*	*c*	*d*	ε
S				$\underline{\bullet}$	$\underline{\bullet}$	
a	$\underline{\bullet}$	<•	<•	$\underline{\bullet}$		
b	$\underline{\bullet}$	<•	<•		$\underline{\bullet}$	
c				•>	•>	•>
d				•>	•>	•>
ε		<•	<•			

Természetesen mind a vízszintes, mind a függőleges fejlécen szerepel az ε mint szimbólum. Ha a veremben nincs még, vagy már átcsúsztatott szimbólum, akkor ezt az üres szimbólummal azonosítjuk. Ugyanakkor, amikor a jelsorozatot teljesen elolvastuk, a még el nem olvasott jelsorozat szintén az üres jelsorozat lesz. Magától értetődő, hogy az ε-ε szimbólumpár az elfogadást, akceptálást jelöli. Ilyenkor ugyanis a vermet kiürítettük, és a teljes jelsorozatot is elolvastuk.

Az ilyen precedencia mátrix használata, a precedencia elemző működése roppant egyszerű.

A jelsorozat olvasásakor állandóan figyeljük a szimbólumpároknak megfelelő precedencia relációkat. Ha a reláció nyél eleje vagy nyél közepe, akkor csúsztatunk.

A nyél vége reláció esetében a nyelet le kell törni. A nyelet, pontosabban potenciális nyelet nagyon könnyen ki tudjuk választani. Hiszen az csak egy nyél eleje és nyél vége közötti jelsorozat lehet, amely természetesen tartalmazhat nyél közepe relációkat is. Ha ugyan nyél ez a potenciális nyél! Sajnos előfordulhat ugyanis, hogy az így kijelölt jelsorozat éppenséggel nem azonos egyik levezetési szabály jobboldalával. Ez esetben hibajelzést kell adnunk.

Zavarban lennénk akkor is, ha a megtalált nyél egynél több levezetési szabály jobboldalára illik. Ez persze csak akkor fordulhat elő, ha a nyelvtanban vannak azonos jobboldalak, ilyenkor azt mondjuk, hogy a nyelvtan nem egyértelműen invertálható. Az ilyen nyelvtanokat nem lehet a precedencia módszerrel elemezni. Akkor sem lehet a nyelvtant precedencia elemezni, ha ε-szabályt tartalmaz.

Ha a mező üres, tehát a szimbólumpár inkompatibilis, hibajelzést adunk.

Gyakorlásképpen elemezzük az előbbi nyelv egy mondatát. Precedencia elemzés esetén a konfigurációt célszerű más sorrendben – veremtartalom, elemzendő szöveg, eredmény – megadni, ilyenkor ugyanis, ha ráadásul a verem legfelső elemét a jobboldalra írjuk, a kérdéses szimbólumpár éppen egymás mellé kerül.

abdc

$\{ \varepsilon, abdc, \varepsilon \} \mapsto \{ a, bdc, \varepsilon \} \mapsto \{ ab, dc, \varepsilon \} \mapsto \{ abd, c, \varepsilon \} \mapsto$

$\mapsto \{ aS, c, \mathbf{4} \} \mapsto \{ aSc, \varepsilon, \mathbf{4} \} \mapsto \{ S, \varepsilon, \mathbf{41} \} \mapsto accept$

A levezetési szabályok sorszámát a megadás sorrendjében használtuk.

A precedencia mátrix felhasználásával kapcsolatosan egy sereg kérdés merülhet fel. Az adott esetben, példánkban, a mátrix minden mezejébe legfeljebb egy relációjel került. Szükségszerű ez?

Sajnos egyáltalában nem. Ha egy nyelvtan egyértelműen invertálható, nem tartalmaz ε-szabályt, és a precedencia relációk diszjunktak, akkor a nyelvtan precedencia nyelvtan. Amennyiben, mint esetünkben, elegendő egy-egy szimbólumot vizsgálnunk, akkor egyszerű precedencia nyelvtanról, illetve elemzőről beszélünk.

Nyilvánvaló, hogy az egyszerű precedencia nyelvtanok az *LR*(**1**) nyelvtanoknak valódi részhalmazát képezik. Ami az előretekintést illeti, mindkét elemző ugyanazt az információt használja fel. Azonban míg az *LR*(**k**) elemzők, és így az *LR*(**1**) elemzők is a múltról, az életképes prefixumról minden lehetséges információt figyelembe vesznek, addig a precedencia elemzők megelégszenek a prefixum utolsó **k** szimbólumának vizsgálatával, egyszerű precedencia elemző esetén pedig a prefixum utolsó szimbólumának vizsgálatával. Olykor ez elegendő az elemzéshez, máskor viszont nem.

Mint említettük, a precedencia elemzők előnye egyszerűségükben van. A példánkban szereplő, és bonyolultnak éppen nem mondható nyelvtannak 26 életképes prefixum osztálya van. Az *LR*(**k**) elemző ezért összehasonlíthatatlan terjedelmesebb és lassúbb, mint a precedencia elemző.

Semmi sem tökéletes azonban, Vegyük például a következő jelsorozatot:

abdd

Ez nyilván nem eleme a nyelvnek. Az *LR*(**1**) elemző ezt azonnal észreveszi, mihelyt a „hibás” szimbólum, a második *d* látótávolságon belül kerül. Nem így a precedencia elemző. Ez szó nélkül átcsúsztatja ezt a karaktert is, és csak akkor eszmél rá a hibára, amikor a precedencia relációk alapján kiadódó potenciális nyelet nem találja a levezetési szabályok jobboldala között.

Vegyük azonban ebből a szempontból még érzékenyebb nyelvtant:

$$S \to ab \mid bc \mid cd$$

Itt a precedencia elemző az

$$abcd$$

jelsorozat elemzésekor szemrebbenés nélkül elgyalogol egészen az utolsó karakterig, és csak akkor ad hibajelzést.

Mint említettük az általános precedencia elemző esetén mind a visszatekintés, mind az előretekintés hossza egynél nagyobb lehet. Így beszélhetünk (**m,n**) precedencia nyelvtanokról illetve elemzőkről, ahol **m** a visszatekintés, **n** pedig az előretekintés hossza. Magától értetődő, hogy az (**m,n**) precedencia nyelvtanok az *LR*(**n**) nyelvtanok általában valódi részhalmazát alkotják.

Lássunk erre az általánosított, tehát nem egyszerű precedencia nyelvtanra egy példát. Legyen ez a prefix lengyel jelölés:

$$E \to +EE \mid *EE \mid a$$

Ha megpróbálkozunk az egyszerű, vagyis (**1,1**) precedencia mátrix felírásával, akkor azt tapasztaljuk, hogy például az *E*-*a* szimbólumpárnál konfliktus helyzet adódik. Ha ugyanis a szóban forgó *E* szimbólum az operátor utáni első szimbólum, akkor itt nyél eleje van, hiszen itt az *a* terminális mint nyél majdani letörésével keletkezett *E* szimbólum egészíti ki az eddigieket nyéllé. Más a helyzet akkor, ha már két *E* szimbólum követi az operátort. Ilyenkor a mondott helyen nyilván a nyél vége reláció érvényes.

A prefix lengyel jelölés nyelvtana tehát nem egyszerű precedencia nyelvtan. Próbálkozzunk most (**2,1**) precedencia elemzéssel, amely az előbbiekből sejthetően eredményre vezet.

A precedencia mátrix függőleges fejléce most a legfeljebb két szimbólumból álló jelsorozatokat tartalmazza. Amennyiben a veremben még nincs két szimbólum, akkor a jelsorozat rövidebb. A fejlécen nem szerepel az összes lehetséges legfeljebb két karakterből álló jelsorozat, hanem csak azok, amelyek az elemzés során valóban előfordulhatnak.

A táblázatból kitűnik, hogy a precedencia relációk diszjunktak, tehát a nyelvtan (**2,1**) precedencia nyelvtan, és így nincs akadálya a (**2,1**) precedencia elemzésnek.

Természetesen nemterminális szimbólum nem lehet az előretekintés tárgya, a táblázatban mégis megjelöltük azokat a párokat is, ahol az *E* nemterminális áll a jobboldalon. Erre azért van szükség, mert a veremben a nyél megállapításánál ismernünk kell azokat a precedencia relációkat is, ahol nemterminális áll a jobboldalon.

	E	+	*	*a*	ε
ε		<●	<●	<●	
+	≐	<●	<●	<●	
*	≐	<●	<●	<●	
a					●>
++	≐	<●	<●	<●	
+*	≐	<●	<●	<●	
+*a*		●>	●>	●>	
+*E*	≐	<●	<●	<●	
*+	≐	<●	<●	<●	
**	≐	<●	<●	<●	
**a*		●>	●>	●>	
**E*	≐	<●	<●	<●	
E+	≐	<●	<●	<●	
*E**	≐	<●	<●	<●	
Ea		●>	●>	●>	●>
EE		●>	●>	●>	●>

Átgondolva a precedencia elemzés folyamatát kiderül, hogy akár nyél eleje, akár nyél belseje relációt találunk, mindkét esetben csúsztatnunk kell. Felmerül a kérdés, miért ragaszkodunk a precedencia relációk diszjunktságához, ha egyszer van két olyan reláció, ahol a teendők azonosak. Természetesen a nyél vége és bármelyik másik reláció esetében ilyen meggondolásnak nincs helye, hiszen itt konfliktus helyzetet okozna a két különböző reláció egyidejű jelenléte.

Visszatérve a nyél eleje, és nyél belseje egyidejű jelenlétére, valóban, a disztinkcióra csak később, akkor lesz szükség, amikor egy nyelet letörünk. Általában a nyél eleje reláció mondja meg, hol kezdődik a nyél. Ha diszjunktak a precedencia relációk, akkor a nyél eleje és nyél vége között csak nyél belseje relációk lehetnek. Ilyenkor egyértelmű, mit kell nyélnek, pontosabban potenciális nyélnek tekinteni.

Amennyiben megengedjük a nyél eleje és nyél belseje relációk egyidejű jelenlétét, akkor a letörés alkalmával találkozhatunk olyan szimbólumpárokkal, amelyre mind a nyél belseje, mind a nyél eleje reláció ráillik. Ilyenkor nem tudhatjuk mi a helyes. Számítsuk-e innen a nyél kezdetét, vagy menjünk tovább nyél eleje relációt keresni? Egyébként a helyzetet bonyolultabbá teheti, ha ez a reláció is kettős, vagyis nyél eleje és nyél belseje is lehet.

A helyzet azért általában nem ennyire kritikus. Gyakran, mint említettem, és mint példánkban is látható volt, a relációkkal kihasított jelsorozat nem egyezik meg egyik levezetési szabály jobboldalával sem. Az ilyen, a

precedencia relációk szabályai szerint nyélnek mutatkozó, de ilyen jobboldal hiányában a nyél szerepét betölteni képtelen fragmensek mint alternatívák szóba sem jöhetnek.

Amennyiben a nyél eleje és nyél belseje relációk nem diszjunkt volta miatt kiadódó potenciális nyelek közül legfeljebb egy egyezik meg egy jobboldallal, így csak egy jelent valódi alternatívát, akkor ez nem okoz zavart az elemzésben.

Ilyen esetekben gyenge precedencia nyelvtanokról beszélünk, szemben az erős precedencia nyelvtanokkal, ahol a precedencia relációk diszjunktak. A gyenge precedencia nyelvtanok esetében tehát a három reláció közül a nyél eleje és nyél belseje esetében bizonyos feltételek mellett nem követeljük meg a két halmaz diszjunkt voltát.

A gyenge precedenciának tett engedmények még tovább bővíthetőek. Tételezzük fel ugyanis, hogy az A és B szimbólumok között mind a nyél eleje, mind a nyél belseje relációk érvényesek. Ez esetben a verem tartalmazhat egy olyan

$$\alpha\ A\ B\ \beta$$

fragmenst, hogy az α fragmens előtt a nyél eleje, a β jelsorozat után a nyél vége reláció áll. Ráadásul lehet a nyelvtannak

$$C \to \alpha AB\beta \qquad \text{és} \qquad D \to B\beta$$

alakú levezetési szabálya is. Itt tehát nem teljesül az a megkötés, hogy a potenciális nyelek közül legfeljebb egy jelent valóságos alternatívát.

Amennyiben azonban az A és D szimbólumok között semmilyen reláció sem érvényes, akkor ez a tény megoldja dilemmánkat. Minthogy az A és D között nincsen reláció, a rövidebb nyél letörésével olyan jelsorozat alakulna ki, amely nem lehet része egyetlen jobboldali levezetéssel előálló mondatszerű formának sem, és így szükségképpen zsákutca.

Ha egy nyelvtan esetében a fenti feltételek teljesülnek, és mindig csak a hosszabb nyéljelölt letörése vezethet eredményre, akkor ezt a nyelvtant is gyenge precedencia nyelvtannak tekintjük. Ilyenkor persze tudnunk kell, hogy dilemma esetében mindig a hosszabb nyelet kell letörni.

A gyenge precedencia nyelvtanra jó példa az ismert és oly gyakran hivatkozott, az infix jelölésű aritmetikai kifejezéseket generáló nyelvtan.

Mint a megadott precedencia mátrixból kitűnik, a nyél eleje és nyél belseje relációk két helyen is ütköznek. Valóban, a (és az E szimbólum között mindkét reláció elképzelhető. Előadódhat ugyanis

$$(E) \qquad \text{és} \qquad (E+T)$$

konstrukció is.

Hasonló a helyzet a + operátor és a T nemterminális esetében. Itt is megadjuk azt a két esetet, amikor különböző relációk lesznek érvényesek:

$$E+T \qquad \text{és} \qquad E+T*F$$

A teljes precedencia mátrix a következő:

	E	*T*	*F*	(	*a*	)	+	*	ε
E						$\doteq$	$\doteq$		*acc*
T						•>	•>	$\doteq$	•>
F						•>	•>	•>	•>
)						•>	•>	•>	•>
a						•>	•>	•>	•>
(	$\doteq$ <•	<•	<•	<•	<•				
+		$\doteq$ <•	<•	<•	<•				
*			$\doteq$	<•	<•				
ε	<•	<•	<•	<•	<•				

Közelebbről megvizsgálva a két konfliktus helyzetet azt tapasztaljuk, hogy mindkettő, ha másképpen is, de eleget tesz a gyenge precedencia követelményeinek.

A kezdőzárójel (és *E* páros esetében, ahol a két reláció a (*E*) illetve a (*E*+ esetekben fordulhat elő, a relációjelekkel leválasztott potenciális nyelek közül legfeljebb egy lehet egy levezetési szabály jobboldala, így csak egyetlen letörési lehetőség jöhet szóba.

A + *T* párosnál más a helyzet. Minthogy a *T* szimbólum egymagában is egy levezetési szabály jobboldalát képviseli, az *E*+*T* jelsorozat esetén elvben akár a teljes kifejezés, akár a *T* nemterminális egyedül is letörhető. Ha azonban a rövidebb lehetőséget választanánk, akkor két olyan szimbólum, a + és az *E* nemterminális kerülne egymás mellé, amelyek között ebben a sorrendben semmiféle reláció nem érvényes. Így a rövidebb nyéljelölt letörése útján nyert jelsorozat nem lehet egyetlen mondatszerű formának sem része.

Ennek következtében itt mindig a hosszabb nyéljelöltet kell letörnünk annak érdekében, hogy egyáltalában reményünk legyen a jobboldali levezetés megtalálására. Ez megfelel a gyenge precedencia nyelvtanok előírásainak.

A gyenge precedencia nyelvtanok mindig átalakíthatóak erős precedencia nyelvtanokká, pontosabban fogalmazva, mindig szerkeszthető egy olyan erős precedencia nyelvtan, amely ugyanazt a nyelvet generálja.

Legyen **G** egy gyenge precedencia nyelvtan, és képezzünk belőle egy **G'** nyelvtant az alábbiak szerint. Erről a nyelvtanról majd kimutatjuk, hogy erős precedencia nyelvtan.

Feleljen meg az eredeti nyelvtan minden produkciós szabályának az új nyelvtan egy produkciós szabálya az alábbiak szerint:

$$A \to \alpha \qquad \text{helyett} \qquad A \to [\alpha] \tag{5.24.}$$

ahol a szögletes zárójelbe tett jelsorozat – a szögletes zárójellel együtt – az új nyelvtanban egyetlen nemterminális szimbólumnak minősül.

Ezen újonnan bevezetett nemterminálisok felbontására definiáljuk a következő szabálytípust:

$$[X\beta] \rightarrow X[\beta] \qquad (5.25.)$$

ahol az X az eredeti nyelvtan tetszőleges, terminális vagy nemterminális szimbóluma, míg β az eredeti nyelvtan szimbólumaiból alkotott tetszőleges, akár üres jelsorozat. Természetesen a szögletes zárójelbe zárt jelsorozat itt is az új nyelvtan egyetlen nemterminális szimbóluma. Amennyiben β az üres jelsorozat, akkor ezt a szögletes zárójellel együtt elhagyjuk. Ilyenkor az új nyelvtan szimbóluma visszaváltozik az eredeti nyelvtan szimbólumává.

Könnyű belátni, hogy az új nyelvtan és a régi nyelvtan ugyanazt a nyelvet generálja. A különbség pusztán annyi, hogy míg a régi nyelvtan egy nemterminálist egyetlen lépésben bonthatott fel egy hosszabb jelsorozattá, addig itt egy kicsit nyögvenyelősen megy, minden karaktert egy külön levezetési szabállyal kell előállítanunk. Igaz, az átalakítás kissé munkaigényes, és helytakarékosnak sem mondható, de látni fogjuk, hogy páronként diszjunkttá teszi a három precedencia relációt.

Elöljáróban szögezzük le, hogy a nyél vége reláció második szimbóluma itt is, mint eddig mindig, terminális szimbólum. Ugyanakkor az új nyelvtanban a nyél belseje reláció második eleme szükségképpen szögletes zárójellel képzett új nemterminális. Ez magától értetődő, ha meggondoljuk, hogy az új nyelvtanban csak az (5.25.) alakú szabályokban szerepel a jobboldalon egynél több, pontosan két szimbólum.

Ebből rögtön következik, hogy az

$$X \doteq Y \qquad \text{és az} \qquad X \bullet\!> Y \qquad (5.26.)$$

relációk egyidejűen nem állhatnak fenn. Az Y szimbólum ugyanis nem lehet egyidejűen terminális és új nemterminális szimbólum. Ezzel a nyél belseje és a nyél vége relációk diszjunkt voltát igazoltuk.

A nyél eleje és a nyél vége relációk diszjunkt voltát a következőképpen láthatjuk be. Tegyük fel, hogy egyidejűen fennáll az

$$X <\!\bullet Y \qquad \text{és az} \qquad X \bullet\!> Y \qquad (5.27.)$$

reláció. Azt már beláttuk, hogy az Y szükségképpen terminális szimbólum, hiszen nyél vége reláció jobboldalán áll. Ugyanakkor az X szimbólumnak is hagyományos szimbólumnak kell lennie, hiszen az új, szögletes zárójellel szerkesztett nemterminálisok csakis a nyél belseje reláció jobboldalán, és a nyél vége reláció baloldalán szerepelhetnek. Itt viszont az X szimbólum a nyél eleje reláció baloldalán is áll.

Az első, tehát a nyél eleje reláció csak akkor lehet igaz, ha létezik egy

$$A \to [\ \alpha BC\beta\ ] \qquad (5.28.)$$

alakú levezetési szabály, amelyet az eredeti nyelvtan

$$A \to \alpha BC\beta$$

szabályából származtattunk.

Ezen kívül legyen

$$B \overset{*}{\Rightarrow} \delta X \text{ és} \qquad C \overset{*}{\Rightarrow} Y\gamma \qquad (5.29.)$$

levezetési sorozat, ahol a $\overset{*}{\Rightarrow}$ jelölés, mint ismeretes, arra utal, hogy a fenti levezetések tetszőleges számú, tehát akár zérus lépésben is történhetnek. Ez utóbbi esetben természetesen helytállóak lehetnek az alábbi összefüggések:

$$B = X \qquad \delta = \varepsilon \qquad C = Y \qquad \gamma = \varepsilon$$

Az (5.28.) kifejezésekben szereplő valamennyi szimbólum „hagyományos” szimbólum, így az összefüggések mind az eredeti, mind a módosított nyelvtan alapján levezethetőek.

Az (5.28.) és (5.29.) összefüggésekből következik, hogy az eredeti nyelvtanban az X és az Y szimbólumok között vagy a nyél eleje, vagy a nyél belseje relációnak fenn kell állnia. Ez utóbbinak akkor, ha az (5.29.) mindkét összefüggés zérus lépésben levezethető.

