

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/308966996>

# Limbaje formale si automate

Book · June 2007

CITATIONS

0

READS

2,772

1 author:



[Adrian Constantin Atanasiu](#)

University of Bucharest

69 PUBLICATIONS 357 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



P Systems [View project](#)



Reaction Systems [View project](#)

# **LIMBAJE FORMALE ȘI AUTOMATE**

Adrian Atanasiu

Editura INFODATA Cluj



# Prefață

Teoria limbajelor formale și a automatelor a devenit un domeniu deja clasic și poate fi studiată sub o formă bine structurată, așa cum sunt de exemplu algebra liniară sau geometria plană. Inițiată ca un aparat pentru modelarea limbajului natural, ea capătă rapid un statut separat odată cu lucrările lui Noam Chomsky (anii '50) și apoi Arto Salomaa (anii '60). De remarcat aportul școlii românești de limbaje formale, începând cu Academicianul Solomon Marcus și continuând cu nume bine cunoscute în domeniu (Gheorghe Păun, Dan Simovici, Alexandru Mateescu, Victor Mitrană, Lucian Ilie și alții).

Importanța limbajelor formale constă în principal în domeniile largi de aplicabilitate. Plecând de la prelucrarea limbajului natural (de la care a preluat terminologia) și continuând cu limbajele de programare (structura unui compilator este aproape în totalitate definită pe baza gramaticilor și a automatelor), modele economice, biologice (calcul bazate pe lanțuri *ADN* sau pe membrane celulare), lista domeniilor de aplicabilitate oferă permanent noi posibilități de investigare bazate pe structuri generative sau de recunoaștere. Analiza complexității unui model de calcul este imposibilă fără o referire la mașinile Turing – modelul teoretic standard al unui computer.

Acestea sunt – pe scurt – motivele existenței încă de la începutul anilor '70 a unui curs de Limbaje formale și automate la secțiunile de informatică. Cartea de față este un suport de curs pentru studenții Facultății de Matematică și Informatică din Universitatea București și preia în mare parte forma unui curs mai vechi de Bazele Informaticii, ținut de autor în perioada 1984 – 1990.

Structura lui urmărește în principal clasificarea făcută de Chomsky limbajelor formale, prezentând pentru fiecare din cele 4 clase de limbaje clasice (limbaje regulate, independente de context, dependente de context, limbaje recursiv enumerabile) atât mecanisme generative (gramatici) cât și de recunoaștere (automate) specifice fiecărei clase.

Astfel, după un prim capitol dedicat automatelor finite, cu rol de a obișnui pe studenți cu modalitatea de abordare a textelor formale, în Capitolul 2 sunt studiate limbajele regulate împreună cu principalele teoreme de reprezentare (lema de pompare și Myhill-Nerode).

Capitolul 3 introduce noțiunea de sistem de rescriere, pe baza căruia se definesc

metodele generative și de recunoaștere folosite la definirea unui limbaj. Clasificarea Chomsky stabilește ierarhia standard a limbajelor ca punct de referință pentru studiile ulterioare.

Capitolul 4 este dedicat principalelor operații pe limbaje. Se studiază proprietăți de închidere la operațiile de reuniune, concatenare, stelare (operațiile regulate pe limbaje), homomorfisme, substituții, intersecție, oglindire și amestecare.

Capitolul 5 introduce câteva mecanisme de recunoaștere noi, legate strâns de automatele finite: gsm-uri, translatorii finiți, automate Moore și Mealy. După ce se arată că automatele finite sunt mecanismele care recunosc limbajele generate de gramaticile regulate, ideea se extinde asupra lui  $\mathcal{L}_2$  – unde se arată că automatele stivă sunt mecanismele de recunoaștere duale gramaticilor independente de context.

Datorită importanței deosebite pe care o are această clasă de gramatici, ei îi este dedicat Capitolul 6 – cel mai extins capitol. Sunt introduse noțiuni noi: arbore de derivare, gramatici proprii, forme normale (Chomsky și Greibach), teoreme de caracterizare.

În Capitolul 7 este făcut un studiu asupra limbajelor dependente de context și a automatelor liniar mărginite. De remarcat că spre deosebire de alte lucrări care tratează această clasă de acceptori ca un caz particular de mașină Turing, am considerat că *LBA*-urile sunt destul de importante pentru a fi construite și analizate separat.

Ultimul capitol se referă la limbajele recursiv numărabile, acceptate de mașinile Turing sau – dual – generate de gramaticile de tip 0.

De menționat că prin acest curs nu se acoperă în totalitate studiul limbajelor formale și al automatelor. Astfel, nu se menționează nimic despre structuri bine dezvoltate cum sunt gramaticile matriciale, indexate, contextuale (cu o mențiune specială pentru gramaticile contextuale Marcus), sisteme de gramatici, sisteme Lindenmayer, *AFL*-uri, sisteme Post, translatorii, modele biologice (automate Watson-Creek, *P* - sisteme<sup>1</sup>) sau despre noțiuni cum sunt vectorii Parikh, operații de ștergere și adăugare, studiul cuvintelor infinite etc, despre care s-ar mai putea elabora alte câteva cursuri.

Prin această carte am intenționat o abordare a elementelor de bază referitoare la cele două domenii. Aceste noțiuni fundamentale sunt necesare atât pentru înțelegerea mecanismelor de lucru folosite în alte discipline care au fost sau vor fi studiate ulterior (arhitectura sistemelor de calcul, elemente de complexitate a calculului, construirea de compilatoare, prelucrare de limbaj natural) precum și o deschidere a interesului de a se alătura colectivelor de cercetare din acest domeniu, colective existente în cadrul Facultăților de Matematică și Informatică din principalele centre universitare românești.

Autorul

---

<sup>1</sup>Litera '*P*' vine de la numele lui Gheorghe Păun.

## Cuprins

Prefață	3
Cap. 1: Automate finite	7
1.1. Prezentare generală	7
1.2. Definiții și notații preliminare	9
1.3. Automate finite deterministe	11
1.4. Automate finite nedeterministe	15
1.5. Echivalența dintre $AFD$ și $AFN$	18
1.6. Automate finite cu $\epsilon$ -mişcări	23
1.7. Exerciții	28
Cap. 2: Teoreme de reprezentare pentru limbajele regulate	31
2.1. Lema de pompare	31
2.1.1. Aplicații ale lemei de pompare	33
2.2. Teorema Myhill - Nerode și aplicații	36
2.2.1. Teorema de caracterizare algebrică a limbajelor regulate	36
2.2.2. Aplicații ale teoremei Myhill - Nerode	38
2.3. Exerciții	44
Cap. 3: Gramatici și limbaje clasificabile Chomsky	45
3.1. Sisteme de rescriere	45
3.2. Gramatici, clasificarea Chomsky	51
3.3. Exerciții	55
Cap. 4: Proprietăți de închidere ale limbajelor Chomsky	57
4.1. Noțiuni introductive	57
4.2. Închiderea limbajelor Chomsky față de operațiile regulate	61
4.3. Închiderea față de homomorfisme și substituții	65
4.4. Închiderea față de alte operații	68
4.4.1. Închiderea la intersecție	68
4.4.2. Închiderea la operația de oglindire	69
4.4.3. Închiderea la operații de amestecare	70
4.5. Exerciții	72

Cap. 5: Metode de recunoaștere pentru limbajele Chomsky	73
5.1. Gramatici regulate și automate finite	73
5.2. Gsm-uri și translatorii finiți	76
5.2.1. Definiții generale	76
5.2.2. Automate finite cu ieșiri	78
5.2.3. Translatorii finiți	80
5.3. Automate stivă	83
5.4. Automate pentru $\mathcal{L}_1$ și $\mathcal{L}_0$	92
5.5. Exerciții	93
Cap. 6: Gramatici independente de context	95
6.1. Introducere	95
6.2. Arbori de derivare și derivări independente de context	96
6.3. Gramatici independente de context proprii	105
6.4. Forme normale	115
6.4.1. Forma normală Chomsky	115
6.4.2. Forma normală Greibach	116
6.5. Teorema $uvwxy$ și aplicații	123
6.5.1. Enunț și demonstrație. Variante	123
6.5.2. Aplicații ale teoremei $uvwxy$	128
6.6. Exerciții	132
Cap. 7. $\mathcal{L}_1$ și automate liniar mărginite	135
7.1. Gramatici "monotone" și gramatici "senzitive de context"	135
7.2. Automate liniar mărginite	140
7.3. Exerciții	146
Cap. 8: Limbaje de tip 0 și mașini Turing	147
8.1. Definiția mașinii Turing	148
8.2. Proprietăți ale mașinilor Turing	151
8.3. Limbaje recursive și recursiv numărabile	154
8.4. Mașina Turing folosită pentru calculul funcțiilor întregi	156
8.5. Tehnici de construcție a mașinilor Turing	157
8.6. Modificări ale mașinilor Turing	162
8.7. Exerciții	168
Bibliografie	169

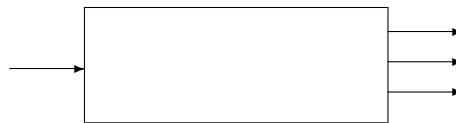
# Capitolul 1

## Automate finite

### 1.1 Prezentare generală

Structurile prezentate în acest prim capitol constituie cel mai simplu model de *automat* – denumire prin care în mod normal se înțelege o ”*cutie neagră*” cu o intrare și mai multe ieșiri (răspunsuri), ca în Figura 1.1.

Figura 1.1:



Complexitatea acestor răspunsuri este în strânsă legătură cu modul de organizare și funcționare a structurii interne a ”*cutiei*”, inaccesibilă (din diverse motive) unui utilizator ocazional.

Spre această direcție ne vom îndrepta atenția, prezentând – sub o formă matematică coerentă – o clasă de astfel de automate, cunoscută sub numele de *automate cu număr finit de stări* sau – mai scurt – *automate finite*.

Începem cu ele datorită simplității extreme sub care se prezintă: automatele finite nu au memorie și sunt formate esențial doar din două componente:

1. Un dispozitiv secvențial de citire (de pe o bandă de intrare),
2. Un registru de control de capacitate finită.



Este remarcabilă însă gama de aplicații – teoretice și practice – în care pot fi folosite aceste automate; de aceea studiul lor capătă un interes deosebit.

Un automat finit ( $AF$ ) poate fi privit ca o modelare matematică a unui sistem<sup>1</sup> cu intrări discrete.

Automatul conține un număr finit de configurații interne (*stări*); caracteristica unei stări este aceea de a păstra informații referitoare la intrările anterioare, necesare pentru determinarea comportării sistemului în funcție de diversele intrări ulterioare.

Există multe exemple în practică, care pot fi simulate cu automate finite, dificultatea constând adesea doar într-un mod de reprezentare cât mai adecvat.

**Exemplul 1.1** *Circuitele logic combinaționale întâlnite în Arhitectura Calculatoarelor – reprezentabile teoretic prin expresii booleene – sunt exemple de  $AF$ -uri. Un astfel de circuit are un număr finit de intrări (variabile logice), fiecare având doar două poziții posibile: închis (notat cu 1) și deschis (notat cu 0).*

*Starea unui circuit cu  $n$  intrări este deci una din cele  $2^n$  secvențe posibile de  $n$  caractere 0 sau 1.*

*Tot automate finite sunt și circuitele de tip 1 sau 2 (memorii, flip-flopuri etc).*

**Exemplul 1.2** *Unele programe utilizate de sistemul de operare (cum ar fi editorul de legături sau analizorul lexical) sunt date sub forma unor automate finite.*

*De exemplu, un analizor lexical parcurge simbolurile unui program de calculator și recunoaște secvențe de caractere corespunzând identificatorilor, constantelor numerice, cuvintelor cheie etc.*

**Exemplul 1.3** *O problemă distractivă foarte cunoscută este aceea a trecerii peste o apă de către un om a unui lup, o capră și o varză.*

*Pentru aceasta el are la dispoziție o barcă în care poate încapa – înafară de om – doar unul din ceilalți trei candidați la traversare.*

*Cum trebuie să acționeze omul, știind că nu are voie să lase singure pe un mal lupul cu capra sau capra cu varza ?*

*Problema poate fi modelată ușor cu un automat finit. Vom nota cu  $O$  - om,  $L$  - lup,  $C$  - capră și  $V$  - varză. Apar astfel  $2^4 = 16$  situații posibile, pe care le vom reprezenta sub forma unor perechi separate printr-o linie, prima parte reprezentând ceea ce rămâne pe un mal, iar a doua – ce a trecut pe malul celălalt.*

*O parte din stări (de exemplu  $CV - OL$ ) sunt excluse de condițiile problemei și nu le vom mai figura.*

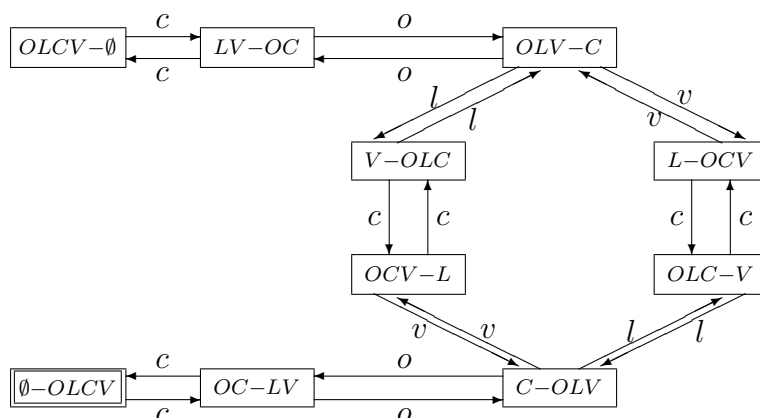
---

<sup>1</sup>Noțiunea de sistem este similară celei definite în Arhitectura Sistemelor de Calcul (Ed. Infodata 2006).

Intrările în automat sunt acțiunile pe care le face omul cu barca; acesta poate trece singur ( $o$ ), sau împreună cu lupul ( $l$ ), capra ( $c$ ) sau varza ( $z$ ). Starea inițială este  $OLCV - \emptyset$ , iar cea finală  $\emptyset - OLCV$ .

O diagramă a problemei este prezentată în Figura 1.2.

Figura 1.2:



Se observă că există două cele mai scurte soluții ale problemei, care pot fi privite ca drumuri de la starea inițială la cea finală. Există de fapt o infinitate de soluții, dar numai acestea două – marcate de arcele "colcvoc" și respectiv "covcloc" – nu conțin cicluri.

## 1.2 Definiții și notații preliminare

Vom considera cunoscute elementele uzuale de algebră și teoria mulțimilor.

Vom nota totdeauna mulțimile cu litere latine mari –  $A, B, C, \dots$ , eventual indexate.

Fie  $A$  o mulțime nevidă (de obicei finită), numită *alfabet* (sau *vocabular*).

Elementele lui  $A$  se numesc *caractere* și vor fi notate cu litere mici de la începutul alfabetului latin, eventual indexate ( $a, b, c_5, \dots$ ).

Dacă  $A$  și  $B$  sunt două mulțimi, definim mulțimea formală

$$AB = \{ab \mid a \in A, b \in B\}$$

Au loc relațiile (evidente)

$$AB \neq BA, \quad A(B \cup C) = AB \cup AC, \quad A(B \cap C) = AC \cap AB$$

Această operație  $(A, B) \longrightarrow AB$  se numește *concatenare* (*compunere*, *alăturare*).

În particular, vom putea construi mulțimile

$$A^2 = AA, \quad A^3 = A^2A, \quad \dots, \quad A^k = A^{k-1}A, \quad \dots$$

Prin convenție

$$A^0 = \{\epsilon\}$$

unde  $\epsilon$  este secvența fără nici un caracter (evident,  $\epsilon a = a\epsilon = a$ ,  $\forall a \in A$ ).

Notăm

$$A^* = \bigcup_{k \geq 0} A^k, \quad A^+ = \bigcup_{k \geq 1} A^k.$$

$A^*$  este *închiderea reflexiv - tranzitivă* a mulțimii  $A$  (sau *închiderea Kleene*).

### Observația 1.1

1.  $A^+ = A^* \setminus \{\epsilon\}$ .
2.  $A^*$  este monoidul liber generat de  $A$ , cu operația de concatenare, având  $\epsilon$  ca element neutru.

Elementele lui  $A^*$  se numesc *cuvinte* ( $\epsilon$  fiind *cuvântul vid*), *secvențe* sau *șiruri* și se notează uzual cu litere grecești mici  $\alpha, \beta, \gamma$  sau litere mici de la sfârșitul alfabetului latin ( $u, v, x, y, z$ ).

Orice alt gen de notație va fi specificat în mod explicit.

Fiind dat un cuvânt  $w \in V^+$ ,  $x \in V^+$  este *subcuvânt* al lui  $w$  dacă  $\exists \alpha, \beta \in V^*$  astfel încât  $w = \alpha x \beta$ .

Dacă  $\alpha = \epsilon$  atunci  $x$  este *prefix* al lui  $w$ , iar dacă  $\beta = \epsilon$ , atunci  $x$  este *sufix* al lui  $w$ .

Pentru un cuvânt  $\alpha \in A^*$ , notăm  $|\alpha|$  numărul de caractere care formează  $\alpha$ ; în particular  $|\epsilon| = 0$ .

**Definiția 1.1** Fie  $A$  o mulțime nevidă. Orice submulțime a lui  $A^*$  se numește *limbaj peste  $A$* .

**Exemplul 1.4** Fie  $A = \{a\}$ . Atunci  $A^2 = \{aa\}$ ,  $A^3 = \{aaa\}$  ș.a.m.d.

Avem  $A^* = \{a^k \mid k \geq 0\}$  (unde  $a^0 = \epsilon$ ) și  $A^+ = \{a^k \mid k \geq 1\}$ .

Mulțimea  $B = \{a^p \mid p \text{ număr prim}\}$  este un limbaj peste alfabetul  $A$ .

În particular,  $A^*$  este el însuși un limbaj.

**Exemplul 1.5** Să considerăm  $A = \{0, 1\}$ . Atunci

$$A^2 = \{0, 1\}\{0, 1\} = \{00, 01, 10, 11\},$$

$$A^3 = \{00, 01, 10, 11\}\{0, 1\} = \{000, 001, 010, 011, 100, 101, 110, 111\}.$$

În  $A^*$  se vor afla toate secvențele binare de lungime arbitrară.

## 1.3 Automate finite deterministe

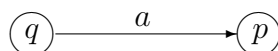
Formal, un automat finit determinist (*AFD*) poate fi definit astfel:

**Definiția 1.2** *Un AFD este o structură  $M = (Q, V, \delta, q_0, F)$  unde*

- $Q$  este o mulțime finită și nevidă de elemente numite "stări";
- $V$  este o mulțime finită și nevidă numită "alfabet de intrare";
- $\delta : Q \times V \longrightarrow Q$  este o aplicație parțial definită, numită "funcție de tranziție";
- $q_0 \in Q$  este "starea inițială" a automatului  $M$ ;
- $F \subseteq Q$  este o mulțime nevidă de elemente numite "stări finale".

Unui *AFD* i se asociază un graf orientat numit *diagramă de tranziție*; corespondența se realizează în felul următor:

- Fiecărei stări a automatului îi corespunde un nod în graf.
- Dacă  $\delta(q, a) = p$  (există o tranziție de la starea  $q$  la starea  $p$ , provocată de apariția caracterului  $a$ ), atunci în graf există un arc marcat cu  $a$  de la nodul  $q$  la nodul  $p$ ;



- Nodul asociat stării inițiale are forma (a), iar pentru o stare finală – forma (b);



- Graful nu conține alte noduri și alte arce înafara celor specificate anterior.

**Exemplul 1.6** *În Exemplul 1.3 este dată o diagramă de tranziție. Reprezentarea AFD-ului corespunzător este următoarea:*

$Q = \{OLCV - \emptyset (1), LV - OC (2), OLV - C (3), V - OLC (4), OCV - L (5), L - OCV, (6), OLC - V (7), C - OLV (8), OC - LV (9), \emptyset - OLCV (10)\}$  (în paranteze au fost date codificări ale stărilor, utile în simplificarea scrierii funcției de tranziție  $\delta$ ),  
 $V = \{o, l, c, v\}, \quad q_0 = 1, \quad F = \{10\}.$

$\delta$	$o$	$l$	$c$	$v$
1	—	—	2	—
2	3	—	1	—
3	2	4	—	6
4	—	3	5	—
5	—	—	4	8
6	—	—	7	3
7	—	8	6	—
8	9	7	—	5
9	8	—	10	—
10	—	—	9	—

Pentru a descrie comportarea unui *AFD* atunci când intrarea este un cuvânt, vom extinde funcția  $\delta$  la  $\tilde{\delta} : Q \times V^* \longrightarrow Q$ , definită recursiv astfel:

$$\tilde{\delta}(q, \epsilon) = q, \quad \tilde{\delta}(q, \alpha a) = \delta(\tilde{\delta}(q, \alpha), a).$$

Deoarece  $\tilde{\delta}(q, a) = \delta(\tilde{\delta}(q, \epsilon), a) = \delta(q, a)$ , putem nota extensia  $\tilde{\delta}$  tot cu  $\delta$  fără a face nici o confuzie.

$\delta(q, \alpha)$  reprezintă starea la care ajunge automatul când se pleacă din starea  $q$  și se parcurge la intrare cuvântul  $\alpha$ .

**Propoziția 1.1**  $\delta(q, \alpha\beta) = \delta(\delta(q, \alpha), \beta), \quad \forall q \in Q, \alpha, \beta \in V^*.$

*Demonstrație:* Prin inducție după lungimea cuvântului  $\beta$ .

*i.*  $|\beta| = 0$ : Deci  $\beta = \epsilon$ ; atunci membrul drept devine succesiv

$\delta(\delta(q, \alpha), \epsilon) = \delta(q', \epsilon) = q' = \delta(q, \alpha) = \delta(q, \alpha\epsilon)$ , unde s-a folosit local notația  $\delta(q, \alpha) = q'$ .

*ii.*  $k \longrightarrow k+1$ : Fie  $\beta \in V^*$  cu  $|\beta| = k+1$ ; deci  $\beta$  este de forma  $\beta = \beta'a$  cu  $a \in V$  și  $|\beta| = k$ .

Atunci

$$\begin{aligned} \delta(q, \alpha\beta) &= \delta(q, \alpha\beta'a) \stackrel{(1)}{=} \delta(\delta(q, \alpha\beta'), a) \stackrel{(2)}{=} \delta(\delta(\delta(q, \alpha), \beta'), a) \stackrel{(3)}{=} \delta(q', \beta'), a) = \delta(q', \beta'a) \stackrel{(3)}{=} \\ &= \delta(\delta(q, \alpha), \beta), \end{aligned}$$

unde s-a folosit:

(1) – definiția extensiei funcției de tranziție;

(2) – ipoteza de inducție;

(3) – notația  $\delta(q, \alpha) = q'$ .

q.e.d.

În unele lucrări se mai utilizează relația următoare (ușor de extins la clase mai largi de automate) definită pe  $Q \times V^*$ :

$$(q, a\alpha) \vdash (p, \alpha) \iff \delta(q, a) = p.$$

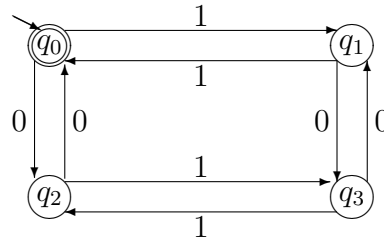
Se arată imediat că închiderea ei reflexiv - tranzitivă este

$$(q, \alpha\beta) \vdash^* (p, \beta) \iff \delta(q, \alpha) = p.$$

**Exemplul 1.7** Fie automatul finit determinist  $M = (Q, V, \delta, q_0, F)$  cu  $Q = \{q_0, q_1, q_2, q_3\}$ ,  $V = \{0, 1\}$ ,  $F = \{q_0\}$  și

$\delta$	0	1
$q_0$	$q_2$	$q_1$
$q_1$	$q_3$	$q_0$
$q_2$	$q_0$	$q_3$
$q_3$	$q_1$	$q_2$

Diagrama de tranziție este



Fie  $110101 \in V^*$  un cuvânt de intrare. Vom avea succesiv:

$$\begin{aligned} \delta(q_0, 110101) &= \delta(\delta(q_0, 1), 10101) = \delta(q_1, 10101) = \delta(\delta(q_1, 1), 0101) = \delta(q_0, 0101) = \\ &= \delta(\delta(q_0, 0), 101) = \delta(q_2, 101) = \delta(\delta(q_2, 1), 01) = \delta(q_3, 01) = \delta(\delta(q_3, 0), 1) = \delta(q_1, 1) = \\ &= q_0 \in F. \end{aligned}$$

Fie  $M = (Q, V, \delta, q_0, F)$  un AFD; definim limbajul acceptat de  $M$  ca fiind mulțimea

$$\tau(M) = \{\alpha \mid \alpha \in V^*, \delta(q_0, \alpha) \in F\}.$$

Revenind la Exemplul 1.7,  $110101 \in \tau(M)$ .

**Exemplul 1.8** Să construim un AFD pentru limbajul  $L$  definit astfel:

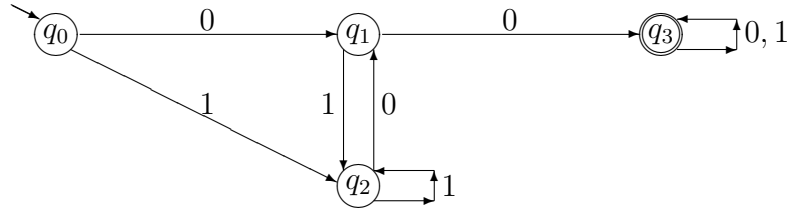
" $\alpha \in L \iff \alpha \in \{0, 1\}^*$  și  $\alpha$  conține cel puțin două zerouri consecutive."

De la început sunt precizate  $V = \{0, 1\}$  și – bineînțeles –  $q_0$ .

Se observă că avem nevoie de stări pentru a codifica următoarele informații:

- $q_0$  - starea inițială;
- $q_1$  - starea în care se ajunge cu un 0 (se are în vedere posibilitatea să urmeze încă un 0);
- $q_2$  - starea în care se ajunge cu 1;
- $q_3$  - starea (finală) în care se ajunge cu al doilea 0 consecutiv.

Diagrama de tranziție a automatului va arăta atunci:



$\delta$	0	1
$q_0$	$q_1$	$q_2$
$q_1$	$q_3$	$q_2$
$q_2$	$q_1$	$q_2$
$q_3$	$q_3$	$q_3$

De aici va rezulta funcția  $\delta$  :

Să arătăm că acest automat acceptă limbajul cerut de problemă.

Fie  $\alpha = \alpha'00\alpha''$ ; fără a micșora generalitatea, putem presupune că aceasta este prima pereche de zerouri consecutive (altfel spus, '00' nu este subcuvânt al lui  $\alpha'0$ ).

Apar două cazuri:

1.  $\boxed{\alpha' = \epsilon}$ : atunci  $\alpha = 00\alpha''$  și  
 $\delta(q_0, 00\alpha'') = \delta(\delta(q_0, 00), \alpha'') = \delta(q_3, \alpha'') = q_3 \in F$ , deci  $\alpha \in \tau(M)$ .
2.  $\boxed{\alpha' = \alpha_1 1}$  (altfel perechea de zerouri din paranteză n-ar mai fi prima): atunci  
 $\delta(q_0, \alpha_1 1 0 0 \alpha'') = \delta(\delta(q_0, \alpha_1 1), 0 0 \alpha'') = \delta(q_2, 0 0 \alpha'') = \delta(\delta(q_2, 00), \alpha'') = \delta(q_3, \alpha'') = q_3 \in F$ .

S-a folosit faptul că pentru orice  $\alpha_1 \notin \tau(M)$ ,  $\delta(q_0, \alpha_1) \in \{q_0, q_1, q_2\}$  (după cum  $\alpha_1$  este  $\epsilon$  sau se termină cu 0 sau 1).

În toate cazurile,  $\delta(q_0, \alpha_1 1) = \delta(\delta(q_0, \alpha_1), 1) = q_2$ .

Invers, fie  $\alpha \in \tau(M)$ ; deci  $\delta(q_0, \alpha) = q_3$ .

Rezultă atunci din construcție că există  $\alpha_1, \alpha_2 \in V^*$  cu

$$\alpha = \alpha_1 \alpha_2 \quad \text{și} \quad \delta(q_0, \alpha_1) = q_1, \quad \delta(q_1, \alpha_2) = q_3.$$

Fără a micșora generalitatea, putem presupune că  $\alpha_1$  și  $\alpha_2$  sunt aleși astfel încât  $q_1$  nu mai apare în parcurgerea lui  $\alpha_2$ .

Deoarece  $\delta(q_0, \alpha_1) = q_1$ ,  $\alpha_1$  nu poate avea ca ultim element decât pe 0; deci  $\alpha_1 = \alpha'0$ .

Similar, cum lui  $\alpha_2$  nu îi corespunde nici un drum în graf care să conțină alt  $q_1$ , înseamnă că acest drum nu va avea nici pe  $q_2$  (deoarece orice drum de la  $q_2$  la  $q_3$  va trece și prin  $q_1$ ).

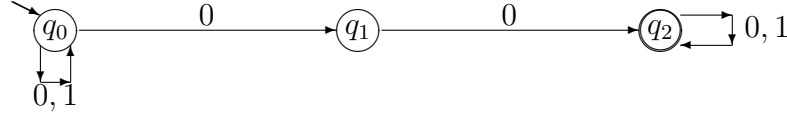
Deci din  $q_1$  nu se poate trece decât în  $q_3$ , adică  $\alpha_2$  trebuie să înceapă cu 0:  $\alpha_2 = 0\alpha''$ .

Rezumând,  $\alpha = \alpha_1 \alpha_2 = \alpha'00\alpha''$ .

**Observația 1.2** În prezentarea de până acum s-a utilizat diagrama de tranziție, nu și funcția  $\delta$  definită prin tabel.

Avantajul utilizării tabelului rezidă din posibilitatea de a decide mai rapid dacă funcția de tranziție  $\delta$  este sau nu bine definită.

De exemplu, diagrama



deși aparent răspunde problemei din Exemplul 1.8, nu corespunde unui AFD deoarece  $\delta(q_0, 0) \notin Q$  (pentru starea  $q_0$  și intrarea 0, funcția  $\delta$  poate lua două valori:  $q_0$  sau  $q_1$ ).

## 1.4 Automate finite nedeterministe

După cum se observă, una din caracteristicile fundamentale ale AFD-urilor este aceea că orice cuvânt de intrare determină un drum unic prin graful asociat automatului.

Pentru orice cuplu  $(q, a) \in Q \times V$  există cel mult un arc notat cu  $a$  în diagrama de tranziție, care pleacă din  $q$  (se ia în considerare și posibilitatea ca  $a \notin \tau(M)$ ; atunci automatul se blochează într-o stare nefinală  $s$ , neexistând nici un arc marcat corespunzător unui caracter din  $\alpha$  care să plece din  $s$ ).

Să renunțăm acum la această condiție și să definim un automat în care, plecând dintr-o stare, un caracter din  $V$  să ofere posibilitatea *alegerii* unei stări în care să treacă automatul.

Această alegere nu mai este determinată apriori, ci se face în funcție de alte criterii, externe automatului.

Noile definiții sunt:

**Definiția 1.3** Un automat finit nedeterminist (AFN pe scurt) este o structură  $M = (Q, V, \delta, q_0, F)$ , unde

- $Q, V, q_0, F$  au aceeași semnificație din Definiția 1.2;
- $\delta : Q \times V \longrightarrow 2^Q$  (mulțimea submulțimilor lui  $Q$ ).

Diagrama de tranziție pentru AFN se definește identic cu cea de la AFD.

De asemenea, funcția de tranziție  $\delta$  se poate extinde la  $\tilde{\delta} : Q \times V^* \longrightarrow 2^Q$  prin:

$$\begin{aligned}\tilde{\delta}(q, \epsilon) &= \{q\}; \\ \tilde{\delta}(q, \alpha a) &= \{p \mid \exists r \in \tilde{\delta}(q, \alpha), p \in \delta(r, a)\}.\end{aligned}$$



În particular, pentru  $\alpha = \epsilon$ , avem

$$\tilde{\delta}(q, a) = \{p \mid \exists r \in \tilde{\delta}(q, \epsilon), p \in \delta(r, a)\} = \{p \mid \exists r \in \{q\}, p \in \delta(r, a)\} = \{p \mid p \in \delta(q, a)\} = \delta(q, a).$$

Deci, fără a micșora generalitatea, putem nota și în cazul *AFN*-urilor,  $\tilde{\delta}$  cu  $\delta$ .