A nyél vége reláció érvényessége az új nyelvtanban viszont megkövetelné, hogy legyen egy

$$A \to [\ \alpha BD\beta] \qquad (5.30.)$$

alakú levezetési szabály, és ugyanakkor legyen lehetséges az alábbi két levezetés

$$B \overset{+}{\Rightarrow} \delta X \qquad \text{és} \qquad D \overset{*}{\Rightarrow} Y\gamma \qquad (5.31.)$$

ahol a $\overset{*}{\Rightarrow}$ jelölés, mint előbb, a tetszőleges számú, míg a $\overset{+}{\Rightarrow}$ pedig a legalább egy lépésben történő levezetést jelenti.

Ha viszont ez mind igaz, akkor az eredeti nyelvtanban is fennáll a két szimbólum között a nyél vége reláció. Ez azonban lehetetlen, hiszen az eredeti nyelvtan gyenge precedencia nyelvtan volt, ahol a nyél eleje és nyél vége illetve a nyél belseje és nyél vége relációk nem ütközhetnek.

Ennek alapján az (5.27.) feltevésünk hamis. Ez tulajdonképpen természetes, hiszen eddig csak azt bizonyítottuk, hogy ami igaz volt az eredeti nyelvtanban, igaz marad a módosítottban is.

Most azonban igazoljuk, hogy az új nyelvtanban a nyél eleje és nyél belseje relációk is diszjunktak. Tegyük fel ugyanis ennek ellenkezőjét, vagyis hogy az

$$X <\bullet\ Y \qquad \text{és} \qquad X \doteq Y \qquad (5.32.)$$

relációk egyidejűen fennállnak.

Az eddigiekből következik, hogy Y az új nyelvtan szögletes zárójellel konstruált nemterminálisa, $Y = [\ \delta]$.

A nyél belseje reláció viszont megköveteli, hogy az új nyelvtanban legyen egy

$$A \to [\alpha X \delta] \tag{5.33.}$$

alakú levezetési szabály.

Amennyiben a δ jelsorozat első szimbóluma Z, akkor az eredeti nyelvtanban az X és a Z szimbólumok között fennáll a nyél belseje reláció.

A nyél eleje relációhoz viszont az szükséges, hogy létezzék egy

$$A \to \alpha X C \beta \tag{5.34.}$$

levezetési szabály, és ugyanakkor lehetséges legyen a következő levezetés:

$$C \overset{*}{\Rightarrow} D\,\gamma \Rightarrow [\delta]\gamma \tag{5.35.}$$

Ez a feltételezés csak akkor igaz, ha az eredeti nyelvtanban van egy $D \to \delta$ levezetési szabály, továbbá az X és D szimbólumok között vagy nyél eleje, vagy nyél belseje reláció áll fenn. Ez utóbbi akkor, ha az (5.35.) összefüggésben az első származtatás zérus lépésben történik. Ilyenkor persze az X és Z szimbólumok között igaz a nyél eleje reláció is, hiszen Z a D szimbólumból levezetett jelsorozat első tagja.

Ez azonban ellentmond annak, hogy az eredeti nyelvtan gyenge precedencia nyelvtan volt. A gyenge precedencia megengedi ugyan, hogy két szimbólum, adott esetben X és Z között a nyél eleje és a nyél belseje reláció egyidejűen igaz legyen, és ezzel a letörendő nyélre látszólag két alternatívát adjon. Ezt a dilemmát azonban feloldja az a megkötés, hogy a rövidebb nyél letörésével nyert szimbólum, és az előtte álló szimbólum között semmiféle reláció nincsen. Itt azonban az adódott, hogy a letöréssel nyert D szimbólum, és az előtte álló X szimbólum között valamilyen, vagy nyél eleje, vagy nyél belseje reláció fennáll. Ez viszont ellentmond a gyenge precedencia nyelvtan szabályainak.

Ezzel az (5.24.) és (5.25.) alapján konstruált új nyelvtanról bebizonyítottuk, hogy erős precedencia nyelvtan.

Ha arra kényszerülünk, hogy egy gyenge precedencia nyelvtant erőssé tegyünk, akkor nem kell a nyelvtant szögletes zárójelekkel teletűzdelnünk, elegendő, ha néhány helyen, a nyelvtan „gyenge” pontjain segítünk.

Ismert és gyengének bizonyult nyelvtanunk esetében például csupán három szabály változtatásával megoldható a nyelvtan megerősítése. Az ilyen szabályokat persze ott alkalmazzuk, ahol szorít a diszjunkció.

Legyenek a megváltoztatott szabályok:

$$E \to E{+}[T] \qquad E \to [T] \qquad F \to ([E])$$

Ezen kívül persze két új szabályt is be kell vezetni:

$$[T] \to T \qquad ([E]) \to E)$$

Az új nyelvtan, amint erről a szorgalmas olvasó meggyőződhet valóban erős precedencia nyelvtan.

Az egyszerűsített jobboldali elemzés további kiterjesztését jelentik a korlátos környezetet vizsgáló elemzők. Ezek a *BRC – Bounded Right Context* – elemzők a feltételezett nyél jobb- és baloldalán néhány szimbólumot vesznek figyelembe, és ennek alapján állapítják meg azt, hogy a szóban forgó fragmens valóban nyél-e, és ha igen, akkor melyik levezetési szabály szerint kell azt letörni. A módszer elnevezése nem túl szerencsés, hiszen a döntéshez mindkét oldali kontextusra szükség van.

A *BRC* elemzők alkalmazásával kiküszöbölhetjük a precedencia elemzőknek azt a hátrányát, hogy segítségükkel csak egyértelműen invertálható nyelvtanok elemezhetőek.

A *BRC*(**m,n**) elemző a nyéltől balra **m**, jobbra pedig **n** szimbólumot vizsgál. Minden nemterminálishoz tartozik egy halmaz, amely tartalmazza azokat az **m** szimbólum – nyél – **n** szimbólum alkotta hármasokat, amely esetekben a nyelet éppen arra a nemterminális szimbólumra kell letörni.

Egy nyelvtan akkor elemezhető *BRC*(**m,n**) elemzővel, ha ezen halmazoknak nincsen összetéveszthető, tehát azonos jelsorozatot tartalmazó eleme. A jobboldali és baloldali kontextus még akkor is disztingválhat két szabály között, ha azok jobboldala azonos. Éppen ez az egyik célja ennek az elemzésnek.

A nemterminális szimbólumokhoz tartozó halmazokon kívül meg kell még adni azokat a kombinációkat is, amikor csúsztatnunk kell. Ezen jelsorozatokat olyan hosszban adjuk meg, mint amilyen a leghosszabb halmazelem. Ez viszont nem más, mint a két kontextus és a leghosszabb jobboldal hosszának összege.

Legyen az egyértelműen nem invertálható nyelvtan:

$$S \to AB \qquad A \to 0A1 \mid \otimes \qquad B \to 1B0 \mid \otimes$$

Próbálkozzunk a lehető legrövidebb kontextussal, legyen az elemző *BRC*(**1,0**). Ez annyit jelent, hogy jobbról nem igénylünk semmiféle információt, és balról is csak egyetlen szimbólumot veszünk figyelembe. Ennek megfelelően a halmazok felírásánál a jobboldali környezet helyét üresen hagyjuk.

Ezzel a nemterminálisokhoz tartozó halmazok:

$\boldsymbol{K}(S) = \{ [\ \varepsilon, AB,]\ \}$
$\boldsymbol{K}(A) = \{ [\ \varepsilon, \mathbf{0}A\mathbf{1},], [\ \mathbf{0}, \mathbf{0}A\mathbf{1},], [\ \varepsilon, \otimes,], [\ \mathbf{0}, \otimes,]\ \}$
$\boldsymbol{K}(B) = \{ [\ \mathbf{1}, \mathbf{1}B\mathbf{0},], [\ \mathbf{1}, \otimes,], [\ A, \mathbf{1}B\mathbf{0},], [\ A, \otimes,]\ \}$

A csúsztatóképes jelsorozatok a következők:

$\boldsymbol{S} = \{\varepsilon, \mathbf{0}, \mathbf{1}, A, \mathbf{00}, \mathbf{0}A, \mathbf{11}, \mathbf{1}B, A\mathbf{1}, \mathbf{000}, \mathbf{00}A, \mathbf{111}, \mathbf{11}B, A\mathbf{11}, \mathbf{A1}B, \mathbf{0000}, \mathbf{000}A, A\mathbf{111}, A\mathbf{1}B, \mathbf{1111}, \mathbf{111}B, A\mathbf{11}B\ \}$

Mint látható, nincs két összetéveszthető szituáció, tehát a minimális információ is elegendő volt arra, hogy két azonos jobboldallal bíró levezetési szabály között különbséget tegyünk.

Példaképpen elemezzük a **00 ⊗111⊗0** jelsorozatot.

{ ε, **00 ⊗111⊗ 0,** ε } ↦ { **0**, **0⊗111⊗ 0**, ε } ↦ { **00**, **⊗ 111⊗ 0**, ε } ↦
↦ { **00⊗**, **111⊗0**, ε } ↦ { **00***A*, **111⊗ 0**, **3**} ↦ { **00** *A***1**, **11⊗ 0**, **3**} ↦
↦ { **0***A*, **11⊗ 0**, **32**} ↦ { **0***A***1**, **1⊗ 0**, **32**} ↦ { *A*, **1⊗ 0**, **322**} ↦
↦ { *A***1↦**, **⊗0**, **322**} ↦ { *A***1⊗**, **0**, **322**} ↦ { *A* **1***B*, **0**, **3225**} ↦
↦ { *A***1***B***0**, ε, **3225**} ↦ { *AB*, ε, **32254**} ↦ { *S*, ε, **322541**} ↦ *accept*

Természetesen a *BRC* elemzők bonyolultabbak, nagyobb hely- és időigényűek, mint a precedencia elemzők. A két módszer előnyeinek ötvözését jelentik a kevert stratégiájú precedencia elemzők, angol rövidítéssel *MSP – Mixed Strategy Precedence* elemzők.

Ezek olyan precedencia elemzők, amelyek általában a precedencia relációk felhasználásával vizsgálják a jelsorozatot, kivéve a nem egyértelmű invertálhatóság okozta vitás helyeken. Erre, de csakis erre az esetre vagy esetekre egy csökkentett környezetet felhasználó elemző is be van építve, amely ilyenkor kisegíti a precedencia elemzőt.

Eddig az egyszerűsített alulról felfelé elemzők tárgyalásakor a precedencia grammatikából kiindulva egyre többet tudó, de ennek megfelelően egyre bonyolultabb elemzési eljárásokat ismertünk meg. Most visszafelé teszünk egy nagy ugrást, amennyiben egy, a precedencia elemzőnél is gyorsabb és kisebb hely- és időigényes módszert tárgyalunk.

Ez az igen hatásos módszer az operátor precedencia elemzés. Természetesen ez az eljárás csak speciális, úgynevezett operátor grammatikák esetében alkalmazható. Egy nyelvtan akkor operátor nyelvtan, ha a levezetési szabályok jobboldalán sehol sem áll két nemterminális szimbólum egymás mellett.

Valóban, általában az olyan kifejezések leírására kimunkált nyelvtanok esetében teljesül ez a megkötés, amelyek monadikus és diadikus operátorokat alkalmaznak, a monadikusoknál prefix, illetve a diadikusoknál infix jelölésmóddal. Ha a jól ismert és unos-untig elkoptatott, az aritmetikai kifejezéseket infix jelöléssel generáló nyelvtanra gondolunk, nos ez is operátor nyelvtan.

Elvben nem szükségszerű, hogy az operátor nyelvtan valóban operátorokat tartalmazzon, hiszen a terminális szimbólumok szemantikáját a környezetfüggetlen nyelvtan alapján nem lehet megállapítani, az elnevezés azonban nem minden alap nélkül való, az operátor nyelvtan többnyire valóban operátorokat tartalmaz.

Az elemzés alapgondolata abban áll, hogy a precedencia relációkat csak a terminális szimbólumok között értelmezzük. A terminálisok közé beékelődő esetleges egyetlen nemterminális szimbólumot egyszerűen nem vesszük tudomásul, úgy tekintjük, mintha ott sem lenne, és a bal- illetve jobboldalán álló terminálisokat mint szomszédokat kezeljük.

A továbbiakban szabályszerű precedencia elemzést végzünk, csupán az igényel néha megfontolást, hogy letörés esetén a szélső nemterminálist beleértsük-e a nyélbe vagy sem.

Példának vegyük az éppen kéznél lévő operátor grammatikát:

$E \to E{+}T \mid T \qquad T \to T{*}F \mid F \qquad F \to (E) \mid a$

Lássuk a precedencia mátrixot:

	(	*a*	)	+	*	ε
)			•>	•>	•>	•>
a			•>	•>	•>	•>
(	<•	<•	≐	<•	<•	
+	<•	<•	•>	•>	<•	•>
*	<•	<•	•>	•>	•>	•>
ε	<•	<•		<•	<•	

Ez a tábla az eredeti, a nemterminális szimbólumokat is tartalmazó precedencia mátrixtól sok relációt örökölt. Azonban most olyan helyeken is találunk relációkat, ahol korábban semmiféle reláció nem volt érvényes. Így például relációban áll az additív és multiplikatív operátor.

Valójában két operátor között mindig van valamilyen szimbólum, itt az *a* terminális. Mihelyt azonban ezt letörjük, belőle nemterminális lesz, amelyet megállapodásunk szerint figyelmen kívül kell hagynunk. Így a két operátor szomszédságba kerül, és relációt értelmezünk közöttük.

Az operátor precedencia mátrixot áttanulmányozva kitűnik, hogy abban mindaz az információ benne foglaltatik, amit a nyelvtan hordoz. Kiolvasható belőle az operátor precedencia – lám jogosan nevezzük precedencia elemzésnek, – és a balról jobbra szabályt is tartalmazza.

A nyelvtannak ezeket a vonásait annak idején több nemterminális alkalmazásával sikerült biztosítanunk. Minthogy mindezt most már az operátor precedencia mátrix szavatolja, nincsen szükség különböző nemterminálisok alkalmazására. Valóban, amennyiben az operátor precedencia elemzés megvalósítható, akkor elegendő egyetlen nemterminális szimbólum.

Egyszerűsítsük hát a nyelvtant:

1 $E \to E{+}E$ **2** $E \to E{*}E$ **3** $E \to (E)$ **4** $E \to a$

A levezetési szabályokat szokás szerint sorszámoztuk. Elemezzük az alábbi jelsorozatot:

$$a{+}a{*}a$$

A konfigurációt kényelmi okokból most is veremtartalom – elemzendő jelsorozat – eredmény sorrendben adjuk meg.

$\{ \varepsilon, a{+}a\ {*}a, \varepsilon\} \mapsto \{ a, +a{*}a, \varepsilon\} \mapsto \{ E, +a{*}a, \mathbf{4}\} \mapsto \{ E{+}, a{*}a, \mathbf{4}\} \mapsto$

$\mapsto \{ E{+}a, {*}a, \mathbf{4}\} \mapsto \{ E{+}E, {*}a, \mathbf{44}\} \mapsto \{ E{+}E{*}, a, \mathbf{44}\} \mapsto \{E{+}E{*}a, \varepsilon, \mathbf{44}\} \mapsto$

$\mapsto \{ E{+}E{*}E, \varepsilon, \mathbf{444}\} \mapsto \{ E{+}E, \varepsilon, \mathbf{4442}\} \mapsto \{ E, \varepsilon, \mathbf{44421}\} \mapsto Accept$

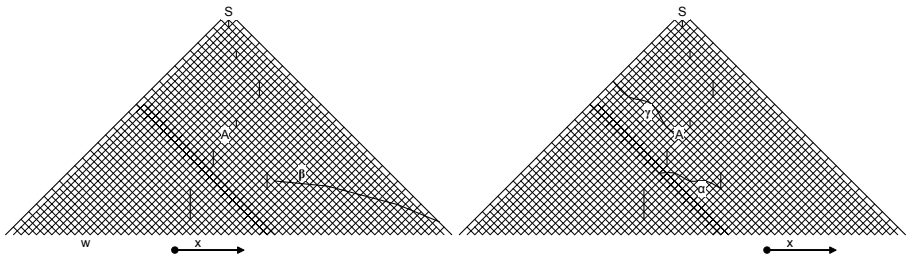
Az elemző, mint látjuk, ügyel az operátor precedenciára.

5.5. Nyelvek és nyelvtanok

A nyelvek és nyelvtanok közötti pontos disztinkció mindig is egyik kényes pontja volt a számítástechnikai nyelvészetnek. A különbségtétel nem triviális voltára már *Chomsky* első alapvető munkája is rámutatott. Nyelvosztályokról beszél, de azokat nyelvtani szabályokkal definiálja.

Az alábbiakban néhány kérdésben megpróbálok tiszta vizet önteni ez esetben a nyelvekre. Minthogy az itt közölt eredmények elsősorban elméleti jelentőségűek, egyes esetekben csak a bizonyítás gondolatmenetét közlöm.

Még mielőtt a nyelvtanokról a nyelvekre áttérnénk, egy fontos tételt érzékeltetnék. Az *LL*(**k**) nyelvtanok az *LR*(**k**) nyelvtanok valódi részhalmazát képezik. Ennek belátására elegendő a balelemzést illetve jobbelemzést szemléltető 5.1 és 5.2. ábrákat együttesen megvizsgálni. Az áttekinthetőség biztosítására az 5.3. ábrán ezek egymás mellett láthatóak.



5.3. ábra

Milyen információ alapján ismerjük fel, hogy a mondat levezetésében szerepel egy $A \rightarrow \alpha$ helyettesítés? Ennek eldöntésére nyilván rendelkezésre áll az elemzendő jelsorozatnak már elolvasott része, továbbá az előretekintés. Ne feledjük, a két elemzési mód jelölésére alkalmazott *L* betű mindkét esetben az angol *left* szót takarja, vagyis mindkét esetben balról jobbra olvasunk, és elemezünk. Mint látható, a jobbelemzés éppen az *A* nemterminális által generált terminális jelsorozat hosszával több szimbólumot használ. Így ugyanannak a kérdésnek az eldöntésére a balelemzésnél felhasznált információ csupán része a jobbelemzésnél felhasználtnak.

Határesetben a két elemzési mód számára hozzáférhető információ csak akkor azonos, ha az *A* nemterminális végül is az üres jelsorozatot generálja. Ebből következik, hogy mindazt a felismerést, amit a balelemzés során általában kisebb, de semmiképpen nem nagyobb információ birtokában megteszünk, a jobbelemzés során is bizonyosan felismerjük.

Ezért az *LL*(**k**) nyelvtanok az *LR*(**k**) nyelvtanoknak részhalmazát alkotják. A részhalmaz valódi részhalmaz, hiszen például vannak balrekurzív *LR*(**k**) nyelvtanok, amelyek viszont nem balelemezhetőek.

A faktorizálás kapcsán említettük, hogy **k > 1** esetén az *LL*(**k**) nyelvtanok előretekintését mindaddig biztosan lehet csökkenteni, ameddig a nyelvtannak nincs ε-szabálya.

Az *LL*(**k**) nyelvek is, és nem csak az *LL*(**k**) nyelvtanok alkotnak hierarchiát.

Ami a jobbelemezhető nyelveket illeti igazolható, hogy minden determinisztikus **CF** nyelvhez szerkeszthető *LR*(**1**) elemző. Ennek az elsősorban elméleti szempontból nagy jelentőségű eredménynek szabatos bizonyítása messzire vezetne, éppen ezért ennek csak gondolatmenetét adom meg.

Mindenekelőtt egy definíció. Amennyiben egy nyelv esetében a nyelv mondatai és azok prefixumai különbözőek, a halmazok diszjunktak, akkor ezt a nyelvet prefix tulajdonságú nyelvnek mondjuk.

Az általunk megismert nyelvek közül csak néhány bír ezzel a tulajdonsággal. Így például az aritmetikai kifejezéseket infix alakban generáló nyelv nem prefix tulajdonságú. Valóban, például az

$$(a+a)$$

jelsorozatról nem dönthető el, hogy önmaga alkot egy mondatot, vagy csupán egy hosszabb mondat prefixuma. Mindkét eset előfordulhat.

Ugyanígy nem prefix tulajdonságú nyelv az aritmetikai kifejezéseket posztfix lengyel formában generáló nyelv sem. Ezzel szemben a prefix lengyel jelölést generáló nyelv rendelkezik a prefix tulajdonsággal.

Valójában bármelyik nyelvből kis módosítással készíthetünk egy prefix tulajdonságú nyelvet. Egészítsük ki ezt a nyelvet egy sohasem volt terminális szimbólummal, és biggyesszük ezt a szimbólumot valamennyi mondat után. Jelölje az eredeti nyelvet ***L***, és legyen a sohasem volt szimbólum ⊗, akkor az ***L***⊗ nyelv nyilván prefix tulajdonságú. Amennyiben egy jelsorozat az új szimbólummal végződik, akkor a jelsorozat mondat, ellenkező esetben csak prefixum lehet.

Vegyük észre, hogy a prefix tulajdonság nyelvhez és nem nyelvtanhoz kötött tulajdonság. Az is triviális, hogy ha az eredeti nyelv **CF** nyelv volt, az lesz a megpatkolt nyelv is.

A prefix tulajdonságú determinisztikus környezetfüggetlen nyelvekre készíthető *LR*(**0**) elemző. Ez annyit jelent, hogy az elemző előretekintés nélkül el tudja dönteni a helyes folytatást.

Az igazolás gondolatmenete szerint a nyelvhez előbb egy úgynevezett normál veremautomatát, majd egy kanonikusnak mondott nyelvtant szerkesztünk.

Minden determinisztikus környezetfüggetlen nyelvnek van determinisztikus veremautomatája. Még azt is feltételezhetjük, hogy ez a veremautomata mentes a végtelen ε-mozgásciklustól, és elfogadó állapotban már nem végez ε-mozgást. Amennyiben nem ez lenne a helyzet, akkor a már ismert módszerekkel az automata átalakítható.

Általában egy mozgás során az automata három funkciót hajthat végre. Elolvashat egy szimbólumot a bemenő szalagról, kitörölheti a verem legfelső szimbólumát, végül beírhat néhány új szimbólumot a verem tetejére. Persze nem szükségszerű, hogy egy mozgás során mindhárom tevékenységre sor kerüljön, mert például ε-mozgás alkalmával nem történik olvasás.

Változtassuk meg kissé a verem kezelésével kapcsolatos szemléletünket. Eddig úgy tekintettük, hogy a verem legfelső szimbólumát mindenképpen kitöröljük, legfeljebb ha szükségünk lesz rá, visszaírjuk. Most, ha a verem tartalma nem változik, vagy csak növekedik, ezt fogjuk fel olymódon, hogy nem végeztünk törlést a veremben. Ezzel a szemlélettel megeshet, hogy egyes mozgásoknál nem végzünk törlést a veremben.

Egy mozgást eddig három tényező határozott meg, az olvasott szimbólum, az automata állapota, végül a verem tetején található szimbólum. Ez sem volt minden esetben így, hiszen például ε-mozgáskor nem is volt olvasott karakter, így az nem is játszhatott szerepet.

Alakítsuk át az automatát olymódon, hogy a mozgás funkcióit válasszuk külön. Egy mozgás során vagy olvasás, vagy törlés, vagy beírás történhet, de mindig csak egy a három funkció közül. További korlátozás, hogy beíráskor a verem tartalma csak egyetlen szimbólummal növelhető, végül megköveteljük, hogy a törlési művelet kivételével a mozgás ne függjön a verem legfelső szimbólumától.

Ezek a követelmények egyszerűen teljesíthetőek. Az utolsó, a verem legfelső szimbólumától való függetlenséget például úgy érhetjük el, hogy az eredeti automata vermének legfelső szimbólumát leemeljük, és az automata állapotterébe olvasztjuk. Ezt a fogást, mint emlékezhetünk, már alkalmaztuk a mélybelátó automatával egyenértékű egyszerű automata szerkesztésekor.

Az új automata állapottere tehát az eredeti állapottér, és a legfelső veremszimbólum direkt szorzata lesz, így a verem tetejére a második veremszimbólum kerül, amitől már valóban nem függ semmi.

Az új állapot tartalmazza tehát mind az eredeti állapotra, mind a verem legfelső szimbólumára vonatkozó valamennyi információt.

Amennyiben egy mozgás több funkciót hajt végre, úgy azt fel lehet bontani több mozgásra olymódon, hogy egy-egy mozgás az eredeti mozgásnak csak egy funkcióját hajtsa végre.

Az így kapott automatát nevezzük normál automatának. Minden determinisztikus nyelvnek van normál automatája.

A normál automata alapján, lényegében az ismert módszerekkel készített nyelvtant nevezzük kanonikus nyelvtannak, pontosabban kanonikus alakú nyelvtannak. Egy nyelvnek ugyanis több kanonikus nyelvtana lehet.

Erről a kanonikus nyelvtanról igazolható, hogy precedencia grammatika. Az igazolás gondolatmenete megfelel annak a bizonyításnak, amikor a gyenge precedencia nyelvtanból alkalmas módon származtatott nyelvtanról igazoltuk, hogy erős precedencia nyelvtan.

Felhasználva a kanonikus nyelvtanok tulajdonságait, itt is igazolni lehet, hogy a precedencia relációk diszjunktak.

Persze elképzelhető, hogy a nyelvtan nem egyértelműen invertálható. Erre ugyanis a nyelvtan kanonikus volta nem ad útmutatást. Ekkor természetesen nem készíthető a nyelvtanra egyszerű precedencia elemző.

Ilyenkor viszont ugyancsak a kanonikus grammatika tulajdonságaiból következően igazolható, hogy az (**1,0**) környezet elegendő információt hordoz a lehetséges alternatívák közül a helyes kiválasztására. Így a nyelvtan *BRC*(**1,0**) nyelvtan. Ebből viszont már következik, hogy a nyelvtan *LR*(**0**) nyelvtan.

Az itt csak gondolatmenetében ismertetett bizonyítás alapján állítható, hogy minden prefix tulajdonságú determinisztikus környezetfüggetlen nyelvnek létezik *LR*(**0**) elemezhető grammatikája.

Nézzünk most egy prefix tulajdonságot nem mutató környezetfüggetlen nyelvet. Mint láttuk, egy sohasemvolt szimbólum hozzátoldásával persze ebből is készíthetünk prefix tulajdonságú változatot. Erre az utóbbira természetesen igaz, hogy van *LR*(**0**) elemezhető nyelvtana. Milyen megállapításokat tehetünk azonban az eredeti nyelvvel kapcsolatosan?

Mi a lényeges különbség a prefix tulajdonságokat mutató, és az ilyen tulajdonsággal nem rendelkező nyelvek között? Míg az előbbi esetben a pusztán az olvasott jelsorozat vizsgálata alapján eldönthető az a kérdés, hogy a jelsorozat mondat-e vagy prefixum, addig az utóbbi nyelveknél van olyan jelsorozat, ahol ez nem válaszolható meg.

Rögtön választ kapunk azonban, ha előretekintünk, ha megnézzük a következő szimbólumot. Ha létezik következő szimbólum, akkor a jelsorozat prefixum, ha nincs, akkor mondat. Nincs mit tennünk tehát, egy karaktert előre kell tekintenünk. Az *LR*(**1**) elemző tehát alkalmas lesz ennek a dilemmának a feloldására, amelyre a nem prefix tulajdonságú nyelvek esetében az *LR*(**0**) elemző nem volt képes.

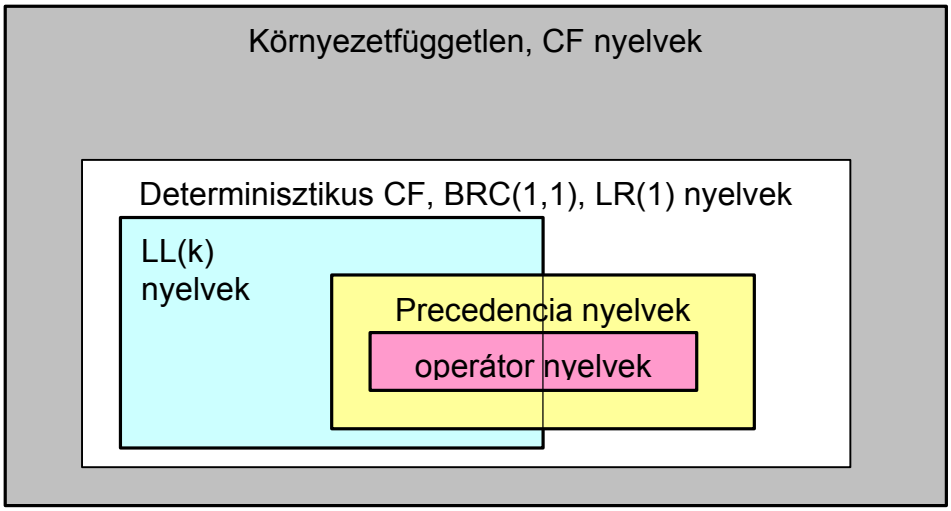
Ezek szerint minden determinisztikus környezetfüggetlen nyelvnek van *LR*(**1**) elemezhető nyelvtana.

Ismét szeretném hangsúlyozni, hogy ez döntően elvi jelentőségű eredmény. Valójában egy kanonikus nyelvtan általában túl terjedelmes ahhoz, hogy gyakorlati felhasználásra igényt tarthatna.

A determinisztikus környezetfüggetlen nyelvek valódi részhalmazát alkotják azok, amelyeknek kanonikus nyelvtana egyértelműen invertálható. Ezek egyszerű precedencia elven elemezhetőek.

Ezen részhalmaz egy további részhalmazát alkotják az operátor precedencia elven elemezhető nyelvek.

Az *LL*(**k**) nyelvek halmaza az előbbi két halmazzal inkommenzurábilis.



5.4. ábra

A determinisztikus környezetfüggetlen nyelvek kapcsolatát tünteti fel az 5.4. ábra. Természetesen az *LL*(**k**) nyelvek nem egyetlen halmazt alkotnak, hanem, mint tisztáztuk, a **k** értékének változtatásával egy végtelen egymásba ágyazott halmazsorozatot. Minél nagyobb a **k** értéke annál nagyobb a halmaz, és természetesen magába foglalja az összes, nála rövidebb előretekintéssel elemezhető nyelveket.

6. Az automataelmélet alapjai

6.1. A *Turing*-gép

Az eddig tárgyalt ismeretanyag a matematikai nyelvészet, vagy ha úgy tetszik számítástechnikai nyelvészet fogalomkörébe tartozott. Ez a tudomány – mint említettük – *Chomsky* 1956-ban megjelent cikkeivel vette kezdetét.

Az automataelmélet jó húsz évvel idősebb fiatalabb rokonánál, ugyanis mint kitűnik, a két diszciplína nagyon szorosan kapcsolódik egymáshoz.

Turing 1936-ban alkotta meg azt a matematikai objektumot, a róla elnevezett automatát vagy gépet, mint olyan „szerkezetet", amellyel matematikai problémák megoldhatók. Az ebben a fejezetben ismertetett más automataelméleti eredmények is jórészt ebből az időből származnak. A *Turing*-gép „mindössze" abban különbözik a kétirányú mozgást végző véges automatától, hogy nem csupán olvasó-, hanem író-olvasó perifériája van. Ennek megfelelően az automata mozgása során az éppen elolvasott szimbólumot felülírja. Amennyiben a *Turing*-gép valamilyen realizálásában gondolkodunk, akkor a véges automatához képest az a különbség, hogy míg az utóbbinál bemeneti berendezésként egy lyukszalag olvasó is megtette, addig itt mágnesszalagos író-olvasó perifériára van szükség.

Itt rögtön említsük meg, hogy mint minden automatát, a *Turing*-gépet is tulajdonképpen a vele megoldható feladatok osztálya jellemez. Gondolom, most már nem lepi meg az olvasót, hogy a *Turing*-gép erejét nagyon sokféle, látszólag eltérő képességű automatával realizálhatjuk. Tulajdonképpen ízlés kérdése, hogy egy szerző melyik automatáról, melyik realizációról jelenti ki: ez a *Turing*-gép.

A *Turing*-gép bemenete természetesen egy jelsorozat, amely a szalag egy véges darabjára van felírva. Induláskor a bemenet legelső szimbólumát helyezzük az író-olvasófej alá. A mágnesszalag többi, induláskor információt nem hordozó része az üres – angolul *blank* – szimbólummal van teleírva. A mágnesszalag potenciálisan végtelen hosszú.

A *Turing*-gépnél a többi automatához képest kissé eltérő az elfogadás kritériuma. A *Turing*-gép egy jelsorozatot akkor fogad el, ha a jelsorozattal, mint bemenettel elindítva a *Turing*-gép elfogadó állapotban megáll. Pontosabban – gondolva a nemdeterminisztikus automatákra is – ha az adott bemenet mellett létezik olyan mozgássorozat, amelyet követően az automata elfogadó állapotban áll meg.

Egy *Turing*-gép – hasonlóan a többi automatához – akkor áll meg, ha egy olyan szituációba kerül, amelyre nézve nincs mozgási szabály.

Ha csak olyan mozgássorozat található, amelynek során az automata vagy nem elfogadó állapotban áll meg, vagy meg sem áll, akkor a *Turing*-gép a jelsorozatot visszautasította.

Matematikailag a *Turing*-gépet egy hetes írja le:

$$\boldsymbol{T} = (Q, \Sigma, \Gamma, \{l,r\}\ \delta, q_0, F) \qquad (6.1.)$$

Itt a Q, Σ, q_0, és F értelmezése a szokásos, vagyis az állapothalmaz, a bemenő alfabéta, az induló állapot, végül az elfogadó állapotok halmaza.

A Γ a szalag szimbólumainak alfabétája. Minthogy induláskor a szalagon a bemeneti jelsorozat található, amely természetesen a bemeneti alfabéta szimbólumaiból áll, így a bemeneti alfabéta a szalag alfabétájának részhalmaza:

$$\Sigma \subset \Gamma \tag{6.2.}$$

A részhalmaz valódi részhalmaz, hiszen az üres szimbólum (*blank*) eleme a Γ-nak, de nem eleme a Σ halmaznak. Az üres szimbólumot jelölje *ы*.

Az $\{l,r\}$ halmaz a mozgás irányát adja meg. Elvben a szalag az automata működése során helyben is maradhat, de – mint ezt a véges automatáknál tisztáztuk – ez nem növeli az automata erejét, így ezt a lehetőséget joggal kihagyhatjuk.

A mozgási szabályok most is egy leképezést reprezentálnak. A mozgás csakis az automata állapottól és az olvasott szimbólumtól függ. A mozgás következményeképpen megváltozik az automata állapota, az olvasott szimbólum felülíródik, végül az író-olvasófej vagy jobbra, vagy balra elmozdul.

A leképezés tehát a következőképpen szemléltethető:

$$Q \times \Gamma \Rightarrow Q \times (\Gamma - \{ы\}) \times \{l,r\} \tag{6.3.}$$

Míg a mozgás meghatározásakor bármely szimbólum, így a *ы* is szerepelhet, addig felülírásra általánosan elfogadott konvenció szerint ez a szimbólum nem használható. A (6.3.) összefüggésben ezért szerepel a baloldalon a Γ, a jobboldalon pedig a $(\Gamma-\{ы\})$ halmaz.

A leképezés nem zárja ki, hogy legyenek olyan mozgási szabályok is, ahol az olvasott szimbólum a *ы*. Ebből következik, hogy az író-olvasó fej elkalandozhat a szalag szűz, a bemeneti jelsorozattal nem érintett részére is. Minthogy megállapodásunk szerint a szalag mindkét irányban potenciálisan végtelen hosszú, az író-olvasófej tetszőlegesen messzire távozhat kiindulási helyzetétől. Ugyanakkor mindig pontosan lehet tudni, melyek a szalag még érintetlen részei, hiszen itt, és csakis itt hordoz a szalag *ы* szimbólumot.

Az természetesen teljesen legális, hogy egy a *ы* szimbólum szemantikájával azonos szemantikájú szimbólumot használunk. Ilyen értelemben informálisan mindenképpen mondhatjuk, hogy egy karaktert a *ы* szimbólummal írtunk felül. Egyébként vannak szerzők, akik szemrebbenés nélkül használják a *ы* szimbólumot felülírásra is.

Persze egy *Turing*-gép specifikálásakor itt sem kell megadni a teljes leírást. Elegendő, ha csak a mozgási szabályokat soroljuk fel, ügyelve arra, hogy a leírás egy q_0 állapotból kiinduló szabállyal kezdődjék, és megadjuk az elfogadó állapotok halmazát.

Végül, ha ez nem egyértelmű, külön fel kell tüntetni a bemeneti alfabetát.

Lássunk egy példát a *Turing*-gépre:

Legyen a nyelv, amelyet a *Turing*-géppel el kívánunk fogadtatni:

$$\boldsymbol{L} = wcw \ \{w \mid w = (a \cup b)^+\}$$

vagyis a nyelv olyan jelsorozatokat tartalmaz, ahol egy tetszőleges nem üres az *a* és *b* szimbólumokból álló jelsorozat a *c* szimbólum után megismétlődik.

A *Turing*-gép működésének alapgondolata az lesz, hogy az eredeti és az ismétlő jelsorozatot szimbólumonként összehasonlítjuk.

Az automata mozgási szabályai a következőek lesznek:

$\delta(q_0,a) = (q_{a1},X, r)$ $\quad$ $\delta(q_0, b) = (q_{b1}, X,r)$

$\delta(q_{a1}, a) = (q_{a1}, a, r)$ $\quad$ $\delta(q_{a1}, b) = (q_{a1}, b,r)$

$\delta(q_{b1}, a) = (q_{b1}, a, r)$ $\quad$ $\delta(q_{b1}, b) = (q_{b1}, b,r)$

$\delta(q_{a1}, c) = (q_{a2}, c, r)$ $\quad$ $\delta(q_{b1}, c) = (q_{b2}, c,r)$

$\delta(q_{a2}, X) = (q_{a2}, X, r)$ $\quad$ $\delta(q_{b2}, X) = (q_{b2}, X,r)$

$\delta(q_{a2}, a) = (q_3, X, l)$ $\quad$ $\delta(q_{b2}, b) = (q_3, X,l)$

$\delta(q_3, X) = (q_3, X, l)$ $\quad$ $\delta(q_3, c) = (q_4, c,l)$

$\delta(q_4, a) = (q_5, a, l)$ $\quad$ $\delta(q_4, b) = (q_5, b,l)$

$\delta(q_5, a) = (q_5, a, l)$ $\quad$ $\delta(q_5, b) = (q_5, b,l)$

$\delta(q_5, X) = (q_0, X, r)$ $\quad$ $\delta(q_4, X) = (q_6, X,r)$

$\delta(q_6, X) = (q_6, X, r)$ $\quad$ $\delta(q_6, c) = (q_6, c,r)$

$$\delta(q_6,ы) = (q_7, ы, r)$$

Az automata egy szimbólum elolvasásával kezdi munkáját. A szimbólumot az automata állapotában memorizálja, ugyanakkor a szalagról törli. Ilyen memorizálást már alkalmaztunk, amikor a mélybelátó veremautomatával egyenértékű automatát konstruáltuk. Akkor a teljes belelátási mélységben található információt az automata állapotában tároltuk.

Ezután az automata végighalad az eredeti szöveg még el sem olvasott, illetve a vízválasztóként szereplő *c* szimbólum után az ismétlő szöveg már elolvasott részén. Amennyiben az ismétlő szöveg első még el nem olvasott szimbóluma megegyezik az állapotban tárolttal, akkor ezt is törli, és visszafordul.

Amikor az automata visszaérkezett az eredeti szöveg utolsó, már elolvasott szimbólumához, az egész folyamat újra kezdődik.

Ha már nincsen olvasatlan szimbólum, amit az automata úgy detektál, hogy nem talál a *c* szimbólum előtt ki nem törölt szimbólumot, akkor meggyőződik arról, hogy az ismétlő szöveg szimbólumai is elfogytak. Ezt úgy érzékeli, hogy az olvasott szimbólumokat *ы* követi.

Ebben és csakis ebben az esetben kerül az automata az elfogadást jelentő q_7 állapotba.

Az automata minden szituációhoz csak egy, pontosabban legfeljebb egy mozgást rendel. Az automata determinisztikus. Könnyű belátni, hogy ez adott esetben a két jelsorozatot elválasztó *c* szimbólum jelenlétének köszönhető.

Említettük már, hogy – mint minden automata, – a *Turing*-gép is egy számítási potenciált képvisel. Vajon milyen mértékű a *Turing*-gép számítási képessége? Erre vonatkozóan *Church* deklarált egy feltételezést, amelyet *Church*-tézis néven emlegetnek.

Ez nem tétel, amely bizonyítható, hanem állítás, amiben hinni kell. Teljes megegyezésben *Church* nevével – angol jelentése egyház, templom – ez a tézis bizonyos értelemben hittételnek tekinthető.