Prin convenție

$$\begin{aligned} \delta(\emptyset, \alpha) &= \emptyset \\ \delta(P, \alpha) &= \bigcup_{q \in P} \delta(q, \alpha), \quad \forall P \subseteq Q, P \neq \emptyset. \end{aligned}$$

În acest fel, definiția extensiei funcției  $\delta$  se poate scrie formal

$$\delta(q, \epsilon) = \{q\}, \quad \delta(q, \alpha a) = \bigcup_{r \in \delta(q, \alpha)} \delta(r, a).$$

### Propoziția 1.2

1.  $\delta(P, \epsilon) = P, \quad \forall P \subseteq Q;$
2.  $\delta(q, \alpha\beta) = \delta(\delta(q, \alpha), \beta), \quad \forall q \in Q, \alpha, \beta \in V^*.$

*Demonstrație:* Este lăsată ca exercițiu (Exercițiul 1.6.).

În particular

$$\delta(q, a\beta) = \delta(\delta(q, a), \beta), \quad \forall q \in Q, a \in V, \beta \in V^*.$$

Limbaajul acceptat de un *AFN*  $M$  este prin definiție:

$$\tau(M) = \{w \mid w \in V^*, \delta(q_0, w) \cap F \neq \emptyset\}.$$

Un cuvânt  $w$  este acceptat de un *AFN* atunci când există o secvență de tranziții corespunzând lui  $w$ , care conduce automatul de la starea inițială la o stare finală.

Sau – dacă se lucrează pe diagrama de tranziție – există cel puțin un drum de la nodul  $q_0$  la un nod final, drum marcat prin cuvântul de intrare.

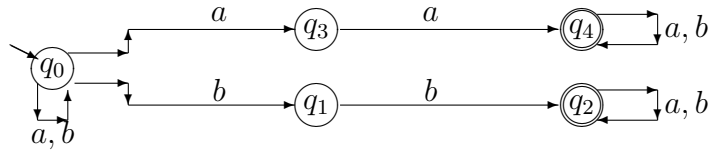
În cazul *AFD*-urilor, acest drum are proprietatea de a fi unic.

**Exemplul 1.9** Fie automatul  $M = (Q, V, \delta, q_0, F)$  definit prin

$$Q = \{q_0, q_1, q_2, q_3, q_4\}, \quad V = \{a, b\}, \quad F = \{q_2, q_4\},$$

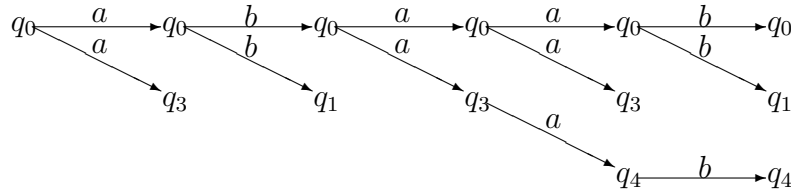
$\delta$	$a$	$b$
$q_0$	$\{q_0, q_3\}$	$\{q_0, q_1\}$
$q_1$	$\emptyset$	$\{q_2\}$
$q_2$	$\{q_2\}$	$\{q_2\}$
$q_3$	$\{q_4\}$	$\emptyset$
$q_4$	$\{q_4\}$	$\{q_4\}$

Diagrama de tranziție va fi



Să luăm de exemplu cuvântul  $\alpha = abaab$ .

Dacă am urmări tranziția stărilor, am obține o figură arborescentă de tipul următor:



Parcurgând toate drumurile posibile, ele duc la una din stările  $q_0$ ,  $q_1$  sau  $q_4$ .

Una din ele ( $q_4$ ) este stare finală; deci există o secvență de stări indicate de cuvântul  $abaab$  care conduce de la  $q_0$  la  $q_4$ , adică  $abaab \in \tau(M)$ .

Pe de-altă parte, folosind Propoziția 1.2, putem scrie formal:

$$\begin{aligned} \delta(q_0, abaab) &= \delta(\delta(q_0, a), baab) = \delta(\{q_0, q_3\}, baab) = \delta(q_0, baab) \cup \delta(q_3, baab) = \\ &= \delta(\delta(q_0, b), aab) \cup \delta(\delta(q_3, b), aab) = \delta(\{q_0, q_1\}, aab) \cup \delta(\emptyset, aab) = \delta(q_0, aab) \cup \delta(q_1, aab) \cup \\ &\emptyset = \delta(\delta(q_0, a), ab) \cup \delta(\delta(q_1, a), ab) = \delta(\{q_0, q_3\}, ab) \cup \delta(\emptyset, ab) = \delta(q_0, ab) \cup \delta(q_3, ab) \cup \emptyset = \\ &\delta(\delta(q_0, a), b) \cup \delta(\delta(q_3, a), b) = \delta(\{q_0, q_3\}, b) \cup \delta(q_4, b) = \delta(q_0, b) \cup \delta(q_3, b) \cup \delta(q_4, b) = \\ &\{q_0, q_1\} \cup \emptyset \cup \{q_4\} = \{q_0, q_1, q_4\}. \end{aligned}$$

Deci  $\delta(q_0, abaab) \cap F = \{q_4\} \neq \emptyset$ .

**Exemplul 1.10** Automatul prezentat în Observația 1.2 este un AFN pentru limbajul din Exemplul 1.13.

El este  $M = (Q, V, \delta, q_0, F)$  cu  $Q = \{q_0, q_1, q_2\}$ ,  $V = \{0, 1\}$ ,  $F = \{q_2\}$  și

$\delta$	0	1
$q_0$	$\{q_0, q_1\}$	$\{q_0\}$
$q_1$	$\{q_2\}$	$\emptyset$
$q_2$	$\{q_2\}$	$\{q_2\}$

Ca un exercițiu, să vedem dacă  $w = 0110$  este un cuvânt din limbaj:

$$\begin{aligned} \delta(q_0, 0110) &= \delta(\delta(q_0, 0), 110) = \delta(\{q_0, q_1\}, 110) = \delta(q_0, 110) \cup \delta(q_1, 110) = \delta(\delta(q_0, 1), 10) \cup \\ &\delta(\delta(q_1, 1), 10) = \delta(q_0, 10) \cup \delta(\emptyset, 10) = \delta(q_0, 10) = \delta(\delta(q_0, 1), 0) = \delta(q_0, 0) = \{q_0, q_1\}. \end{aligned}$$

Cum  $\{q_0, q_1\} \cap F = \emptyset$ , rezultă  $0110 \notin \tau(M)$ .

## 1.5 Echivalența dintre AFD și AFN

**Definiția 1.4** Două automate  $M_1$  și  $M_2$  se numesc "echivalente" dacă  $\tau(M_1) = \tau(M_2)$  (acceptă același limbaj).

Deoarece orice AFD este un caz particular de AFN (când mulțimile de valori se reduc la cel mult câte un element), este evident că limbajele acceptate de AFD-uri sunt acceptate și de AFN-uri.

Mai puțin evidentă este incluziunea inversă, anume:

**Teorema 1.1** Orice limbaj  $L$  acceptat de un AFN este acceptat și de un AFD.

*Demonstrație:* Fie  $M = (Q, V, \delta, q_0, F)$  un AFN cu  $\tau(M) = L$ .

Definim AFD-ul  $M' = (Q', V, \delta', q_0', F')$  astfel:

$$\begin{aligned} Q' &= 2^Q, \quad q_0' = \{q_0\}, \quad F' = \{P \mid P \subseteq Q, P \cap F \neq \emptyset\}, \\ \delta'(\emptyset, a) &= \emptyset, \quad \delta'(P, a) = \bigcup_{q \in P} \delta(q, a), \quad \forall P \subseteq Q, P \neq \emptyset. \end{aligned}$$

Extensia funcției  $\delta'$  se definește în mod natural, conform cu Definiția 1.3.

Vom arăta – prin inducție după lungimea cuvântului  $\alpha$  – relația

$$\delta'(\{q\}, \alpha) = \delta(q, \alpha).$$

Dacă  $|\alpha| = 0$ , atunci  $\alpha = \epsilon$  și deci  $\delta'(\{q\}, \epsilon) = \{q\} = \delta(q, \epsilon)$ .

Să presupunem egalitatea adevărată pentru orice cuvânt de lungime  $k$  și fie  $\beta \in V^+$  cu  $|\beta| = k + 1$ .

Deci  $\beta = \alpha a$  cu  $|\alpha| = k$ ,  $a \in V$ .

Atunci

$$\delta'(\{q\}, \alpha a) \stackrel{(1)}{=} \delta'(\delta'(\{q\}, \alpha), a) \stackrel{(2)}{=} \bigcup_{r \in \delta'(\{q\}, \alpha)} \delta(r, a) \stackrel{(3)}{=} \bigcup_{r \in \delta(q, \alpha)} \delta(r, a) \stackrel{(4)}{=} \delta(q, \alpha a),$$

unde s-a folosit:

- (1) - definiția extensiei lui  $\delta'$ ;
- (2) - definiția lui  $\delta'$ ;
- (3) - ipoteza de inducție;
- (4) - definiția extensiei lui  $\delta$ .

Pe baza acestei relații, se poate construi secvența de echivalențe

$$w \in \tau(M) \iff \delta(q_0, w) \cap F \neq \emptyset \iff \delta'(q_0', w) \cap F \neq \emptyset \iff \delta'(q_0', w) \in F' \iff w \in \tau(M')$$

**Exemplul 1.11** Fie un AFN dat de componentele  $Q = \{q_0, q_1\}$ ,  $V = \{0, 1\}$ ,  $F = \{q_1\}$  și

$\delta$	0	1
$q_0$	$\{q_0, q_1\}$	$\{q_1\}$
$q_1$	$\emptyset$	$\{q_0, q_1\}$

AFD-ul echivalent  $M'$  se va construi astfel:

$Q' = \{p_0, p_1, p_2, p_3\}$ , unde s-a notat  $p_0 = \{q_0\}$ ,  $p_1 = \{q_1\}$ ,  $p_2 = \emptyset$ ,  $p_3 = \{q_0, q_1\}$ .

Apoi,  $q_0' = p_0$ ,  $F' = \{p_1, p_3\}$ .

Să calculăm funcția de tranziție:

$$\begin{aligned} \delta'(p_0, 0) &= \delta(q_0, 0) = \{q_0, q_1\} = p_3; \\ \delta'(p_1, 0) &= \delta(q_1, 0) = \emptyset = p_2; \\ \delta'(p_2, 0) &= \delta'(p_2, 1) = \emptyset = p_2; \\ \delta'(p_3, 0) &= \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\} = p_3; \\ \delta'(p_0, 1) &= \delta(q_0, 1) = \{q_1\} = p_1; \\ \delta'(p_1, 1) &= \delta(q_1, 1) = \{q_0, q_1\} = p_3; \\ \delta'(p_3, 1) &= \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\} = p_3. \end{aligned}$$

Deci

$\delta'$	0	1
$p_0$	$p_3$	$p_1$
$p_1$	$p_2$	$p_3$
$p_2$	$p_2$	$p_2$
$p_3$	$p_3$	$p_3$

Practic, multe din stările unui *AFN* nu sunt accesibile din starea inițială  $q_0$ .

Cum și așa *AFD*-ul asociat are un număr mare de stări, este preferabil să utilizăm în paralel și un algoritm care să rețină în *AFD* numai stările accesibile.

Algoritmul 1.1 este prezentat în Figura 1.3:

Figura 1.3:

**Algoritm 1.1**

**Intrare:** Un *AFD*  $M = (Q, V, \delta, q_0, F)$ .

**Ieșire:** Un *AFD*  $M' = (Q', V, \delta', q_0, F')$  echivalent, cu proprietatea  
 $\forall q \in Q' \quad \exists \alpha \in V^* \quad \text{cu } \delta'(q_0, \alpha) = q.$

**Algoritm:**

1.  $i \leftarrow 0, \quad T_i = \{q_0\};$
2.  $i \leftarrow i + 1, \quad T_i = T_{i-1} \cup \{\delta(q, a) \mid q \in T_{i-1}, a \in V\};$
3. Dacă  $T_i \neq T_{i-1}$  salt la 2;
4.  $Q' = T_i, \quad F' = F \cap T_i, \quad \delta' = \delta|_{Q' \times V} \quad \text{și} \quad M' = (Q', V, \delta', q_0, F').$

**Propoziția 1.3** Automatul  $M'$  obținut prin Algoritmul 1.1

1. are toate stările accesibile din starea inițială;
2. este echivalent cu  $M$ .

*Demonstrație:* Exercițiu (Exercițiul 1.6).

**Exemplul 1.12** Fie  $M = (Q, V, \delta, q_0, F)$  unde  $Q = \{q_0, q_1, q_2, q_3, q_4\}, \quad V = \{a, b\},$   
 $F = \{q_2, q_4\}$  și

$\delta$	$a$	$b$
$q_0$	$\{q_0, q_3\}$	$\{q_0, q_1\}$
$q_1$	$\emptyset$	$\{q_2\}$
$q_2$	$\{q_2\}$	$\{q_2\}$
$q_3$	$\{q_4\}$	$\emptyset$
$q_4$	$\{q_4\}$	$\{q_4\}$

Să construim *AFD*-ul corespunzător.

Teoretic,  $Q' = 2^Q$  va avea  $2^5 = 32$  stări.

Deoarece însă multe din ele – fiind inaccesibile – nu sunt folosite de funcția  $\delta'$ , va fi inutil să le listăm de la început.

Construim funcția de tranziție  $\delta'$  și – pe măsură ce apar stări noi, le vom introduce în  $Q'$ . Inițial, fie

$$p_0 = \{q_0\}, \quad p_1 = \{q_1\}, \quad p_2 = \{q_2\}, \quad p_3 = \{q_3\}, \quad p_4 = \{q_4\}.$$

$$\delta'(p_0, a) = \delta(q_0, a) = \{q_0, q_3\}; \quad \text{notăm } p_5 = \{q_0, q_3\} \text{ noua stare introdusă};$$

$$\delta'(p_0, b) = \delta(q_0, b) = \{q_0, q_1\}, \quad p_6 = \{q_0, q_1\};$$

$$\delta'(p_1, a) = \delta(q_1, a) = \emptyset; \quad p_7 = \emptyset;$$

$$\delta'(p_1, b) = \delta(q_1, b) = \{q_2\} = p_2; \quad \delta'(p_2, a) = \delta(q_2, a) = \{q_2\} = p_2;$$

$$\delta'(p_2, b) = \delta(q_2, b) = \{q_2\} = p_2; \quad \delta'(p_3, a) = \delta(q_3, a) = \{q_4\} = p_4;$$

$$\delta'(p_3, b) = \delta(q_3, b) = \emptyset = p_7; \quad \delta'(p_4, a) = \delta(q_4, a) = \{q_4\} = p_4 = \delta'(p_4, b);$$

$$\delta'(p_5, a) = \bigcup_{p \in p_5} \delta(p, a) = \delta(q_0, a) \cup \delta(q_3, a) = \{q_0, q_3\} \cup \{q_4\} = \{q_0, q_3, q_4\}.$$

Fiind o stare nouă, notăm  $p_8 = \{q_0, q_3, q_4\}$ ;

$$\delta'(p_5, b) = \bigcup_{p \in p_5} \delta(p, b) = \delta(q_0, b) \cup \delta(q_3, b) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\} = p_6;$$

$$\delta'(p_6, a) = \bigcup_{p \in p_6} \delta(p, a) = \delta(q_0, a) \cup \delta(q_1, a) = \{q_0, q_3\} \cup \emptyset = p_5;$$

$$\delta'(p_6, b) = \bigcup_{p \in p_6} \delta(p, b) = \delta(q_0, b) \cup \delta(q_1, b) = \{q_0, q_1\} \cup \{q_2\} = \{q_0, q_1, q_2\}.$$

Configurația  $\{q_0, q_1, q_2\}$  este nouă; deci  $p_9 = \{q_0, q_1, q_2\}$ ;

$$\delta'(p_7, a) = \delta'(\emptyset, a) = \emptyset = p_7 = \delta'(p_7, b);$$

$$\delta'(p_8, a) = \bigcup_{p \in p_8} \delta(p, a) = \delta(q_0, a) \cup \delta(q_3, a) \cup \delta(q_4, a) = \{q_0, q_3\} \cup \{q_4\} \cup \{q_4\} = \{q_0, q_3, q_4\} =$$

$$= p_8;$$

$$\delta'(p_8, b) = \bigcup_{p \in p_8} \delta(p, b) = \delta(q_0, b) \cup \delta(q_3, b) \cup \delta(q_4, b) = \{q_0, q_1\} \cup \emptyset \cup \{q_4\} = \{q_0, q_1, q_4\} = p_{10},$$

$$\text{unde } p_{10} = \{q_0, q_1, q_4\};$$

$$\delta'(p_9, a) = \bigcup_{p \in p_9} \delta(p, a) = \delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) = \{q_0, q_3\} \cup \emptyset \cup \{q_2\} = \{q_0, q_2, q_3\} = p_{11};$$

$$\text{deci } p_{11} = \{q_0, q_2, q_3\};$$

$$\delta'(p_9, b) = \bigcup_{p \in p_9} \delta(p, b) = \delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) = \{q_0, q_1\} \cup \{q_2\} \cup \{q_4\} = \{q_0, q_1, q_2\} =$$

$$= p_9;$$

$$\delta'(p_{10}, a) = \delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_4, a) = \{q_0, q_3\} \cup \emptyset \cup \{q_4\} = \{q_0, q_3, q_4\} = p_8;$$

$$\delta'(p_{10}, b) = \delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_4, b) = \{q_0, q_1\} \cup \{q_2\} \cup \{q_4\} = \{q_0, q_1, q_2, q_4\}$$

$$\text{și notăm } p_{12} = \{q_0, q_1, q_2, q_4\};$$

$$\delta'(p_{11}, a) = \delta(q_0, a) \cup \delta(q_2, a) \cup \delta(q_3, a) = \{q_0, q_3\} \cup \{q_2\} \cup \{q_4\} = \{q_0, q_2, q_3, q_4\}$$

$$\text{și } p_{13} = \{q_0, q_2, q_3, q_4\};$$

$$\begin{aligned}
\delta'(p_{11}, b) &= \delta(q_0, b) \cup \delta(q_2, b) \cup \delta(q_3, b) = \{q_0, q_1\} \cup \{q_2\} \cup \emptyset = \{q_0, q_1, q_2\} = p_9; \\
\delta'(p_{12}, a) &= \delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_4, a) = \{q_0, q_3\} \cup \emptyset \cup \{q_2\} \cup \{q_4\} = p_{13}; \\
\delta'(p_{12}, b) &= \delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_4, b) = \{q_0, q_1\} \cup \{q_2\} \cup \{q_4\} \cup \{q_2\} = p_{12}; \\
\delta'(p_{13}, a) &= \{q_0, q_2, q_3, q_4\} = p_{13}; \quad \delta'(p_{13}, b) = \{q_0, q_1, q_2, q_4\} = p_{12}.
\end{aligned}$$

Deci, în loc de 32 stări, au fost folosite efectiv numai 14, restul (18 stări) fiind inaccesibile din  $q_0$ .

Tabela funcției  $\delta'$  este

$\delta'$	$a$	$b$
$p_0$	$p_5$	$p_6$
$p_1$	$p_7$	$p_2$
$p_2$	$p_2$	$p_2$
$p_3$	$p_4$	$p_7$
$p_4$	$p_4$	$p_4$
$p_5$	$p_8$	$p_6$
$p_6$	$p_5$	$p_9$
$p_7$	$p_7$	$p_7$
$p_8$	$p_8$	$p_{10}$
$p_9$	$p_{11}$	$p_9$
$p_{10}$	$p_8$	$p_{12}$
$p_{11}$	$p_{13}$	$p_9$
$p_{12}$	$p_{13}$	$p_{12}$
$p_{13}$	$p_{13}$	$p_{12}$

Celelalte componente ale lui  $M'$  sunt:

$$Q' = \{p_0, p_1, \dots, p_{13}\}, \quad p_0 \text{ stare inițială, și}$$

$$F' = \{p \mid p \cap F \neq \emptyset\} = \{p_2, p_4, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}\}.$$

Acest AFD mai poate fi redus, el conținând și alte stări inaccesibile.

Să eliminăm aceste stări, utilizând Algoritmul 1.1.

Deci:

$$T_0 = \{p_0\};$$

$$T_1 = T_0 \cup \{p \mid p = \delta(p_0, a) \text{ sau } p = \delta(p_0, b)\} = \{p_0\} \cup \{p_5, p_6\} = \{p_0, p_5, p_6\};$$

$$T_2 = T_1 \cup \{p \mid p = \delta(q, x), q \in T_1, x \in \{a, b\}, p \notin T_1\} = \{p_0, p_5, p_6\} \cup \{p_8, p_9\} = \{p_0, p_5, p_6, p_8, p_9\};$$

$$T_3 = T_2 \cup \{p \mid p = \delta(q, x), q \in T_2, x \in \{a, b\}, p \notin T_2\} = \{p_0, p_5, p_6, p_8, p_9\} \cup \{p_{10}, p_{11}\} = \{p_0, p_5, p_6, p_8, p_9, p_{10}, p_{11}\};$$

$$T_4 = T_3 \cup \{p \mid p = \delta(q, x), q \in T_3, x \in \{a, b\}, p \notin T_3\} = \{p_0, p_5, p_6, p_8, p_9, p_{10}, p_{11}\} \cup \{p_{12}, p_{13}\} = \{p_0, p_5, p_6, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}\};$$

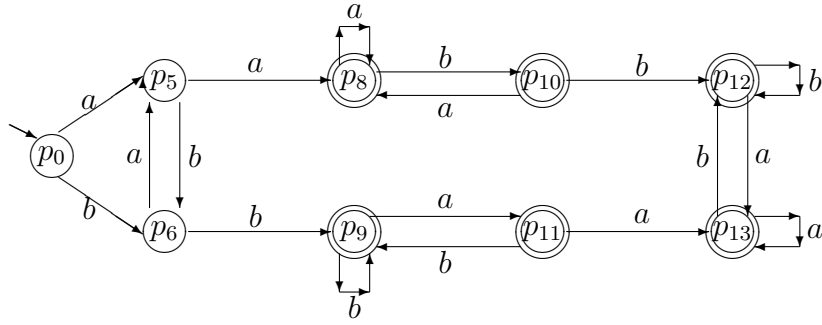
$$T_5 = T_4 \cup \{p \mid p = \delta(q, x), q \in T_4, x \in \{a, b\}, p \notin T_4\} = T_4 \cup \emptyset = T_4.$$

Algoritmul 1.1 s-a terminat și automatul astfel redus are doar 9 stări.

El va fi  $M'' = (Q'', V, \delta'', p_0, F'')$ , unde

$$Q'' = T_4 = \{p_0, p_5, p_6, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}\},$$

$$F'' = T_4 \cap F = \{p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}\}, \text{ și diagrama de tranziție}$$



## 1.6 Automate finite cu $\epsilon$ - mișcări

Conform definiției unui automat finit, în general  $\delta(q, \epsilon) = \{q\}$ .

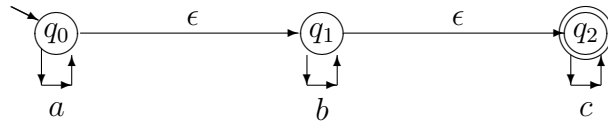
Vom extinde acum această definiție și pentru cazul  $|\delta(q, \epsilon)| > 1$ .

Această variantă de extensie apare frecvent în practică. Există procese care, deși nu sunt activate de nici un mesaj de intrare, după un anumit interval de timp își modifică comportamentul: ecrane de calculator intră în standby, materiale sunt depreciate prin uzură morală sau fizică etc.

Deci un astfel de proces există și trebuie cuprins într-o formalizare a automatelor.

**Definiția 1.5** *Un automat finit nedeterminist cu  $\epsilon$  - mișcări este o structură  $M = (Q, V, \delta, q_0, F)$ , unde  $Q, V, q_0, F$  sunt similare cu cele din Definiția 1.2, iar funcția de tranziție este  $\delta : Q \times (V \cup \{\epsilon\}) \longrightarrow 2^Q$ .*

**Exemplul 1.13** *Fie automatul dat de diagrama de tranziție*



Funcția de tranziție este

$\delta$	$a$	$b$	$c$	$\epsilon$
$q_0$	$\{q_0\}$	$\emptyset$	$\emptyset$	$\{q_1\}$
$q_1$	$\emptyset$	$\{q_1\}$	$\emptyset$	$\{q_2\}$
$q_2$	$\emptyset$	$\emptyset$	$\{q_2\}$	$\emptyset$

Pentru a defini extensia  $\hat{\delta} : Q \times V^* \longrightarrow Q$ , un prim pas constă în aflarea tuturor stărilor accesibile dintr-o anumită stare  $q$ , folosind numai  $\epsilon$  - tranziții. Fie deci, pentru  $q \in Q$ :

$$\langle q \rangle = \{ p \mid \exists r_0, \dots, r_k \in Q, k \geq 0, r_0 = q, r_k = p, r_{i+1} \in \delta(r_i, \epsilon), 0 \leq i \leq k-1 \}$$



Pentru calculul lui  $\langle q \rangle$  se poate da un algoritm – de fapt o variantă simplificată a Algoritmului 1.1 – pe care îl propunem ca exercițiu.

În mod natural,

$$\langle P \rangle = \bigcup_{q \in P} \langle q \rangle, \quad \langle \emptyset \rangle = \emptyset.$$

**Lema 1.1**

1.  $q \in \langle q \rangle$ ;
2.  $p \in \langle q \rangle \implies \langle p \rangle \subseteq \langle q \rangle$ ;
3.  $\langle \langle q \rangle \rangle = \langle q \rangle$ .

*Demonstrație:*

1. În definiția lui  $\langle q \rangle$  se ia  $k = 0$ .

2. Fie  $p \in \langle q \rangle$ ; există deci stările  $r_0, \dots, r_k$  cu  $r_0 = q, r_k = p$  și  $r_i \in \delta(r_{i-1}, \epsilon)$ ,  $i = \overline{1, k}$ .

Dacă  $r \in \langle p \rangle$ , atunci există  $t_0, \dots, t_s \in Q$  cu  $t_0 = p, t_s = r$  și  $t_j \in \delta(t_{j-1}, \epsilon)$ ,  $j = \overline{1, s}$ .

Să considerăm secvența de stări  $r_0, \dots, r_k (= t_0), t_1, \dots, t_s$  (dacă  $s > 0$ );

obținem (conform definiției)  $r \in \langle q \rangle$ .

Deci  $\langle p \rangle \subseteq \langle q \rangle$ .

În cazul  $s = 0$  se ia secvența de stări  $r_0, r_1, \dots, r_k$  și se raționează analog.

3. Se știe că  $\langle \langle q \rangle \rangle = \bigcup_{p \in \langle q \rangle} \langle p \rangle$ .

" $\subseteq$ ": Cu (2),  $p \in \langle q \rangle \implies \langle p \rangle \subseteq \langle q \rangle$ .

Deci  $\bigcup_{p \in \langle q \rangle} \langle p \rangle \subseteq \bigcup_{p \in \langle q \rangle} \langle q \rangle = \langle q \rangle$ , sau  $\langle \langle q \rangle \rangle \subseteq \langle q \rangle$ .

" $\supseteq$ ":  $\langle \langle q \rangle \rangle = \bigcup_{p \in \langle q \rangle} \langle p \rangle \stackrel{(*)}{=} \left( \bigcup_{p \in \langle q \rangle, p \neq q} \langle p \rangle \right) \cup \langle q \rangle \supseteq \langle q \rangle$ ,

unde în egalitatea (\*) s-a utilizat primul punct al Lemei 1.1.

q.e.d.

**Exemplul 1.14** Pentru automatul din Exemplul 1.13 vom avea

$\langle q_0 \rangle = \{q_0, q_1, q_2\}$  deoarece:

- $q_0 \in \langle q_0 \rangle$  conform Lemei 1.1;
- $q_1$  pentru că  $q_1 \in \delta(q_0, \epsilon)$  ( $k = 1$ );
- $q_2$  pentru că  $q_2 \in \delta(q_1, \epsilon)$  ( $k = 2$ ).

Evident, pentru  $q_2$  de exemplu, putem avea și drumuri mai lungi; astfel,

$$q_0 - q_0 - q_0 - q_1 - q_1 - q_2 - q_2$$

poate fi un drum, în care  $k = 6$ .

Importantă este existența unui drum; el nu va fi obligatoriu unic. De obicei convenim ca lungimea lui să fie minimă.

Similar,  $\langle q_1 \rangle = \{q_1, q_2\}$ ,  $\langle q_2 \rangle = \{q_2\}$ .

În acest moment putem defini construcția recursivă a lui  $\hat{\delta}$  astfel:

$$\begin{aligned}\hat{\delta}(q, \epsilon) &= \langle q \rangle; \\ \hat{\delta}(q, a\alpha) &= \{p \mid \exists r \in \delta(\langle q \rangle, a), p \in \hat{\delta}(r, \alpha)\}\end{aligned}$$

**Observația 1.3** Această definiție se mai poate scrie formal și

$$\hat{\delta}(q, a\alpha) = \hat{\delta}(\langle \delta(\langle q \rangle, a) \rangle, \alpha) \quad (1)$$

unde am folosit notația obișnuită:

$$\hat{\delta}(\emptyset, \alpha) = \emptyset, \quad \hat{\delta}(P, \alpha) = \bigcup_{q \in P} \hat{\delta}(q, \alpha).$$

**Propoziția 1.4**  $\hat{\delta}(q, a) = \langle \delta(\langle q \rangle, a) \rangle$ .

*Demonstrație:* În (1) se ia  $\alpha = \epsilon$  și obținem

$$\hat{\delta}(q, a) = \hat{\delta}(\langle \delta(\langle q \rangle, a) \rangle, \epsilon) = \langle \delta(\langle q \rangle, a) \rangle = \langle \delta(\langle q \rangle, a) \rangle. \quad \text{q.e.d.}$$

Deci, spre deosebire de un *AFN* obișnuit, în acest caz  $\hat{\delta}$  nu mai este o generalizare naturală a lui  $\delta$ , între ele existând o relație de incluziune:  $\delta(q, a) \subseteq \hat{\delta}(q, a)$ .

De aceea este necesară o distincție între  $\delta$  și  $\hat{\delta}$ .