A *Church*-tézis azt állítja, hogy minden probléma, amelyre eljárás, procedúra szerkeszthető, *Turing*-géppel megoldható. Ez nagyon súlyos kijelentés, hiszen ez annyit jelent, hogy a *Turing*-gép a matematikai értelemben vett megismerhetőség határa.

Ebből következően az ember azokra és csakis azokra a kérdésekre tud választ adni, amelyekre a *Turing*-gép is képes.

Korábban már szóltunk az algoritmus és a procedúra közötti különbségről, és az algoritmikusan eldönthetetlen problémákról. Ezeknek a kérdéseknek megfogalmazása, tisztázása, és egyáltalában ezen fogalmak definiálása is ennek az időszaknak az eredménye.

Ismeretelméleti szempontból a fenti felismerésnek a jelentőségét lehetetlen túlértékelni. Mióta az emberiség egzakt tudományokkal foglalkozik – és ez az időszak jóval több, mint kétezer év – az volt az általánosan elfogadott vélelem, hogy minden szabatosan megfogalmazott probléma megoldható. A lehetséges megoldások halmazába persze bele kell érteni azt az esetet is, amikor bizonyíthatóan nincs megoldás. Ilyen probléma például a kör négyszögesítése.

Ha valamelyik kérdésre ez idő szerint nem ismerjük a választ, akkor a régi felfogás szerint ez annak tudható be, hogy ismereteink még nem eléggé pallérozottak, a válasz kimunkálása majd az utókor feladata lesz. Valóban, sok korábban válasz nélkül maradt kérdésre adta meg a tudomány később a feleletet.

Ez volt az álláspont egészen az 1930-as évekig. Addig a legnagyobbak is, így kora matematikus fejedelmének tekintett *David Hilbert* is hasonlóan gondolkoztak. Amikor a századfordulón – te jó ég, már specifikálni kell, hogy a XIX. és XX. század fordulóján – megfogalmazta, mintegy megjósolta azokat a feladatokat, amelyeket a XX. század matematikusai majd megoldanak, akkor ő fekete fehér alapon, tehát vagy a megoldás, vagy a megoldhatatlanság bizonyítása gondolkodott. Az, hogy bizonyos feladatokat elvileg sem lehet megoldani, fel sem merült.

Talán kissé patetikus, de szerintem igaz az az állítás, hogy az algoritmikusan eldönthetetlen feladatok létének felismerése alapjában változtatta meg a világ felismerhetőségével kapcsolatos nézeteinket.

Visszatérve a *Church*-tézis mondanivalójára, meglepő lehet, hogy egy csak elfogadás és visszautasítás között választani képes, tehát csak igen-nem válaszra alkalmas gép minden kiszámítható probléma megoldására felhasználható. Ez praktikus szempontból valóban hátrány, de nem elvi akadály.

Gondoljunk csak a barkochba játékra. Az információt tehát igen-nem válaszok útján is megkaphatjuk, csak kissé bonyolultabban.

Egyébként más szaktudományok sokkal rokonszenvesebb formában definiálják a *Turing*-gépet. Van, amelyik a szalagra kiírja a választ, esetleg a működés gyorsítására, és áttekinthetőbbé tételére több szalagot használ. Ezek valóban kényelmesebben adnak választ a feltett kérdésre. Valamennyi ilyen *Turing*-gépről bebizonyítható, hogy ereje, megoldási képessége nem nagyobb a mi barkochbázni kényszerülő *Turing*-gépünknél. Ne higgyenek tehát a definícióknak. Nem **az a** *Turing*-gép, az is **egy** *Turing*-gép.

A *Church*-tézis tétellé való élesítésének lényegében az az akadálya, hogy az algoritmus fogalmának nincs formális, vagyis matematikai szempontból szabatos leírása. Megfogalmazása óta, vagyis több mint 60 éve nem merültek fel kételyek igazságát illetően. Sőt. A 80-as években *Zohar Manna* bebizonyította, hogy mindaz, ami a legbonyolultabb függvények, a parciális rekurzív függvények segítségével megoldható, az *Turing*-géppel is kiszámítható.

Az előbb említett függvények azért parciálisak, mert értelmezési tartományuknak csak egy részében adnak eredményt, és azért rekurzívak, mert a függvényt definiáló kifejezésben rekurzív módon szerepelhet önmaga a függvény is.

Természetesen minden függvény, ha történetesen nem parciális vagy nem rekurzív, felfogható, mint az előbbi függvényosztály speciális esete.

Mindenesetre az elmúlt évek új eredményei csak megerősíthettek minket a *Church*-tézis igazságába vetett hitünkben.

A *Church*-tézis alapján a *Turing*-gépet úgy tekinthetjük, mint a procedúra, vagyis eljárás formális definícióját.

6.2. A *Turing*-gép lehetőségei

Nem kell hozzá különleges fantázia, hogy a számítógépet a *Turing*-gép realizációjának tekintsük. Valóban a közvetlen hozzáférésű memória állapotát feleltethetjük meg a *Turing*-gép állapotainak, míg a háttér memóriák játszhatják a *Turing*-gép szalagjának szerepét.

Az egyetlen különbség az absztrakt *Turing*-gép és a valóságos számítógép között az, hogy az utóbbi háttér memóriája csak korlátos információ tárolására képes.

A *Church*-tézisre gondolva ebből az is következik, hogy bármely algoritmikusan eldönthető kérdés számítógépes megoldásának nincs elvi, hanem csak a számítógép méretét illetően gyakorlati akadálya.

Fordítva, mindazokat a jól ismert és bevált fogásokat, amelyeket számítógéppel meg tudunk valósítani, *Turing*-géppel is megtehetjük. Így például a *Turing*-gép szubrutinként használható.

Definiáljunk ugyanis egy *Turing*-gépet, amely a szubrutintól várt funkciót valósítja meg. Legyen adott egy másik *Turing*-gép, amely az előbbi *Turing*-gép szolgáltatásait szubrutinhívás formájában kívánja igénybe venni.

A két *Turing*-gép állapotai alkossanak diszjunkt halmazokat, és egyesítsük a két gépet egyetlen géppé. Ha ezt az egyesített gépet a hívó gép kezdőállapotából indítjuk el, akkor nyilván csakis a hívó *Turing*-gép állapotai érvényesülhetnek. A szubrutinként szolgáló *Turing*-gép mintha ott sem lenne.

Ha a szubrutint aktivizálni akarjuk, akkor olyan mozgási szabályt kell bevezetnünk, amely az egyesített gépet a szubrutin-gép kezdőállapotába viszi át.

Most viszont a szubrutin *Turing*-gép állapotain történik mozgás, és a gép végrehajtja a kívánt funkciót.

Amikor ez a szubrutin-gép befejezte munkáját, tehát megállna, akkor megállás helyett egy új mozgási szabály ismét a hívó gép alkalmas állapotába viszi át az egyesített *Turing*-gépet. A hívó *Turing*-gép ezután folytatja munkáját.

Persze itt is, éppúgy mint a számítógépek esetében, szubrutin hívásakor gondoskodnunk kell arról, hogy híváskor a hívott, visszatéréskor pedig a hívó hozzáférhessen a működéséhez szükséges valamennyi információhoz. Ez azonban – mint érzékelhető – csupán technikai és nem elvi nehézséget okozhat.

Nagyon könnyen szerkeszthetünk például olyan szubrutint, illetve azt realizáló *Turing*-gépet, amely beszúr, vagy olyat, amely kitöröl egy szimbólumot a szalag alkalmas helyén. Ennek megvalósítása érdekében célszerű az eredeti *Turing*-gépet kissé feltupírozni. Állapotterét bővítsük ki annyira, hogy képes legyen egy szimbólum memorizálására, szalagját pedig szélesítsük ki annyival, hogy azon még egy markerjel elférjen.

Beszúrás esetén tételezzük fel, hogy a beszúrandó szimbólum a kibővített állapotban van tárolva. Markerrel megjegyezzük a beszúrás helyét, és valamelyik irányba, például jobbra lépünk. Ezután a kiolvasott szimbólumot az állapotban tárolttal felülírva, az állapotban a most kiolvasott szimbólumot memorizáljuk. Minden ilyen operáció után jobbra lépve elsétálunk a szalag azon darabjának végéig, amely információt hordoz, vagyis amíg *ы* szimbólumot vagy ezzel egyenértékű jelet nem találunk. Ezután ha szükséges, a szalag tartalmát változatlanul hagyva visszamegyünk a markerjelig, és azt töröljük.

Ha törölni akarunk egy szimbólumot, akkor megjegyezzük a törlés helyét, majd elgyalogolunk a szalag információt hordozó részének végéig. Ezután a *ы* jellel egyenértékű szimbólumot helyezünk el a kibővített állapotban, és ugyanazt a műveletet hajtjuk végre, mint előbb, vagyis a kiolvasott szimbólumot az állapotban tárolttal felülírjuk, miközben a kiolvasott szimbólumot memorizáljuk. Az eltérés csupán annyi, hogy most nem balról jobbra, hanem jobbról balra lépegetünk. Ha elértünk a markerjelig, akkor azt kitöröljük és megállunk.

Igaz, ezek a funkciók nem túl bonyolultak, de remélem, ez a két demonstratív példa is érzékelteti, hogy bonyolultabb, de egyértelműen definiált feladatokra kisebb-nagyobb gonddal, de szerkeszthető szubrutinként működő *Turing*-gép.

Említettük, hogy az általunk megadott definíció a *Turing*-gépnek csak egyik lehetséges megadási módja. Így *Turing*-gép például az a gép is, amelynek író-olvasófeje alatt nem egy szalag, hanem egy mágneses sík helyezkedik el. Itt tehát nem kétirányú – jobbra, balra – hanem négyirányú – jobbra, balra, fel, le – mozgás képzelhető el. Amennyiben hiszünk a *Church*-tézisben, akkor igazolás nélkül el kell fogadnunk azt az állítást, hogy ennek az automatának az ereje sem lehet nagyobb a *Turing*-gépnél. Egyébként nem nehéz belátni, hogy ez a gép sem tud többet, mint szalag mentén mozgó kisöccse.

Foglaljuk ugyanis téglányba a síknak azt a részét, amely információt hordoz. Az így kialakított téglánymátrixot tároljuk a szalagon sorfolytonosan, a sorokat egy új, eddig nem szerepelt szimbólummal elválasztva. A kezdősor elejét, és a zárósor végét két ilyen szimbólummal jelöljük.

Amennyiben a síkbeli automata jobbra vagy balra mozog, akkor az őt szimuláló szalag automata mozgása megegyezik a síkbeli automatáéval. Kínosabb a helyzet, ha a síkbeli automata például felfelé mozog. Ilyenkor a szalagos automata író-olvasófejének természetesen az előző sor ugyanolyan sorszámú szimbóluma fölé kell kerülnie.

Ennek megvalósítására például a következő módszert használhatjuk. Egy markerrel megjegyezzük a kiindulási helyet, és egy másik markert helyezünk el az előző sor elején. Ezután az író-olvasófej ingajáratot végez a két marker között, miközben az eredeti markert mindig egy hellyel balra, az újat pedig egy hellyel jobbra csúsztatjuk. Nyilvánvaló, hogy amikor az eredeti marker a sor elejére ér, a másik marker éppen a kívánt pozícióban lesz.

Amennyiben a síkbeli automata felségterületét felfelé vagy lefelé elhagyná, akkor a mozgás megkezdése előtt a szalagos automata a szalag elejére vagy végére új sort kell írjon. Ha az automata jobbra vagy balra lépne ki az eredeti téglánymátrixból, akkor a szimulálás előtt minden sor elejére illetve végére egy a *ы* szimbólummal egyenértékű szimbólumot kell beszúrnunk.

Nem vitás, a szalagautomata sokkal lassabban működik, mint síkbeli testvére, de ettől függetlenül alkalmas annak szimulálására.

Egy másik kísérlet az általunk megadott definíció szerinti *Turing*-gépnél nagyobb erejű automata megszerkesztésére, ha a gépet nem egy, hanem **n** szalaggal látjuk el. A szalagoknak egymástól független író-olvasófejük van. Az ilyen gép mozgását a gép állapota és valamennyi szalagról leolvasott szimbólum együttesen határozza meg. A mozgás hatására a gép új állapotba megy át, valamennyi olvasott szimbólumot felülírja, végül mindegyik szalag egymástól függetlenül vagy jobbra, vagy balra lép, illetve helyben marad.

Ezzel az **n** szalagos és író-olvasófejes géppel egyenértékű *Turing*-gépet a következő módon szerkeszthetünk.

Az új gép szalagját válasszuk olyan szélesre, hogy rajta mind az **n** szalagon tárolt információ egymás alatt elférjen. Ezen túlmenően legyen hely sávonként egy-egy marker számára is.

A szimulált gép író-olvasófejeinek helyzetét a szalagon markerekkel jelöljük. Így minden sávban található egy marker. Az egyszalagos gép megkeresi valamennyi markert, kiolvassa az ott található információt, és tárolja azt a megfelelően kibővített, és véges sok információ memorizálására alkalmas állapotterében.

Ha minden információ rendelkezésre áll, akkor kiderül, melyik mozgási szabályt kell szimulálni. Az író-olvasófej megint végiglátogatja az összes marker helyét, a szimulált szabálynak megfelelően felülírja a szimbólumot, és a markert jobbra vagy balra csúsztatja. Ennek során célszerűen rögtön ki is olvashatja az új szimbólumot, így az új állapot beállításával egyidejűen összegyűjtheti a következő lépés megtételéhez szükséges információt.

Az állapottérben meglehetősen sok információt kell tárolnunk. Minden szalagról egy szimbólumot memorizálunk. Tudnunk kell, hogy ez a szimbólum még a felülírásra használandó, vagy már az újonnan kiolvasott szimbólum-e. Minden sávra külön meg kell adni, hogy jobbra vagy balra történik-e a csúsztatás, esetleg helyben marad az író-olvasófej. Végül tudnunk kell, melyik szalagon helyezkedik el a baloldali, illetve jobboldali szélső marker.

Kétségtelen munkás dolog egy szalaggal egy **n** szalagos gépet szimulálni, de lehetséges.

A gondolatmenetet ellentétes irányban is megpróbálhatjuk, kereshetünk egy a definiáltnál egyszerűbb, de az eredeti *Turing*-géppel azonos erejű automatát. Így például az a gép, amelynek szalagja csak egyik irányban tartalmazhat tetszőlegesen hosszú információt szintén *Turing*-gép erejű.

Vegyünk ugyanis egy *Turing*-gépet, ahol a szalag természetesen mindkét irányban tetszőleges hosszban felülírható. Féloldalas gépünknek legyen dupla széles a szalagja. Az eredeti *Turing*-gép szalagját hajtsuk ketté, és egyik felének információját az új szalag felső sávján, a másikat pedig az alsó sávján tároljuk. A kettéhajtás helyére írjunk egy új, fordítókorong szemantikájú szimbólumot.

A féloldalas automata állapotában memorizálni kell, hogy a felső vagy az alsó sávon dolgozunk-e. Felülírásnál természetesen a nem használt sáv szimbólumai változatlanok maradnak. A felső sávban az eredeti és az új gép mozgási szabályai azonosak, az alsó sávban az új gép az eredetivel ellenkező irányban lép. Ha a fordítókorongra lépünk, sávot változtatunk, és egyet visszamegyünk.

Amennyiben egy veremautomatát nem egy, hanem két veremmel látunk el, akkor képességei a *Turing*-géppel lesznek azonosak. A kettős verrmű automatával egy *Turing*-gép úgy szimulálható, hogy az egyik verem az író-olvasófejtől balra, a másik a jobbra eső tartalmat tárolja. A fej alatti szimbólumot valamelyik verem tetejére rakjuk.

Induláskor a szalag tartalmát betöltjük az egyik verembe, majd annak érdekében, hogy a sorrend helyes legyen, áttöltjük a másikba. Ezek után a szalaghoz hozzá sem nyúlunk, csupán ε-mozgásokkal szimuláljuk a *Turing*-gép mozgását. Ebben a fázisban az eredeti *Turing*-gép és a kettős vermű automata szituációi egy-egy értelmű megfeleltetésben vannak. A *Turing*-gép szalagján lévő információ megváltoztatása triviális módon vihető át a két verem tartalmának megváltoztatására.

Számláló automatának hívják az olyan automatát, amelynek szalagján csak egyetlen helyen van a többitől megkülönböztető jel. Az információt az olvasófejnek ettől a jeltől lépésekben mért távolsága szolgáltatja.

Könnyű belátni, hogy egy verem mindössze két ilyen szalaggal szimulálható. Az információ tárolására tulajdonképpen elegendő volna egyetlen szalag, a másik szalagra csak a két művelet, a beírás és a kiolvasás miatt van szükség.

Legyen ugyanis a veremautomata vermének alfabetája **k-1** számosságú. Jelöljük az egyes szimbólumokat az **1** … **k-1** intervallumba eső számokkal. Legyen a veremben **m** szimbólum, az első szimbólum van a verem legalján, az **m** sorszámú a verem tetején.

Ezek után a verem tartalmát egyetlen számmal jellemezhetjük:

$$\mathbf{i}_m + \mathbf{i}_{m-1}\mathbf{k} + \mathbf{i}_{m-2}\,\mathbf{k}^2 + \ldots + \mathbf{i}_1\mathbf{k}^{m-1}$$

Itt $\mathbf{i}_j$ a **j** sorszámú verembeli szimbólumnak megfeleltetett számértéket jelöli. Minthogy ez az érték mindig kisebb, mint **k**, a megadott szám valóban egyértelműen jellemzi a verem tartalmát.

A kiolvasás mechanizmusa a következő. Kiinduláskor az egyik szalag olvasója ezt a számot memorizálja, vagyis az olvasófej éppen ennyi lépéssel távolodott el a jeltől, míg a másik szalag olvasófeje éppen a jelen áll. Az automatában van egy számláló, amely a megtett lépéseket számolja. A számot tároló szalag olvasófejét léptessük vissza egészen a jelig. Közben, amikor az olvasófej **k** lépést megtett, eggyel léptejük a másik szalagot, és a számlálót nullázzuk. Mire az információt eredetileg tartalmazó szalagon az olvasófej a jelhez ér, a számláló értéke adja a verem tetején lévő szimbólum értékét, míg a másik szalag a verem új tartalmát memorizálja. A két szalag szerepe tehát felcserélődött.

Beíráskor a kiinduló állapot ugyanaz, mint előbb, hiszen ez a számláló automata alaphelyzete. Most is visszaléptetjük az olvasófejet a jelig, de közben minden visszalépésnél a másik szalag **k** lépést tesz meg. Amikor ezzel végeztünk, akkor a beírandó szimbólum kódjának megfelelő számú lépéssel megtoldjuk a szalag lépésszámát. Könnyű belátni, hogy a tárolt érték ismét a verem új tartalmának felel meg.

Azt már megmutattuk, hogy egy *Turing*-gép egy kettős vermű automatával egyenértékű. Azt is láttuk, hogy két számláló szalaggal egy verem információ tartalma kezelhető. Ebből következik, hogy négy számlálószalaggal egy *Turing*-gépet lehet szimulálni. Ez magától értetődő.

Mint az alábbiakból kitűnik, nem kell bőkezűen négy szalaggal ellátni a számláló gépet, mert kétszalagos változata is ugyanolyan számítási képességű, mint a *Turing*-gép.

Az igazolás gondolatmenetéből csak azt a kritikus elemet érzékeltetjük, amely megmutatja, hogyan lehet két számláló szalaggal két, illetve akárhány verem tartalmát kezelni.

Azt a korábbi eredményünket, hogy egy verem tartalmát hogyan lehet egyetlen számmal jellemezni, itt is felhasználjuk. Legyen a két verem tartalmat jellemző két szám **p** és **q**. Amennyiben az egyik szalagon a

$$2^0 \bullet 3^p \bullet 5^q$$

számot tároljuk, akkor ezzel mindkét számot és így mindkét verem tartalmát hozzáférhetően ismerjük. Itt • szorzás jele. A kívánt információ kinyerése csak technikai kérdés.

Példaképpen azt az esetet részletezzük, amikor valamelyik, mondjuk, a második veremből akarjuk kiolvasni a verem tetején található szimbólumot.

Először a számlálószalag tartalmát öttel osztjuk, vagyis az olvasófejet ötösével visszaléptetjük. A másik szalagot az első szalag minden öt lépése után eggyel továbbítjuk. Amennyiben a szám osztható volt öttel, vagyis az utolsó lépésnél éppen a jelhez értünk, akkor az automata állapotában tárolt számlálón egyet számlálunk, és a két szalag szerepének felcserélésével újból osztunk öttel. Ha bekövetkezik az az állapot, amikor a kapott szám már nem osztható öttel, akkor a szalagon rekonstruáljuk az osztási kísérlet előtti állapotot.

Az állapotban tárolt számláló természetesen nem képes tetszőleges szám tárolására, hiszen az állapothalmaz véges kell legyen. Ez nem is következik be, mert valahányszor a számláló elérte a **k** értéket – **k** mint emlékezünk, a verembeli alfabeta számosságánál eggyel nagyobb szám – a szalag tartalmát megszorozzuk kettővel, és az állapotban elhelyezett számlálót nullázzuk.

Az egész hosszadalmas művelet után a számláló szalagon őrzött szám.

$$2^r \bullet 3^p \bullet 5^0$$

alakú lesz, ahol **r** értéke megfelel a második verem kiolvasás utáni tartalmának, míg az állapotban tárolt számláló állása megadja a kiolvasott szimbólumot.

Az ebben a játékban bizonyos gyakorlatra szert tett olvasó számára nem jelenthet problémát, hogy a **2** kitevőjét az **5** kitevőjének írják át.

Minthogy a prímszámok megszámlálhatóan végtelen sokan vannak, ezzel a módszerrel tetszőleges számú verem információja kezelhető egy két szalaggal bíró számláló automatával.

A *Turing*-gép lehetőségeiről szólva külön ki kell emelnünk a szimulációs képességét. Készíthető ugyanis olyan *Turing*-gép amellyel egy másik *Turing*-gép mozgási szabályait közölve az a továbbiakban ezen mozgási szabályok szerint viselkedik, vagyis szimulálja a megadott *Turing*-gépet. Az ilyen, más tetszőleges *Turing*-gépet szimulálni képes gép neve univerzális *Turing*-gép.

A fenti állításban tulajdonképpen semmi meglepő sincsen. A számítógépeknél ezt a trükköt széles körben alkalmazzák. Minden számítógépnek ugyanis megvan a maga gépi utasítás rendszere, az adott számítógép anyanyelve. Ez a *Turing*-gép mozgási szabályainak analogonja.

Ez a gépi utasításkészlet azonban nagyon nehezen kezelhető, és éppen ezért egy másik utasításkészletet, egy számítógépes nyelv leírását tápláljuk be valamilyen formában a számítógépbe, és ennek hatására a számítógép úgy viselkedik, mintha a számítástechnikai nyelv lenne a számítógép anyanyelve.

Az itt használt terminológiával azt a tényt, hogy a gép nem saját nyelvének, hanem a számítástechnikai nyelvnek az utasításait érti meg, úgy interpretálhatjuk, hogy a gép egy másik gép mozgási szabályai szerint működik.

Az univerzális *Turing*-gép realizálásának egyik lehetséges módja, hogy a szimulálandó gép mozgási szabályainak kódolt leírását a szalagon hagyjuk, és egy szimbólum olvasásakor az univerzális *Turing*-gép a szalagon található specifikációval konzultálja meg a teendőket.

Az alábbiakban vázlatosan ismertetjük az univerzális *Turing*-gép egyik ilyen elvű megvalósítását.

Kódoljuk mind a szimulálandó *Turing*-gép, mind a bemenő adatok leírását ugyanabban a kódrendszerben. Legyen ez az egyes számrendszer. Ebben csak két szimbólumot használunk, például az egyest és a nullát. Ez utóbbit csak mint elválasztó karaktert alkalmazzuk.

Számozzuk meg sorjában mind az automata állapotait, mind a szalag alfabetájának szimbólumait, és kódoljuk azokat számozásuknak megfelelő számú egyessel. Az egyesek ilyen sorozatának végét mindig az elválasztó szimbólum, esetünkben a nulla jelzi.

A *Turing*-gép mozgási szabályainak baloldalán egy állapotból és egy olvasott szimbólumból álló kettős, míg jobboldalán egy hármas áll. Ez utóbbi egy állapotot, egy szimbólumot és egy jobbra vagy balra léptetési utasítást tartalmaz.

A szabályok leírásánál azoknak csak jobboldalát adjuk meg, a baloldalt pozícionálisan kódoljuk.

Egy mozgási szabály jobboldalát három, egyesekből álló, és egymástól egyetlen nullával elválasztott sorozattal jellemezzük. Az első az új állapot, a második a felülírásra használt szimbólum sorszámát adja, míg a harmadik a léptetés irányát jelzi. Például egy egyes jobbra, két egyes balra léptetést jelent.

Az ugyanahhoz az állapothoz tartozó mozgási szabályokat egymás után, az olvasott szimbólum sorszámának megfelelő sorrendben, de most már két nullával elválasztva adjuk meg. Itt ügyelnünk kell arra, hogy a legelső, tulajdonképpen a **0** sorszámhoz tartozó helyet az *ы* szimbólumnak kell fenntartanunk.

Természetesen, mivel a mozgási szabályok baloldalát a pozíció határozza meg, nem hagyhatunk ki egyetlen kombinációt még akkor sem, ha erre nincsen mozgási szabály. Ilyenkor valamilyen jobboldalként nem értelmezhető jelsorozatot alkalmazhatunk. Ugyanígy jeleznünk kell, ha valamely állapothoz nem tartozik egyetlen jobboldal sem, tehát egy egész sor marad el. Ez esetben a teljes sor helyett alkalmazhatunk egy másképpen nem értelmezhető jelölést. Kivételt képez az az eset, amikor a jobboldal utolsó a sorban, illetve a sor utolsó a felsorolásban. Ilyenkor ugyanis a mozgási szabály hiánya detektálható.

Az egyes állapotokhoz tartozó információt három-három nullával elválasztva fűzzük egymáshoz, míg a teljes leírást négy nullával kezdjük és zárjuk.

A mozgási szabályokat követően meg kell adnunk az elfogadó állapotok halmazát egymás után felsorolva az elfogadó állapotok sorszámát.

Ezzel az automata egyértelműen specifikálva van. Itt azzal az egyszerűsítő feltételezéssel éltünk, hogy az automata determinisztikus, hiszen az előbbi leírás szerint egy állapot–olvasott karakter párhoz legfeljebb egy mozgási szabályt vettünk figyelembe. Nemdeterminisztikus automaták ilyen leképezése kissé bonyolultabb, de ez csak technikai és nem elvi problémát jelent.

Egyébként – mint később kitűnik – minden nemdeterminisztikus *Turing*-géphez szerkeszthető egy vele ekvivalens, ugyanazt a nyelvet elfogadó determinisztikus *Turing*-gép.

A szalagnak alkalmasnak kell lennie markerjelek befogadására is.

Az ilyen univerzális *Turing*-gép működési elve a következő.

Induláskor a specifikáció legelején és a bemenő adat első szimbólumánál található marker. Az első feladat az aktuális mozgási szabály megtalálása. Az állapotot ismerjük, hiszen az a kezdőállapot, amelyik markerrel van megjelölve.

Az olvasott szimbólum sorszáma szabja meg a keresett jobboldal helyzetét. Egyenként az adatszimbólum markerének odébb csúsztatásával leolvassuk az adatszalagon az egyesek számát, miközben minden egyesnél visszamegyünk a specifikációhoz, és az ott található markert addig léptetjük jobbra, amíg egy kettős nullán át nem halad. Világos, hogy mire az olvasott szimbólum egyesei elfogynak, a specifikáció markere éppen az aktuális szabály jobboldalának elején áll, feltéve persze, hogy ilyen szabály létezik.

Következik az új állapot kijelölése. Ebből a célból visszamegyünk a specifikáció elejére, amelyet egy újabb markerrel látunk el. A megfelelő állapot megtalálására ugyanazt a technikát alkalmazzuk, mint előbb, csak most a már ismert aktuális jobboldalban számoljuk le az állapotokat jelző egyeseket, és most nem két, hanem három nullából álló sorozatokat kell átlépnünk. A művelet végén a marker az új állapot indulási helyén lesz.

Ezután kerülhet sor az olvasott szimbólum felülírására. Az aktuális jobboldal második elemét – ide-oda vándorolva – összehasonlítjuk az olvasott szimbólum sorszámával. Ha az új szimbólumnak más a sorszáma, akkor az egyeseket beszúró, illetve kitörlő szubrutint hívjuk segítségül.

Utolsó lépésként az adatszalagon kell a markert az aktuális jobboldal harmadik elemének megfelelően, egy szimbólummal jobbra vagy balra léptetnünk.

Vegyük észre, hogy az egész nem túl bonyolult, de meglehetősen hosszadalmas műveletsorozat végén a markerek a helyükön állanak, tehát hozzákezdhetünk a következő mozgás szimulálásához.

Ha valamelyik kombinációhoz nem tartozik mozgási szabály, akkor az automata megáll. Ilyenkor meg kell állapítani, hogy a megállás állapota elfogadó állapot-e vagy sem. Az állapotot a marker helye jelzi. A specifikáció végén szerepel az elfogadó állapotok listája, ennek alapján az előbbiekhez hasonló technikával kideríthetjük, hogy a szimulált gép a jelsorozatot elfogadta-e vagy sem.

A fent vázolt módszerrel persze csak az egyik irányban korlátlan hosszúságú szalaggal bíró *Turing*-gépet modellezhetjük, hiszen a szalag egy része, az, ahol a specifikáció van, tabu, amelyet nem írhatunk felül. Ez azonban nem jelent korlátozást, mert mint láttuk, minden *Turing*-géphez szerkeszthető egy ilyen, szalagját csak az egyik irányban korlátlan hosszban használó, és az eredeti *Turing*-géppel ekvivalens, vagyis ugyanazt a nyelvet elfogadó *Turing*-gép.

Egy ilyen, a fent vázolt megoldású univerzális *Turing*-gép nagyságrendileg 60 mozgási szabállyal megoldható. A szabályok nem bonyolultak, inkább az okozhat problémát, hogy az egyébként egyszerű működésmódot aprólékosan át kell gondolni.

Példaképpen megadjuk egy *Turing*-gép olyan leírását, amelyet a fenti univerzális *Turing*-gép a szimuláció céljaira felhasználhat.

Legyenek a szimulálandó *Turing*-gép mozgási szabályai a következőek:

$$\delta(q_1, a) = (q_1, a, r)$$
$$\delta(q_1, b) = (q_2, b, r)$$
$$\delta(q_2, a) = (q_1, a, r)$$
$$\delta(q_2, b) = (q_2, b, l)$$

Azt hiszem az olvasó felismerte, hogy ez a *Turing*-gép ugyanazt a feladatot látja el, mint amit a kétirányú mozgást végző véges automata példájában adtunk meg. A példa demonstratív jellegű, és az egyszerűsítésre való törekvés céljából használtunk *Turing*-gépet olyan probléma kezelésére, amely véges automatával is megoldható.

A fenti gép specifikációjának leírása a következő:

```
0000 1 00 10101 00 1101101
 000  1 00 10101 00 11011011 0000
       1 0 11                  0000
```

Talán helyes ehhez a specifikációhoz néhány magyarázó szót hozzáfűzni.

A szóközök természetesen nem szignifikánsak, és csakis az áttekinthetőség növelése érdekében szerepelnek. Minthogy csak két állapot van, az állapotokat elválasztó hármas nulla csak egyszer fordul elő. Bár könnyen belátható, hogy az automata sohasem téved *ы* szimbólumokkal jelzett területre, a leírásban erre is fel kell készülni. Ezért áll a nulladik szimbólum helyén az **1 00** jelsorozat.

Minthogy mindkét állapot elfogadó állapot, sorszámuk szerepel az elfogadó állapotok listáján.

Vegyük észre, hogy csupa elfogadó állapota ellenére ez az automata sem fogad el minden jelsorozatot. A visszautasítás módja itt azonban a végtelen ciklus, vagyis az automata sohasem áll meg.

6.3. A megállási probléma

Az előbbi példánk *Turing*-gépéről szinte ránézésre meg tudjuk állapítani, hogy bizonyos bemeneti jelsorozatokra végtelen ciklusba megy, sohasem fog megállni.

Felvetődik a gondolat, vajon ez a kérdés általánosságban megválaszolható-e vagy sem? Legyen adott egy *Turing*-gép és egy jelsorozat. Eldönthető-e bármely *Turing*-gép, és tetszőleges jelsorozat esetében, hogy ez a gép az adott jelsorozatra valaha is megáll. A nemdeterminisztikus gépekre is gondolva fogalmazzunk szabatosabban. Adott a gép, és adott a bemeneti jelsorozat. A kérdés így hangzik: létezik-e olyan mozgássorozat, amelyre a *Turing*-gép megáll.

A probléma alapvető ismeretelméleti elveket érint, és jelentősége messze túlmutat azon a feladaton, amelynek kapcsán a kérdést megfogalmaztuk.

Mielőtt ennek tárgyalásába belefognánk, tekintsük át azokat az explicite ki nem mondott eredményeket, amelyeket az univerzális *Turing*-gép vizsgálata során kaptunk.

A lehetséges *Turing*-gép specifikációk, vagyis azok a jelsorozatok, amelyek egy gép specifikációjának tekinthetőek, egy nyelvet, mégpedig környezetfüggetlen nyelvet alkotnak. Minthogy minden *Turing*-géphez szerkeszthető ilyen specifikáció, ez a nyelv valamennyi *Turing*-gép leírását tartalmazza. Ebből viszont következik, hogy az összes *Turing*-gép száma megszámlálhatóan végtelen, és így beszélhetünk első, második, stb. *Turing*-gépről.

Egy másik fontos tanulság, hogy a gép leírása és bemeneti adatai ugyanabban a kódban jeleníthetőek meg. Szeretném itt felhívni a figyelmet a számítógépek híres neumanni elvére, amely szerint a számítógép nem tesz különbséget program jellegű és adat jellegű információ között.

Valóban, itt sincs semmiféle akadálya annak, hogy egy gép leírását adatnak tekintsük, és feldolgoztassuk egy *Turing*-géppel. Nyilvánvaló, hogy a gép semmi módon nem érzékeli, hogy a beadott jelsorozat akár egy *Turing*-gép leírása is lehet, hiszen az információ funkcióját nem az információ jellege, hanem a felhasználás módja szabja meg. Itt pedig az információ adatként szerepel.

Megtehetjük azt is, hogy egy *Turing*-gépnek saját leírását adjuk be adatként. Vegyük sorra a *Turing*-gépeket, és bemenetként kapják meg saját specifikációjukat. Kérdés, hogyan reagálnak erre a bemenetre a gépek?

Nyilván lesznek olyanok, amelyek elfogadják, de olyanok is, amelyek visszautasítják ezt a bemenetet.

Azok a leírások, amelyeket saját gépük elfogad, de azok is, amelyek visszautasításban részesülnek, egy-egy nyelvet alkotnak. Legyen a két nyelv neve $\boldsymbol{L}_{01}$ és $\boldsymbol{L}_{02}$. A **0** index itt arra utal, hogy a *Turing*-gépek a **0**-ás nyelvosztály nyelveit fogadják el, bár ennek igazolásával még adós vagyok.

Az $\boldsymbol{L}_{01}$ nyelv tehát azon *Turing*-gépek specifikációit tartalmazza, amelyek elfogadják saját leírásukat. Vajon készíthető-e olyan *Turing*-gép, amely ezt a nyelvet fogadja el?

Erre a kérdésre a válasz pozitív. Egyszerű eszközökkel készíthetünk olyan gépet, amely éppen ezt a nyelvet fogadja el.

Először is szűrjük ki azokat a jelsorozatokat, amelyek nem tekinthetőek *Turing*-gép leírásnak. Minthogy a *Turing*-gép leírásokat egy környezetfüggetlen nyelvtan generálja, ezt a feladatot egy veremautomata is elláthatja.

A továbbiakban már csak olyan jelsorozatokkal foglalkozunk, amelyek gépek leírásai, a többit a veremautomata kiszűrte.

Másoljuk le ezután a szalagon található specifikációt még egy példányban, és azt tápláljuk be bemenetként egy univerzális *Turing*-gépnek. Az univerzális gép a leírás első példányát valóban leírásnak, a másodikat azonban adat fogja tekinteni.

Így az univerzális *Turing*-gép a specifikációban szereplő gépet szimulálva, feldolgozza annak leírását. Nyilvánvaló, hogy az ily módon konstruált gép éppen az $\boldsymbol{L}_{01}$ nyelvet fogja elfogadni.

Mi a helyzet a másik, az $\boldsymbol{L}_{02}$ nyelvvel, amely a saját specifikációjukat visszautasító *Turing*-gépek leírását tartalmazza? Vajon erre a nyelvre is szerkeszthető *Turing*-gép?

Sajnos nem. Tételezzük fel ugyanis az ellenkezőjét, vagyis azt, hogy létezik egy az $\boldsymbol{L}_{02}$ nyelvet elfogadó *Turing*-gép. Ez a gép persze szükségképpen egyike a megszámlálhatóan sok *Turing*-gépnek. Természetesen ennek is van specifikációja, és ennek a gépnek is odaadható a saját leírása.

Kérdés, hogyan reagál ez a gép a saját leírására? Ha elfogadja, az hiba, mert ekkor a leírás nem mondata az $\boldsymbol{L}_{02}$ nyelvnek, és így a gép nem fogadhatná el. Viszont az is baj, ha nem fogadja el, hiszen ilyenkor ez egy olyan *Turing*-gép

leírása, amelyet saját gépe nem fogadott el, tehát mondata az $\boldsymbol{L}_{02}$ nyelvnek, és így ezt el kellene fogadnia. Mindkét esetben ellentmondásra jutottunk, így ezek szerint eredeti feltevésünk volt helytelen, Nincsen tehát az $\boldsymbol{L}_{02}$ nyelvet elfogadó *Turing*-gép, ez a nyelv nem generatív nyelv.

Az ilyen feloldhatatlan dilemmák kérdése már régóta, az antik görög korból is ismeretes. Had említsek meg egy ismert, későbbi esetet, a katonai borbély problémáját. A borbély azt a parancsot kapja, hogy borotváljon meg mindenkit, aki nem maga borotválkozik. Ki borotválja a katonai borbélyt?

Emlékeztetnék arra, hogy minden nyelvtannal generálható nyelvre, vagyis generatív nyelvre készíthető *Turing*-gép. Annak idején a formális nyelvek definiálásakor megállapítottuk, hogy egy adott alfabeta felett kontinuum számosságú nyelv létezik. A generatív nyelvek ennek a halmaznak egy megszámolhatóan végtelen számosságú részhalmazát képezik.

Eddig mindig olyan nyelvekkel volt dolgunk, amelyek ebbe a megszámlálható számosságú halmazba estek. Az $\boldsymbol{L}_{02}$ nyelv nem tartozik bele ebbe a sokaságba, nem generatív nyelv. Felhívnám a figyelmet, hogy ezzel egy Rubicont léptünk át. Minthogy többet ilyen, nem generatív nyelvvel nem lesz dolgunk, talán helyesebb úgy mondani, hogy átnéztünk a Rubicon túlsó oldalára.

Ez után az előtanulmány után, térjünk vissza eredeti kérdésünkhöz. Eldönthető-e a *Turing*-gép megállási problémája, vagyis meg tudjuk-e állapítani, hogy egy adott *Turing*-gép egy adott bemenetre megáll-e vagy sem? A válasz negatív.

Tételezzük fel ugyanis az ellenkezőjét. Ez esetben szükségképpen létezik egy olyan algoritmus, amellyel közölve a *Turing*-gép leírását, és a bemenetet, mindenkor megállapíthatjuk, megáll-e a gép vagy sem.

Minthogy hiszünk a *Church*-tézisben, erre az algoritmusra *Turing*-gépet is szerkeszthetünk. Becézzük ezt a hipotetikus gépet szuperokos *Turing*-gépnek. Ha létezik ilyen szuperokos *Turing*-gép, akkor viszont könnyen szerkeszthetünk egy olyan *Turing*-gépet, amely az $\boldsymbol{L}_{02}$ nyelvet fogadja el.

Legyen adott egy tetszőleges *Turing*-gép specifikációja. Adjuk ezt át két példányban a szuperokos *Turing*-gépnek, egyszer mint gépleírást, egyszer mint adatot.

A szuperokos gép megmondja megáll-e a specifikált gép a bemeneti adatra, vagyis jelen esetben megáll-e a gép saját leírását elemezve. Ha nem áll meg, ez visszautasítást jelent, ez a leírás tehát mondata az $\boldsymbol{L}_{02}$ nyelvnek. Éppen ezért a szuperokos géptől nyert információ alapján a nyelv elfogadására szerkesztett gépünket elfogadó állapotban állítjuk meg.