Din (1) și Propoziția 1.4 rezultă egalitatea

$$\hat{\delta}(q, a\alpha) = \hat{\delta}(\hat{\delta}(q, a), \alpha) \quad (2)$$

**Lema 1.2**  $\hat{\delta}(q, a) = \hat{\delta}(\langle q \rangle, a)$ .

$$\text{Demonstrație: } \hat{\delta}(\langle q \rangle, a) = \bigcup_{p \in \langle q \rangle} \hat{\delta}(p, a) = \bigcup_{p \in \langle q \rangle} \langle \delta(\langle p \rangle, a) \rangle =$$

$$= \langle \delta(\langle \langle q \rangle \rangle, a) \rangle = \langle \delta(\langle q \rangle, a) \rangle = \hat{\delta}(q, a). \quad \text{q.e.d.}$$

Definim în mod natural (ca la *AFN*) limbajul acceptat de automatul cu  $\epsilon$  - mișcări  $M = (Q, V, \delta, q_0, F)$  ca fiind

$$\tau(M) = \{w \mid w \in V^*, \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

**Exemplul 1.15** Reluând automatul din Exemplul 1.13, să cercetăm dacă  $w = aab$  este sau nu în limbajul acceptat  $\tau(M)$ . Avem:

$$\begin{aligned}\hat{\delta}(q_0, a) &= \langle \delta(\langle q_0 \rangle, a) \rangle = \langle \delta(\{q_0, q_1, q_2\}, a) \rangle = \langle \delta(q_0, a) \rangle \cup \langle \delta(q_1, a) \rangle \cup \\ &\quad \cup \langle \delta(q_2, a) \rangle = \langle \{q_0\} \rangle \cup \langle \emptyset \rangle \cup \langle \emptyset \rangle = \langle q_0 \rangle = \{q_0, q_1, q_2\}; \\ \hat{\delta}(q_1, a) &= \langle \delta(\langle q_1 \rangle, a) \rangle = \langle \delta(\{q_1, q_2\}, a) \rangle = \langle \delta(q_1, a) \rangle \cup \langle \delta(q_2, a) \rangle = \\ &= \langle \emptyset \rangle \cup \langle \emptyset \rangle = \emptyset \cup \emptyset = \emptyset;\end{aligned}$$

$$\begin{aligned}
\hat{\delta}(q_2, a) &= \langle \delta(\langle q_2 \rangle, a) \rangle = \langle \delta(q_2, a) \rangle = \langle \emptyset \rangle = \emptyset; \\
\hat{\delta}(q_0, b) &= \langle \delta(\langle q_0 \rangle, b) \rangle = \langle \delta(\{q_0, q_1, q_2\}, b) \rangle = \langle \delta(q_0, b) \rangle \cup \langle \delta(q_1, b) \rangle \cup \\
&\quad \cup \langle \delta(q_2, b) \rangle = \langle \emptyset \rangle \cup \langle \{q_1\} \rangle \cup \langle \emptyset \rangle = \langle q_1 \rangle = \{q_1, q_2\}; \\
\hat{\delta}(q_1, b) &= \langle \delta(\langle q_1 \rangle, b) \rangle = \langle \delta(\{q_1, q_2\}, b) \rangle = \langle \delta(q_1, b) \rangle \cup \langle \delta(q_2, b) \rangle = \\
&\quad = \langle \{q_1\} \rangle \cup \langle \emptyset \rangle = \langle q_1 \rangle = \{q_1, q_2\}; \\
\hat{\delta}(q_2, b) &= \langle \delta(\langle q_2 \rangle, b) \rangle = \langle \delta(q_2, b) \rangle = \langle \emptyset \rangle = \emptyset.
\end{aligned}$$

Putem acum calcula

$$\begin{aligned}
\hat{\delta}(q_0, aab) &= \hat{\delta}(\hat{\delta}(q_0, a), ab) = \hat{\delta}(\{q_0, q_1, q_2\}, ab) = \hat{\delta}(q_0, ab) \cup \hat{\delta}(q_1, ab) \cup \hat{\delta}(q_2, ab) = \\
&\hat{\delta}(\hat{\delta}(q_0, a), b) \cup \hat{\delta}(\hat{\delta}(q_1, a), b) \cup \hat{\delta}(\hat{\delta}(q_2, a), b) = \hat{\delta}(\{q_0, q_1, q_2\}, b) \cup \hat{\delta}(\emptyset, b) \cup \hat{\delta}(\emptyset, b) = \\
&\hat{\delta}(q_0, b) \cup \hat{\delta}(q_1, b) \cup \hat{\delta}(q_2, b) \cup \emptyset \cup \emptyset = \{q_1, q_2\} \cup \{q_1, q_2\} \cup \emptyset = \{q_1, q_2\}.
\end{aligned}$$

$$\text{Deci } \hat{\delta}(q_0, aab) \cap F = \{q_1, q_2\} \cap \{q_2\} = \{q_2\} \neq \emptyset, \quad \text{adică } aab \in \tau(M).$$

**Propoziția 1.5**  $\hat{\delta}(q, \alpha\beta) = \hat{\delta}(\hat{\delta}(q, \alpha), \beta), \quad \forall \alpha, \beta \in V^*.$

*Demonstrație:* Vom proceda prin inducție după lungimea lui  $\alpha$ :

Dacă  $\alpha = \beta = \epsilon$ , atunci  $\hat{\delta}(q, \epsilon) = \langle q \rangle$  și respectiv  $\hat{\delta}(\hat{\delta}(q, \epsilon), \epsilon) = \hat{\delta}(\langle q \rangle, \epsilon) = \langle \langle q \rangle \rangle = \langle q \rangle$ , deci egalitate.

Dacă  $|\alpha| = 1$ , deci  $\alpha = a$ , avem  $\hat{\delta}(q, a\beta) = \hat{\delta}(\hat{\delta}(q, a), \beta)$  (cf. (2)).

Să presupunem proprietatea adevărată pentru orice cuvânt de lungime  $n$  și fie  $\alpha \in V^*$ ,  $|\alpha| = n + 1$ ; deci  $\alpha = a\alpha'$  cu  $|\alpha'| = n$ .

Atunci

$$\hat{\delta}(q, a\alpha'\beta) = \hat{\delta}(\hat{\delta}(q, a), \alpha'\beta) \stackrel{(i)}{=} \hat{\delta}(\hat{\delta}(\hat{\delta}(q, a), \alpha'), \beta) = \hat{\delta}(\hat{\delta}(q, a\alpha'), \beta) \stackrel{(ii)}{=} \hat{\delta}(\hat{\delta}(q, \alpha), \beta),$$

unde s-a folosit:

(i) - ipoteza de inducție;

(ii) - egalitatea (1).

q.e.d.

În particular, vom avea  $\hat{\delta}(\alpha a) = \hat{\delta}(\hat{\delta}(q, \alpha), a)$ .

**Teorema 1.2** Dacă  $L$  este un limbaj acceptat de un AFN cu  $\epsilon$  - mișcări, atunci  $L$  este acceptat de un AFN fără  $\epsilon$  - mișcări.

*Demonstrație:* Fie  $M = (Q, V, \delta, q_0, F)$  un AFN cu  $\epsilon$  - mișcări.

Construim un AFN  $M' = (Q, V, \delta', q_0, F')$  în care  $\delta'(q, a) = \hat{\delta}(q, a)$  și

$$F' = \begin{cases} F \cup \{q_0\} & \text{dacă } \langle q_0 \rangle \cap F \neq \emptyset \\ F & \text{altfel} \end{cases}$$

Prin construcție,  $M'$  este un AFN fără  $\epsilon$  - mișcări.

Deci se poate folosi notația  $\delta'$  în loc de  $\hat{\delta}'$  (păstrându-se însă diferențierea între  $\delta$  și  $\hat{\delta}$ ).

Vom arăta întâi, prin inducție după lungimea lui  $\alpha \in V^+$  că

$$\delta'(q_0, \alpha) = \hat{\delta}(q_0, \alpha) \quad (3)$$

$|\alpha| = 1$ : atunci  $\alpha = a \in V$  și (3) se verifică banal folosind definiția lui  $\delta'$ .

Fie acum  $\alpha$  cu  $|\alpha| > 1$ ; atunci  $\alpha = \beta a$  cu  $a \in V$ . Vom avea

$$\begin{aligned} \delta'(q_0, \alpha) &= \delta'(q_0, \beta a) \stackrel{(i)}{=} \delta'(\delta'(q_0, \beta), a) \stackrel{(ii)}{=} \delta'(\hat{\delta}(q_0, \beta), a) \stackrel{(iii)}{=} \hat{\delta}(\hat{\delta}(q_0, \beta), a) \stackrel{(iv)}{=} \hat{\delta}(q_0, \beta a) = \\ &= \hat{\delta}(q_0, \alpha), \end{aligned}$$

unde s-a folosit:

- (i) – extensia lui  $\delta'$
- (ii) – ipoteza de inducție
- (iii) – definiția lui  $\delta'$
- (iv) – Propoziția 1.5

Folosind acum relația (3), vom deduce egalitatea  $\tau(M) = \tau(M')$  (adică echivalența celor două automate).

" $\subseteq$ ": Fie  $\alpha \in \tau(M)$ ; deci  $\hat{\delta}(q_0, \alpha) \cap F \neq \emptyset$ .

Dacă  $\alpha = \epsilon$ , atunci  $\hat{\delta}(q_0, \epsilon) \cap F \neq \emptyset$ , deci  $q_0 \in F' \delta'(q_0, \epsilon) = \{q_0\} \subseteq F' \implies \epsilon \in \tau(M')$ .

Dacă  $\alpha \neq \epsilon$ , atunci  $\hat{\delta}(q_0, \alpha) = \delta'(q_0, \alpha)$  (cu (3)), deci  $\delta'(q_0, \alpha) \cap F' \supseteq \hat{\delta}(q_0, \alpha) \cap F \neq \emptyset$ .

" $\supseteq$ ": Considerăm  $\alpha \in \tau(M')$ .

Dacă  $\alpha = \epsilon$ , atunci  $q_0 \in F'$ .

Din construcția lui  $F'$  rezultă  $\langle q_0 \rangle \cap F \neq \emptyset$  sau  $\hat{\delta}(q_0, \epsilon) \cap F \neq \emptyset$ ; deci  $\epsilon \in \tau(M)$ .

Să presupunem  $\alpha \neq \epsilon$ .

Atunci – cu (3) – rezultă imediat  $\alpha \in \tau(M)$ .

q.e.d.

**Exemplul 1.16** Să reluăm automatul definit în Exemplul 1.13 și să construim un automat echivalent, fără  $\epsilon$  - mișcări.

Închiderile celor trei stări au fost construite în Exemplul 1.14.

Deoarece  $\langle q_0 \rangle \cap F \neq \emptyset$ , vom avea  $F' = \{q_0, q_2\}$ .

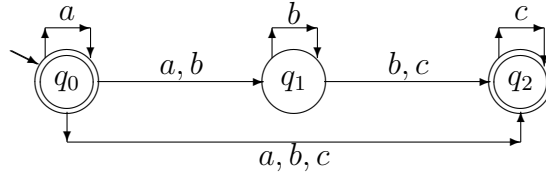
Funcția de tranziție a noului automat este determinată prin calcul:

$$\begin{aligned} \delta'(q_0, a) &= \hat{\delta}(q_0, a) = \{q_0, q_1, q_2\}; \\ \delta'(q_0, b) &= \hat{\delta}(q_0, b) = \{q_1, q_2\}; \\ \delta'(q_0, c) &= \hat{\delta}(q_0, c) = \langle \delta(\langle \tau_0 \rangle, c) \rangle = \langle \delta(\{q_0, q_1, q_2\}, c) \rangle = \\ &= \langle \delta(q_0, c) \cup \delta(q_1, c) \cup \delta(q_2, c) \rangle = \langle \emptyset \cup \emptyset \cup \{q_2\} \rangle = \langle q_2 \rangle = \{q_2\}; \\ \delta'(q_1, a) &= \hat{\delta}(q_1, a) = \emptyset; \\ \delta'(q_1, b) &= \hat{\delta}(q_1, b) = \{q_1, q_2\}; \\ \delta'(q_1, c) &= \hat{\delta}(q_1, c) = \langle \delta(\langle \tau_1 \rangle, c) \rangle = \langle \delta(\{q_1, q_2\}, c) \rangle = \langle \delta(q_1, c) \cup \delta(q_2, c) \rangle = \\ &= \langle \emptyset \cup \{q_2\} \rangle = \langle q_2 \rangle = \{q_2\}; \\ \delta'(q_2, a) &= \delta'(q_2, b) = \emptyset; \\ \delta'(q_2, c) &= \hat{\delta}(q_2, c) = \langle \delta(\langle \tau_2 \rangle, c) \rangle = \langle \delta(q_2, c) \rangle = \langle q_2 \rangle = \{q_2\}. \end{aligned}$$

Deci

$\delta'$	$a$	$b$	$c$
$q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$q_1$	$\emptyset$	$\{q_1, q_2\}$	$\{q_2\}$
$q_2$	$\emptyset$	$\emptyset$	$\{q_2\}$

Diagrama de tranziție a automatului  $M' = (Q, V, \delta', q_0, F')$  este



## 1.7 Exerciții

**Exercițiul 1.1** Fie  $V = \{a, b, c\}$ ,  $W = \{x, y, z, u\}$ . Să se scrie  $VW$ ,  $WV$ ,  $WW$ .

**Exercițiul 1.2** Fie  $A$  o mulțime nevidă.  $A$  este finită dacă și numai dacă  $A^* = \{\epsilon\}$ .

**Exercițiul 1.3** Dacă  $A$  are  $n$  elemente, câte elemente va avea  $A^k$  ( $k \geq 0$ ) ?

**Exercițiul 1.4** Să se construiască AFD-uri pentru următoarele limbaje:

1.  $L = \{0^{3k+1} \mid k \geq 0\}$ ;
2.  $L = \{\alpha \mid \alpha \in \{a, b\}^*, \alpha = \alpha'bbab\}$ ;
3.  $L = \{\alpha \mid \alpha \in \{0, 1\}^*; \text{ scris în binar, } \alpha \text{ este un multiplu de } 5\}$ ;
4.  $L = \{a_1a_2 \dots a_n \mid n \geq 3, a_i \in \{x, y\}, a_{n-2} = y\}$ ;
5.  $L = \{a^ib^jc^k \mid i, j, k \geq 1\}$ ;
6.  $L$  conține toate cuvintele peste alfabetul  $\{a, b\}$  care au un număr par de  $a$ -uri și un număr impar de  $b$ -uri;
7.  $L$  conține toate cuvintele peste alfabetul  $\{0, 1\}$  care au cel puțin trei zerouri consecutive.

**Exercițiul 1.5** Fie  $M = (Q, V, \delta, q_0, F)$  un AFD și  $\alpha, \beta \in V^*$  cu  $\delta(q_0, \alpha) = \delta(q_0, \beta)$ .

Să se arate că  $\forall \gamma \in V^*, \quad \alpha\gamma \in \tau(M) \iff \beta\gamma \in \tau(M)$ .

**Exercițiul 1.6** Să se demonstreze Propozițiile 1.2 și 1.3.

**Exercițiul 1.7** Să se construiască AFN-uri pentru următoarele limbaje:

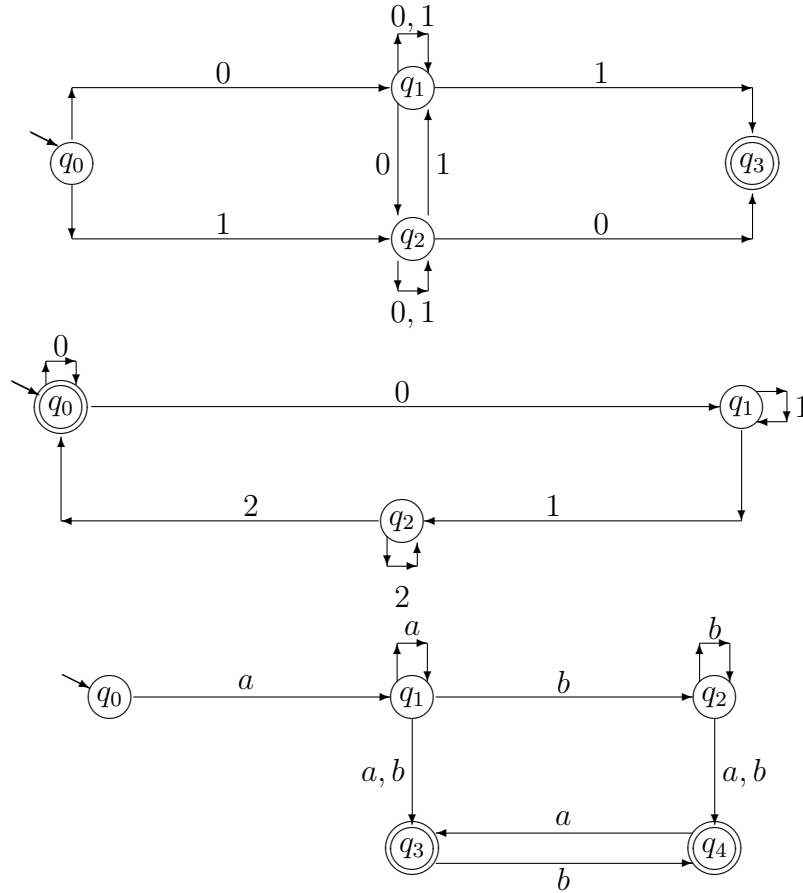
1. Limbajul cuvintelor peste  $\{a, b, c\}$  cu proprietatea că ultimul simbol a mai apărut anterior;
2. Limbajul numerelor în care cel puțin o cifră se repetă;
3.  $L = \{w \mid w \in \{a, b\}^*, \exists w_1, w_2 \text{ cu } w = w_1 a a w_2 \text{ sau } w = w_1 b b w_2\}$ .

**Exercițiul 1.8** Fie  $M = (Q, V, \delta, q_0, F)$  un AFD cu  $F = \{q'\}$ .

Construim AFD-ul  $M' = (Q, V, \delta', q', F')$  unde  $F' = \{q_0\}$ ,  $\delta'(q, a) = \{p \mid \delta(p, a) = q\}$ .

Să se exprime  $\tau(M')$  în funcție de  $\tau(M)$ .

**Exercițiul 1.9** Să se construiască AFD-uri pentru AFN-urile date prin diagramele de tranziție:



**Exercițiul 1.10** Să se construiască un AFD echivalent cu automatul dat în Exemplul 1.16.

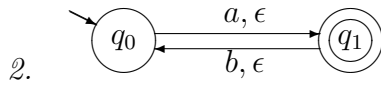
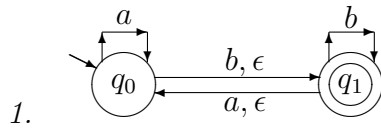
**Exercițiul 1.11** Să se construiască un algoritm pentru calculul mulțimilor  $\langle q \rangle$ .

**Exercițiul 1.12** Fie automatul  $M = (Q, V, \delta, q_0, F)$  unde  $Q = \{q_0, q_1, q_2, q_3\}$ ,  $V = \{0, 1\}$ ,  $F = \{q_2\}$  și

$\delta$	0	1	$\epsilon$
$q_0$	$\{q_1\}$	$\emptyset$	$\emptyset$
$q_1$	$\emptyset$	$\emptyset$	$\{q_2, q_3\}$
$q_2$	$\emptyset$	$\{q_2\}$	$\{q_3\}$
$q_3$	$\{q_1\}$	$\{q_0\}$	$\emptyset$

Să se construiască un AFD echivalent.

**Exercițiul 1.13** Să se construiască AFD echivalente cu AFN cu  $\epsilon$  - mișcări date de diagramele de tranziție



**Exercițiul 1.14** Este adevărată egalitatea

$$\delta(\langle q_0 \rangle, \alpha) = \langle \hat{\delta}(q_0, \alpha) \rangle ?$$

**Exercițiul 1.15** Să se arate că:

1. Dacă  $p \in \langle q \rangle$  atunci  $\forall a \in V [\langle \delta(p, a) \rangle \subseteq \langle \delta(q, a) \rangle]$ ;
2.  $\forall A, B \subseteq Q$  avem  $\langle A \cup B \rangle = \langle A \rangle \cup \langle B \rangle$ .

# Capitolul 2

## Teoreme de reprezentare pentru limbajele regulate

Începem cu o definiție:

**Definiția 2.1** *Limbaajul acceptat de un automat finit se numește limbaj regulat.*

Conform celor demonstrate în Capitolul 1, limbajele regulate sunt limbajele acceptate de automatele finite (deterministe).

Atunci când avem un automat finit  $M$ , știm că  $\tau(M)$  este un limbaj regulat.

Să considerăm însă și problema reciprocă: fiind dat un limbaj  $L$ , cum putem ști dacă acesta este regulat sau nu: merită să căutăm un automat finit  $M$  cu  $\tau(M) = L$ , sau putem stabili de la început că acest lucru este imposibil ?

Capitolul 2 va încerca să dea un răspuns acestei probleme.

### 2.1 Lema de pompare

**Lema 2.1** (*Lema de pompare*): *Fie  $L$  un limbaj regulat. Există atunci un număr natural  $n$  astfel încât pentru orice cuvânt  $\alpha \in L$ ,  $|\alpha| \geq n$ , putem scrie  $\alpha = uvw$ , cu proprietățile:*

1.  $|uv| \leq n$ ;
2.  $|v| \geq 1$ ;
3.  $uv^i w \in L, \quad \forall i \geq 0$ .



*Demonstrație:* Fiind regulat, limbajul  $L$  este acceptat de un AFD  $M = (Q, V, \delta, q_0, F)$ ; fie  $n = |Q|$  valoarea pe care o considerăm în Lemă.

Să luăm un cuvânt  $\alpha \in L$  cu  $|\alpha| = m \geq n$ ; putem scrie deci  $\alpha = a_1 a_2 \dots a_m$ .

Fie

$$\delta(q_0, a_1 \dots a_i) = q_i \quad (1 \leq i \leq m).$$

Evident,  $q_m \in F$  (cuvântul  $\alpha$  este în limbajul  $L$ ).

În secvența de  $m + 1$  stări  $q_0, q_1, \dots, q_m$  există cel puțin două stări care coincid (sunt numai  $n$  stări distincte și  $m + 1 > n$ ).

Fie  $q_j$  și  $q_k$  două astfel de stări, cu  $q_j = q_k$ ,  $0 \leq j < k \leq m$ .

Vom nota  $u = a_1 \dots a_j$ ,  $v = a_{j+1} \dots a_k$ ,  $w = a_{k+1} \dots a_m$ .

Fără a micșora generalitatea, putem presupune că în secvența  $q_0, \dots, q_k$  nu mai sunt alte stări care se repetă (se ia  $j$  minim cu proprietatea că există  $k > j$  cu  $q_k = q_j$ ).

Să arătăm că cele trei cerințe din lema sunt îndeplinite:

1. Secvența  $q_0, \dots, q_{k-1}$  fiind formată din stări distincte ( $q_k$  este prima stare care se repetă), are cel mult  $n$  elemente. Cum ea generează cuvântul  $uv$ , rezultă  $|uv| \leq n$ .
2. Deoarece  $j < k$ , cuvântul  $v = a_{j+1} \dots a_k$  are lungimea cel puțin 1.
3.  $\delta(q_0, u) = q_j$ ,  $\delta(q_j, v) = q_k = q_j$ ,  $\delta(q_k, w) = q_m$ . Se poate scrie atunci

$$\delta(q_0, uv^i w) = \delta(\delta(q_0, u), v^i w) = \delta(q_j, v^i w) = \dots = \delta(q_j, w) = q_m \in F$$

deci  $uv^i w \in L$ ,  $\forall i \geq 0$ .

q.e.d.

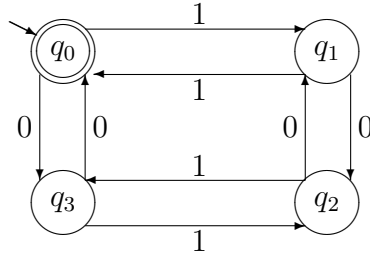
### Observația 2.1

- Lema de pompare arată că dacă o mulțime regulată conține un cuvânt  $uvw$  de lungime suficient de mare, atunci există o infinitate de cuvinte de forma  $uv^i w$ .

Aceasta nu înseamnă însă că orice cuvânt de lungime suficient de mare este de forma  $uv^i w$  pentru  $u, v, w \in V^*$  și un anumit  $i$  întreg.

- Se poate demonstra similar o variantă a lemei de pompare, în care (1) se înlocuiește cu (1')  $|uw| \leq n$  sau cu (1'')  $|vw| \leq n$ .

**Exemplul 2.1** Fie automatul finit determinist definit prin diagrama de tranziție



Se poate arăta ușor că el acceptă toate cuvintele cu un număr par de 0 și un număr par de 1. Conform demonstrației din Lema de pompare,  $n = 4$ .

Să luăm un cuvânt arbitrar  $\alpha \in \tau(M)$  cu  $|\alpha| \geq 4$ ; de exemplu  $\alpha = 101101$ .

Secvența de stări a automatului, trasată de acest cuvânt este

$$q_0 \xrightarrow{1} q_1 \xrightarrow{0} \underline{q_2} \xrightarrow{1} q_3 \xrightarrow{1} \underline{q_2} \xrightarrow{0} q_1 \xrightarrow{1} q_0$$

Prima stare care se repetă este  $q_2$  (apare pe pozițiile 3 și 5). Deci  $j = 2$  și  $k = 4$ .

Cuvintele în care se descompune  $\alpha$  vor fi  $u = 10$ ,  $v = 11$ ,  $w = 01$ .

Deci  $\alpha = uvw$  și

1.  $|uv| = |1011| = 4 \leq 4$ ;
2.  $|v| = 2 \geq 1$ ;
3.  $uv^i w = 10(11)^i 01 \in \tau(M)$ ,  $\forall i \geq 0$ .

În particular, pentru  $i = 0$ ,  $1001 \in \tau(M)$ , iar pentru  $i = 2$ ,  $10111101 \in \tau(M)$ .

Evident, limbajul  $L_1 = \{10(11)^i 01 \mid i \geq 0\}$  este inclus strict în  $\tau(M)$ , acesta conținând și alte cuvinte în afară de cele din  $L_1$ .

### 2.1.1 Aplicații ale Lemei de pompare

Una din principalele aplicații ale Lemei de pompare constă în posibilitatea de a demonstra că anumite limbaje  $L$  nu sunt regulate (deci nu există automate finite  $M$  astfel ca  $\tau(M) = L$ ).

Vom arăta metoda utilizată folosind două exemple:

#### Exemplul 2.2 Fie

$$L = \{a^{i^2} \mid i \geq 1\}.$$

Vom presupune prin absurd că  $L$  este un limbaj regulat, și fie  $n$  dat de Lema de pompare.

Alegem un cuvânt  $\alpha$  cu  $|\alpha| = n^2$  (acest lucru este posibil deoarece  $n^2 > n$ , iar orice cuvânt din  $L$  are ca lungime un pătrat perfect).

Conform Lemei,  $\alpha = uvw$ , unde  $1 \leq |v| \leq n$  și  $uv^i w \in L$ ,  $\forall i \geq 0$ .

În particular, fie  $i = 2$ .

Atunci, cum  $|uv^2 w| = |uvw| + |v| = n^2 + |v|$ , vom avea  $n^2 < |uv^2 w| \leq n^2 + n < (n+1)^2$ . Deci  $uv^2 w \notin L$  (lungimea lui nu este pătrat perfect), infirmând Lema de pompare.

#### Exemplul 2.3 Să considerăm limbajul

$$L = \{a^k b^k \mid k \geq 1\}.$$

La fel, vom presupune că  $L$  este limbaj regulat, și fie  $n$  dat de Lema de pompare.

Vom lucra cu  $\alpha = a^n b^n$ .

Conform Lemei,  $a^n b^n = uvw$ , cu proprietățile corespunzătoare.

Din  $|uv| \leq n$  rezultă că  $uv$  este formată numai din 'a'-uri ( $uv$  este un prefix al primilor  $n$  caractere din  $\alpha$ ).

Deci  $v = a^p$ ,  $0 < p \leq n$  ( $v \neq \epsilon$ ).

Luând acum  $i = 0$  în proprietatea (3) din Lema de pompare, obținem  $uv^0w = a^{n-p}b^n$ ; acest cuvânt nu este în  $L$ , deoarece are mai puține 'a'-uri decât 'b'-uri, ceea ce contrazice Lema de pompare.

O altă aplicație importantă a Lemei de pompare este posibilitatea de a construi algoritmi care să decidă unele proprietăți ale limbajelor regulate.

**Propoziția 2.1** Fie  $L$  un limbaj acceptat de un AFD cu  $n$  stări. Atunci

1.  $L \neq \emptyset \iff \exists \alpha \in L, |\alpha| < n$ ;
2.  $L$  infinit  $\iff \exists \alpha \in L, n \leq |\alpha| < 2n$ .

*Demonstrație:*

1. " $\Leftarrow$ ": Evident.

" $\Rightarrow$ ": Fie  $\alpha \in L$  ( $\alpha$  există deoarece  $L \neq \emptyset$ ).

Dacă  $|\alpha| < n$ , demonstrația s-a încheiat.

În caz contrar, conform Lemei de pompare,  $\alpha = uvw$ .

Cum  $|v| \geq 1$ , vom avea  $uw \in L$  (cazul  $i = 0$ ) și  $|uw| < |\alpha|$ .

Argumentația se reia pentru cuvântul  $\alpha' = uw$ .

Se obține în acest fel un șir de cuvinte din  $L$ , de lungimi strict descrescătoare.

După un număr finit de pași, se va ajunge la un cuvânt din  $L$  de lungime mai mică decât  $n$ .

2. " $\Leftarrow$ ": Fie  $\alpha \in L$ ,  $|\alpha| \geq n$ .

Conform Lemei de pompare,  $\alpha = uvw$ ,  $v \neq \epsilon$  și  $uv^i w \in L$ ,  $\forall i \geq 0$ .

Deci  $L$  conține o infinitate de cuvinte, ceea ce înseamnă în particular că este infinit.

" $\Rightarrow$ ":  $L$  este infinit, deci există  $\alpha \in L$  cu  $|\alpha| \geq n$ .

Dacă  $|\alpha| < 2n$ , demonstrația este încheiată.

Dacă nu, se aplică Lema de pompare:  $\alpha = uvw$  și deci  $\alpha' = uw \in L$ ,  $|\alpha'| < |\alpha|$ ;  $\alpha'$  este mai scurt decât  $\alpha$  cu cel mult  $n$  caractere ( $|\alpha| \leq n$ ), deci  $|\alpha'| \geq n$ .

Raționamentul se reia pentru  $\alpha'$  și – după un număr finit de pași – se ajunge la un cuvânt care verifică condițiile cerute. q.e.d.

Se poate construi un algoritm care să testeze dacă limbajul acceptat de un automat finit este finit sau nu:

**Algoritmul 2.1:**

**Intrare:** Un  $AFD$  :  $M = (Q, V, \delta, q_0, F)$ ;

**Ieșire:** DA dacă  $\tau(M)$  este infinit, NU altfel.

**Algoritm:**

1.  $i := 0$ ,  $T_i := \{q_0\}$ ;
2.  $i := i + 1$ ;  $T_i := \{p \mid \delta(q, a) = p, \quad q \in T_{i-1}, a \in V\}$ ;
3. Dacă  $i \neq n$ , salt la (2).
4. Dacă  $T_i \cap F \neq \emptyset$ , atunci DA, STOP.
5.  $i := i + 1$ ; dacă  $i = 2n$ , atunci NU, STOP.
6.  $T_i := \{p \mid \delta(q, a) = p, \quad q \in T_{i-1}, a \in V\}$ , salt la (4).

Algoritmul se bazează pe o idee diferită de cea folosită în demonstrația Propoziției 2.1. Cu primii trei pași se obțin toate stările la care poate ajunge  $AFD$ -ul după parcurgerea tuturor cuvintelor de lungime  $n$ .

În continuare, timp de  $n$  pași, se testează dacă în mulțimile astfel obținute s-a ajuns la o stare finală; dacă acest lucru este îndeplinit, înseamnă că există un cuvânt în limbajul  $\tau(M)$  având lungimea în intervalul  $[n, 2n)$ , deci  $\tau(M)$  este infinit.

**Exemplul 2.4** *Să aplicăm algoritmul de sus automatului definit în Exemplul 2.1.*

*Vom avea  $n = 4$  și se obține succesiv:*

*Pas 1 :  $i := 0$ ,  $T_0 := \{q_0\}$ .*

*Pas 2 :  $i := 1$ ,  $T_1 := \{q_1, q_3\}$ ; cum  $i \neq n$ , se reia pasul 2.*

*Pas 2 :  $i := 2$ ,  $T_2 := \{q_0, q_2\}$ ; la fel, pentru că  $2 \neq 4$ , revenim la pasul 2.*

*Pas 2 :  $i := 3$ ,  $T_3 := \{q_1, q_3\}$ .*

*Pas 2 :  $i := 4$ ,  $T_4 := \{q_0, q_2\}$ . Acum  $i = n$  și se trece mai departe.*

*Pas 4 :  $T_4 \cap F = \{q_0, q_2\} \cap \{q_0\} = \{q_0\} \neq \emptyset$ .*

*Deci  $\tau(M)$  este un limbaj conținând o infinitate de cuvinte.*

În mod similar se poate construi și un algoritm care să decidă posibilitatea ca un limbaj să fie vid sau nu. Lăsăm ca exercițiu construcția unui astfel de algoritm (Exercițiul 2.1).

Propoziția 2.1 rezolvă două probleme de decidabilitate. Pe baza ei spunem că problema vidării (*emptiness*) și problema finitudinii (*finitness*) sunt decidabile pentru limbajele regulate.

## 2.2 Teorema Myhill - Nerode și aplicații

### 2.2.1 Teorema de caracterizare algebrică a limbajelor regulate

Fie  $V$  un alfabet și  $L \subseteq V^*$  un limbaj peste  $V$ .

Pentru  $\alpha, \beta \in V^*$  definim relația

$$\alpha \equiv_L \beta \iff \forall \gamma \in V^* [\alpha\gamma \in L \iff \beta\gamma \in L]$$

În particular, pentru  $\gamma = \epsilon$  obținem  $\alpha \in L \iff \beta \in L$ .