Más a helyzet, ha a szuperokos gép vizsgálata szerint a gép megáll az adott bemenetre. Ekkor az univerzális gépet hívjuk segítségül, és a gép leírását ennek adjuk át két példányban.

A szimuláció során az univerzális gépnek meg kell állnia, hiszen erre kötelezi őt a szuperokos gép ítélete. Megállás után el tudjuk dönteni, elfogadná-e vagy sem a gép saját leírását.

Ha elfogadja, akkor a jelsorozat nem mondata az $\boldsymbol{L}_{02}$ nyelvnek, míg ha visszautasítja, akkor a nyelv egy mondatát kaptuk. Ennek megfelelően az $\boldsymbol{L}_{02}$ nyelvre szerkesztett *Turing*-gép az első esetben visszautasít, a másodikban elfogad.

Persze ezek mind csak feltételezések. Mi más okokból már tudjuk, hogy az $\boldsymbol{L}_{02}$ nyelvre nem lehet *Turing*-gépet szerkeszteni. Az az okoskodás, amely ilyen gép kialakítására vezet, valahol hibázik. Adott esetben ez a szuperokos *Turing*-gép létének feltételezése volt. Nincs tehát szuperokos gép, a megállási problémára nem készíthető algoritmus.

Az a tény, hogy valamely problémára, pontosabban problémaosztályra nem készíthető algoritmus annyit jelent, hogy erre a feladattípusra nem adható meg olyan módszer, amely általános, vagyis minden ebbe a kategóriába eső feladatnál alkalmazható. Ez nem zárja ki, hogy egyes konkrét feladatokat egy szellemes ötlet, vagy heurisztika igénybevételével meg ne oldjunk.

Így például ismeretes, hogy negyedfokúnál magasabb kitevőjű polinomokra nem szerkeszthető megoldó képlet. Ezzel együtt bizonyos polinomok megoldása ügyeskedéssel vagy akár ránézésre is megadható. Így az $x^5 = 32$ egyenletnek egyik megoldása nyilván 2.

Visszatérve a *Turing*-gép megállási problémájához, azt már láttuk, hogy vannak olyan gépek, amelyek nem állnak meg bizonyos bemenetekre. Ilyen volt például az az univerzális géppel szimulált *Turing*-gép is, amely minden visszautasított jelsorozat esetén végtelen ciklusba került.

Az adott nyelvnél a végtelen ciklus nem szükségszerű. Habkönnyen szerkeszthetnénk olyan, ugyanezt a nyelvet elfogadó gépet, amely minden bemenetre megáll, persze nem mindig elfogadó állapotban. Itt tehát nem a probléma természete, hanem a *Turing*-gép szerkesztőjének felületessége vagy ügyetlensége miatt kaptunk egy nem minden esetben megálló gépet.

Felmerül a kérdés, elkerülhetetlen a megállni képtelen *Turing*-gépek tárgyalása, nem lehet-e minden feladatra mindig megálló *Turing*-gépet szerkeszteni. Sajnos nem. Vannak olyan nyelvek, amelyeket elvben sem lehet egy minden bemenetre megálló *Turing*-géppel elfogadtatni.

Ilyen nyelv például az előbb tárgyalt $\boldsymbol{L}_{01}$ nyelv. Tételezzük fel ugyanis, hogy létezik egy minden bemenetre megálló és ezt a nyelvet elfogadó automata. Mint tisztáztuk azokat a jelsorozatokat, amelyek nem egy *Turing*-gép leírását adják, könnyű kiszűrni. Ha most csak az olyan jelsorozatokra szorítkozunk, amelyek egy gép specifikációját adják, akkor az $\boldsymbol{L}_{01}$ nyelv mondataira a gép elfogadó, az $\boldsymbol{L}_{02}$ nyelv mondataira pedig visszautasító állapotban fog megállni.

Ha most egy olyan *Turing*-gépet szerkesztünk, amely a fenti, feltételezésünk szerint mindig megálló gépnek a válaszait megfordítja, vagyis ha az visszautasít, mi elfogadunk, ha elfogad, mi visszautasítunk, akkor ez a gép éppen az $\boldsymbol{L}_{02}$ nyelvet fogadja el. Minthogy – mint tudjuk – erre a nyelvre *Turing*-gép nem szerkeszthető, feltételezésünk nem állja meg a helyét. Nincs tehát olyan az $\boldsymbol{L}_{01}$ nyelvet elfogadó *Turing*-gép, amely minden bemenetre megállna.

A *Turing*-gép megállása híres, nevezetes probléma. Ez volt ugyanis az első olyan feladat, amelynek algoritmikusan eldönthetetlen voltát felismerték. Ez volt a kérdés, amely az addig kétpólusú igen-nem válaszokon alapuló, megoldható, nem megoldható lehetőségeket ismerő világunkat megzavarta.

Tehát most már három lehetőségben kell gondolkoznunk, megoldható, nem megoldható és eldönthetetlen. Sajnos nem, világunk nem hárompólusú. Vannak ugyanis olyan problémák, amelyeknek az eldönthetetlen voltát sem tudjuk megállapítani. Sőt ez a gondolatmenet folytatható. Elképzelhető olyan feladat, amelynél az eldönthetetlenség kimutatásának lehetőségét sem vagyunk képesek megállapítani. És ezt vég nélkül tovább lehet folytatni. A megismerhetőség horizontja végtelen.

Elnézést, hogy itt kissé patetikus, sőt talán szentimentális szólamokra ragadtattam el magam, de megpróbáltam érzékeltetni, milyen hatással volt rám ennek a diszciplínának a megismerése.

6.4. A *Turing*-gép és a 0-ás osztályú nyelvek

Régi becsületbeli adósságunkat rójuk le akkor, amikor igazoljuk a *Turing*-gépek és a **0**-ás osztályú nyelvek ekvivalenciáját. Ehhez két dolgot kell belátnunk. Egyrészt be kell bizonyítanunk, hogy minden **0**-ás osztályú nyelvhez szerkeszthető egy olyan *Turing*-gép, amely pontosan a nyelvtan által generált mondatokat, és csakis azokat fogadja el. Másrészt viszont igazolnunk kell, hogy minden *Turing*-gépnek megfeleltethető egy olyan 0 osztályú nyelvtan, amely éppen a gép által elfogadott jelsorozatokat, és csakis azokat generálja.

Ami az állítás első részét illeti, ez számunkra, akik hiszünk a *Church*-tézisben, trivialitás. A generatív nyelvtanok segítségével egy mondat generálása algoritmizálható feladat, és így hivatalból létezik a megszámlálhatóan végtelen sok *Turing*-gép között egy olyan *Turing*-gép, amely éppen az adott nyelvre pontosan ezt a feladatot végzi el.

Pusztán annak érdekében, hogy az olvasót egy ilyen *Turing*-gép feltalálásának fáradságos munkájától megkíméljük, az alábbiakban megadjuk egy ilyen gép szerkesztésének lehetséges megoldását, helyesebben annak vázlatát.

Először azt kell elképzelnünk, hogyan képezhetőek le egy nyelvtan levezetési szabályai *Turing*-gép segítségével. Tételezzük fel, hogy a gép

egy mondatszerű forma van. Az automata elolvassa ezt a mondatszerű formát, és megkeresi azokat a fragmenseket, amelyek valamely levezetési szabály baloldalán állnak. Ezek közül egyiket – a választás rajtunk áll – a szabály jobboldalával, pontosabban – ha netán több szabály baloldalán szerepelne ugyanaz a fragmens – egyik jobboldalával felülírjuk.

Ez a művelet, vagyis a baloldal felismerése, és annak a jobboldallal való felülírása az automata állapothalmazától bizonyos memorizáló képességet követel meg. Ez azonban csak technikai probléma, lévén a levezetési szabályok száma és hosszúsága is véges, tehát véges a memorizálandó információ mennyisége is.

Ennek egyik lehetséges megoldása, ha a helyettesítési szabályokat kódolt formában szalagra írjuk, és olyan gépet szerkesztünk, amely – hasonlóan az univerzális *Turing*-géphez – a szalagon található információ alapján teszi meg lépéseit.

Természetesen, amennyiben a levezetési szabály baloldala és jobboldala nem egyforma hosszú, akkor a felülírási művelet még szimbólumok törlésével, illetve beszúrásával járhat, de ez már megoldott kérdés. Az a tény, hogy egy adott mondatszerű forma esetében az alkalmazható levezetési szabályok közül csak egyet helyettesítünk, azt jelenti, hogy automatánk nemdeterminisztikus automata lesz. Nyilvánvaló, hogy egy ilyen automata a nyelvtan bármely mondatát előállíthatja.

Egy tetszőleges **0**-ás osztályú nyelvtannal ekvivalens *Turing*-gép felépítése ennek alapján a következő lehet.

Induláskor a szalagon természetesen az a jelsorozat van, amelyet a szóban forgó nyelv egy mondatának hiszünk. Írjunk a jelsorozat után egy minden nyelvtani szimbólumtól különböző elválasztó jelet, majd a mondatszimbólumot.

Ezek után adjuk át a teret a levezetési szabályokat leképező automatának, amely a mondatszimbólumból kiindulva mondatszerű formák sorozatán keresztül egy mondatot generál, ha generál. Elképzelhető ugyanis, hogy a mondatszerű formák sorozata sohasem eredményez mondatot.

Amennyiben a generálás eredménye egy mondat, akkor összevetjük a bemenetként megadott jelsorozattal. Ha a két jelsorozat azonos, az automata elfogadó, ha eltérő, visszautasító állapotban áll meg.

Amennyiben olyan mondatszerű formához érünk, amely tartalmaz ugyan nemterminális szimbólumokat, de tovább nem deriválható, más szóval a levezetés fiaskóval végződött, az automata akkor is visszautasító állapotban áll meg.

Világos, hogy az így definiált *Turing*-gép csakis azokat a jelsorozatokat fogadja el, amelyek a nyelvtan által generált mondatok.

A fentiekben leírt *Turing*-gép nemdeterminisztikus automata. Kis módosítással azonban elkészíthetjük ennek determinisztikus változatát.

Ez a lépés nagy horderejű, mert igazolja a nemdeterminisztikus és determinisztikus *Turing*-gépek ekvivalenciáját, és megmutatja, hogy minden **0**-ás

osztályú nyelvhez szerkeszthető determinisztikus automata. Ilyen élményben mióta elhagytuk a reguláris nyelveket nem volt részünk.

A determinisztikus változat működését éppen úgy kezdi, mint a nemdeterminisztikus megfelelője. Induláskor a bemeneti jelsorozat után odaírja, persze elválasztva a mondatszimbólumot.

Ezt követően azonban a vizsgált mondatszerű formából – jelen esetben induláskor a mondatszimbólumból – az egy levezetési lépésben származtatható valamennyi mondatszerű formát sorban, egymástól elválasztva felírja a szalagra, majd továbblép a következő mondatszerű formára, és megismétli ezt a műveletet. Így a szalag egy sereg, a későbbiekben elemzendő mondatszerű formát tartalmaz, amelyek aztán sorban vizsgálatra kerülnek.

Amint az aktuálisan vizsgált mondatszerű formában egy levezetési szabály baloldalát fedezzük fel, akkor a lista végére írjuk azt a mondatszerű formát, amely az eredetiből a megtalált baloldal helyettesítésével keletkezik. Amennyiben a szóban forgó baloldal többször szerepel az elemzett mondatszerű formában, akkor a szabály valamennyi alkalmazásával előálló mondatszerű formát sorban leírjuk, majd visszatérve az aktuális mondatszerű formához, abban újabb baloldalakat keresünk.

Ha már valamennyi deriválási lehetőséget kimerítettünk, vagyis nincs már több felderítetlen baloldal a mondatszerű formában, akkor továbblépünk a következő elemzendő mondatszerű formára.

Így a szalagra kerül a mondatszimbólumból egy lépésben, majd két lépésben, stb. elérhető valamennyi mondatszerű forma, illetve ha az nem tartalmaz nemterminális szimbólumot, akkor mondat. Így folytatva a procedúrát egyetlen mondatszerű forma sem marad ki.

Ha a derivátum mondatnak bizonyul, akkor összevetjük a bemenetként megadott jelsorozattal. Ha a két jelsorozat azonos, akkor elfogadó állapotban megállunk. Eltérés esetén folytatjuk az elemzést.

Ezzel nem csak állításunk első részét tettük érzékletesebbé, hanem mint melléktermékeket, egy fontos tételt is igazoltunk a nemdeterminisztikus és determinisztikus gépek ekvivalenciájáról.

Ami állításunk második részének igazolását illeti bizonyításunk konstruktív lesz. Egy tetszőleges *Turing*-gépből kiindulva kialakítjuk annak a **0**-ás osztályú nyelvtannak a szabályait, amely éppen az automata által elfogadott nyelvet generálja.

Mielőtt a tárgyalás részleteiben elmerülnénk, ismerkedjünk meg a *Turing*-gép konfigurációjának leírására használt egyszerű és szellemes jelöléssel, amelyet nem csak most, de később is alkalmazni fogunk.

Valamely automata konfigurációja definíció szerint tartalmazza mindazokat az információkat, amelyek alapján az automata további működését egyértelműen el lehet dönteni.

A *Turing*-gép esetében a konfiguráció leírásának tartalmaznia kell az automata állapotát, a szalagon található érdemi, tehát a *ы* szimbólumtól eltérő szimbólumok sorozatában megtestesített információt, végül meg kell mondania, hol áll az automata író-olvasófeje a szalagon.

Ezt a hármas információt egyetlen jelsorozattal is megadhatjuk. Írjuk le ugyanis a szalag tartalmát, és szúrjuk be ebbe a jelsorozatba az automata állapotát jelző, a szalagbeli szimbólumoktól természetesen eltérő, szimbólumot. Ha most azzal a konvencióval élünk, hogy az automata író-olvasófeje mindig az állapotot jelző szimbólum jobboldali szomszédja felett áll, akkor ezzel az egyetlen jelsorozattal a teljes konfigurációt jellemezni tudtuk.

Példaképpen, ha egy *Turing*-gép kezdőállapota q_0 és a bemeneti jelsorozat w, akkor az induló konfigurációt leíró jelsorozat

$$q_0\, w$$

amely nem csak a szalag tartalmát és az automata állapotát mondja meg, de azt az információt is közli, hogy az író-olvasófej a legelső szimbólum felett van.

Ennek alapján egy *Turing*-gép működését konfigurációk sorozataként is megadhatjuk, ahol a konfigurációkat a nekik megfelelő jelsorozatokkal fogjuk azonosítani.

Egy ilyen konfigurációt jellemző jelsorozatból a következő konfiguráció jelsorozatát úgy kapjuk, hogy az olvasott szimbólum helyébe a felülíró szimbólumot írjuk, míg az új állapotnak megfelelő szimbólumot – a mozgás irányától függően – az eredeti szimbólum helyétől eggyel jobbra vagy balra szúrjuk be.

Erre a leírási módra gondolva a *Turing*-gép működési szabályait felfoghatjuk úgy is, mint jelsorozatok átírására szolgáló szabályokat. Az eredeti és az új átírási szabályok között a megfeleltetés triviális.

Az átírási szabályokra a grammatikák leírásánál alkalmazott jelöléseket használva:

$$\delta(q, a) = (p, b, r) \qquad qa \to bp \tag{6.4.}$$

$$\delta(q, a) = (p, b, l) \qquad Xqa \to pXb \tag{6.5.}$$

ahol X tetszőleges szalagbeli szimbólum, beleértve a *ы* szimbólumot is.

Most, hogy a *Turing*-gép specifikációját grammatikai szabályokkal tudjuk megadni, vázolhatjuk a *Turing*-gép nyelvét generáló **0** osztályú nyelvtan szerkesztésének elvét.

Először is módosítsuk a *Turing*-gépet olymódon, hogy csak egyetlen elfogadó állapota legyen. Ennek megoldása nem okozhat gondot. Vezessünk be ugyanis egy új, elfogadó állapotot, és amikor az automata elfogadó állapotában

nincs mozgási szabály, tehát a gép megállna, tegyünk még egy lépést, és menjünk át az új elfogadó állapotba. Hogy ennél a lépésnél mivel írjuk felül az olvasott szimbólumot, és merre lépünk teljesen közömbös, az a lényeges, hogy az új elfogadó állapothoz nem tartozik mozgási szabály, vagyis az automata megáll.

Ezzel a megoldással a régi állapotok átminősítését is megtakaríthatjuk, hiszen ezekben az állapotokban a *Turing*-gép sohasem fog megállni, így az állapotok hovatartozása ki sem derül.

A mondatok generálása három részre osztható, és ennek megfelelően a nyelvtani szabályok is három részre partícionálhatóak. A nyelvtan először egy a *Turing*-gép kezdeti konfigurációjának megfelelő jelsorozatot generál. Ennek a jelsorozatnak az első szimbóluma természetesen a kezdőállapot, de a bemeneti jelsorozat tetszőleges, attól a nyilvánvaló megkötéstől eltekintve, hogy szimbólumai a Σ alfabeta elemei lesznek.

Ezt a funkciót megvalósító szabályok univerzálisak, vagyis függetlenek a leképezendő *Turing*-gép specifikációjától.

A következőkben a kezdeti konfigurációból mint mondatszerű formából kiindulva, a *Turing*-gép mozgását tükröző (6.4.) és (6.5.) típusú szabályok segítségével lépünk mondatszerű formáról mondatszerű formára. A mondatszerű formáknak ez a sorozata lényegében a *Turing*-gép konfigurációinak sorozatát adja.

A levezetésnek ez az a része, amely a leképezendő automata specifikumától függ.

Amennyiben a *Turing*-gép működésének szimulálása során a gép elfogadó állapotban áll meg, akkor a bemenetként beadott jelsorozatot a gép elfogadta.

A folyamat harmadik részének lefutására csak ebben az esetben kerül sor, tehát akkor, ha a bemenetet a *Turing*-gép elfogadta. A nyelvtan ilyenkor előállítja az eredeti jelsorozatot, mint a nyelv egy mondatát.

Az ezt a célt szolgáló nyelvtani szabályok is univerzálisak, függetlenek a konkrét *Turing*-géptől.

Ezen áttekinthető és világos módszer alkalmazásánál problémát okoz az a körülmény, hogy a *Turing*-gép működésének szimulálásakor meg kell változtatnunk azt a jelsorozatot, amelyet később esetleg mondatként generálnunk kell.

A megoldás kulcsa az, hogy a bemeneti jelsorozatot két példányban generáljuk. A szimuláció alkalmával csak az egyik példányt módosítjuk, a jelsorozat másik példánya változatlan marad. Így képesek leszünk, ha úgy hozza a sors, hogy az eredeti bemenetet rekonstruáljuk.

Nyelvtanunk nemterminális szimbólumai így két részből tevődnek össze. Lesznek a *Turing*-gép állapotát tükröző szimbólumok, és lesznek két-két szalagbeli jelet tartalmazók. Ezeken kívül lesz még néhány magától értetődő funkciójú nemterminális is.

A levezetési folyamat első fázisában csakis olyan nemterminálisokat használunk, amelyeknek két szimbóluma azonos. Ez felel meg annak, hogy a jelsorozatot két példányban generáljuk.

A folyamat második részében a szimbólumpárok közül csak az egyik, mondjuk a jobboldali a mértékadó. A *Turing*-gép szimulálásakor ezek játszzák a konfiguráció szerepét, a baloldaliak megtartják értéküket.

Ha sor kerül a folyamat harmadik részére, akkor az viszont csak a baloldali szimbólumokat érinti, hiszen azok alapján kell rekonstruálnunk az eredeti jelsorozatot.

Most már felírhatóak a nyelvtani szabályok. Íme az első csoport

$$\begin{aligned} S &\rightarrow [ы, ы]\, A_1 \\ A_1 &\rightarrow [ы, ы]\, A_1 \mid q_0 A_2 \\ A_2 &\rightarrow [x, x]\, A_2 \mid A_3 \qquad \forall\, x \in \Sigma \\ A_3 &\rightarrow [ы, ы]\, A_3 \mid \varepsilon \end{aligned} \tag{6.6.}$$

Itt q_0 a kezdeti állapot, x pedig a Σ alfabeta tetszőleges eleme.

Mint látható a kiindulási jelsorozat előtt és után bizonyos számú ы szimbólumot helyeztünk el. Az itt közölt nyelvtan ugyanis megköveteli, hogy a *Turing*-gép által igénybe vett szalag teljes hosszában szerepeljen a kiindulási mondatszerű formában. Így célszerű nagyvonalúan bánni a [ы, ы] szimbólumok elhelyezésével. Amennyiben a *Turing*-gép lelépne a mondatszerű formáról, a levezetés nem volna folytatható.