**Definiția 2.2**  $S \subset V^*$  este o mulțime de bază pentru limbajul  $L$  dacă:

1.  $S$  este finită.
2.  $\forall w \in V^*$ , există  $\alpha \in S$  cu  $\alpha \equiv_L w$ .

În acest cadru, putem enunța

**Teorema 2.1** (Myhill - Nerode) *Un limbaj  $L$  este regulat dacă și numai dacă admite o mulțime de bază.*

*Demonstrație:*

" $\implies$ ": Fie  $L \subseteq V^*$  un limbaj regulat.

Atunci există un AFD  $M = (Q, V, \delta, q_0, F)$  cu  $\tau(M) = L$ , în care funcția  $\delta$  este extinsă în așa fel încât să fie total definită<sup>1</sup>.

Folosind Algoritmul 1.1. din Capitolul 1, putem presupune că toate stările din  $Q$  verifică proprietatea

$$\forall q \in Q, \exists \alpha_q \in V^* \text{ cu } \delta(q_0, \alpha_q) = q.$$

Evident, acest  $\alpha_q$  nu este unic.

Putem însă selecta pentru fiecare  $q \in Q$  un cuvânt arbitrar cu această proprietate, pe care îl vom nota  $\alpha_q$ .

Fie  $S = \{\alpha_q \mid q \in Q\}$ .

Cum  $|S| = |Q|$ , rezultă că  $S$  este o mulțime finită.

Pentru a arăta că  $S$  este mulțime de bază, să luăm un cuvânt arbitrar  $w \in V^*$  și să arătăm că există  $\alpha \in S$  cu  $w \equiv_L \alpha$ .

Fie  $\delta(q_0, w) = q$ . Atunci  $\alpha = \alpha_q \in S$ .

Din construcția lui  $S$ ,  $\delta(q_0, \alpha_q) = q$ .

Acum,  $\forall u \in V^*$ ,  $\delta(q_0, wu) = \delta(\delta(q_0, w), u) = \delta(q, u) = \delta(\delta(q_0, \alpha_q), u) = \delta(q_0, \alpha_q u)$ .

Deci  $\delta(q_0, wu) \in F \iff \delta(q_0, \alpha_q u) \in F$ , sau  $wu \in L \iff \alpha_q u \in L$ , adică  $w \equiv_L \alpha$ .

---

<sup>1</sup>Se adaugă eventual o stare nouă  $q_e$  numită de obicei "stare eroare" și pentru orice pereche  $(q, a) \in Q \times V$  pentru care  $\delta(q, a)$  nu era definită, se completează cu  $\delta(q, a) = q_e$ .

" $\Leftarrow$ ": Fie  $S \subset V^*$  o mulțime de bază pentru limbajul  $L$ .

Să notăm cu  $\phi$  aplicația canonică  $\phi : S \longrightarrow V^* / \equiv_L$ , definită

$$\phi(\alpha) = [\alpha]$$

(s-a notat  $[\alpha] = \{w \mid w \in V^*, \alpha \equiv_L w\}$ ).

În particular, pentru  $\epsilon \in V^*$ , există  $\alpha_0 \in S$  cu  $\epsilon \in \phi(\alpha_0)$ .

Vom construi un  $AFN$   $M = (Q, V, \delta, q_0, F)$  astfel:

$$Q = S, \quad q_0 = \alpha_0, \quad F = \{\alpha \mid \alpha \in S, \phi(\alpha) \cap L \neq \emptyset\},$$

și

$$\delta(\alpha, a) = \{\beta \mid \beta \in S, \alpha a \in \phi(\beta)\}, \quad \forall \alpha \in Q, a \in V.$$

### Observația 2.2

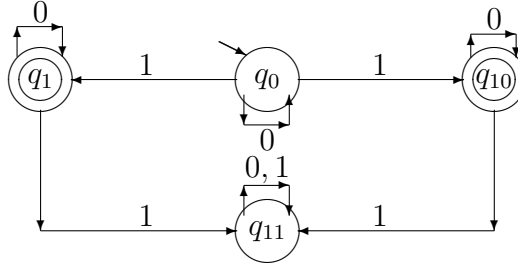
1. Deoarece  $S$  este o mulțime de bază,  $\phi$  va fi o aplicație surjectivă.
2. Aplicația  $\phi$  nu este în general injectivă; de aici rezultă că  $M$  este un automat nedeterminist.

De exemplu, pentru limbajul  $L = \{0^i 10^j \mid i, j \geq 0\}$ , o mulțime de bază este  $S = \{0, 1, 10, 11\}$ .

Vom avea

$$\phi(0) = L_1, \quad \phi(1) = \phi(10) = L, \quad \phi(11) = V^* \setminus (L \cup L_1), \quad \text{unde } L_1 = \{0^i \mid i \geq 0\}.$$

Evident,  $\phi$  nu este injectivă, iar automatul  $M$  este definit prin diagrama de tranziție



3.  $\phi(\alpha) \cap L \neq \emptyset \iff \phi(\alpha) \subseteq L$ .

Într-adevăr, dacă  $w \in \phi(\alpha) \cap L$ , atunci  $w \in L$  și  $w \equiv_L \alpha$ . Fie  $u \in \phi(\alpha)$ ; deci  $u \equiv_L \alpha \implies u \equiv_L w$ .

Va rezulta

$$u\gamma \in L \iff w\gamma \in L, \quad \forall \gamma \in V^*$$

Luând  $\gamma = \epsilon$ , avem  $u \in L \iff w \in L$ , deci  $u \in L$ .

Implicația inversă este banală.

Revenind la demonstrarea Teoremei 2.1, rămâne de arătat că  $\tau(M) = L$ .

Pentru aceasta, demonstrăm întâi prin inducție după  $k = |w|$  aserțiunea

$$\beta \in \delta(\alpha, w) \iff \alpha w \in \phi(\beta), \quad \forall \alpha, \beta \in Q, \quad w \in V^*.$$

$k = 0$ : asta înseamnă  $w = \epsilon$ ; atunci  $\delta(\alpha, \epsilon) = \{\alpha\}$  și  $\alpha \in \phi(\alpha)$  (evident).

Fie  $w \in V^*$  cu  $|w| = k + 1$ . Atunci  $w = ua$  cu  $|u| = k$ .

Conform definiției lui  $\phi$ , există  $\gamma \in Q$  cu  $\alpha u \in \phi(\gamma)$ .

Folosind ipoteza de inducție, avem  $\gamma \in \delta(\alpha, u)$ .

” $\Leftarrow$ ”: Cum  $\delta(\gamma, a) \neq \emptyset$  (prin ipoteză,  $\delta$  este total definită), putem alege un  $\beta \in \delta(\gamma, a)$ . Atunci  $\beta \in \delta(\gamma, a) \subset \delta(\delta(\alpha, u), a) = \delta(\alpha, w)$ .

” $\Rightarrow$ ”: Avem următoarele rezultate obținute până acum:

$\alpha u \in \phi(\gamma) \iff \alpha u \equiv_L \gamma$  (definiția aplicației canonice  $\phi$ )

$\beta \in \delta(\gamma, a) \iff \gamma a \equiv_L \beta$  (definiția lui  $\delta$ )

Cum relația  $\equiv_L$  este invariantă la dreapta (Exercițiul 2.6), avem  $\alpha u a \equiv_L \gamma a \equiv_L \beta$ .

Deci  $\alpha w \equiv_L \beta$ , adică  $\alpha w \in \phi(\beta)$ .

Să arătăm acum egalitatea  $\tau(M) = L$ .

” $\subseteq$ ”: Fie  $w \in \tau(M)$ ; atunci  $\delta(\alpha_0, w) \cap F \neq \emptyset$ .

Să luăm  $\beta \in \delta(\alpha_0, w) \cap F$ , arbitrar.

Din  $\beta \in \delta(\alpha_0, w)$  rezultă  $\alpha_0 w \in \phi(\beta)$ ; din  $\beta \in F$  avem  $\phi(\beta) \cap L \neq \emptyset$ , deci  $\phi(\beta) \subseteq L$ .

Combinând aceste două rezultate, se obține  $\alpha_0 w \in L$ .

Din  $\alpha_0 \equiv_L \epsilon$  rezultă  $\alpha_0 w \equiv_L w$ .

Cum  $\alpha_0 w \in L$ , vom avea și  $w \in L$ .

” $\supseteq$ ”: Fie  $w \in L$ . Similar ca mai sus, se obține și  $\alpha_0 w \in L$ .

Din definiția lui  $\phi$ , există un  $\beta \in S$  cu  $\alpha_0 w \in \phi(\beta)$ ; deci  $\beta \in \delta(\alpha_0, w)$ .

Cum  $\alpha_0 w \in \phi(\beta) \cap L$ , vom avea  $\beta \in F$ .

Deci  $\delta(\alpha_0, w) \cap F \neq \emptyset$ , adică  $w \in \tau(M)$ .

q.e.d.

### 2.2.2 Aplicații ale Teoremei Myhill-Nerode

Similar Lemei de pompare, și folosind Teorema 2.1, putem demonstra că unele limbaje nu sunt regulate.

Să exemplificăm acest lucru.

**Exemplul 2.5** Să arătăm că limbajul

$$L = \{\alpha \mid \alpha \in \{a, b\}^*, |\alpha|_a = |\alpha|_b\}$$

nu este regulat (s-a notat cu  $|\alpha|_X$  numărul de caractere  $X$  existente în secvența  $\alpha$ ).

Vom presupune prin absurd că există o mulțime de bază  $S$  pentru  $L$ ; fie aceasta  $S = \{\alpha_{i_1}, \dots, \alpha_{i_n}\}$  unde  $|\alpha_{i_k}|_a - |\alpha_{i_k}|_b = i_k$ ,  $i_k \in \mathbb{Z}$ ,  $(1 \leq k \leq n)$ .

Mulțimea  $\{i_1, \dots, i_n\}$  este finită; deci există  $i \in \mathbb{Z} \setminus \{i_1, \dots, i_n\}$ ,  $i > 0$ .

Atunci cuvântul  $\alpha \in V^*$  cu  $|\alpha|_a - |\alpha|_b = i$  nu este echivalent cu nici un cuvânt din  $S$ .

Într-adevăr, dacă există  $j$  cu  $\alpha_{i_j} \equiv_L \alpha$ , deci  $\alpha_{i_j}w \in L \iff \alpha w \in L$ , alegem  $w = b^i$ .

Vom avea  $|\alpha w|_a - |\alpha w|_b = 0$  (și  $\alpha w \in L$ ), dar  $|\alpha_{i_j}w|_a - |\alpha_{i_j}w|_b = i_j - i \neq 0$  (deci  $\alpha_{i_j} \notin L$ ).

Ca o remarcă, dacă încercăm să obținem acest rezultat folosind Lema de pompă, nu vom reuși, deoarece limbajul  $L$  verifică această lemă !

Din acest punct de vedere Teorema Myhill - Nerode completează Lema de pompă.

Un alt rezultat important obținut pe baza Teoremei Myhill - Nerode este următorul:

**Fiind dat un AFD  $M$ , se poate construi un AFD  $M'$  echivalent, având un număr minim de stări.**

Să demonstrăm acest lucru.

**Propoziția 2.2** *Aplicația  $\phi$  este injectivă dacă și numai dacă automatul  $M$  corespunzător are un număr minim de stări.*

*Demonstrație:* Fiind injectivă și surjectivă,  $\phi$  este o bijecție între  $S$  și  $V^*/\equiv_L$ .

Una din consecințele acestui rezultat va fi  $|S| = |V^*/\equiv_L| = n$ .

Să presupunem prin absurd că limbajul regulat  $L$  este acceptat de un automat  $M$  cu mai puțin de  $n$  stări.

Din demonstrația Teoremei 2.1 rezultă că atunci există o mulțime de bază  $S'$  cu  $|S'| = |Q| < n$ . Aplicația

$$\phi' : S' \longrightarrow V^*/\equiv_L$$

nu va mai fi surjectivă; va exista atunci  $w \in V^*$  cu  $w \notin \phi'(\alpha)$ ,  $\forall \alpha \in S'$ .

Deci  $w \not\equiv_L \alpha$ ,  $\forall \alpha \in S'$ , ceea ce contrazice definiția mulțimii de bază.

În consecință, numărul minim de stări ale unui automat  $M$  care acceptă limbajul regulat  $L$  va fi  $|V^*/\equiv_L|$ , tocmai cazul când  $\phi$  este o aplicație injectivă.

Pe mulțimea  $Q$  a stărilor unui AF vom introduce următoarea relație de echivalență:

$$p \equiv q \iff \alpha_p \equiv_L \alpha_q$$

Două stări  $p$  și  $q$  care nu sunt echivalente, se numesc *separabile*.



**Lema 2.2** *Un automat finit cu toate stările separabile este determinist.*

*Demonstrație:* Să presupunem că  $AF\ M = (Q, V, \delta, q_0, F)$  are stările separabile dar nu este determinist; deci  $\exists q, q_1, q_2 \in Q$  și  $a \in V$  cu  $q_1, q_2 \in \delta(q, a)$ ,  $q_1 \neq q_2$ .

Atunci  $\alpha_{q_1} \equiv_L \alpha_q$ ,  $\alpha_{q_2} \equiv_L \alpha_q$  (s-au folosit notațiile din demonstrația Teoremei 2.1), deci  $\alpha_{q_1} \equiv_L \alpha_{q_2}$ , contradicție pentru că  $q_1$  și  $q_2$  sunt separabile. q.e.d.

**Propoziția 2.3** *Fie  $M = (Q, V, \delta, q_0, F)$  un AFD cu stări echivalente. Atunci există un AFD  $M'$  echivalent cu el, având toate stările separabile.*

*Demonstrație:* Fie  $p_0, p_1 \in Q$  două stări echivalente.

Construim  $AFD\ M' = (Q', V, \delta', q_0, F')$  astfel:  $Q' = Q \setminus \{p_1\}$ ,  $F' = F \setminus \{p_1\}$ ,  $\delta'(q, a) = p_0$  dacă  $\delta(q, a) = p_1$ ,  $\delta'(q, a) = \delta(q, a)$  în rest.

Să arătăm că  $\tau(M) = \tau(M')$ .

” $\subseteq$ ”: Fie  $w \in \tau(M)$ ,  $w = a_1 \dots a_n$ .

Dacă  $\forall i\ (1 \leq i \leq n)$ ,  $\delta(q_0, a_1 \dots a_i) \neq p_1$ , atunci aceeași secvență de stări se va găsi și în automatul  $M'$ , deci  $w \in \tau(M')$ .

Fie  $i\ (1 \leq i \leq n)$  cu  $\delta(q_0, a_1 \dots a_i) = p_1$ ; putem presupune că  $i$  este minim cu această proprietate.

Din ipoteză, cum  $p_1 \equiv p_0$ , rezultă că există  $\alpha \in V^*$  cu  $\delta(q_0, \alpha) = p_0$  și  $a_1 \dots a_i \equiv_L \alpha$ .

Cum  $a_1 \dots a_n \in L$ , vom avea și  $\alpha a_{i+1} \dots a_n \in L$ , deci  $\delta(q_0, \alpha a_{i+1} \dots a_n) \in F$ , de unde rezultă  $\delta(p_0, a_{i+1} \dots a_n) \in F$ .

Din definiția lui  $M'$  avem  $\delta'(q_0, a_1 \dots a_i) = p_0$ .

Folosind acum un argument de inducție asupra lungimii cuvântului, se obține

$$\delta'(p_0, a_{i+1} \dots a_n) \in F$$

Deci  $\delta'(q_0, a_1 \dots a_n) \in F$ , adică  $w \in \tau(M')$ .

” $\supseteq$ ”: Fie  $w \in \tau(M')$ ,  $w = a_1 \dots a_n$ .

Considerăm vectorii  $v' = (i_1, \dots, i_n)$  și  $v = (j_1, \dots, j_n)$ , unde  $q_0, q_{i_1}, \dots, q_{i_n}$  reprezintă secvența de stări din automatul  $M'$  notată de caracterele  $a_1, \dots, a_n$ , iar  $q_0, q_{j_1}, \dots, q_{j_n}$  secvența similară de stări din  $M'$ .

Comparăm lexicografic vectorii  $v$  și  $v'$ . Dacă  $v = v'$ , atunci cele două drumuri sunt identice, deci  $w \in \tau(M)$ .

În caz contrar, fie  $k$  primul indice cu  $i_k \neq j_k$ .

Din construcția automatului  $M'$  rezultă  $q_{i_k} = p_0$ ,  $q_{j_k} = p_1$  și  $\delta(q_{i_{k-1}}, a_k) = p_1$ .

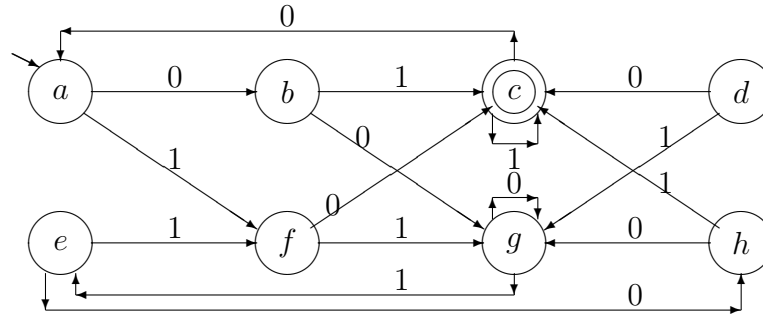
În  $M$ , stările  $p_0$  și  $p_1$  sunt echivalente, deci  $\delta(p_0, a_{k+1} \dots a_n) \in F \iff \delta(p_1, a_{k+1} \dots a_n) \in F$ ; de aici rezultă  $\delta(p_1, a_{k+1} \dots a_n) \in F$ , adică  $q_{j_n} \in F$  și  $\delta(q_0, a_1 \dots a_n) = q_{j_n}$ , ceea ce înseamnă  $w = a_1 \dots a_n \in \tau(M)$ .

Aplicăm această modalitate de eliminare a stărilor echivalente cât timp este posibil. Automatul obținut în final va avea proprietățile cerute. q.e.d.

Pe baza Propoziției 2.3 se poate construi un algoritm de aflare a AFD-ului cu număr minim de stări, echivalent cu un AFD dat  $M = (Q, V, \delta, q_0, F)$ .

Deoarece în acest caz o formalizare ar complica mult prezentarea, vom schița algoritmul pe un exemplu.

**Exemplul 2.6** Fie un AFD dat prin diagrama de tranziție



Vom construi o tabelă de stări, în care marcăm cu  $\times$  faptul că stările corespunzătoare sunt separabile.

b	$\times$							
c	$\times$	$\times$						
d	$\times$	$\times$	$\times$					
e		$\times$	$\times$	$\times$	$\times$			
f	$\times$	$\times$	$\times$	$\times$		$\times$		
g	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
h	$\times$		$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
	a	b	c	d	e	f	g	

Inițial,  $\times$  se plasează în căsuțele aflate la intersecția unei stări finale cu una nefinală (deci  $(a, c), (b, c), (c, d), (c, e), (c, f), (c, g), (c, h)$ ).

Apoi, pentru fiecare pereche  $(p, q)$  de stări despre care nu se știe dacă sunt separabile, luăm toate perechile de stări  $(r, s)$  unde  $r = \delta(p, a)$ ,  $s = \delta(q, a)$ ,  $\forall a \in V$ .

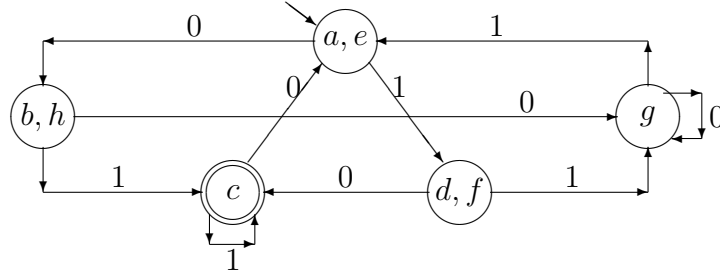
Dacă  $r$  și  $s$  sunt separabile, atunci  $p$  și  $q$  sunt separabile (și se pune  $\times$  în căsuța corespunzătoare lor).

Revenind la exemplu, pentru  $(a, b)$  avem  $(\delta(b, a), \delta(a, 1)) = (c, f)$ , separabilă. Deci se pune  $\times$  în locația  $(a, b)$ .

Se procedează astfel cât timp este posibil.

În final se obțin stările echivalente  $a \equiv e$ ,  $b \equiv h$ ,  $d \equiv f$ .

Folosind Propoziția 2.3, se ajunge la automatul



Corectitudinea acestui algoritm este dată de

**Propoziția 2.4** Fie  $M = (Q, V, \delta, q_0, F)$  un AFD. Atunci stările  $p$  și  $q$  sunt separabile dacă și numai dacă poziția din tabel a lui  $(p, q)$  este marcată.

*Demonstrație:*

” $\implies$ ”: Fie  $p, q \in Q$  două stări separabile și  $\alpha \in V^*$  cel mai scurt drum care le face separabile ( $\delta(p, \alpha) \in F$  și  $\delta(q, \alpha) \notin F$ , sau invers).

Vom arăta – prin inducție după lungimea lui  $\alpha$  – că poziția din tabel a lui  $(p, q)$  este marcată.

Dacă  $\alpha = \epsilon$ , atunci una și numai una din stările  $p, q$  este finală, și, conform construcției tabelului, se face marcarea.

Presupunem afirmația adevărată pentru  $|\alpha| < i$ ,  $\forall i \geq 1$ , și să considerăm  $|\alpha| = i$ .

Atunci  $\alpha = a\beta$  și fie  $t = \delta(p, a)$ ,  $u = \delta(q, a)$ .

Stările  $t$  și  $u$  sunt distinse prin  $\beta$  ( $\delta(t, \beta) = \delta(p, \alpha)$ ,  $\delta(u, \beta) = \delta(q, \alpha)$ ) și  $|\beta| = i - 1$ .

Conform ipotezei de inducție,  $(t, u)$  este marcată și, folosind regula de marcare, marcăm și  $(p, q)$ .

” $\Leftarrow$ ”: Se face o inducție similară după numărul de perechi marcate. q.e.d.

Dacă renunțăm la condiția ca automatul  $M$  să aibă funcția de tranziție total definită, este posibil să mai micșorăm numărul stărilor sale, renunțând la stările inaccesibile (cu Algoritmul 1.1 – Capitolul 1) și la stările care nu sunt legate de stări finale prin nici un drum (cum sunt de exemplu stările eroare).

Acest ultim tip de eliminare se bazează pe rezultatul următor:

**Lema 2.3** Fie  $M = (Q, V, \delta, q_0, F)$  un AFD cu toate stările accesibile. Există atunci un AFD  $M'$  echivalent, cu proprietatea

$$\forall q \in Q, \exists \alpha \in V^* \text{ cu } \delta(q, \alpha) \in F.$$

*Demonstrație:* Pentru fiecare stare  $q_k \in Q$  se construiește automatul  $M_k = (Q, V, \delta, q_k, F)$ . Cu ajutorul Propoziției 2.1 putem decide dacă  $\tau(M_k) \neq \emptyset$ .

Automatul căutat este  $M' = (Q', V, \delta', q_0, F)$ , unde

$$Q' = \{q_k \mid q_k \in Q \setminus F, \tau(M_k) \neq \emptyset\} \cup F, \quad \delta' = \delta|_{Q' \times V}.$$

Egalitatea  $\tau(M') = \tau(M)$  este evidentă dacă se folosesc grafurile de tranziție ale celor două automate.

**Exemplul 2.7** Fie AFD definit prin graful de tranziție

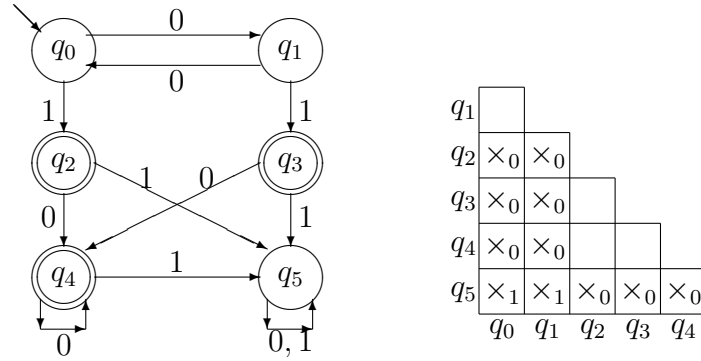
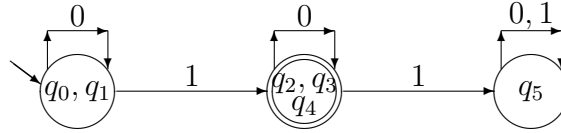


Tabela pentru construcția AFD-ului cu număr minim de stări este reprezentată în dreapta.

Am notat cu  $\times_0$  marcarea inițială și cu  $\times_1$  marcarea la prima trecere prin tabel.

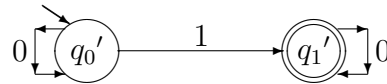
La a doua trecere nu s-a mai efectuat nici o marcarea.

Deci  $q_0 \equiv q_1$ ,  $q_2 \equiv q_3 \equiv q_4$ . Graful de tranziție al noului automat este



Acesta nu are însă efectiv un număr minim de stări, deoarece  $\tau(M_5) = \emptyset$ ; deci starea  $q_5$  – împreună cu toate arcele legate de ea – se poate elimina.

În final se obține automatul de mai jos (unde stările au fost renotate):



**Observația 2.3** Teorema Myhill - Nerode completează Lema de pompare. Astfel, valoarea minimă lui  $n$  din Lema de pompare este  $n_0 = |Q|$ , unde  $Q$  este mulțimea stărilor automatului minimal construit pe baza Teoremei Myhill - Nerode.

## 2.3 Exerciții

**Exercițiul 2.1** Să se construiască un algoritm care să decidă dacă  $\tau(M) = \emptyset$ ,  $M$  fiind un automat finit determinist.

**Exercițiul 2.2** Să se arate că problema apartenenței este decidabilă pentru limbajele regulate: se poate construi un algoritm care, pentru un limbaj regulat  $L \subseteq V^*$  și un cuvânt  $w \in V^*$ , poate decide dacă  $w \in L$ .

**Exercițiul 2.3** Să se demonstreze Lema de pompare cu variantele (1') și (1'') din Observația 2.1.

**Exercițiul 2.4** Să se arate că limbajele  $\{a^n \mid n \text{ prim}\}$  și  $\{a^n b^{2n} \mid n \geq 0\}$  nu sunt regulate.

**Exercițiul 2.5** Să se arate că limbajul  $L = \{\alpha \mid \alpha \in \{a, b\}^*, |\alpha|_a = |\alpha|_b\}$  verifică Lema de pompare.

**Exercițiul 2.6** Să se arate că  $\equiv_L$  este o relație de echivalență, invariantă la dreapta.

**Exercițiul 2.7** Fie  $S$  o mulțime de bază și  $x, y \in S$  cu  $x \equiv_L y$ . Să se arate că  $S' = S \setminus \{x\}$  este tot o mulțime de bază.

**Exercițiul 2.8** Să se construiască mulțimi de bază pentru limbajele  

$$L_1 = \{a^n b^m c^p \mid m, n, p \geq 1\}, \quad L_2 = \{\alpha 00\beta \mid \alpha, \beta \in \{0, 1\}^*\}$$

**Exercițiul 2.9** Folosind Teorema Myhill - Nerode, să se arate că limbajele definite în Exercițiul 2.4 nu sunt regulate.

**Exercițiul 2.10** Să se formalizeze algoritmul utilizat în demonstrația Propoziției 2.3.

# Capitolul 3

## Gramatici clasificabile Chomsky

### 3.1 Sisteme de rescriere

Fie  $V$  un alfabet finit. După cum am văzut, orice submulțime  $L \subseteq V^*$  se numește limbaj.

În primele două capitole am dezvoltat un studiu al modalității de definire și de recunoaștere pentru limbajele regulate.

În acest capitol vom extinde acest studiu pentru o clasă arbitrară de limbaje.

Un limbaj este *finit* dacă are un număr finit de cuvinte; altfel, limbajul se consideră *infinit*.

Pentru limbajele finite, o manieră des folosită de definire constă în listarea tuturor cuvintelor care le conțin.

Aceasta nu este bineînțeles posibil totdeauna.

De exemplu, limbajul tuturor secvențelor binare de lungime 10 cuprinde  $2^{10} = 1024$  cuvinte; listarea lor este mult mai complicată decât am scrie

$$L = \{\alpha \mid \alpha \in \{0,1\}^*, |\alpha| = 10\}.$$

Aici dificultatea a constat în numărul de cuvinte care – deși finit – este foarte mare.

Dar aceasta nu este singura dificultate ! Astfel, să construim limbajul următor:

Fie

$$W = \{(m, n, p, k) \mid m, n, p, k \in \mathcal{N}, k > 2, m^k + n^k = p^k\} \quad \text{și}$$

$$L = \{x \mid x = 1 \text{ dacă } W \neq \emptyset, x = 0 \text{ dacă } W = \emptyset\}.$$

$L$  este un limbaj finit, cu un singur element. Dar acest element nu a putut fi specificat decât în 1995 când a fost rezolvată marea teoremă a lui Fermat !

Deci și pentru limbajele finite apar probleme în reprezentare, nu numai pentru limbajele infinite.

În capitolele anterioare am folosit două modalități de definire a limbajelor:

1. Definirea cu ajutorul mulțimilor: se specifică o secvență de proprietăți, după care limbajul asociat este considerat ca fiind mulțimea tuturor cuvintelor care satisfac acele proprietăți.

Este o metodă folosită aproape universal, toate celelalte maniere de definire făcând referire la ea.

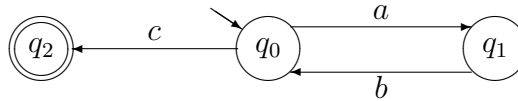
### Exemplul 3.1

$$\begin{aligned} L_1 &= \emptyset, & L_2 &= \{a^n \mid n \text{ prim}\}, \\ L_3 &= \{\alpha^{|\alpha|} \mid \alpha \in \{a, b\}^*\}, & L_4 &= \{\alpha \mid \alpha \in \{a, b\}^*, |\alpha|_a = |\alpha|_b\}. \end{aligned}$$

2. Metodă de recunoaștere: constă dintr-un procedeu care – aplicat unui cuvânt – îl acceptă dacă acesta este în limbaj, sau îl respinge – dacă nu este.

Automatele finite sunt o astfel de manieră de definire a unei clase de limbaje: limbajele regulate.

**Exemplul 3.2** Fie limbajul regulat  $L$  acceptat de AFD-ul:



Să cercetăm dacă un cuvânt fixat  $\alpha$  este sau nu în limbajul  $L$ ; algoritmul aplicat pentru a decide acest fapt este cel din Capitolul 1: testăm dacă  $\delta(q_0, \alpha) \in F$ ; în caz afirmativ,  $\alpha \in L$ , altfel răspunsul este  $\alpha \notin L$ .

De exemplu, pentru  $\alpha = (ab)^2c$  se obține  $\delta(q_0, (ab)^2c) = q_2 \in F$ , deci  $(ab)^2c \in L$ .

În schimb, pentru  $\alpha = aba$ ,  $\delta(q_0, aba) = q_1 \notin F$ , deci  $aba \notin L$ .

La fel, pentru  $\alpha = abb$ ,  $\delta(q_0, abb) = \delta(q_0, b) = \text{nedefinit}$ , deci  $abb \notin L$ .

**Exemplul 3.3** Fie  $L$  un limbaj definit în felul următor:

" $\alpha \in L \iff \alpha$  poate fi redus la  $\epsilon$  printr-un șir de înlocuiri succesive ale lui  $ab$  cu  $\epsilon$ ."

Deci, în particular,  $\alpha = aabbab \in L$  deoarece – aplicând regula dată – se obține pe rând:  $aabbab \rightarrow abab \rightarrow ab \rightarrow \epsilon$ .

Similar,  $\epsilon$ ,  $aabb$ ,  $a(ba)^5b$  sunt cuvinte din limbaj.

În schimb  $\alpha = baabba$  nu este în  $L$ , pentru că reducerea nu este totală:  $baabba \rightarrow baba \rightarrow ba$  și regula de reducere nu se mai poate aplica.

În afară de aceste modalități de definire a limbajelor, foarte des folosită este și

3. *Metoda generativă*: este o metodă care, folosind ca bază un set de axiome, efectuează generarea recursivă a limbajului.

**Exemplul 3.4** Fie următoarea definire recursivă a limbajului  $L$ :

1.  $\epsilon \in L$ ;
2. Dacă  $\alpha \in L$  atunci  $a\alpha b, b\alpha a \in L$ ;
3. Dacă  $\alpha, \beta \in L$  atunci  $\alpha\beta \in L$ ;
4.  $L$  nu conține alte cuvinte.

Să arătăm că  $L = L_4$  ( $L_4$  dat în Exemplul 3.1).

" $\subseteq$ ":  $\epsilon$  are proprietatea cerută de  $L_4$ . Cu regulile (2) și (3) se obțin numai cuvinte în care 'a' și 'b' apar de un număr egal de ori (deci cuvinte din  $L_4$ ).

" $\supseteq$ ": Prin inducție după lungimea lui  $\alpha$ :

Dacă  $|\alpha| = 0$ , atunci  $\alpha = \epsilon$  care – conform cu (1) – este în  $L$ .

Presupunem acum că  $L$  conține toate cuvintele de lungime cel mult  $2k$ , cu un număr egal de 'a' și 'b', și fie  $\alpha \in L_4$  cu  $|\alpha| = 2(k+1)$ .

Apar patru posibilități:

$$\alpha = a\alpha'b, \quad \alpha = b\alpha'a, \quad \alpha = a\alpha''a, \quad \alpha = b\alpha''b.$$

Primele două cazuri sunt similare și vom trata numai unul din ele.

La fel ultimele două cazuri.

- $\boxed{\alpha = a\alpha'b}$ :

Aici  $|\alpha'|_a = |\alpha|_a - 1 = k$ ,  $|\alpha'|_b = |\alpha|_b - 1 = k$ , deci  $\alpha' \in L_4$ .

Ipoteza de inducție asigură  $\alpha' \in L$ .

Utilizând acum axioma (2) de construcție a lui  $L$ , obținem  $a\alpha'b = \alpha \in L$ .

- $\boxed{\alpha = a\alpha''a}$ :

Fie funcția  $f : V^* \longrightarrow \mathbb{Z}$  definită  $f(w) = |w|_a - |w|_b$ .

Evident,  $f(xy) = f(x) + f(y)$ .

În cazul nostru, să considerăm  $\alpha = a_1a_2 \dots a_{2k+1}a_{2k+2}$  unde  $a_1 = a_{2k+2} = a$ .

Din ipoteză, vom avea  $f(a_1a_2 \dots a_{2k+1}a_{2k+2}) = 0$ .

În plus,  $\forall n$  ( $1 \leq n \leq 2k+1$ ),  $f(a_1 \dots a_{n+1}) = f(a_1 \dots a_n) \pm 1$ , pentru că  $f(a_{n+1}) = 1$  dacă  $a_{n+1} = a$ ,  $f(a_{n+1}) = 1$  dacă  $a_{n+1} = b$ .



Această proprietate de "continuitate" a lui  $f$ , la care se adaugă

$$f(a_1) = 1, f(a_1 \dots a_{2k+1}) = -1,$$

duce la concluzia că există cuvintele  $\alpha_1, \alpha_2 \in V^*$  cu  $\alpha = \alpha_1 \alpha_2$  și  $f(\alpha_1) = 0$ .

Deci, cum  $f(\alpha) = 0$ , vom avea și  $f(\alpha_2) = 0$ , ceea ce înseamnă că  $\alpha_1 \alpha_2 \in L_4$ , cu  $|\alpha_1| \leq 2k$ ,  $|\alpha_2| \leq 2k$ .

Ipoteza de inducție asigură că  $\alpha_1, \alpha_2 \in L$ , iar cu axioma (3) din definirea lui  $L$ ,  $\alpha = \alpha_1 \alpha_2 \in L$ .

Dintre cele trei modalități de descriere a limbajelor, ne vom îndrepta atenția asupra ultimelor două procedee.

Definirea cu ajutorul mulțimilor este o definire amorfă, statică, foarte utilă pentru referințe, nu și pentru aplicarea în practică, pentru lucrul efectiv cu limbaje.

Ambele maniere de definire a limbajelor (generativă și de recunoaștere) au o trăsătură fundamentală comună: sunt proceduri în care cuvintele se obțin unele din altele prin *rescrierea* anumitor subcuvinte.

Astfel, în Exemplul 3.3, fiecare subcuvânt  $ab$  era rescris  $\epsilon$ ; în Exemplul 3.4, fiecare  $\alpha$  era rescris  $\epsilon$ ,  $aab$ ,  $baa$  sau  $\alpha\alpha$ .

Să formalizăm această observație:

**Definiția 3.1** *Un sistem de rescriere este o pereche ordonată  $SR = (W, P)$  unde  $W$  este un alfabet finit și nevid, iar  $P \subset W^* \times W^*$  este o mulțime finită de perechi de cuvinte, numite reguli de rescriere.*

**Definiția 3.2** *Fie  $SR = (W, P)$  un sistem de rescriere și  $\alpha, \beta \in W^*$ .*

- $\alpha$  generează direct  $\beta$  și scriem  $\alpha \Rightarrow \beta$  dacă și numai dacă  $\exists \alpha_1, \beta_1, \gamma_1, \gamma_2 \in W^*$  astfel încât  $\alpha = \gamma_1 \alpha_1 \gamma_2$ ,  $\beta = \gamma_1 \beta_1 \gamma_2$  și  $(\alpha_1, \beta_1) \in P$ .
  - $\alpha$  generează  $\beta$  și scriem  $\alpha \xRightarrow{*} \beta$  dacă și numai dacă  $\exists \alpha_0, \alpha_1, \dots, \alpha_k \in W^*$  ( $k \geq 0$ ) astfel încât  $\alpha_0 = \alpha$ ,  $\alpha_k = \beta$  și  $\alpha_i \Rightarrow \alpha_{i+1}$  ( $0 \leq i \leq k-1$ ).
- Secvența  $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \dots \Rightarrow \alpha_k$  se numește "derivare" a lui  $\alpha$ .*

Deci  $\Rightarrow$  este o relație binară pe  $W^*$ , iar  $\xRightarrow{*}$  este închiderea ei reflexivă și tranzitivă<sup>1</sup>.

<sup>1</sup>O relație  $\tau^*$  este închiderea reflexiv-tranzitivă a relației  $\tau$  pe mulțimea  $A$  dacă și numai dacă sunt verificate condițiile:

1.  $x\tau^*x$ ,  $\forall x \in A$ ;
2.  $x\tau^*y \ \& \ y\tau z \implies x\tau^*z$ ;
3. Relația  $x\tau^*y$  poate fi stabilită numai prin aplicarea condițiilor (1) și (2) de un număr finit de ori.

**Exemplul 3.5** Fie sistemul de rescriere  $SR = (W, P)$  unde  $W = \{a, b, c\}$  și  $P = \{(ba, ab), (ca, ac), (cb, bc)\}$ .

Să construim câteva derivări în acest context.

Din  $\alpha = bacb$  se pot efectua două generări directe:

1.  $\underline{bacb} \Rightarrow abcb$  (s-a folosit notația  $\beta = abcb$ ,  $\gamma_1 = \epsilon$ ,  $\gamma_2 = cb$ ,  $\alpha_1 = ba$ ,  $\beta_1 = ab$ )
2.  $\underline{bacb} \Rightarrow babc$  (aici  $\gamma_1 = ba$ ,  $\gamma_2 = \epsilon$ ,  $\alpha_1 = cb$ ,  $\beta_1 = bc$ ).

Deci cuvântul  $bacb$  poate genera direct atât pe  $abcb$  cât și pe  $babc$ .

Să construim acum toate derivările posibile plecând de la cuvântul  $\alpha = ccbca$ .

Vom avea

$$\begin{aligned}
 ccbca &\Rightarrow cbcca \Rightarrow bccca \Rightarrow bccac \Rightarrow bcacc \Rightarrow baccc \Rightarrow abccc \\
 ccbca &\Rightarrow cbcca \Rightarrow cbcac \Rightarrow bccac \Rightarrow bcacc \Rightarrow baccc \Rightarrow abccc \\
 ccbca &\Rightarrow cbcca \Rightarrow cbcac \Rightarrow cbacc \Rightarrow bcacc \Rightarrow baccc \Rightarrow abccc \\
 ccbca &\Rightarrow cbcca \Rightarrow cbcac \Rightarrow cbacc \Rightarrow cabcc \Rightarrow acbcc \Rightarrow abccc \\
 ccbca &\Rightarrow ccbac \Rightarrow cbcac \Rightarrow bccac \Rightarrow bcacc \Rightarrow baccc \Rightarrow abccc \\
 ccbca &\Rightarrow ccbac \Rightarrow cbcac \Rightarrow cbacc \Rightarrow bcacc \Rightarrow baccc \Rightarrow abccc \\
 ccbca &\Rightarrow ccbac \Rightarrow cbcac \Rightarrow cbacc \Rightarrow cabcc \Rightarrow acbcc \Rightarrow abccc \\
 ccbca &\Rightarrow ccbac \Rightarrow ccabc \Rightarrow cacbc \Rightarrow cabcc \Rightarrow acbcc \Rightarrow abccc \\
 ccbca &\Rightarrow ccbac \Rightarrow ccabc \Rightarrow cacbc \Rightarrow accbc \Rightarrow acbcc \Rightarrow abccc
 \end{aligned}$$

Acest sistem de rescriere efectuează de fapt o ordonare lexicografică a caracterelor cuvântului  $\alpha$ . Formal,  $\forall \alpha \in W^*$  cu  $|\alpha|_a = i$ ,  $|\alpha|_b = j$ ,  $|\alpha|_c = k$ , este posibilă derivarea  $\alpha \xRightarrow{*} a^i b^j c^k$ .

**Exemplul 3.6** Fie sistemul de rescriere  $SR = (W, P)$  cu  $W = \{a, A, B, \cdot\}$  și regulile  $P = \{(\underbrace{Ba, aB}_1), (\underbrace{Aa, aBA}_2), (\underbrace{A, \epsilon}_3), (\underbrace{a \cdot, \cdot A}_4), (\underbrace{\cdot a, \cdot}_5), (\underbrace{\cdot, \epsilon}_6), (\underbrace{B, a}_7)\}$ .

Să arătăm că este posibilă derivarea  $a^i \cdot a^j \xRightarrow{*} a^{ij}$ ,  $\forall i, j \geq 0$ .

Pentru ușurința referirii, regulile de rescriere au fost numerotate.

Dacă  $i = 0$ , avem  $\underbrace{a^j \xRightarrow{*}}_5 \cdot \xRightarrow{*}_6 \epsilon = a^0 = a^{0 \cdot j}$

Similar, pentru  $j = 0$ :  $a^i \cdot \xRightarrow{*}_4 A^i \xRightarrow{*}_6 A^i \xRightarrow{*}_3 \epsilon = a^0 = a^{i \cdot 0}$

Să presupunem  $i > 0$ ,  $j > 0$ . Atunci vom avea

$$a^i \cdot a^j \xRightarrow{*}_4 a^{i-1} \cdot Aa^j \xRightarrow{*}_2 a^{i-1} \cdot aBAa^{j-1} \xRightarrow{*}_2 a^{i-1} \cdot (aB)^j A \xRightarrow{*}_3 a^{i-1} \cdot (aB)^j \xRightarrow{*}_1 a^{i-1} \cdot a^j B^j$$

Procedeul este reluat pentru cuvântul  $a^{i-1} \cdot a^j$  și se obține similar  $a^{i-2} \cdot a^j B^j$ . Deci  $a^i \cdot a^j \xRightarrow{*} a^{i-2} \cdot a^j B^{2j}$ . Repetând procedeul până la epuizarea a-urilor din fața punctului, se ajunge la

$$a^i \cdot a^j \xRightarrow{*} \cdot a^j B^{ij} \xRightarrow{*}_5 \cdot B^{ij} \xRightarrow{*}_6 B^{ij} \xRightarrow{*}_7 a^{ij}$$

**Definiția 3.3** Se numește metodă generativă a limbajului  $L$  structura  $M_g(L) = (SR, AX, X)$  unde:

- $SR = (W, P)$  este un sistem de rescriere;
- $AX \subset W^*$  este o mulțime finită de elemente numite "axiome";
- $X \subset W$  este o mulțime numită "mulțime de bază",  $AX \cap X = \emptyset$ ;
- $L = \{\alpha \mid \alpha \in X^*, \exists \beta \in AX, \beta \xRightarrow{*} \alpha\}$ .

**Definiția 3.4** Se numește metodă de recunoaștere a limbajului  $L$  structura  $M_r(L) = (SR, AX, X)$ , unde  $SR, AX$  și  $X$  au aceeași semnificație ca mai sus, iar

$$L = \{\alpha \mid \alpha \in X^*, \exists \beta \in AX, \alpha \xRightarrow{*} \beta\}$$

**Exemplul 3.7** Să construim o metodă de recunoaștere care să accepte toate numerele naturale divizibile cu 3.

Ideea construcției se bazează pe criteriul de divizibilitate cu 3: vom transforma fiecare număr în restul său modulo 3. Dacă în final se obține  $\epsilon$ , numărul va fi acceptat.

Deci  $W = \{0, 1, 2, \dots, 9\}$ ,  $X = W$ ,  $AX = \{\epsilon\}$ ,  
 $P = \{(0, \epsilon), (3, \epsilon), (4, 1), (5, 2), (6, \epsilon), (7, 1), (8, 2), (9, \epsilon), (11, 2), (12, \epsilon), (21, \epsilon), (22, 1)\}$ .

De exemplu,  $265 \Rightarrow 2\bar{5} \Rightarrow \underline{22} \Rightarrow 1$ , deci 265 nu se divide cu 3.

În schimb  $8457 \Rightarrow 24\bar{5}7 \Rightarrow \underline{21}57 \Rightarrow \bar{5}7 \Rightarrow 2\bar{7} \Rightarrow \underline{21} \Rightarrow \epsilon$ , deci 8457 este un număr divizibil cu 3.

### Exemplul 3.8

- Limbajul definit în Exemplul 3.3 poate fi scris ca o metodă de recunoaștere astfel:  
 $W = \{a, b\}$ ,  $P = \{(ab, \epsilon)\}$ ,  $AX = \{\epsilon\}$ ,  $X = W$ .
- Limbajul  $L_4$  (tratat și în Exemplul 3.4) poate fi definit prin metoda generativă:  
 $W = \{a, b, c\}$ ,  $P = \{(x, \epsilon), (x, axb), (x, bxa), (x, xx)\}$ ,  $AX = \{x\}$ ,  $X = \{a, b\}$ .

**Propoziția 3.1** AF sunt metode de recunoaștere a limbajelor regulate.

*Demonstrație:* Fie  $M = (Q, V, \delta, q_0, F)$  un AFD cu  $L(M) = L$ ; fără a micșora generalitatea putem presupune  $Q \cap V = \emptyset$ . Construim  $M_r(L_1)$  astfel:

$$W = Q \cup V, \quad P = \{(pa, q) \mid \delta(p, a) = q\} \cup \{(a, q_0a) \mid a \in V\}, \quad AX = F, \quad X = V.$$

Mai rămâne de arătat că  $L = L_1$ , unde – conform Definiției 3.4,

$$L_1 = \{\alpha \mid \alpha \in V^*, \exists q \in F \text{ cu } \alpha \xRightarrow{*} q\}.$$

" $\subseteq$ ": Fie  $\alpha \in L$ ; deci  $\alpha = a_1a_2 \dots a_k$  și există o secvență de stări  $q_0, q_1, \dots, q_k$  astfel încât  $q_k \in F$  și  $\delta(q_i, a_{i+1}) = q_{i+1}$  ( $0 \leq i \leq k-1$ ).

Din construcția lui  $M_r(L_1)$  avem  $(q_i a_{i+1}, q_{i+1}) \in P$  ( $0 \leq i \leq k-1$ ). În plus,  $(a_1, q_0 a_1) \in P$ . Deci avem generarea

$$\alpha = a_1 a_2 \dots a_k \implies q_0 a_1 a_2 \dots a_k \implies \dots \implies q_{k-1} a_k \implies q_k$$

de unde rezultă concluzia  $\alpha \in L_1$ .

" $\supseteq$ ": Fie  $\alpha \in L_1$ ; deci există  $q \in AX$  cu  $\alpha \xRightarrow{*} q$ .

Primul pas al generării folosește obligatoriu o regulă de rescriere de forma  $(a, q_0 a)$ , care introduce  $q_0$  în fața lui  $\alpha$ . Orice inserare a lui  $q_0$  cu o astfel de regulă în interiorul cuvântului conduce la un blocaj, deoarece cuvintele de forma  $\beta q \gamma$  cu  $\beta \neq \epsilon$ , nu pot fi reduse la o singură stare, ci – în cel mai bun caz – la o secvență de minimum două stări. Din același motiv, o astfel de regulă se aplică o singură dată, la prima generare directă.

În continuare, raționamentul este similar celui de la implicația inversă. q.e.d.

Pentru  $AFN$ -uri modificările din demonstrație sunt minore.

## 3.2 Gramatici, clasificarea Chomski

Așa cum – în general – metodele de recunoaștere pentru diverse clase de limbaje sunt automatele sau modelări similare, cele mai cunoscute metode generative utilizate în teoria limbajelor formale sunt *gramaticile*.

**Definiția 3.5** *Se numește gramatică o metodă generativă  $M_g(L) = (SR, AX, X)$  cu următoarele proprietăți:*

- $W$  este partiționat în două mulțimi nevide  $V_N$  și  $V_T$ , umite "alfabet de neterminale" și respectiv "alfabet de terminale";
- $X = V_T$ ;
- $\exists S \in V_N$  și  $AX = \{S\}$ ;
- $\forall (\alpha, \beta) \in P$ ,  $\alpha$  este un cuvânt nevid care conține cel puțin un element din  $V_N$ .

**Observația 3.1** *O gramatică se notează de obicei  $G = (V_N, V_T, S, P)$ .*

*Noțiunile de derivare, generare directă, generare, se păstrează.*

*O componentă  $\alpha_i$  a unei derivări se numește "formă sentențială".*

*$S$  mai poartă numele de "simbol de start".*

*$(\alpha, \beta) \in P$  se notează  $\alpha \longrightarrow \beta$  și se numește "regulă de producție" sau – mai simplu – "producție".*

*$L = L(G) = \{w \mid w \in V_T^*, S \xRightarrow{*} w\}$  se numește "limbajul generat" de gramatica  $G$ .*

Prin convenție – dacă nu există specificații speciale – elementele din  $V_N$  (numite și *atribute*) se notează cu litere latine mari ( $A, B, C, \dots$ ), iar cele din  $V_T$  cu litere latine mici de la începutul alfabetului ( $a, b, c, \dots$ ). Cuvintele (elementele lui  $W^*$ ) se notează cu litere latine mici de la sfârșitul alfabetului ( $w, x, y, \dots$ ) sau cu litere grecești mici ( $\alpha, \beta, \gamma, \dots$ ). Eventual literele pot fi indexate.

**Exemplul 3.9** Să considerăm gramatica  $G = (V_N, V_T, S, P)$  în care

$$V_N = \{S\}, \quad V_T = \{a, b\}, \quad P = \{S \rightarrow aSb | \epsilon\}$$

(adesea se utilizează notația  $\alpha \rightarrow \beta_1 | \beta_2 | \dots | \beta_n$  ca o prescurtare pentru setul de producții  $\alpha \rightarrow \beta_1, \alpha \rightarrow \beta_2, \dots, \alpha \rightarrow \beta_n$ ; caracterul  $|$  este un metasimbol care nu apare în  $W$ , folosit frecvent drept separator cu rol de alternativă).

Conform definiției, limbajul generat este  $L = \{\alpha \mid \alpha \in \{a, b\}^*, S \xRightarrow{*} \alpha\}$ .

Să studiem proprietățile unei derivări în această gramatică.

Fie o derivare de lungime  $n$  ( $n \geq 1$  arbitrar fixat)

$$s = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_n = \alpha$$

- Toate formele sentențiale  $\alpha_i$  (în afară de ultima) sunt cuvinte formate dintr-o singură apariție a neterminalului  $S$  (producția  $S \rightarrow aSb$  nu introduce mai multe neterminale decât erau anterior), însoțite de apariții ale terminalelor 'a' și 'b'.
- Generarea  $\alpha_i \Rightarrow \alpha_{i+1}$ , ( $0 \leq i \leq n-2$ ) se face numai prin folosirea producției  $S \rightarrow aSb$ . Cu regula  $S \rightarrow \epsilon$  se șterge singurul neterminal existent în formele sentențiale, ajungându-se la un cuvânt din  $V_T^*$  (deci din  $L$ ). Rezultă că  $S \rightarrow \epsilon$  este ultima producție utilizată în derivare.

Cu aceste observații tragem concluzia că este posibilă o singură derivare de lungime  $n$ :

$$\begin{aligned} S &\Rightarrow aSb \Rightarrow a^2Sb^2 \Rightarrow \dots \Rightarrow a^{n-1}Sb^{n-1} \Rightarrow a^{n-1}b^{n-1} & \text{dacă } n > 1, \\ S &\Rightarrow \epsilon & \text{dacă } n = 1. \end{aligned}$$

Deci

$$L = L(G) = \{a^n b^n \mid n \geq 0\}.$$

**Exemplul 3.10** Limbajul generat de gramatica  $G = (\{S\}, \{a\}, S, \{S \rightarrow aS\})$  este  $L(G) = \emptyset$ . Aceasta se datorește observației că prin singura producție existentă ( $S \rightarrow aS$ ) neterminalul  $S$  se păstrează permanent, deci nu se ajunge niciodată la un cuvânt  $w \in V_T^*$  obținut prin derivarea  $S \xRightarrow{*} w$ .

**Exemplul 3.11** Să considerăm mulțimea

$$L = \{a^n b^n c^n \mid n \geq 1\}$$

O gramatică pentru generarea ei este  $G = (V_N, V_T, S, P)$  unde

$$V_N = \{S, B, C\}, \quad V_T = \{a, b, c\} \text{ și}$$

$$P = \begin{cases} 1. S \longrightarrow aSBC & 3. CB \longrightarrow BC & 5. bB \longrightarrow bb & 7. cC \longrightarrow cc \\ 2. S \longrightarrow aBC & 4. aB \longrightarrow ab & 6. bC \longrightarrow bc \end{cases}$$

(producțiile au fost numerotate pentru a ne referi mai ușor la ele).

Să arătăm că  $L = L(G)$ .

" $\subseteq$ ": Fie  $a^n b^n c^n \in L$ ; aplicând regula 1. de  $n - 1$  ori, obținem  $S \xRightarrow{*} a^{n-1} S(BC)^{n-1}$ . Apoi, cu producția 2.,  $S \xRightarrow{*} a^n S(BC)^n$ ; cu 3. neterminalele  $B$  și  $C$  se rearanjează în așa fel încât toate  $B$ -urile să precedă  $C$ -urile:  $S \xRightarrow{*} a^n B^n C^n$ . Producția 4. utilizată odată și 5. utilizată de  $n - 1$  ori dau  $S \xRightarrow{*} a^n b^n C^n$  (deoarece  $n \geq 1$ , regula 4. se aplică în mod cert). În sfârșit, cu producția 6. utilizată odată, urmată de  $n - 1$  aplicări ale regulii 7. se ajunge la  $a^n b^n c^n$ .

" $\supseteq$ ": În orice derivare care pleacă din  $S$ , producțiile 4. – 7. nu pot fi folosite decât după aplicarea regulii 2. (aceasta aduce terminalul 'a' la stânga lui  $B$ ). Utilizarea producției 1. de  $n$  ori duce la forma sentențială  $a^n S(BC)^n$ , ( $n \geq 1$ ). Apoi, cu regula 2. (care nu poate fi aplicată decât odată) se obține forma sentențială  $a^{n+1}(BC)^{n+1}$ .

Dacă  $n = 0$ , se începe direct cu aplicarea producției 2. și avem  $aBC$ .

Deci, după dispariția lui  $S$  se ajunge la forma sentențială  $a^n(BC)^n$  cu  $n \geq 1$ . Din acest moment producțiile 1. și 2. nu mai pot fi folosite, iar orice aplicare a uneia din regulile 3. – 7. duce la cuvinte de forma  $\alpha\beta$  cu  $\alpha \in V_T^*$ ,  $\beta \in V_N^*$ . Mai mult, producțiile 4. – 7. se pot aplica numai subcuvintelor de lungime 2, care constituie interfața dintre  $\alpha$  și  $\beta$ , având ca efect transformarea unui neterminal din  $\beta$  (un  $B$  în 'b' sau un  $C$  în 'c').

Deci cuvântul final  $w$  va satisface condiția  $|w|_a = |w|_b = |w|_c$ .

Să presupunem acum că un  $C$  este transformat în 'c' înainte ca toate  $B$ -urile să fi fost transformate în 'b'-uri. Atunci forma sentențială la care se ajunge este  $a^n b^i c \alpha$  cu  $i < n$  și  $|\alpha|_B > 0$ . În acest moment se mai pot aplica doar producțiile 3. sau 7.; regula 7. la interfața terminal/neterminal, iar regula 3. între două neterminale consecutive. Producția 3. reordonează neterminalele, dar nu poate să-l elimine pe  $B$ . Producția 3. generează 'c'-uri până când apare subcuvântul  $cB$ ; în acest moment nu se mai poate aplica nici o regulă 3. și derivarea se blochează fără a se obține nici un cuvânt din  $L(G)$ . Deci trebuie transformate în 'b'-uri toate neterminalele  $B$  aflate pe prima poziție în secvența de neterminale; când acest lucru nu mai este posibil, s-a ajuns la un cuvânt de forma  $a^n b^n C^n$ . Transformarea ulterioară a  $C$ -urilor în 'c'-uri (cu producțiile 6. și 7.) duce în final la un cuvânt din  $L$ .

După anumite restricții impuse regulilor de producție, se poate defini următoarea clasificare a gramaticilor – clasificare cunoscută sub numele de *clasificare Chomsky*:

**Definiția 3.6** O gramatică  $G = (V_N, V_T, S, P)$  este de tip  $i$  ( $0 \leq i \leq 3$ ) dacă sunt verificate restricțiile (i) asupra regulilor din  $P$ :

- (0) : Nici o restricție;
- (1) : Orice producție  $\alpha \rightarrow \beta \in P$  verifică condiția  $|\alpha| \leq |\beta|$ ; singura excepție posibilă o constituie regula  $S \rightarrow \epsilon$ ; apariția ei impune condiția suplimentară ca simbolul de start  $S$  să nu apară în membrul drept al nici unei producții din  $P$ ;
- (2) : Toate regulile din  $P$  sunt de forma  $A \rightarrow \alpha$ ,  $A \in V_N$ ,  $\alpha \in (V_N \cup V_T)^*$ ;
- (3) :  $P$  are numai producții de forma  $A \rightarrow bB$  sau  $A \rightarrow a$ , unde  $A, B \in V_N$ ,  $b \in V_T$ ,  $a \in V_T \cup \{\epsilon\}$ .

### Observația 3.2

- Tipurile de gramatici definite în acest mod se numesc gramatici clasificabile Chomsky, pentru a le deosebi de alte clase de gramatici cu proprietăți generative dar construite diferit, cum ar fi gramaticile matriciale, indexate, programate, contextuale etc.
- Gramaticile de tipul 3 se numesc și **gramatici regulate**, dintr-un motiv care va fi explicat mai târziu.
- Gramaticile de tipul 2 se numesc și **independente de context** (cfg – context-free grammars), deoarece o producție de forma  $A \rightarrow \alpha$  rescrie neterminalul  $A$  prin cuvântul  $\alpha$ , independent de contextul în care apare  $A$ .
- Gramaticile de tipul 1 se numesc **dependente (senzitive) de context** (csg) deoarece ele mai pot fi definite în felul următor:  $P$  conține numai producții de forma  $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$  cu  $A \in V_N$ ,  $\alpha_1, \alpha_2 \in (V_N \cup V_T)^*$ ,  $\beta \neq \epsilon$ . O regulă  $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$  arată că neterminalul  $A$  se rescrie  $\beta$  numai dacă  $A$  apare în contextul  $(\alpha_1, \alpha_2)$  (aceste noțiuni vor fi detaliate în Capitolul 7).

Un limbaj  $L$  este de tip  $i$  dacă este generat de o gramatică de tip  $i$ .

Familia limbajelor de tip  $i$  este notată  $\mathcal{L}_i$ . Din definiție rezultă șirul de incluziuni

$$\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$$

Una din principalele probleme – pe care vom încerca să o rezolvăm ulterior – este de a decide dacă aceste incluziuni sunt stricte sau nu.

### 3.3 Exerciții

**Exercițiul 3.1** Să se definească metode de recunoaștere și metode generative pentru limbajele  $L_1 = \{a^n \mid n \geq 0\}$ ,  $L_2 = \{a^n b^n \mid n \geq 0\}$ ,  $L_3 = \{a^n b^n c^n \mid n \geq 0\}$ .

**Exercițiul 3.2** Fie metoda de recunoaștere:  $\alpha \in L$  dacă și numai dacă

1.  $\alpha \in \{a, b\}^*$ ,
2.  $|\alpha|$  divizibil cu 3,
3.  $\alpha = a\beta b$ ,
4.  $a^3$  nu este subcuvânt al lui  $\alpha$ .

Să se construiască un AF care să accepte acest limbaj.

**Exercițiul 3.3** Fie metoda de recunoaștere:  $\alpha \in L$  dacă și numai dacă

1.  $\alpha = a^n$  ( $n \geq 1$ ),
2.  $\exists k$  impar unic astfel ca  $\alpha' = a^{n-k} \in L$ .

Să se arate că limbajul  $\{a^{p^2} \mid p \in \mathcal{N}\}$  este recunoscut de această metodă.

**Exercițiul 3.4** Fie metoda generativă pentru limbajul  $L \subseteq \{a, b\}^*$ :

1.  $\epsilon \in L$ ;
2. Dacă  $\alpha \in L$  atunci  $a\alpha, \alpha b \in L$ ;
3.  $L$  nu conține alte cuvinte.

Să se determine limbajul  $L$ .

**Exercițiul 3.5** Se dă metoda generativă  $M_g(L) = (SR, AX, X)$  unde  $AX = \{.X\}$ ,  $W = \{X, Y, Z, ., a\}$ ,  $X = \{a\}$ ,  $P = \{(X, XY), (X, a), (aY, Zaa), (aZ, Zaa), (.Z, .), (., \epsilon)\}$ . Să se arate că  $L = \{a^{2^n} \mid n \geq 0\}$ .

**Exercițiul 3.6** Pentru sistemul de rescriere definit în Exemplul 3.6 să se arate că este posibilă derivarea  $a^i.a^j \xRightarrow{*} a^{i+j}$ ,  $\forall i, j \geq 0$ .

**Exercițiul 3.7** Se dă sistemul de rescriere  $SR = (W, P)$  unde  $W = \{a, A, B, C, .\}$  și  $P = \{(aA, Aa), (a.a, A.), (a., .B), ((B, a), (A, C), (c, a), (., \epsilon))\}$ .

Să se arate că este posibilă derivarea  $a^i.a^j \xRightarrow{*} a^k$  unde  $k = \text{cmmdc}(i, j)$ .

**Exercițiul 3.8** Să se construiască o metodă de recunoaștere care să accepte toate cuvintele  $\alpha \in \{a, b, c\}^*$  care pot fi reduse la  $\epsilon$  dacă se șterg toate tripletele posibile  $abc$  (nu neapărat consecutive) care apar în  $\alpha$  (adică  $L = \{\alpha \mid \alpha \in \{a, b, c\}^*, |\alpha|_a = |\alpha|_b = |\alpha|_c\}$ ).

**Exercițiul 3.9** Să se construiască o metodă de recunoaștere pentru numerele pare.



**Exercițiul 3.10** Să se arate că limbajul din Exemplul 3.9 poate fi generat și de gramaticile  $G = (V_N, V_T, S, P)$  unde

- a)  $V_N = \{S, A\}$ ,  $V_T = \{a, b\}$ ,  $P = \{S \rightarrow aA| \epsilon, S \rightarrow Sb\}$   
b)  $V_N = \{S, A, B\}$ ,  $V_T = \{a, b\}$ ,  $P = \{S \rightarrow AB| \epsilon, A \rightarrow a, B \rightarrow SB|b\}$ .