Persze lehetne olyan, kissé bonyolultabb nyelvtant szerkeszteni, ahol nincs szükség előzetesen valamennyi, pontosabban hasra ütéssel becsült számú szimbólum generálására.

A fenti szabályok valósítják meg a folyamat első részét, készen áll a *Turing*-gép induló konfigurációja.

A *Turing*-gép mozgási szabályainak helyettesítési szabályokká való átírását, tekintettel a szimbólumpárokból alkotott nemterminálisokra, megismételjük:

$$\delta(q, a) = (p, b, r) \qquad q\,[x, a] \rightarrow [x, b]\, p \tag{6.7.}$$

$$\delta(q, a) = (p, b, l) \qquad [y, Y]\, q\,[x, a] \rightarrow p[y, Y]\,[x, b] \tag{6.8.}$$

Itt x és y a Σ alfabeta tetszőleges eleme, míg Y tetszőleges szalagbeli szimbólum.

Az új szabályok, ha a szimbólumpárokban csak a jobboldali elemet nézzük, megegyeznek a korábban felírt szabályokkal.

Ezeket a szabályokat alkalmazzuk a folyamat második részében, amikor a *Turing*-gép működését szimuláljuk.

Az eredeti bemenet rekonstruálására a következő szabályok szolgálnak.

$$
\begin{aligned}
&q_E \rightarrow A_4\, A_4\\
&A_4\, [x,\, Y] \rightarrow xA_4\\
&[x,\, Y]\, A_4 \rightarrow A_4 x\\
&A_4\, [\text{ы},\, Y] \rightarrow A_4\\
&[\text{ы},\, Y]\, A_4 \rightarrow A_4\\
&A_4 \rightarrow \varepsilon
\end{aligned}
\qquad (6.9.)
$$

Itt q_E a *Turing*-gép elfogadó állapota, x a Σ alfabeta egy tetszőleges - eleme, míg Y a szalag tetszőleges szimbóluma. Minden további magyarázat nélkül gondolom világos, hogy ezekkel a szabályokkal a nyelvtan kihámozza az eredeti bemenetet, és a felesleges sallangokat lenyesegeti. Így végül valóban az eredeti jelsorozatot állítja elő mondatként. Ezzel a *Turing*-gépek és a **0**-ás osztályú nyelvek ekvivalenciáját igazoltuk.

6.5. Lineárisan korlátos automata

Rekurzíve felsorolható és rekurzív halmazok

A négy nyelvosztály közül három esetben már megtaláltuk a hozzá tartozó automataosztályt. A még hiányzó **1**-es osztályú, vagyis környezetfüggő nyelvekre ez a lineárisan korlátos automata.

A lineárisan korlátos automata nagyon hasonlít a *Turing*-gépre. Van egy véges állapothalmaza, van egy író-olvasófejjel ellátott szalagja, a különbség csupán annyi, hogy a szalag korlátos, pontosabban olyan terjedelmű, amilyen a bemeneti jelsorozat.

A lineárisan korlátos automata tehát nem kalandozhat el a szalag szűz területére – amely tulajdonképpen nem is létezik – illetve ha ezt mégis megkísérelné, akkor az automata megáll, és a bemenetet nem fogadja el.

Ez a korlátozás nagyon természetesnek tűnik, ha meggondoljuk, hogy az **1**-es nyelvosztályt éppen a nem csökkentés jellemzi.

Mindenekelőtt igazoljuk a nem csökkentő nyelvtanok és a lineárisan korlátos automaták ekvivalenciáját. Egy adott nem csökkentő nyelvtanhoz a következőképpen szerkeszthetünk egy lineárisan korlátos automatát.

Alkalmazzunk az automatán olyan széles szalagot, amelyen egymás alatt két szimbólum is elfér. Induláskor a szalag felső felén helyezkedik el a bemeneti jelsorozat, a szalag alsó fele üres. Az automata ezután a szalag alsó felének első helyére beírja a mondatszimbólumot, majd a már ismert módon a mondatszerű formákban levezetési szabályokat keresve, azok egyikét behelyettesíti. Így nemdeterminisztikus módon játssza le a nyelvtan levezetéseit.

Mindezeket a műveleteket csak a szalag alsó felén végzi el, a szalag felső felének tartalma változatlan marad.

A *Turing*-gép hasonló rendeltetésű működtetéséhez képest itt annyi a különbség, hogy törlésre soha sincs szükség csak beszúrásra, ugyanakkor ha a mondatszerű forma kiszaladna a rendelkezésre álló szalag területről, akkor az elemzés negatív eredménnyel leáll.

Amennyiben a levezetés csupa terminálisból álló jelsorozatot eredményez, akkor azt összevetjük az eredeti bemenettel, és ha a két jelsorozat azonos, akkor elfogadó állapotban megállunk. Egyébként nem elfogadó állapotban maradunk.

Az ily módon kialakított lineárisan korlátos automata a nyelvtan által generált nyelvet fogadja el.

Az eddigiek alapján nem jelent problémát a másik irányú megfeleltetés sem. A *Turing*-gépnél alkalmazott megoldással itt is célt érünk. A lineárisan korlátos automatával egyenértékű nyelvtan megszerkesztésénél csupán néhány dologra kell ügyelnünk.

Először is itt nincs szükség a bemenet körülbástyázására *ы* szimbólumokkal, hiszen az automata nem léphet szűz területre.

Bonyolultabbá teszi a megoldást az a körülmény, hogy a nyelvtan nem tartalmazhat ε-szabályokat, hiszen nem csökkentő nyelvtanról van szó. Éppen ezért az állapot jelzésére használt szimbólumok nem szerepelhetnek külön nemterminális szimbólumként a nyelvtanban, mert nincs mód egy különálló szimbólum megsemmisítésére.

A megoldás például az lehet, hogy az állapotot jelző szimbólumot az író-olvasófej alatti szimbólummal egyetlen nemterminálissá olvasztjuk össze. Ez viszont rontja az áttekinthetőséget, ezért is nem éltünk ezzel a lehetőséggel a *Turing*-gép tárgyalásánál.

Hasonló okok miatt még néhány kisebb változtatásra van szükség a bemenet rekonstruálására szolgáló szabályoknál is. Ez után az előzetes magyarázkodás után minden további indoklást mellőzve adjuk meg a nyelvtani szabályokat.

$$
\begin{array}{lll}
 & S \to [x,\ q_0\,x] \mid [x,\ q_0\,x]\,A_1 & \\
 & A_1 \to [x,\ x] \mid [x,\ x]\,A_1 & \forall\, x \in \Sigma \\
\delta(q,\ a) = (p,\ b,\ r) & \Rightarrow \quad [x,\ qa]\,[y,\ Y] \to [x,\ b]\,[y,\ pY] & \\
\delta(q,\ a) = (p,\ b,\ l) & \Rightarrow \quad [y,\ Y]\,[x,\ qa] \to [y,\ pY]\,[x,\ b] & \\
 & [x,\ q_E X] \to [x,\ A_2] & \\
 & [y,\ Y]\,[x,\ A_2] \to [y,\ A_2]\,x & \qquad (6.10.) \\
 & [x,\ A_2]\,[y,\ Y] \to x\,[y,\ A_2] & \\
 & [x,\ A_2] \to x &
\end{array}
$$

Ezzel igazoltuk a lineárisan korlátos automaták és a nem csökkentő, vagyis az **1**-es nyelvosztályba tartozó nyelvtanok ekvivalenciáját.

Azokat a gondolatkísérleteket, amelyeket a *Turing*-géppel annak idején elvégeztünk, megtehetjük a lineárisan korlátos automatával is.

Az automata leírását és a bemeneti jelsorozatot itt is előállíthatjuk ugyanabban a kódban. Persze a lineárisan korlátos automaták száma is megszámlálhatóan végtelen, mint volt a *Turing*-gépeké. A lineárisan korlátos automaták halmaza természetesen a *Turing*-gépek halmazának valódi részhalmaza.

Mivel a lineárisan korlátos automaták megszámolhatóak, itt is adhatunk minden ilyen automatának egy sorszámot.

Most is megvizsgálhatjuk, hogy az automata elfogadja-e saját leírását. Nyilván lesznek, amelyek elfogadják, és lesznek olyanok, amelyek nem.

Így a lineárisan korlátos automaták leírásait itt is két halmazba sorolhatjuk aszerint, hogy elfogadják, vagy visszautasítják saját specifikációjukat. Az így kialakított két nyelvet jelölje $\boldsymbol{L}_{11}$ és $\boldsymbol{L}_{12}$. Az első index a nyelvosztályra utal.

Emlékezzünk arra a gondolatmenetre, amelynek segítségével beláttuk, hogy a saját specifikációjukat visszautasító *Turing*-gépek leírásából alkotott nyelv nem lehet egy *Turing*-gép nyelve.

Ezt az érvelést szóról szóra megismételve juthatunk arra a következtetésre, hogy az $\boldsymbol{L}_{12}$ nyelv, a saját leírásukat visszautasító lineárisan korlátos automaták specifikációinak halmaza nem lehet egy lineárisan korlátos automata nyelve.

Turing-gép azonban készíthető erre a nyelvre. Emlékeztetek arra, hogy korábbi eredményünk szerint a tartalmazás problémája az **1**-es nyelvosztályra nézve algoritmizálható, vagyis véges sok lépésben megoldható feladat. Ezen túlmenően vegyük figyelembe, hogy a lineárisan korlátos automaták szituációinak száma, egy konkrét jelsorozat elemzésekor korlátos. Így ha egy ilyen automata „ügyetlen” konstrukciója következtében végtelen ciklusba kerülne, az detektálható.

Ebből következik, hogy minden lineárisan korlátos automatához szerkeszthető egy vele egyenértékű olyan automata, amely minden bemenetre megáll.

Amennyiben az univerzális *Turing*-géppel szimuláljuk a lineárisan korlátos automatákat, illetve azok minden bemenetre megálló egyenértékeseit, akkor az univerzális *Turing*-gép is minden esetben meg fog állni, természetesen abban az esetben is, ha bemenetére a saját leírásukat adjuk.

Ha most a megálláskor érvényes állapotot változatlanul hagyjuk, akkor gépünk éppen az $\boldsymbol{L}_{11}$ nyelvet fogadja el. Változtassuk meg azonban a megállási állapot értelmét, és akkor fogadjunk el, ha a szimulált automata visszautasít, és fordítva, akkor a gép nyelve az $\boldsymbol{L}_{12}$ nyelv lesz.

Persze ugyanúgy, mint a *Turing*-gép esetében, itt is előzetesen ki kell szűrnünk azokat a jelsorozatokat, amelyek nem lineárisan korlátos automata leírásai.

A halmazelméletben rekurzíve felsorolható halmaznak nevezik az olyan halmazokat, amelyek elemeit valamilyen algoritmus segítségével rendre fel lehet sorolni. A generatív nyelvek rekurzíve felsorolható halmazok.

Ezt azonnal beláthatjuk, hiszen a felsorolást szolgáltató algoritmust már kialakítottuk. Amikor ugyanis a **0**-ás osztályú nyelvtanhoz szerkesztettünk *Turing*-gépet, akkor előbb levezettük a mondatszimbólumból egy lépésben levezethető mondatszerű formákat, majd az ezekből egy lépésben, tehát a mondatszimbólumból két lépésben levezethetőeket, és így tovább.

A levezetés során adódnak olyan mondatszerű formák, amelyek csakis terminális szimbólumokat tartalmaznak. Ezek a mondatok. Ez adja a felsorolást. Minthogy egy generatív nyelv valamennyi mondata véges számú lépésben levezethető, ezzel a módszerrel a nyelv összes mondatát, a halmaz minden elemét felsoroljuk.

Természetesen csak legfeljebb megszámlálhatóan végtelen számosságú halmaz lehet rekurzíve felsorolható.

Rekurzív az olyan halmaz, ahol nem csak a halmaz, hanem komplemense is rekurzíve felsorolható. A definícióból következik, hogy ez esetben mindkét halmaz, az eredeti is meg a komplemense is rekurzív.

Rekurzív halmazoknál a tartalmazás feladata mindig algoritmizálható. Soroljuk fel ugyanis felváltva a halmaznak és a halmaz komplemensének elemeit. Ha most arra vagyunk kíváncsiak, hogy az univerzum egy eleme eleme-e a halmaznak, akkor meg kell várnunk, amíg ebben a felsorolásban ez az elem elő nem fordul. Ha mint a halmaz eleme került elő, akkor a vizsgált elem beletartozik a halmazba, ha viszont a komplemens elemeként, akkor a válasz nemleges.

Abban biztosak lehetünk, hogy a kérdésre választ kapunk. A halmaz elemeit ugyan itt is fel tudjuk sorolni, de akkor nem tudhatjuk, hogy azért hiányzik, mert nincs benne a halmazban, vagy azért, mert még nem jutottunk el a felsorolásban a keresett elemhez.

Az **1**-es nyelvosztály nyelvei rekurzív halmazok. Ugyanakkor azt is tudjuk, hogy a **0**-ás nyelvosztály nyelvének mondatai felsorolhatók, vagyis ezek a nyelvek legalábbis rekurzíve felsorolhatók.

Kérdés, vajon bővebb-e a rekurzív halmazok osztálya az **1**-es nyelvosztály nyelveinél? Van-e olyan – szükségképpen **0**-ás nyelvosztályba tartozó nyelv, amelyik rekurzív, és nem csupán rekurzíve felsorolható.

A válasz pozitív. Az imént definiált $\boldsymbol{L}_{12}$ nyelv, amelyről beláttuk, hogy nem fogadható el egy lineárisan korlátos automatával, rekurzív, hiszen komplemense is rekurzív.

Uggyanakkor az $\boldsymbol{L}_{01}$ nyelv, amely a saját leírásukat elfogadó *Turing*-gépek specifikációit tartalmazza, és amelyre *Turing*-gépet szerkesztettünk, tehát

0 osztályú nyelv, csak rekurzíve felsorolható. Amennyiben ugyanis a nyelv rekurzív lenne, akkor arra a bizonyos ominózus $\boldsymbol{L}_{02}$ nyelvre is lehetne *Turing*-gépet készíteni. A rekurzív halmazok határa így valahol a **0**-ás nyelvosztály tartományán belül halad, és természetesen teljes mértékben magába foglalja az **1**-es nyelvosztályt.

6.6. A számítástechnikai nyelvészet algoritmikusan eldönthetetlen feladatairól

Az első olyan probléma, amelyről beláttuk, hogy algoritmikusan eldönthetetlen a *Turing*-gép megállási problémája volt. Kiderült ugyanis, hogy amennyiben létezne erre a kérdésre algoritmus, akkor egy korábban már megoldhatatlannak bizonyult feladatra is volna megoldás, nevezetesen az $\boldsymbol{L}_{02}$ nyelvre szerkeszthetnénk *Turing*-gépet. Ebből az ellentmondásból következett, hogy a *Turing*-gép megállási problémája szükségszerűen algoritmikusan eldönthetetlen kérdés.

A számítástechnikában egy sereg nagyon kézenfekvő, és a gyakorlat szempontjából nagyon fontos feladatról derült ki, hogy algoritmikusan eldönthetetlen.

Az algoritmikus eldönthetetlenség tényét rendszerint indirekt bizonyítással igazoljuk. Kimutatjuk, ha létezne az adott feladatra algoritmus, akkor egy másik korábban más okokból algoritmikusan eldönthetetlennek bizonyult feladatra is volna algoritmus.

Az alábbiakban egy sor problémáról fogjuk belátni, hogy algoritmikusan eldönthetetlen. Kezdjük a *Post* problémával.

Post problémáját még a *Turing*-gép bevezetésekor, tehát 1936-ban fogalmazta meg, és mutatta ki, hogy az algoritmikusan eldönthetetlen.

Első látásra a *Post* probléma kissé hajánál fogva előráncigált problémának tűnik, de mint látni fogjuk, segítségével sok más, gyakorlati szempontból fontos és lényeges feladatról lehet kimutatni, hogy algoritmikusan eldönthetetlen.

Post problémája a következő:

Legyen adott valamely Σ alfabeta, és legyen

$$(x_i, y_i) \tag{6.11.}$$

ezen alfabeta szimbólumaiból alkotott jelsorozatpárok véges halmaza.

A kérdés a következő: összeállítható-e ezen jelsorozatpárok mondjuk baloldali elemeiből konkatenációval, vagyis egymás után írással olyan eredő jelsorozat, amelyben minden elemet jobboldali párjával kicserélve ugyanazt a jelsorozatot kapjuk.

Az eredő jelsorozat kialakításánál teljes szabadságunk van, egyes elemeket megismételhetünk, másokat meg kihagyhatunk a felsorolásból.

Amennyiben ez lehetséges, vagyis sikerül az x_i elemekből egy olyan sorozatot összehozni, hogy a megfelelő y_i elemek ugyanazt a sorozatot adják, akkor az adott *Post* problémának van megoldása, ha ez lehetetlen, akkor nincsen.

Ahogy az ilyenkor lenni szokott, egyes *Post* problémáknak van megoldása, másoknak meg nincsen.

Lássunk egy példát a *Post* problémára. Legyen négy jelsorozatpárunk:

x	y
1	**101**
1	**110**
101	**1**
110	**011**

Ennek a *Post* problémának van megoldása. Ha ugyanis az indexek sorrendjét a **34123** sorozatnak megfelelően választjuk, akkor akár a baloldalakból, akár a jobboldalakból állítjuk is össze az eredő jelsorozatot, ugyanazt kapjuk. Valóban

$$x_3x_4x_1x_2x_3 = y_3y_4y_1y_2y_3 = \mathbf{10111011101}$$

Az alábbi *Post* problémának viszont nincs megoldása. Ezt azonnal belátjuk, ha meggondoljuk, hogy már az első elemet sem tudjuk úgy kiválasztani, hogy a két sorozat azonosan kezdődjék:

x	y
10	**01**
110	**010**
00	**111**

Ebben a két esetben a megoldhatóság illetve megoldhatatlanság mintegy ránézésre eldönthető volt. Vajon létezik-e algoritmus, amely általános esetben, tehát tetszőleges *Post* probléma esetében képes választ adni erre a kérdésre.

Bizonyára nem hat a meglepetés erejével az az állítás, hogy ilyen algoritmus nem létezik, a *Post* probléma algoritmikusan eldönthetetlen feladat. Ezt a megállapításunkat az alábbiakban bebizonyítjuk.

Elöljáróban egy újabb feladatot definiálunk, amelyet súlyosbított *Post* problémának nevezhetünk. Ez egy olyan *Post* probléma, ahol még azt is kikötjük, hogy melyik párral kezdődjék az eredő jelsorozat.

Első pillanatra úgy tűnik, hogy ezzel egy új, az eredeti *Post* problémánál nehezebben megoldható feladatosztályt hoztunk létre. A valóságban nem ez a helyzet. Minden súlyosbított *Post* problémához szerkeszthető ugyanis egy egyszerű, tehát nem súlyosbított, de az eredetivel egyenértékű *Post* probléma.

Az egyenértékűség jelen esetben azt jelenti, hogy az egyszerű *Post* problémának akkor, és csakis akkor van megoldása, ha a súlyosbítottnak is van.

Vezessünk be ugyanis két új szimbólumot. Legyenek ezek # és &. Szúrjuk be az első szimbólumot a baloldali elemeknél minden szimbólum elé, a jobboldaliaknál minden szimbólum után.

Ezen kívül egészítsük ki a problémát két új párral. Az egyik a súlyosbítás tárgyát képező, vagyis a kezdésül kijelölt párból keletkezik az előbbi módon, de azzal az eltéréssel, hogy itt mind a baloldali, mind a jobboldali elem az új első szimbólummal kezdődik. A másik új pár a sorozat lezárására szolgál.

Tételezzük fel, hogy az első példánkban szereplő *Post* problémát azzal súlyosbítottuk, hogy a harmadik párt jelöltük ki kezdő párként. Az ehhez a súlyosbított *Post* problémához tartozó egyszerű *Post* probléma a következő lesz:

x	*y*
#1	**1#0#1#**
#1	**1#1#0#**
#1#0#1	**1#**
#1#1#0	**0#1#1#**
#1#0#1	**#1#**
#&	**&**

Az egyenértékűség nem igényel magyarázatot.

Egyetlen olyan pár van, amelyik ugyanazzal a szimbólummal kezdődik, és ez éppen a kezdésre kijelölt pár. Így – tetszik, nem tetszik – csakis ezzel az elempárral kezdhetünk.

A továbbiakban a keresést nem befolyásolja, hogy az eredeti, súlyosbított probléma jelsorozataihoz képest itt a szimbólumok között mindig van egy # szimbólum.