**Exercițiul 3.11** Să se arate că limbajul  $L = \{a^n b^n c^n \mid n \geq 1\}$  este generat de gramaticile având producțiile

- a)  $P_1 = \{S \rightarrow aSBC|abC, CB \rightarrow BC, bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc\}$   
b)  $P_2 = \{S \rightarrow aSA|aB, cA \rightarrow Ac, BA \rightarrow bBc, B \rightarrow bc\}$

**Exercițiul 3.12** Se dă gramatica  $G = (V_N, V_T, S, P)$  cu  $P = \{S \rightarrow aAcbA, bA \rightarrow ca, Sc \rightarrow cb, Acb \rightarrow aSb\}$ .

Să se determine dacă  $aacba \in L(G)$ .

**Exercițiul 3.13** Să se construiască gramatici pentru generarea limbajelor

$$L_1 = \{a^n \mid n \geq 0\}, \quad L_2 = \{w \mid w \in \{a, b\}^*, |w|_a = |w|_b\}, \quad L_3 = \{a^i c a^j \mid i \geq j \geq 0\}.$$

**Exercițiul 3.14** Fie gramatica  $G = (V_N, V_T, S, P)$  unde  $V_N = \{S, A\}$ ,  $V_T = \{a, b\}$ ,  $P = \{S \rightarrow aAb, aA \rightarrow aaAb, A \rightarrow \epsilon\}$ . Să se determine  $L(G)$ .

**Exercițiul 3.15** Fie gramatica  $G_X = (V_N, V_T, X, P)$  cu  $V_N = \{A, B\}$ ,  $V_T = \{;, a, b\}$ ,  $P = \{A \rightarrow B|b; A, B \rightarrow a|B; a\}$ .

Să se afle  $L(G_A)$  și  $L(G_B)$ .

**Exercițiul 3.16** Similar exercițiului precedent, fie  $V_N = \{A, B, C\}$ ,  $V_T = \{a, b\}$  și  $P = \{A \rightarrow a|aC|bAA, B \rightarrow b|bC|aBB, C \rightarrow aB|bA\}$ .

Să se afle  $L(G_A)$ ,  $L(G_B)$ ,  $L(G_C)$ .

**Exercițiul 3.17** Fie gramatica  $G = (V_N, V_T, E, P)$  unde  $V_N = \{E, T, F\}$ ,  $V_T = \{+, *, (, ), a\}$ ,  $P = \{E \rightarrow E + T|T, T \rightarrow T * F|F, F \rightarrow (E)|a\}$ . Să se arate că:

1.  $(a * a + (a + a)) * a, (a) + (a), (a + a) * a \in L(G)$ .
2.  $L(G)$  conține toate expresiile aritmetice formate cu paranteze, operatorii  $+, *$  și operandul  $'a'$ .

**Exercițiul 3.18** Să se arate că limbajul generat de gramatica cu producțiile

$S \rightarrow ACaB, Ca \rightarrow aaC, CB \rightarrow DB, CB \rightarrow E, aD \rightarrow AC, AE \rightarrow \epsilon, aE \rightarrow Ea$  este  $L = \{a^{2^n} \mid n \geq 1\}$ .

**Exercițiul 3.19** Să se arate că limbajul generat de o gramatică care are numai producții de forma  $A \rightarrow Ba$  sau  $A \rightarrow a$  este regulat.

**Exercițiul 3.20** Fie gramatica de producții

$$P = \{S \rightarrow aAB|b, A \rightarrow bBS|c, B \rightarrow cSA|a\}.$$

Câte cuvinte de lungime 5 sunt în  $L(G)$  ? Dar de lungime 100 ?

# Capitolul 4

## Proprietăți de închidere ale limbajelor Chomsky

### 4.1 Noțiuni introductive

Fie  $\mathcal{L}$  o familie arbitrară de limbaje peste un alfabet finit  $V$ .

**Definiția 4.1**  $\mathcal{L}$  este închisă față de o operație  $\circ$  dacă

$$\forall L_1, L_2 \in \mathcal{L} \implies L_1 \circ L_2 \in \mathcal{L}.$$

O problemă deosebit de importantă în teoria limbajelor formale este aceea a închiderilor unei clase de limbaje față de anumite operații. Proprietățile de închidere studiate aici nu sunt singurele posibile; ne vom limita numai la operațiile uzuale; de asemenea, vom insista asupra celor mai utilizate clase de limbaje Chomsky:  $\mathcal{L}_2$  și  $\mathcal{L}_3$ .

De menționat că analiza anumitor proprietăți de închidere necesită noțiuni suplimentare; acestea vor fi abordate ulterior, în alte capitole.

**Definiția 4.2** Fie  $L_1$  și  $L_2$  două limbaje arbitrare peste un alfabet  $V$ . Definim operațiile

$$\begin{aligned} L_1 \cup L_2 &= \{w \mid w \in L_1 \text{ sau } w \in L_2\} && (\text{reuniune}) \\ L_1 \cap L_2 &= \{w \mid w \in L_1 \text{ și } w \in L_2\} && (\text{intersecție}) \\ L_1 \setminus L_2 &= \{w \mid w \in L_1 \text{ și } w \notin L_2\} && (\text{diferență}) \\ L_1 L_2 &= \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\} && (\text{concatenare}) \end{aligned}$$

**Definiția 4.3** Fie  $L$  un limbaj peste  $V$ . Se definesc operațiile

$$\begin{aligned} \bar{L} &= V^* \setminus L && (\text{complementară}) \\ L^* &= \bigcup_{i \geq 0} L^i && (\text{închidere Kleene sau "stelare"}) \\ L^+ &= \bigcup_{i \geq 1} L^i \end{aligned}$$

Reuniunea, concatenarea și stelarea se numesc *operații regulate*.

**Definiția 4.4** Două gramatici  $G_1$  și  $G_2$  se numesc *echivalente* dacă  $L(G_1) = L(G_2)$ .

Pentru demonstrarea multor proprietăți de închidere sunt necesare câteva leme de caracterizare.

**Lema 4.1** Pentru orice gramatică  $G = (V_N, V_T, S, P)$  de tip  $i$  ( $0 \leq i \leq 3$ ), există o gramatică echivalentă  $G' = (V_N', V_T, S, P')$  de același tip, în care toate producțiile lui  $P'$  care conțin terminale sunt de forma  $A \rightarrow a$ .

*Demonstrație:* Pentru fiecare  $a \in V_T$  se definește un neterminal nou  $A_a$ . Fie acum  $V_N' = V_N \cup \{A_a \mid a \in V_T\}$ , iar  $P'$  obținut din  $P$  prin înlocuirea tuturor terminalelor ' $a$ ' (oriunde apar) cu  $A_a$ , și adăugarea în final a producțiilor  $A_a \rightarrow a$ ,  $\forall a \in V_T$ .

Evident,  $G'$  este tot o gramatică de tip  $i$ . Rămâne de arătat egalitatea  $L(G) = L(G')$ .

" $\subseteq$ ": Fie  $w = a_1 a_2 \dots a_n \in L(G)$ ; deci există derivarea  $S \xRightarrow{*} w$ . Aceeași derivare construită în  $G'$  va da forma sentențială  $A_{a_1} A_{a_2} \dots A_{a_n}$ . Mai departe, aplicând noile reguli introduse în  $G'$ , fiecare  $A_{a_j}$  va merge în  $a_j$  ( $1 \leq j \leq n$ ) și se va obține tot  $w$ .

" $\supseteq$ ": Fie homomorfismul  $h : (V_N' \cup V_T)^* \rightarrow (V_N \cup V_T)^*$  definit astfel:

$$h(A_a) = a, \forall a \in V_T, \quad h(x) = x, \forall x \in V_N \cup V_T.$$

Să considerăm o generare directă  $\alpha \Rightarrow \beta$ , în gramatica  $G'$ . Dacă s-a folosit o producție  $A_a \rightarrow a$ , atunci  $h(\alpha) = h(\beta)$ ; altfel  $h(\alpha) \Rightarrow h(\beta)$ .

În ambele cazuri avem  $h(\alpha) \xRightarrow{*} h(\beta)$  (derivare în  $G$ ).

Deci, pentru orice  $w \in L(G')$ , vom avea o derivare în gramatica  $G$ :

$$S = h(S) \xRightarrow{*} h(w) = w$$

adică  $w \in L(G)$ .

q.e.d.

**Lema 4.2** Fie  $G = (V_N, V_T, S, P)$  o gramatică independentă de context. Dacă  $AB \xRightarrow{*} \alpha$  atunci există  $\alpha_1, \alpha_2 \in (V_N \cup V_T)^*$  astfel încât:

$$1. \alpha = \alpha_1 \alpha_2;$$

$$2. A \xRightarrow{*} \alpha_1, \quad B \xRightarrow{*} \alpha_2.$$

*Demonstrație:* Vom folosi o inducție după lungimea  $n$  a derivării  $AB \xRightarrow{n} \alpha$ .

Pentru  $n = 0$  avem  $AB \xRightarrow{0} AB$  și – dacă luăm  $\alpha_1 = A$ ,  $\alpha_2 = B$  – afirmația este evidentă.

Să presupunem lema adevărată pentru orice derivare de lungime  $n$  și fie  $AB \xRightarrow{n+1} \alpha$ . Punând în evidență ultimul pas al derivării, avem  $AB \xRightarrow{n} \gamma \Rightarrow \alpha$ , și fie  $C \rightarrow \beta$  a  $(n+1)$ -a producție aplicată.

Conform ipotezei de inducție,  $\gamma = \alpha_1' \alpha_2'$  și  $A \xRightarrow{*} \alpha_1'$ ,  $B \xRightarrow{*} \alpha_2'$ . Neterminalul  $C$  se află în unul din cele două cuvinte:  $\alpha_1'$  sau  $\alpha_2'$ .

Să presupunem că  $\alpha_1' = \beta_1 C \beta_2$  (dacă  $C$  se află în  $\alpha_2'$ , se procedează analog). Atunci

$$AB \xRightarrow{n} \beta_1 C \beta_2 \alpha_2' \xRightarrow{*} \beta_1 \beta_2 \alpha_2' = \alpha$$

Vom lua  $\alpha_1 = \beta_1 \beta_2$ ,  $\alpha_2 = \alpha_2'$ ; se obține

$$AB \xRightarrow{n+1} \alpha \quad \text{și} \quad \alpha = \alpha_1 \alpha_2, \quad A \xRightarrow{*} \beta_1 C \beta_2 \xRightarrow{*} \beta_1 \beta_2 = \alpha_1, \quad B \xRightarrow{*} \alpha_2' = \alpha_2. \quad \text{q.e.d.}$$

**Exemplul 4.1** Să luăm gramatica de producții

$$S \longrightarrow aBC|b, \quad B \longrightarrow bCS|c, \quad C \longrightarrow cSB|a$$

În derivarea  $SB \xRightarrow{*} accbcbab$  vom avea  $\alpha_1 = accbc$ ,  $\alpha_2 = bab$  și

$$\begin{aligned} S &\xRightarrow{*} aBC \xRightarrow{*} acC \xRightarrow{*} accSB \xRightarrow{*} accbB \xRightarrow{*} accbc, \\ B &\xRightarrow{*} bCS \xRightarrow{*} baS \xRightarrow{*} bab. \end{aligned}$$

**Observația 4.1**

- Cuvintele  $\alpha_1$  și  $\alpha_2$  nu sunt unic determinate. Astfel, pentru gramatica de producții  $S \longrightarrow SS|a$ , avem  $SS \xRightarrow{*} aaa$  și sunt două alegeri posibile: ( $\alpha_1 = a$ ,  $\alpha_2 = aa$ ), sau ( $\alpha_1 = aa$ ,  $\alpha_2 = a$ ).

- Lema 4.2 poate fi generalizată pentru  $p$  neterminale; anume:

$$\text{Dacă } A_1 A_2 \dots A_p \xRightarrow{*} \alpha, \text{ atunci } \alpha = \alpha_1 \alpha_2 \dots \alpha_p \text{ și } A_i \xRightarrow{*} \alpha_i, \quad (1 \leq i \leq p).$$

**Lema 4.3** Orice gramatică regulată este echivalentă cu o gramatică care conține numai producții de forma

$$A \longrightarrow \alpha B, \quad A \longrightarrow \alpha, \quad \text{unde} \quad A, B \in V_N, \quad \alpha \in V_T^*.$$

*Demonstrație:* " $\implies$ ": Fie  $G = (V_N, V_T, S, P)$  o gramatică regulată; atunci ea va satisface în mod banal condițiile cerute de lema.

" $\impliedby$ ": Fie  $A \longrightarrow \alpha B$  o producție de primul tip în gramatica  $G = (V_N, V_T, S, P)$ . Se disting trei cazuri:

- $|\alpha| = 1$ . Atunci producția verifică condiția cerută de o gramatică regulată și rămâne nemodificată.
- $|\alpha| \geq 2$ . Dacă  $\alpha = a_1 a_2 \dots a_n$  ( $n \geq 2$ ), se introduc  $n - 1$  neterminale noi  $B_1, B_2, \dots, B_{n-1}$  și regula analizată se înlocuiește cu producțiile

$$A \longrightarrow a_1 B_1, \quad B_1 \longrightarrow a_2 B_2, \quad \dots \quad B_{n-1} \longrightarrow a_n B.$$

- iii.  $\alpha = \epsilon$ . Se construiește mulțimea  $N(B) = \{D \mid B \xRightarrow{*} D, D \in V_N\}$ ; regula  $A \longrightarrow B$  se înlocuiește cu mulțimea de producții  $\{A \longrightarrow \gamma C \mid D \longrightarrow \gamma C \in P, D \in N(B), \gamma \neq \epsilon\} \cup \{A \longrightarrow \gamma \mid D \longrightarrow \gamma \in P, D \in N(B)\}$ .

De remarcat că este posibil ca mulțimea de reguli care înlocuiește producția  $A \longrightarrow B$  să fie vidă.

Pentru o producție de forma  $A \longrightarrow \alpha$  (tipul doi) apar două situații:

- iv.  $|\alpha| \leq 1$ . Producția rămâne nemodificată.
- v.  $|\alpha| \geq 2$ . Dacă  $\alpha = b_1 b_2 \dots b_n$ , atunci se introduc  $n - 1$  neterminale noi  $C_1, C_2, \dots, C_{n-1}$  și regula  $A \longrightarrow \alpha$  se înlocuiește cu setul de producții

$$A \longrightarrow b_1 C_1, \quad C_1 \longrightarrow b_2 C_2, \quad \dots \quad C_{n-1} \longrightarrow b_n.$$

Se prelucrează pe rând fiecare regulă de tipul doi, apoi fiecare regulă de primul tip, obținându-se în final gramatica  $G' = (V_N', V_T, S, P')$  (eventualele producții identice din  $P'$  se elimină). Să arătăm că  $L(G) = L(G')$ .

" $\subseteq$ ": Rezultă imediat din construcția gramaticii  $G'$ .

" $\supseteq$ ": Fie  $w \in L(G')$ ,  $w = a_1 a_2 \dots a_n$ , ( $n > 1$ ). Deci există o derivare în  $G'$  pentru  $w$ :

$$S \Longrightarrow a_1 X_1 \Longrightarrow a_1 a_2 X_2 \Longrightarrow \dots \Longrightarrow a_1 a_2 \dots a_{n-1} X_{n-1} \Longrightarrow a_1 a_2 \dots a_n$$

Putem găsi atunci mulțimea tuturor neterminalelor  $X_i$  folosite în această derivare, cu proprietatea  $X_i \in V_N$ . Punând în evidență numai aceste neterminale  $X_i$ , avem

$$S \xRightarrow{*} a_1 a_2 \dots a_{i_1} X_{i_1} \xRightarrow{*} a_1 a_2 \dots a_{i_2} X_{i_2} \xRightarrow{*} \dots \xRightarrow{*} a_1 a_2 \dots a_{i_k} X_{i_k} \xRightarrow{*} a_1 a_2 \dots a_n$$

Din construcția gramaticii  $G'$  va rezulta că producțiile

$$S \longrightarrow a_1 a_2 \dots a_{i_1} X_{i_1}, \quad X_{i_1} \longrightarrow a_{i_1+1} \dots a_{i_2} X_{i_2}, \quad \dots \quad X_{i_k} \longrightarrow a_{i_k+1} \dots a_n$$

sunt sau în  $P$ , sau – dacă  $X_{i_j} \longrightarrow a_{i_j+1} \dots a_{i_{j+1}} X_{i_{j+1}} \notin P$ , atunci  $\exists Y_1, Y_2, \dots, Y_r \in V_N$  ( $r \geq 1$ ) cu  $X_{i_j} \longrightarrow Y_1$ ,  $Y_1 \longrightarrow Y_2, \dots, Y_r \longrightarrow a_{i_j+1} \dots a_{i_{j+1}} X_{i_{j+1}}$  producții din  $P$ .

În acest fel, derivarea anterioară se poate scrie ca o derivare în gramatica  $G$ , deci  $w \in L(G)$ .

Dacă  $n = 1$ , atunci  $S \longrightarrow w \in P'$ , deci  $S \longrightarrow w \in P$  sau – altfel spus,  $w \in L(G)$ . q.e.d.

**Exemplul 4.2** Fie gramatica de producții

$$S \longrightarrow A|abc, \quad A \longrightarrow aA|B, \quad B \longrightarrow bbS|b$$

Conform Lemei 4.3, ea este echivalentă cu o gramatică regulată, construită astfel:

Producțiile  $A \rightarrow aA$  și  $B \rightarrow b$  rămân nemodificate.

Pentru  $S \rightarrow abc$  se introduc două neterminale noi  $C, D$  și producția se înlocuiește cu regulile  $S \rightarrow aC$ ,  $C \rightarrow bD$ ,  $D \rightarrow c$ .

Pentru  $B \rightarrow bbS$  este nevoie de un singur neterminal suplimentar –  $E$ , și apar noile reguli  $B \rightarrow bE$ ,  $E \rightarrow bS$ .

Deci noua mulțime de neterminale este  $V_N' = \{S, A, B, C, D, E\}$ , iar producțiile sunt

$$S \rightarrow A|aC, \quad A \rightarrow aA|B, \quad B \rightarrow bE|b, \quad C \rightarrow bD, \quad D \rightarrow c, \quad E \rightarrow bS.$$

Mai trebuie eliminate două reguli:  $S \rightarrow A$  și  $A \rightarrow B$ .

Se construiesc mulțimile  $N(A) = \{A, B\}$ ,  $N(B) = \{B\}$ .

Producția  $S \rightarrow A$  se înlocuiește cu  $S \rightarrow aA|bE|b$ , iar  $A \rightarrow B$  cu  $A \rightarrow bE|b$ .

Gramatica regulată obținută în final este  $G' = (V_N', V_T, S, P')$  unde

$$P' = \left\{ \begin{array}{ll} S \rightarrow aC|aA|bE|b, & C \rightarrow bD \\ A \rightarrow aA|bE|b, & D \rightarrow c \\ B \rightarrow bE|b, & E \rightarrow bS \end{array} \right\}$$

Fie cuvântul  $w = bbabc$ .

El este obținut în gramatica  $G$  cu derivarea  $S \Rightarrow A \Rightarrow B \Rightarrow bbS \Rightarrow bbabc$ ,

iar în gramatica  $G'$  cu derivarea  $S \Rightarrow bE \Rightarrow bbS \Rightarrow bbaC \Rightarrow bbabD \Rightarrow bbabc$ .

## 4.2 Înciderea limbajelor Chomsky față de operațiile regulate

Ne vom referi în această secțiune la familiile de limbaje Chomsky  $\mathcal{L}_i$  ( $0 \leq i \leq 3$ ).

**Teorema 4.1**  $\mathcal{L}_i$  conține limbajele finite.

*Demonstrație:* Fie  $L = \{x_1, x_2, \dots, x_n\}$  un limbaj finit peste un vocabular  $V$ . Gramatica  $G = (\{S\}, V, S, \{S \rightarrow x_1|x_2|\dots|x_n\})$  este echivalentă cu o gramatică regulată (Lema 4.3) și  $L(G) = L$ , deci  $L \in \mathcal{L}_3$ . Cum  $\mathcal{L}_3 \subseteq \mathcal{L}_i$  ( $0 \leq i \leq 3$ ), rezultă că orice clasă de limbaje Chomsky conține limbajele finite. q.e.d.

Fie acum  $L, L' \in \mathcal{L}_i$ , două limbaje de tip  $i$ , generate de gramaticile  $G = (V_N, V_T, S, P)$  respectiv  $G' = (V_N', V_T', S', P')$ ; fără a micșora generalitatea, putem presupune

$$V_N \cap V_N' = \emptyset.$$

**Teorema 4.2**  $\mathcal{L}_i$  ( $0 \leq i \leq 3$ ) este închisă față de reuniune.

*Demonstrație:* Fie gramatica  $G'' = (V_N \cup V_N' \cup \{S_0\}, V_T \cup V_T', S_0, P \cup P' \cup \{S_0 \rightarrow S|S'\})$  unde  $S_0 \notin V_N \cup V_N'$  este un neterminal nou.  $G''$  este o gramatică de tip  $i$  pentru  $i = 0, 1, 2$ ; pentru  $i = 3$ , Lema 4.3 asigură existența unei gramatici regulate echivalente.

Să arătăm că  $L(G'') = L \cup L'$ .

" $\subseteq$ ": Fie  $w \in L(G'')$ , deci există o derivare  $S_0 \xRightarrow{*} w$  în  $G''$ . Prima producție care se poate aplica este  $S_0 \rightarrow S$  sau  $S_0 \rightarrow S'$  (nu există alte reguli cu  $S_0$  în membrul stâng). Să presupunem că s-a folosit  $S_0 \rightarrow S$  (pentru  $S_0 \rightarrow S'$  se procedează analog).

Deci  $S_0 \Rightarrow S \xRightarrow{*} w$ . În derivarea  $S \xRightarrow{*} w$  apar numai neterminale din  $V_N$ , deci nu se pot aplica decât producții din  $P$ . Rezultă că aceasta va fi o derivare și în gramatica  $G$ , deci  $w \in L(G) = L \subseteq L \cup L'$ .

" $\supseteq$ ": Dacă  $w \in L \cup L'$ , să presupunem că  $w \in L'$  (cazul  $w \in L$  se tratează similar). Deci  $S' \xRightarrow{*} w$ , derivare care utilizează numai reguli din  $P'$ . Cum  $P' \subseteq P''$ , rezultă că aceasta va fi o derivare și în gramatica  $G''$ . Putem construi atunci derivarea în  $G''$ :  $S_0 \Rightarrow S' \xRightarrow{*} w$ . Deci  $w \in L(G'')$ . q.e.d.

**Teorema 4.3**  $\mathcal{L}_i$  ( $0 \leq i \leq 3$ ) este închisă față de concatenare.

*Demonstrație:*

- $i = 3$ : Construim gramatica  $G'' = (V_N \cup V_N', V_T \cup V_T', S, P'')$ , unde

$$P'' = P' \cup \{A \rightarrow aB \mid A \rightarrow aB \in P\} \cup \{A \rightarrow aS' \mid A \rightarrow a \in P\}$$

Lăsăm ca exercițiu demonstrarea egalității  $L(G'') = LL'$ .

- $i \leq 2$ : Limbajul  $LL'$  este generat de gramatica

$$G'' = (V_N \cup V_N' \cup \{S_0\}, V_T \cup V_T', S_0, P \cup P' \cup \{S_0 \rightarrow SS'\})$$

unde  $S_0 \notin V_N \cup V_N'$  este un simbol de start nou.

Egalitatea  $L(G'') = LL'$  se arată folosind Lema 4.2 pentru cazul  $i = 2$ , sau pe baza Lemei 4.1 și a condiției  $V_N \cap V_N' = \emptyset$  pentru cazurile  $i = 0$  și  $i = 1$ . q.e.d.

**Teorema 4.4**  $\mathcal{L}_i$  ( $0 \leq i \leq 3$ ) este închisă față de stelare.

*Demonstrație:* Fie  $L \in \mathcal{L}_i$  un limbaj generat de  $G = (V_N, V_T, S, P)$ , gramatică de tip  $i$ .

- $i = 3$ : Să considerăm  $S' \notin V_N$  un simbol de start nou; construim  $V_N' = V_N \cup \{S'\}$  și

$$P' = \{A \rightarrow aB \mid A \rightarrow aB \in P\} \cup \{A \rightarrow aS' \mid A \rightarrow a \in P\} \cup \{S' \rightarrow S \mid \epsilon\}.$$

Gramatica  $G' = (V_N', V_T, S', P')$  este echivalentă cu o gramatică regulată (Lema 4.3), iar limbajul generat de ea este  $L(G') = L^*$ . Într-adevăr:

" $\subseteq$ ": Fie  $w \in L(G')$  și  $S' \xRightarrow{*} w$  o derivare în gramatica  $G'$ .

Dacă  $w = \epsilon$ , atunci  $S' \Rightarrow \epsilon$  (pe baza producției  $S' \rightarrow \epsilon$ ) și evident  $\epsilon \in L^*$ .

Să presupunem  $w \neq \epsilon$ ; punând în evidență în derivarea  $S' \xRightarrow{*} w$  fiecare apariție a lui  $S'$  (singurul neterminal din  $V_N' \setminus V_N$ ) obținem

$$S' \xRightarrow{*} \alpha_1 S' \xRightarrow{*} \alpha_1 \alpha_2 S' \xRightarrow{*} \dots \xRightarrow{*} \alpha_1 \alpha_2 \dots \alpha_k S' \xRightarrow{*} \alpha_1 \alpha_2 \dots \alpha_k \quad (k \geq 1)$$

unde, datorită formei gramaticii  $G'$ ,  $\alpha_1, \alpha_2, \dots, \alpha_k \in V_T^*$ .

Deci, cu excepția ultimei derivări, toate celelalte derivări folosite sunt de forma  $S' \xRightarrow{*} \alpha_i S'$ . Să detaliam o astfel de derivare:

Dacă  $\alpha_i = a_1^i a_2^i \dots a_{n_i}^i$ , avem

$$S' \Rightarrow S \Rightarrow a_1^i A_1 \Rightarrow a_1^i a_2^i A_2 \Rightarrow \dots \Rightarrow a_1^i a_2^i \dots a_{n_i-1}^i A_{n_i-1} \Rightarrow a_1^i a_2^i \dots a_{n_i}^i S'.$$

Înafara ultimei producții utilizate, toate celelalte sunt reguli din  $P$ . Ultima regulă  $A_{n_i-1} \rightarrow a_{n_i}^i S'$  corespunde prin construcție producției  $A_{n_i-1} \rightarrow a_{n_i}^i \in P$ . Deci în  $G$  există derivarea  $S \xRightarrow{*} \alpha_i$ , adică  $\alpha_i \in L$ .

Rezumând,  $w = \alpha_1 \alpha_2 \dots \alpha_k \in L^k \subseteq L^*$ .

" $\subseteq$ ": Fie  $w \in L^*$ ; deci există un întreg  $k \geq 0$  cu  $w \in L^k$ .

Pentru  $k = 0$ ,  $w = \epsilon$  și  $S' \Rightarrow \epsilon$ .

Pentru  $k \geq 1$  există  $\alpha_1, \alpha_2, \dots, \alpha_k \in L$  cu  $w = \alpha_1 \alpha_2 \dots \alpha_k$ .

Deci pentru  $i = 1, 2, \dots, k$ , în gramatica  $G$  există derivările  $S \xRightarrow{*} \alpha_i$ .

Vom construi derivarea în gramatica  $G'$

$$\begin{aligned} S' \Rightarrow S \xRightarrow{*} \alpha_1 S' \Rightarrow \alpha_1 S \xRightarrow{*} \alpha_1 \alpha_2 S' \Rightarrow \dots \xRightarrow{*} \alpha_1 \alpha_2 \dots \alpha_k S' \Rightarrow \\ \Rightarrow \alpha_1 \alpha_2 \dots \alpha_k = w \end{aligned}$$

Detaliile derivării  $S \xRightarrow{*} \alpha_i S'$  sunt similare celor de la incluziunea anterioară.

- $i \leq 2$ : Folosind Lema 4.1, putem presupune că orice producție din  $P$  are sau forma  $A_a \rightarrow a$ , sau conține numai neterminal. Fără a micșora generalitatea, putem aranja ca într-o derivare, regulile de forma  $A_a \rightarrow a$  să fie aplicate ultimele (acest lucru este totdeauna posibil, regulile fiind de tip independent de context).

Cu aceste prezumții, se demonstrează într-o manieră similară celei anterioare, că  $L^*$  este generat de gramatica

$$G' = (V_N \cup \{S'\}, V_T, S', P \cup \{S \rightarrow S'S|\epsilon\})$$

q.e.d.



**Exemplul 4.3** Fie limbajul  $L = \{a^i b^j c^k \mid i \neq j \text{ sau } j \neq k, \quad i, j, k \geq 1\}$ .

Să construim o gramatică independentă de context  $G$  care să genereze  $L$ . Pentru aceasta ne vom baza pe Teoremele 4.2 și 4.3 ca să descompunem  $L$  în limbaje mai simple.

$L$  poate fi scris  $L = L_1 \cup L_2$  unde

$$L_1 = \{a^i b^j c^k \mid i \neq j, \quad i, j, k \geq 1\}, \quad L_2 = \{a^i b^j c^k \mid j \neq k, \quad i, j, k \geq 1\}.$$

Mai departe,  $L_1 = L_3 L_4$  și  $L_2 = L_5 L_6$ , unde

$$L_3 = \{a^i b^j \mid i \neq j, \quad i, j \geq 1\}, \quad L_4 = \{c^k \mid k \geq 1\},$$

$$L_5 = \{a^i \mid i \geq 1\}, \quad L_6 = \{b^j c^k \mid j \neq k, \quad j, k \geq 1\}.$$

Pentru  $L_4$  și  $L_5$  gramaticile sunt ușor de scris; ele au seturile de producții

$$P_4 = \{S_4 \longrightarrow cS_4|c\} \text{ respectiv } P_5 = \{S_5 \longrightarrow aS_5|a\}.$$

Pentru  $L_3$  și  $L_6$  mai efectuăm o descompunere:  $L_3 = L_3' \cup L_3''$ ,  $L_6 = L_6' \cup L_6''$  unde

$$L_3' = \{a^i b^j \mid 1 \leq i < j\}, \quad L_3'' = \{a^i b^j \mid 1 \leq j < i\},$$

$$L_6' = \{b^j c^k \mid 1 \leq j < k\}, \quad L_6'' = \{b^j c^k \mid 1 \leq k < j\}.$$

Avem gramaticile de producții

$$P_3' = \{S_3' \longrightarrow aS_3'b|S_3'b|ab^2\}, \quad P_3'' = \{S_3'' \longrightarrow aS_3''b|aS_3''|a^2b\},$$

$$P_6' = \{S_6' \longrightarrow bS_6'c|S_6'c|bc^2\}, \quad P_6'' = \{S_6'' \longrightarrow bS_6''c|bS_6''|b^2c\}.$$

cu  $L_i' = L(G_i')$ ,  $L_i'' = L(G_i'')$ , ( $i = 3, 6$ ) (detalierea acestor egalități este lăsată ca exercițiu).

Acum începem să compunem pas-cu-pas gramatica pentru limbajul

$$L = (L_3' \cup L_3'')L_4 \cup L_5(L_6' \cup L_6'')$$

Pentru  $L_3$  și  $L_6$  seturile de producții sunt

$$P_3 = \begin{cases} S_3 \longrightarrow S_3'|S_3'' \\ S_3' \longrightarrow aS_3'b|S_3'b|ab^2 \\ S_3'' \longrightarrow aS_3''b|aS_3''|a^2b \end{cases} \quad \text{respectiv} \quad P_6 = \begin{cases} S_6 \longrightarrow S_6'|S_6'' \\ S_6' \longrightarrow bS_6'c|S_6'c|bc^2 \\ S_6'' \longrightarrow bS_6''c|bS_6''|b^2c \end{cases}$$

Setul de producții pentru  $L_1$  este  $P_1 = \{S_1 \longrightarrow S_3 S_4\} \cup P_3 \cup P_4$ , iar pentru  $L_2$ :

$$P_2 = \{S_2 \longrightarrow S_5 S_6\} \cup P_5 \cup P_6.$$

În sfârșit, pentru limbajul  $L$  avem gramatica  $G = (V_N, V_T, S, P)$ , unde

$$V_N = \{S, S_1, S_2, S_3, S_3', S_3'', S_4, S_5, S_6, S_6', S_6''\},$$

$$P = \{S \longrightarrow S_1|S_2, S_1 \longrightarrow S_3 S_4, S_2 \longrightarrow S_5 S_6, S_3 \longrightarrow S_3'|S_3'', S_3' \longrightarrow aS_3'b|S_3'b|ab^2,$$

$$S_3'' \longrightarrow aS_3''b|aS_3''|a^2b, S_4 \longrightarrow cS_4|c, S_5 \longrightarrow aS_5|a, S_6 \longrightarrow S_6'|S_6'',$$

$$S_6' \longrightarrow bS_6'c|S_6'c|bc^2, S_6'' \longrightarrow bS_6''c|bS_6''|b^2c\}.$$

Evident, în această gramatică, derivarea unui cuvânt nu este totdeauna unică.

De exemplu, pentru  $w = ab^2 c^3$  sunt posibile cel puțin două derivări:

$$S \Longrightarrow S_1 \Longrightarrow S_3 S_4 \Longrightarrow S_3 c S_4 \Longrightarrow S_3 c^2 S_4 \Longrightarrow S_3 c^3 \Longrightarrow S_3'' c^3 \Longrightarrow ab^2 c^3 \text{ și}$$

$$S \Longrightarrow S_2 \Longrightarrow S_5 S_6 \Longrightarrow a S_6 \Longrightarrow a S_6' \Longrightarrow ab S_6' c \Longrightarrow ab^2 c^3.$$

**Exemplul 4.4** Fie limbajul  $L = \{a(bc)^n \mid n \geq 0\}$ . Ne propunem să construim o gramatică regulată pentru limbajul  $L^*$ .

$L$  este generat de gramatica regulată  $G = (V_N, V_T, S, P)$  unde  $V_N = \{S, A, B\}$ ,  $V_T = \{a, b, c\}$ ,  $P = \{S \rightarrow aA, A \rightarrow bB|\epsilon, B \rightarrow cA\}$ .

Conform Teoremei 4.4,  $L^*$  va fi generat de gramatica  $G' = (V_N', V_T, S', P')$  cu  $V_N' = \{S', S, A, B\}$  și

$$P' = \{S' \rightarrow \epsilon|S, S \rightarrow aA, A \rightarrow bB|S', B \rightarrow cA\}.$$

Utilizând Lema 4.3,  $G'$  este echivalentă cu o gramatică regulată. Singurele producții care trebuie modificate în acest scop sunt  $S' \rightarrow S$  și  $A \rightarrow S'$ .

Vom avea  $N(S) = \{S\}$ ,  $N(S') = \{S', S\}$ .

$S' \rightarrow S$  se înlocuiește cu  $S' \rightarrow aA$ , iar  $A \rightarrow S'$  cu  $A \rightarrow aA|\epsilon$ .

Deci setul de producții  $P'$  va deveni

$$P' = \{S' \rightarrow aA|\epsilon, S \rightarrow aA, A \rightarrow bB|aA|\epsilon, B \rightarrow cA\}.$$

De remarcat că regula  $S \rightarrow aA$  nu poate fi folosită în nici o derivare, și deci ea poate fi eliminată din gramatică, împreună cu simbolul  $S$  (care nu mai este simbol de start).

**Corolarul 4.1** Dacă  $L$  este un limbaj de tip  $i$  ( $0 \leq i \leq 3$ ), atunci  $L^+$  este de tip  $i$ .

*Demonstrație:* Din gramatica pentru  $L^*$  construită în demonstrația Teoremei 4.4, se elimină regula  $S' \rightarrow \epsilon$ . q.e.d.

O clasă  $\mathcal{L}$  de limbaje cu proprietatea  $L \in \mathcal{L} \implies L^+ \in \mathcal{L}$  este închisă la stelare  $\epsilon$ -free.

## 4.3 Închiderea față de homomorfisme și substituții

**Definiția 4.5** Fie  $V, W$  două alfabet finite. O aplicație  $f : V \rightarrow 2^{W^*}$  se numește substituție.

Aplicația  $f$  se poate extinde la  $V^*$  astfel:

1.  $f(\epsilon) = \epsilon$ ;
2.  $f(aw) = f(a)f(w) \quad \forall a \in V, w \in V^*$ .

Pentru un limbaj  $L \subseteq V^*$ ,  $f(L) = \bigcup_{x \in L} f(x)$ .

**Exemplul 4.5** Fie  $f(0) = \{a\}$ ,  $f(1) = \{w\tilde{w} \mid w \in \{b, c\}^*\}$ . Substituția  $f$  aplică cuvintele din limbajul  $\{0^n 1^n \mid n \geq 1\}$  în limbajul  $\{a^n w_1 \tilde{w}_1 w_2 \tilde{w}_2 \dots w_n \tilde{w}_n \mid w_i \in \{b, c\}^*, 1 \leq i \leq n\}$ .

**Definiția 4.6** Fie  $V$  un alfabet finit,  $\mathcal{L}$  o clasă de limbaje peste  $V$  și  $f : V \longrightarrow \mathcal{L}$  o substituție.  $\mathcal{L}$  este închisă față de  $f$  dacă  $\forall L \in \mathcal{L}$  avem  $f(L) \in \mathcal{L}$ .

Substituția  $f$  peste  $V$  este  $\epsilon$ - liberă dacă  $\forall a \in V, \epsilon \notin f(a)$ .

Substituția  $f$  este finită dacă  $\forall a \in V, f(a)$  este un limbaj finit.

Dacă  $\forall a \in V, f(a) \in W$ ,  $f$  se numește *homomorfism*.

Pentru un homomorfism  $f : V \longrightarrow W$ , aplicația  $h : W \longrightarrow 2^V$  definită  $h(x) = \{a \mid f(a) = x\}$  se numește *homomorfism invers* al lui  $f$  (notat cu  $h = f^{-1}$ ).

**Teorema 4.5**  $\mathcal{L}_3$  este închisă la substituție.

*Demonstrație:* Fie  $L \in \mathcal{L}_3$  un limbaj,  $G = (V_N, \{a_1, \dots, a_n\}, S, P)$  gramatica regulată care-l generează, iar  $f : V \longrightarrow 2^{W^*}$  o substituție. Fie  $G_i = (V_N^i, V_T^i, S_i, P_i)$  gramaticile regulate cu  $L(G_i) = f(a_i)$ , ( $1 \leq i \leq n$ ). Fără a micșora generalitatea, putem presupune că

- Toate alfabetele de neterminale  $V_N^i$  sunt disjuncte două câte două.
- În fiecare  $G_i$  există  $X_i \in V_N^i$  unic, folosit ca membru stâng în regula  $X_i \longrightarrow \epsilon$ . Cu excepția acestei producții, toate regulile din  $P_i$  sunt de forma  $A \longrightarrow aB^1$ .
- Similar, în  $G$  există un neterminal  $X$  cu aceeași proprietate.

Construim gramatica  $G' = (V_N', V_T', S, P')$  unde  $V_N' = \left( \bigcup_{1 \leq i \leq n} V_N^i \right) \cup V_N$ ,  $V_T' = \bigcup_{1 \leq i \leq n} V_T^i$ , iar  $P'$  este generată în felul următor:

- $X \longrightarrow \epsilon \in P'$ ;
- Pentru fiecare producție  $A \longrightarrow a_i B \in P$ , în  $P'$  se introduce setul de reguli

$$(P_i \setminus \{X_i \longrightarrow \epsilon\}) \cup \{A \longrightarrow S_i, \quad X_i \longrightarrow B\}.$$

Conform Lemei 4.3, gramatica  $G'$  este echivalentă cu o gramatică regulată.

Demonstrarea egalității  $L(G') = f(L)$  este lăsată ca exercițiu.

q.e.d.

**Exemplul 4.6** Să considerăm substituția regulată  $f : \{a, b\} \longrightarrow 2^{\{0,1\}^*}$  definită

$$f(a) = (0+1)^*11(0+1)^*, \quad f(1) = 10^*1$$

<sup>1</sup>Restricția este ușor de îndeplinit. Se introduce un neterminal nou  $X_i$ , și fiecare producție de forma  $A \longrightarrow a$  se înlocuiește cu  $A \longrightarrow aX_i$ . În plus, în  $P_i$  este introdusă și producția  $X_i \longrightarrow \epsilon$ .

Limbaajul regulat  $f(a)$  este generat de gramatica  $G_1 = (\{S_1, A_1, B_1, Y\}, \{0, 1\}, S_1, P_1)$ , unde  $P_1 = \{S_1 \rightarrow 0S_1|1S_1|1A_1, A_1 \rightarrow 1B_1|1Y, B_1 \rightarrow 0B_1|1B_1|0Y|1Y, Y \rightarrow \epsilon\}$ .

Similar,  $f(b)$  este generat de  $G_2 = (\{S_2, A_2, Z\}, \{0, 1\}, S_2, P_2)$  cu

$$P_2 = \{S_2 \rightarrow 1A_2, A_2 \rightarrow 0A_2|1Z, Z \rightarrow \epsilon\}.$$

Fie acum limbaajul regulat  $L = (ab)^+ba$  și  $G = (\{S, A, B, C, X\}, \{a, b\}, S, P)$  gramatica care îl generează, cu

$$P = \begin{cases} (1) & S \rightarrow aA, & (4) & B \rightarrow bC \\ (2) & A \rightarrow bS & (5) & C \rightarrow aX \\ (3) & A \rightarrow bB & (6) & X \rightarrow \epsilon \end{cases}$$

Limbaajul regulat  $f(L)$  este generat de gramatica  $G' = (V_N', V_T', S, P')$ , unde  $V_N' = \{S, S_1, S_2, A, A_1, A_2, B, B_1, C, X, Y, Z\}$ ,  $V_T' = \{0, 1\}$ , iar  $P'$  este formată din producțiile

- $X \rightarrow \epsilon$
- $S_1 \rightarrow 0S_1|1S_1|1A_1, S_2 \rightarrow 1A_2, A_1 \rightarrow 1B_1|1Y, A_2 \rightarrow 0A_2|1Z, B_1 \rightarrow 0B_1|1B_1|0Y|1Y$  (regulile nefinale din  $P_1 \cup P_2$ )
- $\{S \rightarrow S_1, Y \rightarrow A\}$  (generate de producția (1)),  
 $\{A \rightarrow S_2, Z \rightarrow S\}$  (generate de producția (2)),  
 $\{A \rightarrow S_2, Z \rightarrow B\}$  (generate de producția (3)),  
 $\{B \rightarrow S_2, Z \rightarrow C\}$  (generate de producția (4)),  
 $\{C \rightarrow S_1, Y \rightarrow X\}$  (generate de producția (5)).

Să considerăm cuvântul  $abba \in L$ . Atunci, de exemplu  $w = w_1w_2w_3w_4 = 01^40^21^50 \in f(L)$ , unde  $w_1 = 0111$ ,  $w_2 = 1001$ ,  $w_3 = 11$ ,  $w_4 = 110$  și  $w_1, w_4 \in f(a)$ ,  $w_2, w_3 \in f(b)$ . Acest cuvânt poate fi obținut direct folosind derivarea

$$\begin{aligned} S &\Rightarrow S_1 \Rightarrow 0S_1 \Rightarrow 01A_1 \Rightarrow 01^2B_1 \Rightarrow 01^3Y \Rightarrow 01^3A \Rightarrow 01^3S_2 \Rightarrow 01^4A_2 \Rightarrow \\ &\Rightarrow 01^40A_2 \Rightarrow 01^40^2A_2 \Rightarrow 01^40^21Z \Rightarrow 01^40^21B \Rightarrow 01^40^21S_2 \Rightarrow 01^40^21^2A_2 \Rightarrow \\ &\Rightarrow 01^40^21^3Z \Rightarrow 01^40^21^3C \Rightarrow 01^40^21^3S_1 \Rightarrow 01^40^21^4A_1 \Rightarrow 01^40^21^5B_1 \Rightarrow \\ &\Rightarrow 01^40^21^50Y \Rightarrow 01^40^21^50X \Rightarrow 01^40^21^50. \end{aligned}$$

**Teorema 4.6**  $\mathcal{L}_2$  este închisă la substituție.

*Demonstrație:* Fie  $L \in \mathcal{L}_2$  un limbaaj și  $G = (V_N, \{a_1, \dots, a_n\}, S, P)$  gramatica independentă de context care-l generează, iar  $f$  o substituție. Fie  $G_i = (V_N^i, V_T^i, S_i, P_i)$  gramaticile independente de context cu  $L(G_i) = f(a_i)$  ( $1 \leq i \leq n$ ). Similar demonstrației anterioare, vom presupune că toate alfabetele de neterminale  $V_N^i$  sunt disjuncte două câte două.

Construim gramatica  $G' = (V_N', V_T', S, P')$  unde

$$V_N' = \left( \bigcup_{1 \leq i \leq n} V_N^i \right) \cup V_N, \quad V_T' = \bigcup_{1 \leq i \leq n} V_T^i.$$

Fie  $h$  substituția definită prin

$$h(a_i) = \{S_i\} \quad (1 \leq i \leq n), \quad h(A) = \{A\} \quad \forall A \in V_N.$$

$$\text{Atunci } P' = \left( \bigcup_{1 \leq i \leq n} P_i \right) \cup \{A \longrightarrow h(\alpha) \mid A \longrightarrow \alpha \in P\}.$$

$G'$  este evident tot independentă de context; se arată ușor că  $f(L(G)) = L(G')$ . q.e.d.

**Exemplul 4.7** Fie limbajul  $L = \{0^n 1^n \mid n \geq 1\}$ , generat de gramatica cu producțiile

$$S \longrightarrow 0S1 \mid 01$$

și să luăm substituția  $f$  definită în Exemplul 4.5.

$f(0) = \{a\}$  este generat de gramatica cu producția  $S_1 \longrightarrow a$ , iar

$f(1) = \{w\tilde{w} \mid w \in \{b, c\}^*\}$ , de gramatica cu producțiile  $S_2 \longrightarrow bS_2b \mid cS_2c \mid \epsilon$ .

Atunci limbajul  $f(L)$  va fi generat de gramatica  $G = (\{S, S_1, S_2\}, \{a, b, c\}, S, P)$  unde  $P = \{S_1 \longrightarrow a, \quad S_2 \longrightarrow bS_2b \mid cS_2c \mid \epsilon, \quad S \longrightarrow S_1SS_2 \mid S_1S_2\}$ .

**Corolarul 4.2**  $\mathcal{L}_2, \mathcal{L}_3$  sunt închise la substituții finite, homomorfisme și homomorfisme inverse.

Teorema rămâne adevărată și pentru  $\mathcal{L}_0$ . În schimb, pentru limbajele dependente de context, problema este mai dificilă. Se arată ([7], pp. 125 – 128 sau [14] pagina 90) că  $\mathcal{L}_1$  nu este închisă la substituții, substituții finite și homomorfisme, dar este închisă la substituții și homomorfisme  $\epsilon$  - libere.

## 4.4 Închiderea față de alte operații

### 4.4.1 Închiderea la intersecție

**Teorema 4.7**  $\mathcal{L}_3$  este închisă la intersecție.

*Demonstrație:* Fie limbajele  $L_1$  și  $L_2$  peste  $V$ , generate de gramaticile regulate  $G_i = (V_N^i, V_T^i, S_i, P_i)$ ,  $i = 1, 2$ .

Limbajul  $L_1 \cap L_2$  este generat de gramatica  $G' = (V_N^1 \times V_N^2, V_T^1 \cap V_T^2, [S_1, S_2], P')$  unde

$$P' = \{[A, B] \longrightarrow a[C, D] \mid A \longrightarrow aC \in P_1, B \longrightarrow aD \in P_2\} \cup \{[A, B] \mid A \longrightarrow a \in P_1, B \longrightarrow a \in P_2\}$$

Lăsăm ca exercițiu demonstrarea egalității  $L(G') = L_1 \cap L_2$ .

q.e.d.

**Teorema 4.8**  $\mathcal{L}_0$  este închisă la intersecție.

*Demonstrație:* Fie  $G_i = (V_N^i, V_T^i, S_i, P_i)$  două gramatici de tip 0 cu  $L(G_i) = L_i$ . Putem presupunem – așa cum mai mai făcut – că  $V_N^1 \cap V_N^2 = \emptyset$ .

Construim gramatica  $G' = (V_N^1 \cup V_N^2 \cup V_N^3, V_T^1 \cup V_T^2, S_0, P_1 \cup P_2 \cup P_3)$ , unde

- $V_N^3 = \{A_a \mid a \in V_T^1 \cup V_T^2\} \cup \{S_0, Y, Z\}$ ;
- $P_3 = \begin{cases} S_0 \longrightarrow YS_1ZS_2Y & (1) \\ Za \longrightarrow A_aZ & \text{dacă } a \in V_T^1 \cup V_T^2 & (2) \\ bA_a \longrightarrow A_ab & \text{dacă } a, b \in V_T^1 \cup V_T^2 & (3) \\ YA_aa \longrightarrow aY & \text{dacă } a \in V_T^1 \cup V_T^2 & (4) \\ YZY \longrightarrow \epsilon & (5) \end{cases}$

Folosind (1) precum și producțiile din  $P_1 \cup P_2$  vom obține derivări de forma

$$S_0 \xRightarrow{*} Yw_1Zw_2Y$$

cu  $w_i \in L_i$  ( $i = 1, 2$ ).

Cu regulile (2) – (4), această formă sentențială poate deriva în  $wYZY$  dacă și numai dacă  $w = w_1 = w_2$ . Regula (5) va șterge sufixul  $YZY$ , rămânând numai  $w \in V_T^1 \cap V_T^2$ .

Cum aceasta este singura posibilitate de a obține un cuvânt terminal plecând din  $S_0$ , rezultă  $L(G') = L_1 \cap L_2$ . q.e.d.

Vom studia mai târziu problemele închiderii lui  $\mathcal{L}_2$  și  $\mathcal{L}_1$  la intersecție.

#### 4.4.2 Înciderea la oglindire

Fie  $L \subseteq V^*$  un limbaj nevid. Considerăm limbajul  $\tilde{L} = \{\tilde{w} \mid w \in L\}$ .

**Teorema 4.9** *Dacă  $L \in \mathcal{L}_i$  ( $0 \leq i \leq 3$ ) atunci  $\tilde{L} \in \mathcal{L}_i$ .*

*Demonstrație:* Dacă  $L \in \mathcal{L}_i$ , atunci există o gramatică  $G = (V_N, V_T, S, P)$  de tip  $i$  cu  $L(G) = L$  ( $0 \leq i \leq 3$ ).

Construim gramatica  $G_1 = (V_N, V_T, S, P_1)$  unde  $P_1 = \{\tilde{\alpha} \longrightarrow \tilde{\beta} \mid \alpha \longrightarrow \beta \in P\}$ .

Gramatica  $G_1$  este evident de același tip cu  $G$ . Mai rămâne de arătat că  $L(G_1) = \tilde{L}$ .

Este suficient să demonstrăm că  $x \Longrightarrow y$  în gramatica  $G$  dacă și numai dacă  $\tilde{x} \Longrightarrow \tilde{y}$  în  $G_1$ .

Fie  $x \Longrightarrow y$  în gramatica  $G$ ; deci  $x = x_1\alpha x_2$ ,  $y = y_1\beta y_2$  și  $\alpha \longrightarrow \beta \in P$ . Dar atunci vom avea  $\tilde{x} = \tilde{x}_2\tilde{\alpha}\tilde{x}_1$ ,  $\tilde{y} = \tilde{y}_2\tilde{\beta}\tilde{y}_1$  și  $\tilde{\alpha} \longrightarrow \tilde{\beta} \in P_1$ , informații din care se deduce  $\tilde{x} \Longrightarrow \tilde{y}$  în gramatica  $G_1$ .

Implicația inversă se demonstrează similar.

Printr-un simplu procedeu de inducție va rezulta că  $S \xRightarrow{*} w$  este o derivare în gramatica  $G$  dacă și numai dacă  $S \xRightarrow{*} \tilde{w}$  în gramatica  $G_1$ . q.e.d.

### 4.4.3 Închiderea la operații de amestecare

Fie  $V$  un alfabet finit nevid și  $\alpha, \beta \in V^*$ . Se definește operația

$$Shuf(\alpha, \beta) = \{\alpha_1\beta_1\alpha_2\beta_2 \dots \alpha_p\beta_p \mid \alpha_i, \beta_i \in V^*, \alpha_1\alpha_2 \dots \alpha_p = \alpha, \beta_1\beta_2 \dots \beta_p = \beta, p \geq 1\}.$$

Pentru două limbaje  $L_1, L_2 \subseteq V^*$ ,

$$Shuf(L_1, L_2) = \{Shuf(\alpha, \beta) \mid \alpha \in L_1, \beta \in L_2\}.$$

În sfârșit, pentru  $L \subseteq V^*$  se definesc mulțimile

$$Shuf^0(L) = \{\epsilon\}, \quad Shuf^{n+1}(L) = Shuf(Shuf^n(L), L), \quad n \geq 0$$

și

$$Shuf^*(L) = \bigcup_{n=0}^{\infty} Shuf^n(L).$$

Deoarece clasa  $\mathcal{L}_2$  a limbajelor independente de context nu este închisă la operațiile  $Shuf$  și  $Shuf^*$  (demonstrarea acestor afirmații necesită noțiuni mai elaborate de limbaje formale), ne vom îndrepta atenția asupra proprietăților de închidere ale celorlalte clase de limbaje Chomsky.

**Teorema 4.10**  $\mathcal{L}_i$  ( $i = 0, 1, 3$ ) sunt închise la operația  $Shuf$ .

*Demonstrație:*

$i = 0, 1$ : Fie limbajele  $L_j \subseteq V_j^*$  de tip  $i$ , generate de gramaticile  $G_j = (V_N^j, V_j, S_j, P_j)$  ( $j = 1, 2$ ). Vom presupune  $V_N^1 \cap V_N^2 = \emptyset$ . Considerăm două alfabetele noi disjuncte  $U_1$  și  $U_2$ , precum și două homomorfisme bijective  $h_j : V_j \rightarrow U_j$  ( $j = 1, 2$ ).

Acum, pentru  $j = 1, 2$ :

- Extindem  $h_j$  la  $V_j \cup V_N^j$  punând  $h_j(A) = A$ ,  $\forall A \in V_N^j$ ;
- Fie  $h_j(P_j) = \{h_j(\alpha) \rightarrow h_j(\beta) \mid \alpha \rightarrow \beta \in P_j\}$ .

Construim gramatica  $G = (V_N, V_T, S, P)$  definită prin

$$V_N = V_N^1 \cup V_N^2 \cup U_1 \cup U_2 \cup \{S\},$$

$$V_T = V_1 \cup V_2,$$

$$P = h_1(P_1) \cup h_2(P_2) \cup \{S \rightarrow S_1S_2\} \cup \{AB \rightarrow BA \mid A \in U_1, B \in U_2\} \cup \{A \rightarrow a \mid A \in U_1, a \in V_1, h_1(a) = A\} \cup \{A \rightarrow a \mid A \in U_2, a \in V_2, h_2(a) = A\}.$$

Evident  $G$  este o gramatică de același tip cu  $G_1$  și  $G_2$ .

Orice derivare în gramatica  $G$  decurge astfel: după folosirea producției  $S \rightarrow S_1S_2$  se derivează independent în gramaticile  $G_1$  și  $G_2$ , obținându-se limbajul  $h_1(L_1)h_2(L_2)$ . Cu

regulile de forma  $AB \rightarrow BA$  simbolurile din  $U_2$  sunt transportate spre stânga, iar cele din  $U_1$ , spre dreapta. Când se trece la o secvență de terminale, se obține un cuvânt din limbajul  $Shuf(L_1, L_2)$ .

Invers, orice cuvânt  $w \in Shuf(L_1, L_2)$  se poate obține în acest fel, deci are loc egalitatea  $L(G) = Shuf(L_1, L_2)$ .

$i = 3$ : Dacă  $L_j \subseteq V_j^*$  sunt limbaje generate de gramaticile regulate  $G_j = (V_N^j, V_j, S_j, P_j)$  ( $j = 1, 2$ ), construim gramatica regulată  $G = (V_N, V_T, S, P)$ , unde  $V_N = (V_N^1 \times V_N^2) \cup (V_N^1 \times \{F\}) \cup (\{F\} \times V_N^2)$ ,  $F \notin V_N^1 \cup V_N^2$ ,  $V_T = V_1 \cup V_2$ ,  $S = (S_1, S_2)$ ,  $P = \{(A, B) \rightarrow a(C, B) \mid A, C \in V_N^1, B \in V_N^2 \cup \{F\}, a \in V_1, A \rightarrow aC \in P_1\} \cup \{(A, B) \rightarrow a(F, B) \mid A \in V_N^1, B \in V_N^2, a \in V_1, A \rightarrow a \in P_1\} \cup \{(A, F) \rightarrow a \mid A \in V_N^1, a \in V_1, A \rightarrow a \in P_1\} \cup \{(A, B) \rightarrow a(A, C) \mid A \in V_N^1 \cup \{F\}, B, C \in V_N^2, a \in V_2, B \rightarrow aC \in P_2\} \cup \{(A, B) \rightarrow a(A, F) \mid A \in V_N^1, B \in V_N^2, a \in V_2, B \rightarrow a \in P_2\} \cup \{(F, B) \rightarrow a \mid B \in V_N^2, a \in V_2, A \rightarrow a \in P_2\}$

Orice derivare în gramatica  $G$  este compusă din două derivări: una în gramatica  $G_1$  pe prima componentă a neterminalelor din  $V_N$  și una în gramatica  $G_2$  pe a doua componentă; cele două derivări se desfășoară în paralel, intercalând pașii lor de derivare.

Invers, din oricare două derivări – una în raport cu  $G_1$  și cealaltă în raport cu  $G_2$  – se pot obține toate derivările intercalate.

În concluzie,  $L(G) = Shuf(L_1, L_2)$ .

q.e.d.

**Teorema 4.11**  $\mathcal{L}_3$  nu este închisă la operația  $Shuf^*$ . Mai mult, există limbaje finite  $L$  pentru care  $Shuf^*(L) \notin \mathcal{L}_2$ .

*Demonstrație:* Fie limbajul finit  $L = \{abc\}$ ; evident, acesta este regulat. Considerăm limbajele

$$L_1 = \{a^n b^m c^p \mid n, m, p \geq 1\}, \quad L_2 = \{a^n b^n c^n \mid n \geq 1\}.$$

Limbajul  $L_1$  este evident regulat, iar  $L_2$  nu este independent de context (afirmație care se va demonstra în Capitolul 6). Avem egalitatea  $Shuf^*(L) \cap L_1 = L_2$ .

Într-adevăr, să observăm că în  $Shuf^*(L)$ , orice șir conține exact  $n$  simboluri ' $a$ ',  $n$  simboluri ' $b$ ' și  $n$  simboluri ' $c$ '. Deci  $Shuf^*(L) \cap L_1 = \{a^n b^n c^n\}$ . Dar

$$L_2 = \bigcup_{n=1}^{\infty} \{a^n b^n c^n\} = \bigcup_{n=1}^{\infty} (Shuf^n(L) \cap L_1) = \left( \bigcup_{n=1}^{\infty} Shuf^n(L) \right) \cap L = Shuf^*(L) \cap L_1.$$

$\mathcal{L}_3$  este închisă la intersecție. Dacă  $Shuf^*(L)$  ar fi limbaj regulat, atunci și intersecția lui cu  $L_1$  ar fi regulat, contradicție.

q.e.d.



**Teorema 4.12**  $\mathcal{L}_0$  este închisă la operația  $Shuf^*$ .

*Demonstrație:* Fie limbajul  $L \in \mathcal{L}_0$  generat de gramatica  $G = (V_N, V_T, S, P)$ ; construim limbajul

$$L_1 = \{R\}L\{C\}(\{A\}L)^+\{D\} \cup \{R\}L\{D\} \cup \{\lambda\}$$

unde  $R, C, A, D \notin V_N$ . Evident,  $L_1 \in \mathcal{L}_0$ .

Fie  $G' = (V_N', V_T, S, P')$  gramatica de tip 0 definită:

$$V_N' = V_N \cup \{R, D, C, A, B, E\} \cup \{A_a, a' \mid a \in V_T\} \quad (B, E \notin V_N),$$

$$P' = P_1 \cup \{S \rightarrow \lambda, D \rightarrow \lambda, CA \rightarrow AB\} \cup \{BA_a \rightarrow a'B \mid a \in V_T\} \cup \\ \cup \{Aa' \rightarrow a'A \mid a \in V_T\} \cup \{A_b a' \rightarrow a'A_b \mid a, b \in V_T\} \cup \{ABA \rightarrow EA, ABD \rightarrow D\} \cup \\ \cup \{A_a E \rightarrow EA_a \mid a \in V_T\} \cup \{SE \rightarrow SC\} \cup \{a'E \rightarrow EA_a \mid a \in V_T\} \cup \\ \cup \{CA_a \rightarrow A_a C \mid a \in V_T\} \cup \{A_a \rightarrow a \mid a \in V_T\},$$

unde  $P_1$  se obține din  $P$  prin înlocuirea tuturor terminalelor  $a \in V_T$  cu  $A_a$ .

Se poate arăta că  $L(G') = Shuf^*(L)$ .

q.e.d.

Clasa  $\mathcal{L}_1$  este de asemenea închisă la operația  $Shuf^*$ , dar demonstrația necesită elemente specifice acestei clase de limbaje.

## 4.5 Exerciții

**Exercițiul 4.1** Să se construiască gramatici regulate echivalente cu gramaticile de producții

- a)  $S \rightarrow abS|baS|A, \quad A \rightarrow aabb|S|\epsilon;$
- b)  $E \rightarrow a + T|T, \quad T \rightarrow a * F|F, \quad F \rightarrow (E)|a;$
- c)  $S \rightarrow A|B, \quad A \rightarrow abc|aA|\epsilon, \quad B \rightarrow abC|C, \quad C \rightarrow cB|\epsilon.$

**Exercițiul 4.2** Să se construiască gramatici regulate pentru limbajele  $L_1 \cup L_2$  și  $L_1 L_2$ , unde

$$L_1 = \{a^i \mid i \geq 0\}, \quad L_2 = \{(ab)^i \mid i \geq 1\}$$

**Exercițiul 4.3** Să se construiască gramatici pentru limbajele  $L_1 \cup L_2$  și  $L_1 L_2$ , unde

$$L_1 = \{0^i 10^j \mid i \geq j \geq 1\}, \quad L_2 = \{0^i 10^j \mid j \geq i \geq 1\}$$

**Exercițiul 4.4** Să se construiască gramatici independente de context pentru limbajele

$$L_1 = \{a^n b^{2n} \mid n \geq 0\} \cup \{a^n b^n \mid n \geq 0\}, \quad L_2 = \{a^i b^j c^k \mid i = j \text{ sau } j = k, \quad i, j, k \geq 1\}$$

**Exercițiul 4.5** Fie limbajul  $L = \{w \mid w \in \{a, b\}^*, |w|_a = |w|_b\}$ , generat de o gramatică independentă de context  $G$ . Să se construiască o gramatică pentru  $L^*$  și să se arate că această gramatică este echivalentă cu  $G$ .

# Capitolul 5

## Metode de recunoaștere pentru limbajele Chomsky

În clasificarea Chomsky, pentru fiecare  $i$  ( $0 \leq i \leq 3$ ) s-a notat cu  $\mathcal{L}_i$  clasa limbajelor care pot fi generate de gramaticile de tip  $i$ . Similitudinea dintre metodele generative și cele de recunoaștere sugerează că pentru fiecare limbaj  $L$  de tip  $i$ , înafara gramaticii de tip  $i$  care îl generează, există și o metodă de recunoaștere – un anumit tip de automat specific clasei  $\mathcal{L}_i$ .

Deci clasificarea Chomsky este posibilă nu numai pe baza gramaticilor, ci și pe baza automatelor.

**Definiția 5.1** *O gramatică  $G$  este echivalentă cu un automat  $M$  dacă  $L(G) = \tau(M)$ .*

### 5.1 Gramatici regulate și automate finite

În cazul  $\mathcal{L}_3$ , tipul de automat folosit este automatul finit ( $AF$ ) definit în Capitolul 1. În Capitolul 3 am arătat că automatele finite sunt metode de recunoaștere. Mai rămâne de arătat un singur lucru; anume:

**Teorema 5.1** *Pentru fiecare gramatică regulată  $G = (V_N, V_T, S, P)$  există un automat finit  $M = (Q, V, \delta, q_0, F)$  echivalent, și reciproc.*

*Demonstrație:*

” $\implies$ ”: Fie  $G = (V_N, V_T, S, P)$  o gramatică regulată cu  $L(G) = L$ . Fără a micșora generalitatea, putem presupune că ea are numai reguli de forma  $A \longrightarrow aB$  sau  $A \longrightarrow \epsilon$  (producțiile  $A \longrightarrow a$  cu  $a \in V_T$  se înlocuiesc cu  $A \longrightarrow aX$ , unde  $X$  este un neterminal nou, și în plus se introduce și regula  $X \longrightarrow \epsilon$ ).