Amikor az eredeti probléma megoldásához érünk, akkor itt a két jelsorozat között egy # szimbólumnyi különbség van. Ennek pótlására hivatott az utolsó elempár.

A fenti eredmények birtokában az általánosság csorbítása nélkül használhatunk a jövőben súlyosbított *Post* problémát.

A *Post* probléma algoritmikus eldönthetetlenségét a következő gondolatmenet alapján bizonyítjuk.

Legyen adott egy tetszőleges *Turing*-gép, és annak valamilyen tetszőleges bemenete. Ehhez a tetszőleges *Turing*-géphez, és tetszőleges bemenethez szerkesszünk egy olyan *Post* problémát, amely a *Turing*-gép mozgása során felvett konfigurációk leírásának sorozatát tartalmazza.

Ennek az adott *Turing*-géphez és adott bemenethez illesztett *Post* problémának akkor és csak akkor van megoldása, ha a *Turing*-gép megáll.

Amennyiben a *Post* problémáról el tudnánk dönteni, hogy van-e megoldása vagy nincsen, akkor ezzel együtt azt is eldöntjük, hogy a *Turing*-gép az adott bemenetre megáll-e vagy sem.

Minthogy az ismertetendő eljárással bármely *Turing*-gépre és bármely bemenetre szerkeszthetünk egy illeszkedő *Post* problémát, ha létezne algoritmus a *Post* problémára, akkor a *Turing*-gép megállási problémájára készíthetnénk algoritmust.

A tetszőleges *Turing*-géppel szemben támasszuk az alábbi speciális követelményeket.

A csak két olyan állapota legyen, amelyben megáll, egy elfogadó q_E és egy visszautasító q_N. Ezekben az állapotokban viszont már ne végezzen mozgást.

Ha valamelyik *Turing*-gép eredetileg nem teljesítené a fenti feltételeket, mint ismeretes, könnyen szerkeszthetünk egy vele egyenértékű olyan automatát, amely már tudja ezeket a követelményeket. Így, bár nem illendő egy tetszőleges automatával szemben bármilyen követelményt támasztani, ez a feltétel az automata általános voltát nem csorbítja.

A *Post* probléma jelsorozatpárjai három csoportra oszthatóak.

Az első csoportba azok az elempárok tartoznak, amelyek biztosítják a konfigurációk sorozatos leírását. Ezen elempárok alakja független a *Turing*-gép mozgási szabályaitól.

Az elempárok másik része éppen a *Turing*-gép mozgási szabályait tükrözi.

A harmadik csoportba tartozó elempárok alkalmazására akkor kerül sor, amikor a mozgásban követett *Turing*-gép megállt. Feladatuk, hogy ebben az esetben biztosítsák a *Post* probléma megoldását. Funkciójuk hasonló ahhoz, amikor egy sakkozó már nyert állást ért el, de hátra van még a lebonyolítás, a szerzett előny érvényesítése.

A *Post* probléma alfabetája bővebb, mint a *Turing*-gép szalagjáé. Ezen az alfabetán kívül tartalmazza az állapotok leírására szolgáló szimbólumokat, valamint még egy elválasztójelet, legyen ez **#**, végül egy, a megoldás kialakításában fontos szerepet játszó szimbólumot. Legyen ez **&**.

Az első csoportba három elempár típus tartozik:

x	y	
#	$\#q_0w\#$	
X	X	$\forall X \in \Gamma$
#	**#**	

Mint említettük, egy súlyosbított *Post* problémát szerkesztünk. Az első elempár éppen a súlyosbító körülményt jelenti, ez a kezdésre kijelölt elempár. Az elempár jobboldali eleme egy kezdő konfiguráció, míg baloldala gyakorlatilag üres.

A további két elempárcsoport lehetővé teszi a *Turing*-gép szalagján található szimbólumok meg az elválasztó jel másolását. Vegyük észre, hogy a *Turing*-gép állapotainak jelzésére használt szimbólumok ezen szabályok segítségével nem másolhatóak.

Ez utóbbi szimbólumokat a *Turing*-gép mozgási szabályait tükröző második csoportbeli elempárok kezelik. Ezek az elempárok „illesztik” a *Post* problémát az adott *Turing*-géphez.

A kétféle, jobbra illetve balra lépő szabálytípusnak két-két elempártípus felel meg. Legyenek az ismert mozgási szabályok:

$$\delta(q, a) = (p, b, r)$$
$$\delta(q, a) = (p, b, l)$$

Ezeknek a mozgási szabályoknak a következő elempárok felelnek meg:

x	*y*
qaX	***bpX***
qa#	***bpы#***
Xqa	***pXb***
#qa	***#pыb***

ahol - ***X*** tetszőleges szalagbeli szimbólum.

Természetesen itt is elempárok halmazáról van szó, hiszen minden mozgási szabályhoz tartozik egy elempárkettős. A második és negyedeik szabálytípus arra szolgál, hogy a *Turing*-gép író-olvasófeje ne fusson le a szalagról.

Az adott kezdés mellett csak akkor reménykedhetünk abban, hogy a *Post* problémát megoldjuk, ha ezt a kezdő konfigurációt – mintegy futva a pénzünk után – beírjuk a baloldali elemek által generált jelsorozatba is.

A szalag szimbólumait és az elválasztó jelet egyszerűen átmásolhatjuk. Az állapotnak megfelelő szimbólumot viszont csakis a második csoport elempárjaival tudjuk átírni. Azt hiszem további magyarázat nélkül is érzékelhető, hogy amikor nagy gonddal beírjuk a baloldalra a jobboldalon szereplő konfigurációt, akkor a jobboldalon a *Turing*-gép új, következő konfigurációja generálódik.

A *Post* probléma megoldását keresve persze ezt a konfigurációt is le kell másolnunk, és így kerül a *Post* probléma jelsorozatába mindig újabb és újabb konfiguráció.

Ez a folyamat, ha a *Turing*-gép nem áll meg, vég nélkül folytatódik. Más a helyzet, ha a *Turing*-gép akár a $\boldsymbol{q}_{\mathrm{E}}$ elfogadó, akár a $\boldsymbol{q}_{\mathrm{N}}$ visszautasító állapotban megáll.

Ekkor kerül sor a harmadik elempár csoport felhasználására.

x	*y*
$\boldsymbol{q}_{\mathrm{E}}$	**&**
$\boldsymbol{q}_{\mathrm{N}}$	**&**
&X	**&**
X&	**&**
&##	**#**

Első lépésben az állapotot jelző szimbólum helyébe egy új, a szimbólumok törlésére hivatott **&** szimbólum kerül. A megállás pillanatában ugyanis a jobboldali jelsorozat egy teljes konfigurációval „vezet” a baloldali előtt.

Nos a törlő szimbólum minden egyes másolásnál egy szimbólummal lerövidíti az eredeti konfiguráció leírását. A végén az egész konfigurációból már csak a törlő szimbólum marad. Ekkor alkalmazhatjuk az utolsó elempárt, amely behozza a baloldal lemaradását.

Így végül is sikerült a *Post* problémára egy megoldást összehoznunk, persze akkor és csakis akkor, ha a *Turing*-gép megáll. Ezzel egy olyan *Post* problémát konstruáltunk, amelynek megoldási feladata nyilván algoritmikusan eldönthetetlen, hiszen ellenkező esetben a *Turing*-gép megállási problémáját is el tudnánk dönteni.

Ebből következik, hogy a *Post* probléma megoldási feladata általánosságban algoritmikusan eldönthetetlen. Egyes konkrét *Post* problémák esetében persze – mint láttuk – adhatunk választ, éppúgy, mint egyes *Turing*-gépek megállását is el tudtuk dönteni.

Mint említettem, maga a *Post* probléma számítástechnikai szempontból nem túl izgalmas kérdés. Segítségével azonban egy sor nagyon is húsbavágó feladat eldönthetetlenségét tudjuk tisztázni.

Lássunk rögtön egy alapvető problémát.

Az a kérdés, hogy egy környezetfüggetlen nyelv egyértelmű-e algoritmikusan eldönthetetlen.

Legyen adott egy tetszőleges *Post* probléma:

$$(x_i, y_i)$$

Legyen **CF** nyelvtanunk a következő:

$$S \to X \mid Y$$
$$X \to x_i X \mathbf{i} \mid x_i \mathbf{i}$$
$$Y \to y_j Y \mathbf{j} \mid y_j \mathbf{j}$$

Itt **i** és **j** a *Post* probléma indexhalmazának elemei, míg x_i és y_j a *Post* probléma megfelelő jelsorozatai. Itt persze megköveteljük, hogy az indexhalmaz és a *Post* probléma alfabetája diszjunkt legyen.

Könnyen belátható, hogy a nyelvtan a *Post* probléma baloldali illetve jobboldali jelsorozatait generálja, hozzáfűzve a választott elemek indexeit fordított sorrendben.

Nyilvánvaló, hogy a nyelvnek csak akkor lehet olyan mondata, amelynek két lényegesen különböző levezetése, vagyis különböző levezetési fája van, ha a *Post* probléma megoldható. Ekkor ugyanis a baloldali és jobb oldali elemeken alapuló levezetés ugyanazt a jelsorozatot, mondatot eredményezi.

Annak érdekében, hogy az adott nyelvtan egyértelmű voltát megállapíthassuk, az alapul szolgáló *Post* probléma megoldhatóságát kell megmondanunk. Ez viszont algoritmikusan elédönthetetlen.

Hasonlóan nem dönthető el, hogy egy **CF** nyelvtan generálja-e a Σ^* teljes univerzumot.

A feladatot ismét a *Post* problémára vezetjük vissza.

Definiáljunk a *Post* probléma alapján két nyelvet az alábbi nyelvtanokkal:

$$\boldsymbol{L}_X \qquad S \rightarrow x_i S\,\mathbf{i} \mid x_i\,\mathbf{i}$$
$$\boldsymbol{L}_Y \qquad S \rightarrow \mathbf{j} S y_j^{-1} \mid \mathbf{j} y_j^{-1}$$

A **-1** hatványkitevő – mint korábban – az inverzet, vagyis a fordított sorrendű jelsorozatot jelöli. A konstrukció hasonlít az előbbihez, itt is az elemek és indexeik szerepelnek. A különbség csupán annyi, hogy míg a baloldali elemeknél előbb jönnek az elemek és utána az indexek, addig a jobboldali elemeknél a sorrend minden tekintetben fordított.

Definiáljunk további két nyelvet:

$$\boldsymbol{L}_P = \boldsymbol{L}_X c \boldsymbol{L}_Y$$
$$\boldsymbol{L}_Q = wcw^{-1}$$

Az $\boldsymbol{L}_P$ nyelv mondatai így a *Post* probléma baloldali elemeiből és indexeiből, illetve a jobboldali elemek indexeiből és az elemek inverzeiből alkotott jelsorozatok, ahol a két komponenst egy *c* szimbólum választ el.

Itt nem követelmény, hogy a baloldali, illetve jobboldali elemek ugyanahhoz az indexhez tartozzanak, sőt az is lehetséges, hogy baloldalon és jobboldalon az elemek száma sem azonos.

Az $\boldsymbol{L}_Q$ nyelvben *w* tetszőleges jelsorozat, amelynek alfabetája a *Post* probléma alfabetájának és az indexek jelkészletének uniója. A jelsorozatot és inverzét itt is a *c* szimbólum választja el. Itt biztosak lehetünk abban, hogy a *c* szimbólumtól balra és jobbra eső sorozatok egymásnak tükörképei.

További magyarázat nélkül belátható, hogy mind az $\boldsymbol{L}_P$, mind az $\boldsymbol{L}_Q$ nyelv determinisztikus környezetfüggetlen nyelv. Ebből következően negáltjaik is determinisztikus környezetfüggetlen nyelvek lesznek. Sőt emiatt negáltjaik uniója is környezetfüggetlen nyelv lesz:

$$\boldsymbol{L}_R = \overline{\boldsymbol{L}_P} \cup \overline{\boldsymbol{L}_Q}$$

A negáltak uniója azonban a *de Morgan* összefüggések alapján nem más, mint a metszet negáltja.

Gondoljuk csak át, mit tartalmaz a fenti két nyelv, $\boldsymbol{L}_P$ és $\boldsymbol{L}_Q$ metszete.

Az $\boldsymbol{L}_P$ nyelv definíciója szerint a baloldal a *Post* probléma valamelyik baloldali jelsorozatából, és a hozzátartozó indexekből áll, míg a jobboldal ugyanannak a *Post* problémának szintén valamelyik, esetleg ugyanahhoz az indexsorozathoz tartozó jobboldali jelsorozat és a neki megfelelő indexsorozat, azzal az eltéréssel, hogy itt a sorrend fordított.

Az $\boldsymbol{L}_Q$ nyelv definíciója szerint viszont a mondat baloldal és jobboldala egymás tükörképe.

Amennyiben egy mondat mindkét nyelvben előfordul, vagyis eleme a két nyelv metszetének, akkor az előbbiek szerint ez csakis a *Post* probléma megoldása lehet.

Amennyiben a *Post* problémának nincs megoldása, akkor a két nyelv metszete üres, így negáltja a teljes univerzum. Így az

$$\boldsymbol{L}_R = \overline{\boldsymbol{L}}_P \cup \overline{\boldsymbol{L}}_Q = \overline{\boldsymbol{L}_P \cap \boldsymbol{L}_Q}$$

nyelvtana attól függően generálja vagy nem generálja a teljes univerzumot, hogy a *Post* problémának van-e megoldása vagy nincsen. Ez utóbbi azonban nem dönthető el.

Egy másik lényeges, de sajnos algoritmikusan eldönthetetlen feladat: Adott két környezetfüggetlen nyelvtan, kérdés ugyanazt a nyelvet generálják-e.

Ennek a feladatnak egy szigorúbb változata is igaz, nevezetesen ha az egyik nyelvtan reguláris, akkor sem lehet erre a kérdésre választ adni.

Az igazolást tulajdonképpen már el is végeztük, hiszen mi sem egyszerűbb, mint a teljes univerzumot generáló reguláris nyelvtant szerkeszteni.

Íme:

$$S \rightarrow xS \mid \varepsilon \qquad x \in \Sigma$$

ahol - az x alfabeta tetszőleges szimbóluma.

Miután erről a nyelvtanról tudjuk, hogy a teljes univerzumot generálja, összevetve ezt azzal az előbbi nyelvtannal, amely vagy generálja, vagy sem az univerzumot, állításunkat igazoltuk.

Szószedet

acceptance by empty stack	üres veremmel elfogadó
acceptance by final state	állapottal elfogadó
accepting state	elfogadó állapot
algorithm	algoritmus
alphabet	alfabeta, ábécé
ambiguous grammar	többértelmű nyelvtan
automaton	automata
Baar–Hillel lemma	→ pumping lemma
Backus–Naur form	Backus–Naur jelölés
Backus–Naur notation	Backus–Naur jelölés
bottom-up	alulról felfelé
BNF	→ Backus–Naur form
Bounded Right Context	BRC
characterizing	jellemző
CF	→ context free
Chomsky language classes	Chomsky-féle nyelvosztályok
Chomsky normal form	Chomsky normálalak, Chomsky normálforma
Church thesis	Church tézis, Church tétel
closure	zártság
CNF	→ Chomsky normal form
Coke–Younger–Kasami method	Coke–Younger–Kasami módszer
complement	komplemens, kiegészítő
computational linguistics	számítástechnikai nyelvészet
concatenation	konkatenáció, egymás után fűzés
configuration	konfiguráció
containment	tartalmazás
context free	környezetfüggetlen

context sensitive	környezetfüggő
CS	→ context sensitive
counter machine	számláló automata
deterministic finite automaton	determinisztikus véges automata
deterministic push-down automaton	determinisztikus veremautomata
DFA	→ deterministic finite automaton
distinguishable states	megkülönböztethető állapotok
DPDA	→ deterministic push-down automaton
Earley algorithm	Earley algoritmus
empty language	üres nyelv
empty stack	üres verem
empty string	üres jelsorozat
ε	→ empty string
FA	→ finite (state) automaton
final state	elfogadó állapot
finite (state) automaton	véges (állapotú) automata
finite (state) transducer	véges (állapotú) fordító
FSA	→ finite (state) automaton
FST	→ finite (state) transducer
FT	→ finite (state) transducer
grammar	nyelvtan
Greibach normal form	Greibach normálalak
GNF	→ Greibach normal form
halting of a Turing-machine	a Turing gép megállási problémája
homomorphism	homomorfizmus
initial state	kezdőállapot
intersection	metszet
Kleene closure	→ transitive closure
LBA	→ linear bounded automaton

leaf	levél
leftmost derivation	baloldali levezetés
linear bounded automaton	lineárisan korlátos automata
λ	→ empty string
minimal automaton	minimálautomata
NFA	→ nondeterministic finite (state) automaton
NFSA	→ nondeterministic finite (state) automaton
nondeterministic finite (state) automaton	nemdeterminisztikus véges automata
nondeterministic push-down automaton	nemdeterminisztikus veremautomata
nonterminal symbol	nemterminális szimbólum, nyelvtani szimbólum
NPDA	→ nondeterministic push-down automaton
palindrome	palindróma, palindrom
parser	szintaktikus elemző
PDA	→ push-down automaton
Polish notation	lengyel jelölés
Post problem	Post probléma
push-down automaton	veremautomata
procedure	procedúra, eljárás
proper grammar	jólfésült nyelvtan
pumping lemma	pumpálási lemma
recursive language	rekurzív nyelv
recursively enumerable language	rekurzíve felsorolható nyelv
regular expression	reguláris kifejezés
regular language	reguláris nyelv
rightmost derivation	jobboldali levezetés
sentence	mondat
sentential form	mondatszerű forma
sink state	csapdaállapot
string	jelsorozat, karakterfüzér

substate	alállapot
substitution	helyettesítés
terminal symbol	terminális szimbólum
TM	→ Turing-machine
Trap	csapdaállapot
Turing-machine	Turing-gép
universal Turing-machine	univerzális Turing-gép
UTM	→ universal Turing-machine
useless symbol	felesleges szimbólum
wcw^{-1}	→ palindrome
ww^{-1}	→ palindrome

Név- és tárgymutató

Irodalom

Bánkfalvy J., Bánkfalvy Zs., Bolgár :
A formális nyelvek szintaktikus elemzése
Közgazdasági és Jogi Kiadó 1978

Demetrovics - Denev - Pavlov
A számítástudomány matematikai alapjai
Tankönyvkiadó 1989

Révész György
Bevezetés a formális nyelvek elméletébe
Akadémiai Kiadó 1979

Csörnyei Zoltán
Bevezetés a fordítóprogramok elméletébe
Tankönyvkiadó 1992

E. Horovitz
Magasszintű programnyelvek
Műszaki Könyvkiadó 1987

Zohar Manna
Programozáselmélet
Műszaki Könyvkiadó 1981

Noam Chomsky
Three models for the Description of Language
Transactions on Information Theory 1956

John E. Hopcroft - Jeffrey D. Ullman
Formal Languages and their Relation to Automata
Addison-Wesley 1969

John E. Hopcroft – Rajeev Motwani - Jeffrey D. Ullman
Itroduction to the Theory, Languages, and Computation
Addison Wesley 2001X

Harry R. Lewis – Christos H. Papamdimitrou
Elements of the Therory of Computation
Prentice-Hall 1998

Alfred V. Aho - Jeffrey D. Ullman
The Theory of Parsing, Translation and Compiling
Prentice-Hall 1972

John E. Hopcroft - Jeffrey D. Ullman
Introduction to Automata Theory, Languages and Compilation
Addison-Wesley 1979

David Gries
Compiler Construction for Digital Computers
John Wiley & Sons

Arto Salomaa
Formal Languages
Academic Press 1973

Arto Salomaa
Computation and Automata
Cambridge University Press 1985

Antony J. Davie - Ronald Morrison
Recursive Descent Compiling
John Wiley & Sons 1981

Anton Nijholt
Computers and Languages
North Holland 1988

Jean-Paul Tremblay – Paul G. Sorensen
The Theory and Practice of Compiler Writing
McGraw-hill 1985

K. John Gough
Syntax Analysis and Software Tools
Addison-Wesley 1988

Robert W. Sebesta
Concept of Programming Languages
Benjamin Cummings 1989

David A. Watt
Programming Language Syntax and Semantics
Prentice-Hall 1991

William M. Waite - Gerhard Goos
Compiler Construction
Springer-Verlag 1984

M. Ben-Ari
Understanding Programming Languages
John Wiley & Sons 1996

Alexander Meduna
Automata and Languages
Springer Verlag 2000