Construim automatul  $M = (Q, V, \delta, q_0, F)$  astfel:

$Q = V_N$ ,  $V = V_T$ ,  $q_0 = S$ ,  $F = \{A \mid A \longrightarrow \epsilon \in P\}$  și  $\delta(A, a) = \{B \mid A \longrightarrow aB \in P\}$ .

Rămâne de arătat egalitatea  $L = \tau(M)$ .

" $\subseteq$ ": Fie  $w = a_1 a_2 \dots a_n \in L$ ; deci există derivarea

$$S \Longrightarrow a_1 A_1 \Longrightarrow a_1 a_2 A_2 \Longrightarrow \dots \Longrightarrow a_1 a_2 \dots a_n A_n \Longrightarrow a_1 a_2 \dots a_n.$$

Regulilor  $S \longrightarrow a_1 A_1$ ,  $A_1 \longrightarrow a_2 A_2$ ,  $\dots$ ,  $A_{n-1} \longrightarrow a_n A_n \in P$  le corespund prin construcție apartenențele

$A_1 \in \delta(S, a_1)$ ,  $A_2 \in \delta(A_1, a_2)$ ,  $\dots$ ,  $A_n \in \delta(A_{n-1}, a_n)$ ,  $A_n \in F$ ; atunci

$\delta(S, a_1 a_2 \dots a_n) = \delta(\delta(S, a_1), a_2 \dots a_n) \stackrel{(*)}{=} V_1 \cup \delta(A_1, a_2 \dots a_n) = \dots \stackrel{(*)}{=} V_n \cup \delta(A_n, \epsilon) = V_n \cup \{A_n\}$  și  $A_n \in F$ ; deci  $w \in \tau(M)$ .

Pentru fiecare egalitate  $(*)$  s-au notat cu  $V_i \subseteq Q$  ( $1 \leq i \leq n$ ) mulțimi de stări (posibil mulțimi vide).

" $\supseteq$ ": Vom demonstra prin inducție după  $n$  afirmația

$$\forall A \in Q \quad [\delta(A, a_1 a_2 \dots a_n) \cap F \neq \emptyset \implies A \xRightarrow{*} a_1 a_2 \dots a_n].$$

$\boxed{n=1}$ : Avem  $\delta(A_1, a_1) \cap F \neq \emptyset$ ; dacă  $\delta(A, a_1) = \{B \mid A \longrightarrow a_1 B \in P\}$  și  $F = \{B \mid B \longrightarrow \epsilon \in P\}$ , ipoteza de inducție asigură existența unui element  $B \in \delta(A, a_1) \cap F$ , deci  $A \Longrightarrow a_1 B$  și  $B \Longrightarrow \epsilon$ . Concluzie:  $A \xRightarrow{*} a_1$ .

$\boxed{n \rightarrow n+1}$ : Fie  $\delta(A, a_1 a_2 \dots a_{n+1}) \cap F \neq \emptyset$ . Detaliind această relație, avem

$$\begin{aligned} \delta(A, a_1 a_2 \dots a_{n+1}) \cap F &= \delta(\delta(A, a_1), a_2 \dots a_{n+1}) \cap F = \\ &= \left[ \bigcup_{B \in \delta(A, a_1)} \delta(B, a_2 \dots a_{n+1}) \right] \cap F = \bigcup_{b \in \delta(A, a_1)} [\delta(B, a_2 \dots a_{n+1}) \cap F] \neq \emptyset. \end{aligned}$$

Deoarece reuniunea este nevidă, există cel puțin un  $B \in \delta(A, a_1)$  cu  $\delta(B, a_2 \dots a_{n+1}) \cap F \neq \emptyset$ . Ipoteza de inducție asigură atunci că  $B \xRightarrow{*} a_2 \dots a_{n+1}$ .

Pe de altă parte, din  $B \in \delta(A, a_1)$  avem  $A \Longrightarrow a_1 B$ . Din cele două rezultate se obține  $A \xRightarrow{*} a_1 a_2 \dots a_{n+1}$ .

În particular, pentru  $A \equiv S$  avem  $[\delta(S, w) \cap F \neq \emptyset \implies S \xRightarrow{*} w]$ . Deci

$$w \in \tau(M) \implies w \in L.$$

" $\Leftarrow$ ": Fie  $M = (q, V, \delta, q_0, F)$  un AFD cu  $L = \tau(M)$ .

Gramatica echivalentă  $G = (V_N, V_T, S, P)$  se definește astfel:  $V_N = Q$ ,  $V_T = V$ ,  $S = q_0$ , iar pentru  $\epsilon \notin L$ , mulțimea  $P$  a producțiilor este

$$P = \{p \longrightarrow aq \mid \delta(p, a) = q\} \cup \{p \longrightarrow a \mid \delta(p, a) \in F\}.$$

Gramatica  $G$  este evident regulată. Rămâne de arătat egalitatea  $L = L(G)$ , pe care o vom demonstra folosind echivalența

$$\delta(p, \alpha) = q \iff p \xRightarrow{*} \alpha q \quad (1)$$

" $\subseteq$ ": Fie  $\alpha a \in L$  (se folosește faptul că  $\alpha a \neq \epsilon$ ). Deci  $\delta(q_0, \alpha) = p$ ; aceasta înseamnă – cu  $(a)$  – că există derivarea  $q_0 \xRightarrow{*} \alpha p$ . În plus, cum  $\delta(p, a) \in F$ , rezultă că  $p \rightarrow a$  este o regulă de producție. Deci  $q_0 \xRightarrow{*} \alpha p \xRightarrow{*} \alpha a$ .

" $\supseteq$ ": Fie  $q_0 \xRightarrow{*} \beta$ . Atunci  $\beta = \alpha a$  ( $\beta \neq \epsilon$ ) și – punând în evidență ultimul pas al derivării,  $q_0 \xRightarrow{*} \alpha p \xRightarrow{*} \alpha a$  pentru un anumit  $p \in V_N$ . Revenind la automatul inițial  $M$ ,  $\delta(q_0, \alpha) = p$  și  $\delta(p, a) \in F$ , deci

$$\delta(q_0, \alpha a) = \delta(\delta(q_0, \alpha), a) = \delta(p, a) \in F \implies \alpha a = \beta \in L$$

În cazul când  $\epsilon \in L$ , avem  $q_0 \in F$ . La gramatica  $G$  construită mai sus (care generează limbajul  $L \setminus \{\epsilon\}$ ) se introduce un simbol de start nou  $S$  și se adaugă producțiile  $S \rightarrow q_0 | \epsilon$ . Conform cu Lema 4.3, această gramatică este echivalentă cu o gramatică regulată  $G'$ . q.e.d.

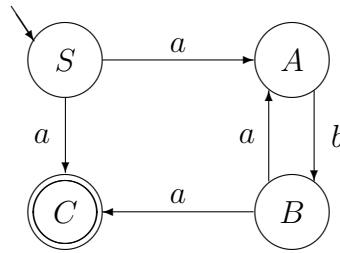
**Exemplul 5.1** Fie gramatica regulată  $G = (V_N, V_T, S, P)$  cu  $V_N = \{S, A, B\}$ ,  $V_T = \{a, b\}$  și  $P = \{S \rightarrow aA | a, A \rightarrow bB, B \rightarrow aA | a\}$  care generează limbajul

$$L = \{a(ba)^n \mid n \geq 0\} = a(ba)^*.$$

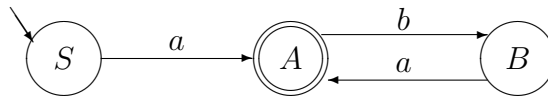
Pentru a construi automatul finit echivalent, se introduce un neterminat nou  $C$  și se transformă setul de producții astfel:

$$P = \{S \rightarrow aA | aC, A \rightarrow bB, B \rightarrow aA | aC, C \rightarrow \epsilon\}$$

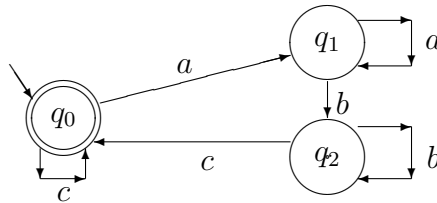
Automatul finit este descris atunci de diagrama de tranziție



sau – dacă se solicită să fie determinist:



**Exemplul 5.2** Fie automatul finit descris de diagrama de tranziție:



Gramatica regulată asociată va fi în prima etapă (când  $\epsilon \notin L$ ):  $V_N = \{q_0, q_1, q_2\}$ ,  $V_T = \{a, b, c\}$ ,  $S = q_0$ ,  $P_1 = \{q_0 \rightarrow aq_1|cq_0|c, q_1 \rightarrow aq_1|bq_2, q_2 \rightarrow bq_2|cq_0|c\}$ .

Deoarece  $\epsilon \in L$ , se introduce un simbol de start nou –  $S$ , și regulile  $S \rightarrow q_0|\epsilon$ . Rebotezând neterminalele în notația uzuală, avem gramatica  $G = (V_N, V_T, S, P)$  cu  $V_N = \{S, A, B, C\}$ ,  $V_T = \{a, b, c\}$ ,

$$P = \{S \rightarrow A|\epsilon, A \rightarrow aB|cA|c, B \rightarrow aB|bC, C \rightarrow bC|cA|c\}.$$

Dacă eliminăm regula  $S \rightarrow A$  (nespecifică unei gramatici regulate), se ajunge la gramatica regulată, de producții

$$P = \{S \rightarrow aB|cA|c|\epsilon, A \rightarrow aB|cA|c, B \rightarrow aB|bC, C \rightarrow bC|cA|c\}.$$

De remarcat că ea este echivalentă cu gramatica regulată de producții

$$A \rightarrow aB|cA|\epsilon, B \rightarrow aB|bC, C \rightarrow bC|cA$$

și simbol de start  $A$ .

## 5.2 Gsm-uri și translatori finiți

### 5.2.1 Definiții generale

**Definiția 5.2** Se numește mașină secvențială generalizată (gsm) structura  $M = (Q, V_i, V_e, \delta, q_0, F)$  unde:

- $Q, V_i, V_e$  sunt mulțimi finite, nevide numite respectiv mulțime de stări, alfabet de intrare, alfabet de ieșire;
- $q_0 \in Q$  este starea inițială;
- $F \subseteq Q$ ,  $F \neq \emptyset$  este mulțimea stărilor finale;
- $\delta : Q \times V_i \rightarrow 2^{Q \times V_e^*}$  este funcția de tranziție.

Aserțiunea  $(p, w) \in \delta(q, a)$  poate fi interpretată în felul următor: automatul aflat în starea  $q$ , având la intrare caracterul  $a$ , poate trece în starea  $p$ , emițând la ieșire cuvântul  $w$ .

Funcția de tranziție  $\delta$  poate fi extinsă la  $Q \times V_i^*$  astfel:  $\forall q \in Q, a \in V_i, \alpha \in V_i^*$ ,

$$i. \delta(q, \epsilon) = \{(q, \epsilon)\};$$

$$ii. \delta(q, \alpha a) = \{(p, w) | w = w_1 w_2, \exists r \in Q, (r, w_1) \in \delta(q, \alpha), (p, w_2) \in \delta(r, a)\}.$$

Extensia este naturală, fapt arătat de cazul particular  $\alpha = \epsilon$ .

Dacă  $\delta : Q \times V_i \rightarrow 2^{Q \times V_e^+}$ , atunci  $M$  este un gsm  $\epsilon$  - liber.

Similar automatelor finite, putem reprezenta un gsm ca un graf având drept noduri elementele lui  $Q$ , iar arcele de tipul:

$$\textcircled{q} \xrightarrow{a/w} \textcircled{p} \iff (p, w) \in \delta(q, a)$$

Fie  $M = (Q, V_i, V_e, \delta, q_0, F)$  un gsm și  $\alpha \in V_i^*$ . Definim

$$M(\alpha) = \{\beta \mid \exists p \in F, (p, \beta) \in \delta(q_0, \alpha)\}.$$

Dacă  $L \subseteq V_i^*$  este un limbaj, atunci putem defini

$$M(L) = \{\beta \mid \exists \alpha \in L, \beta \in M(\alpha)\} = \bigcup_{\alpha \in L} M(\alpha).$$

Spunem în acest caz că  $M(L)$  este o *aplicație gsm* (de variabilă  $L$ ).

Dacă  $M$  este  $\epsilon$  - liber, atunci  $M(L)$  este o aplicație gsm  $\epsilon$  - liberă.

Similar vom defini

$$M^{-1}(\beta) = \{\alpha \mid \beta \in M(\alpha)\}$$

respectiv

$$M^{-1}(L) = \{\beta \mid \exists \alpha \in L, \alpha \in M(\beta)\}.$$

$M^{-1}$  este o aplicație *gsm - inversă*. De remarcat că – în ciuda denumirii – egalitățile

$$M^{-1}(M(L)) = M(M^{-1}(L)) = L$$

nu sunt în general adevărate.

Pentru a facilita operarea cu mașinile secvențiale generalizate, putem lucra cu *descrieri instantanee* (*DI*) care sunt triplete de forma  $(q, \alpha, \beta)$  unde:

$$q \in Q;$$

$$\alpha \in V_i^* \text{ este o stivă cu vârful la stânga;}$$

$$\beta \in V_e^* \text{ este o stivă cu vârful la dreapta.}$$

Pe aceste triplete se definește relația  $\vdash$  astfel:

$$\forall p, q \in Q, a \in V_i, \alpha \in V_i^*, \beta, y \in V_e^* [(q, a\alpha, \beta) \vdash (p, \alpha, \beta y) \iff (p, y) \in \delta(q, a)].$$

Dacă  $\vdash^*$  este închiderea reflexiv - tranzitivă a acestei relații, putem defini *translatarea* unui gsm  $M$  ca fiind:

$$\tau(M) = \{(\alpha, \beta) \mid \alpha \in V_i^*, \beta \in V_e^*, (q_0, \alpha, \epsilon) \vdash^* (p, \epsilon, \beta), p \in F\}.$$

Atunci definițiile anterioare se exprimă astfel:

$$M(\alpha) = \{\beta \mid (\alpha, \beta) \in \tau(M)\}, \quad M^{-1}(\beta) = \{\alpha \mid (\alpha, \beta) \in \tau(M)\}.$$

**Exemplul 5.3** Să considerăm  $M = (\{q_0, q_1\}, \{0, 1\}, \{a, b\}, \delta, q_0, \{q_1\})$  unde:

$$\begin{aligned} \delta(q_0, 0) &= \{(q_0, aa), (q_1, b)\}, & \delta(q_1, 0) &= \emptyset, \\ \delta(q_0, 1) &= \{(q_0, a)\}, & \delta(q_1, 1) &= \{(q_1, e)\}. \end{aligned}$$

Fie  $L = \{0^n 1^n \mid n \geq 1\}$ . Se arată prin inducție după  $n$  că

$$(q_0, 0^n 1^n, \epsilon) \vdash^* (q_1, \epsilon, a^{2n-2}b).$$

Deci  $M(L) = \{a^{2n}b \mid n \geq 0\}$ .

Să luăm acum  $L_1 = \{a^{2n}b \mid n \geq 0\}$ . Căutând toate cuvintele  $w \in \{0, 1\}^*$  cu proprietatea  $(q_0, w, \epsilon) \vdash^* (q_1, \epsilon, a^{2n}b)$ , găsim  $w \in \{\alpha 01^k \mid k \geq 0, |\alpha|_1 = \mathcal{M}2\}$ .

Deci  $M^{-1}(L) = \{w 01^k \mid k \geq 0, |w|_2 = \mathcal{M}2\}$ .

Se observă că  $M^{-1}(M(L)) \supset L$ .

Aplicațiile gsm constituie o modalitate uzuală de exprimare a unui limbaj în termenii unui alt limbaj având aceeași structură, dar cu elemente exterioare distincte.

De exemplu,  $L_1 = \{a^n b^n \mid n \geq 1\}$  și  $L_2 = \{a^n b a^n \mid n \geq 1\}$  au – într-un anumit sens – aceeași structură, ușor modificată. Putem exprima două gsm-uri care transformă – unul pe  $L_1$  în  $L_2$ , celălalt pe  $L_2$  în  $L_1$ :



Dintre cazurile cele mai cunoscute de extensie a mașinilor secvențiale generalizate, prezentăm două: *automatele finite cu ieșiri* (Moore/Mealy) și *translatorii finiți*.

### 5.2.2 Automate finite cu ieșiri

În general, automatele finite privite ca metode de recunoaștere pot fi considerate ca având la ieșire un semnal binar: *acceptat/neacceptat*. Pentru o rafinare a semnalului de ieșire, s-au dezvoltat două tipuri distincte de automate: cele în care ieșirea corespunde unei stări (automat sau mașină Moore) și cele în care ieșirea corespunde unei tranziții (automat sau mașină Mealy).

**Definiția 5.3** Un automat Moore este o structură  $AM = (Q, V_i, V_e, \delta, \lambda, q_0)$  unde  $Q$ ,  $V_i$ ,  $V_e$ ,  $q_0$  au semnificațiile uzuale,  $\delta : Q \times V_i \longrightarrow Q$  este funcția de tranziție, iar  $\lambda : Q \longrightarrow V_e$  este funcția de ieșire.

Ieșirea unui automat Moore  $AM$  ca răspuns la cuvântul de intrare  $w = a_1 a_2 \dots a_n \in V_i^*$  ( $n \geq 0$ ) este

$$\lambda(q_0)\lambda(q_1)\dots\lambda(q_n) \in V_e^*$$

unde  $q_0, q_1, \dots, q_n$  este secvența de stări definită  $\delta(q_{i-1}, a_i) = q_i$ , ( $1 \leq i \leq n$ ).

#### Observația 5.1

1. La intrarea  $\epsilon$ , răspunsul de ieșire este  $\lambda(q_0)$ ;

2. Automatul finit determinist este un caz particular de automat Moore în care  $V_e = \{0, 1\}$  și  $F = \{q \mid \lambda(q) = 1\}$ .

**Definiția 5.4** Un automat Mealy este o structură  $AM = (Q, V_i, V_e, \delta, \lambda, q_0)$  unde  $Q, V_i, V_e, \delta, q_0$  au semnificațiile de la automatul Moore, iar funcția de ieșire este  $\lambda : Q \times V_i \longrightarrow V_e$ .

$\lambda(q, a)$  dă ieșirea asociată tranziției din starea  $q$  cu intrarea  $a$ . Ieșirea unui automat Mealy  $AM$  ca răspuns la cuvântul de intrare  $a_1 a_2 \dots a_n$  ( $n \geq 0$ ) este

$$\lambda(q_0, a_1) \lambda(q_1, a_2) \dots \lambda(q_{n-1}, a_n)$$

unde  $q_0, q_1, \dots, q_n$  este secvența de stări definită prin  $\delta(q_{i-1}, a_i) = q_i$ , ( $1 \leq i \leq n$ ).

**Observația 5.2**

1. Pentru un cuvânt de intrare de lungime  $n$ , ieșirea este un cuvânt de lungime  $n + 1$  la un automat Moore și de lungime  $n$  la un automat Mealy.
2. Pentru automatele Mealy, la intrarea  $\epsilon$ , răspunsul este  $\epsilon$ .

Ambele tipuri de automate (Moore și Mealy) pot fi tratate în mod unitar ca particularizări ale automatelor Starke, a căror definiție este:

**Definiția 5.5** Un automat Starke este o structură  $AS = (Q, V_i, V_e, h, q_0)$  unde  $Q, V_i, V_e, q_0$  au aceleași semnificații ca mai sus, iar  $h : Q \times V_i \longrightarrow 2^{Q \times V_e}$ .

Un automat Starke diferă de un gsm prin faptul că:

1. Nu are stări finale;
2. Dacă  $(p, x) \in \delta(q, a)$  atunci  $|x| = 1$ .

De fapt, un automat Starke este un gsm în care  $F = Q$  și  $\delta : Q \times V_i \longrightarrow 2^{Q \times V_e}$ .

Automatul Moore este un caz particular de automat Starke (gsm) în care funcția  $h$  este definită

$$h(q, a) = (\delta(q, a), \lambda(q)).$$

Pentru automatul Mealy,

$$h(q, a) = (\delta(q, a), \lambda(q, a)).$$

Reciproc, un automat Starke permite obținerea unui automat Mealy (sau Moore) astfel:

Fie  $A_1, A_2, \dots, A_n$  mulțimi finite nevide; definim funcția *proiecție*

$$pr_k : A_1 \times A_2 \times \dots \times A_n \longrightarrow A_k \quad (1 \leq k \leq n)$$

în felul următor: dacă  $a = (a_1, a_2, \dots, a_n)$  atunci  $pr_k(a) = a_k$ .



Acum, pentru un automat Starke  $AS = (Q, V_i, V_e, h, q_0)$  se construiește automatul Mealy  $AM = (Q, V_i, V_e, \delta, \lambda, q_0)$  unde  $\delta(q, a) = pr_1(h(q, a))$ ,  $\lambda(q, a) = pr_2(h(q, a))$ .

Evident, comportările celor două automate sunt identice.

Pentru automatul Moore singura modificare este  $\lambda(q) = pr_2(h(q, a))$ .

Fie  $M = (Q, V_i, V_e, \delta, \lambda, q_0)$  un automat Mealy sau Moore; definim aplicația  $f_M : V_i^* \rightarrow V_e^*$  astfel:

$$f_M(x) = y \iff (x, y) \in \tau(M).$$

Dacă  $M$  este un automat Mealy și  $N$  este un automat Moore, nu va fi niciodată identitate între funcțiile  $f_M$  și  $f_N$  deoarece, pentru același  $x \in V_i^*$ ,  $|f_N(x)| = |f_M(x)| + 1$ .

Totuși, dacă facem abstracție de răspunsul unui automat Moore la intrarea  $\epsilon$  (caracteristică fiecărui automat), putem conveni că automatul Mealy  $M$  este echivalent cu automatul Moore  $N$  dacă  $bf_M(x) = f_N(x)$  unde  $b$  este ieșirea specifică lui  $N$  din starea inițială.

Cu această precizare, putem afirma:

**Teorema 5.2** *Pentru un automat Moore  $M_1$  există un automat Mealy  $M_2$  echivalent și reciproc.*

*Demonstrație:* O soluție poate fi dată trecând prin automatele Starke. Vom prezenta însă și o rezolvare directă.

" $\implies$ ": Fie  $M_1 = (Q, V_i, V_e, \delta, \lambda, q_0)$  un automat Moore; se definește automatul Mealy  $M_2 = (Q, V_i, V_e, \delta, \lambda', q_0)$ , unde singura modificare este funcția de ieșire:

$$\lambda'(q, a) = \lambda(\delta(q, a)).$$

Echivalența este imediată.

" $\impliedby$ ": Fie  $M_1 = (Q, V_i, V_e, \delta, \lambda, q_0)$  un automat Mealy; se definește automatul Moore  $M_2 = (Q \times V_e, V_i, V_e, \delta', \lambda', [q_0, b_0])$  unde  $b_0 \in V_e$  este arbitrar fixat, astfel:

$$\delta'([q, b], a) = [\delta(q, a), \lambda(q, a)], \quad \lambda'([q, b]) = b.$$

Se arată ușor, prin inducție după  $n$  că, dacă pentru  $a_1 a_2 \dots a_n \in V_i^*$ ,  $M_1$  trece prin stările  $q_0, q_1, \dots, q_n$  și emite ieșirile  $b_1, b_2, \dots, b_n$ , atunci  $M_2$  trece prin stările  $[q_0, b_0], [q_1, b_1], \dots, [q_n, b_n]$  și emite ieșirile  $b_0, b_1, \dots, b_n$ . q.e.d.

### 5.2.3 Translatori finiți

Un translator finit este un gsm care permite mișcări (ieșiri și treceri în altă stare) și în cazul unei  $\epsilon$  - intrări. Formal, un translator finit este o structură  $M = (Q, V_i, V_e, \delta, q_0, F)$  unde  $Q, V_i, V_e, q_0, F$  au aceeași semnificație ca la gsm, iar  $\delta : Q \times (V_i \cup \{\epsilon\}) \rightarrow 2^{Q \times V_e^*}$ .

Notățiile și construcțiile prezentate la mașinile secvențiale generalizate se păstrează și în cazul translatorilor finiți. Operația de traducere în acest caz nu se va numi aplicație gsm ci *traducere finită*.

**Exemplul 5.4** *Translatorul  $M = (Q, V_i, V_e, \delta, q_0, F)$  unde  $Q = \{q_0, q_1, q_2, q_3\}$ ,  $V_i = \{0, 1\}$ ,  $V_e = \{0, 1, 2, 3\}$ ,  $F = \{q_1\}$  și funcția de tranziție*

$$\begin{aligned} \delta(q_0, a) &= (q_1, a), \quad \forall a \in \{\epsilon, 0, 1\} \\ \delta(q_1, 0) &= (q_2, \epsilon), \quad \delta(q_1, 1) = (q_3, \epsilon), \quad \delta(q_2, 0) = (q_1, 0), \\ \delta(q_2, 1) &= (q_1, 1), \quad \delta(q_3, 0) = (q_1, 2), \quad \delta(q_3, 1) = (q_1, 3) \end{aligned}$$

*are ca efect traducerea numerelor întregi din baza 2 în baza 4.*

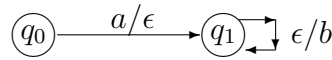
*Demonstrația se bazează pe următoarea observație: un cuvânt  $w \in V_i^*$  determină în graful asociat un drum unic până la starea finală  $q_1$  astfel:*

- dacă  $|w| = 2k + 1$ , se trece de la  $q_0$  la  $q_1$  citind primul caracter;
  - dacă  $|w| = 2k$ , trecerea la  $q_1$  se face cu o  $\epsilon$  - mișcare
- (alte variante determină terminarea cuvântului  $w$  în una din stările  $q_2, q_3$ , nefinale).*

Un translator finit  $M$  este *determinist* dacă pentru orice stare  $q$  este verificată una din condițiile:

- $\forall a \in V_i, |\delta(q, a)| \leq 1$  și  $\delta(q, \epsilon) = \emptyset$ ;
- $|(q, \epsilon)| = 1$  și  $\delta(q, a) = \emptyset, \forall a \in V_i$ .

De remarcat că și în acest caz, un translator finit poate defini mai multe traducări pentru o singură intrare. De exemplu



permite traducările  $(q_0, a, \epsilon) \vdash (q_1, \epsilon, b) \vdash \dots \vdash (q_1, \epsilon, b_{n+1}) \quad \forall n \geq 0$ . Deci

$$\tau(M) = \{(a, b^n) | n \geq 1\}.$$

Pot fi construite diverse modificări simple ale definiției determinismului pentru translatorii finiți, care să asigure unicitatea ieșirii. De exemplu, o sugestie poate fi condiția

$$|\delta(q, \epsilon)| = 0 \quad \forall q \in F.$$

Pentru alte proprietăți simple ale translatorilor finiți, recomandăm de exemplu [1], vol. 1.

Toate clasele de limbaje Chomsky sunt închise la aplicațiile gsm, gsm inverse și translatorii finiți. Pentru exemplificare, vom arăta acest lucru în cazul clasei limbajelor regulate.

**Teorema 5.3**  $\mathcal{L}_3$  este închisă la aplicații gsm și translatori finiți.

*Demonstrație:* Fie  $M = (Q, V_i, V_e, \delta, q_0, F)$  un gsm și  $L \subseteq V_i^*$  un limbaj regulat; deci există o gramatică regulată  $G = (V_N, V_T, S, P)$  cu  $L(G) = L$ . Evident  $V_T \subseteq V_i$ . Putem presupune, fără a micșora generalitatea, că  $P$  are numai reguli de forma  $A \longrightarrow aB$  și există  $X \in V_N$  unic cu  $X \longrightarrow \epsilon$ .

Construim gramatica  $G' = (V_{N'}, V_{T'}, S', P')$  astfel:

$V_{N'} = Q \times V_N$ ,  $S' = [q_0, S]$  și  $P'$  are numai producții de forma:

- i.  $\forall A \longrightarrow aB \in P, \forall (p, w) \in \delta(q, a), [q, A]w \longrightarrow [p, B] \in P'$ ;
- ii.  $\forall q \in F, [q, X] \longrightarrow \epsilon \in P'$ .

Evident,  $G'$  va genera un limbaj regulat. În plus, se arată folosind un procedeu de inducție, că  $L(G') = M(L)$ .

Pentru închiderea la translatările finite, se adaugă regula:

- iii.  $\forall (p, w) \in \delta(q, \epsilon) \forall A \in V_N, [q, A] \longrightarrow w[p, A] \in P'$ . q.e.d.

Pentru a arăta închiderea față de aplicațiile gsm inverse avem nevoie de un rezultat suplimentar:

**Lema 5.1** Orice gsm (translator finit)  $M = (Q, V_i, V_e, \delta, q_0, F)$  este echivalent cu un translator finit  $M' = (Q', V_i, V_e, \delta', q_0, F)$  unde  $\delta' : Q \times (V_i \cup \{\epsilon\}) \longrightarrow 2^{Q \times (V_e \cup \{\epsilon\})}$ .

*Demonstrație:* Pentru fiecare  $(p, w) \in \delta(q, a)$  cu  $w = x_1 x_2 \dots x_n$ , ( $n \geq 2$ ) se introduc  $n - 1$  stări noi  $r_1, r_2, \dots, r_{n-1}$  și se definește  $(r_1, x_1) \in \delta'(q, a)$ ,  $(r_i, x_i) \in \delta'(r_{i-1}, \epsilon)$ , ( $2 \leq i \leq n$ ) (prin convenție,  $r_n = p$ ).

În rest (dacă  $|w| \leq 1$ ),  $(p, w) \in \delta'(q, a) \iff (p, w) \in \delta(q, a)$ . q.e.d.

**Teorema 5.4**  $\mathcal{L}_3$  este închisă la aplicații gsm inverse.

*Demonstrație:* Pentru gsm-ul  $M = (Q, V_i, V_e, \delta, q_0, F)$  se construiește translatorul finit  $M_1 = (Q_1, V_i, V_e, \delta_1, q_0, F)$  dat de Lema 5.1. Pe baza lui, putem construi în etapa a doua gsm - ul  $M' = (Q_1, V_e, V_i, \delta', q_0, F)$  unde  $\delta'$  este definit:

$$(p, x) \in \delta'(q, a) \iff (p, a) \in \delta(q, x)$$

Se arată imediat că acesta este un gsm și  $M'(L) = M^{-1}(L)$ .

Restul aserțiunii din teoremă rezultă din teorema anterioară. q.e.d.

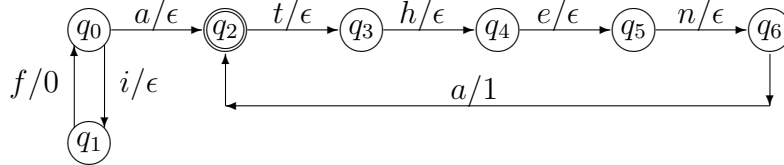
O consecință a acestei proprietăți de închidere este posibilitatea de a arăta că unele limbaje nu sunt regulate.

**Exemplul 5.5** Limbajul generat de gramatica de producții  $S \longrightarrow \text{if } S \text{ then } S \mid a$  nu este regulat.

Pentru a arăta aceasta, intersectăm limbajul  $L(G)$  generat de gramatica dată mai sus cu limbajul regulat  $(\text{if})^*a(\text{thena})^*$  și obținem

$$L_1 = L(G) \cap (\text{if})^*a(\text{thena})^* = \{(\text{if})^n a (\text{thena})^n \mid n \geq 0\}$$

Construim aplicația gsm dată de diagrama de tranziție



Se obține  $M(L_1) = \{0^k 1^k \mid k \geq 0\}$ , care nu este un limbaj regulat.

Pentru că  $\mathcal{L}_3$  este închisă la intersecție și la aplicații gsm, rezultă că nici  $L(G)$  nu este regulat.

## 5.3 Automate stivă

Automatul stivă (sau **pushdown**) este un mecanism destul de asemănător cu automatul finit. Ca și acesta, el este capabil să citească un cuvânt de intrare în mod secvențial (stânga-dreapta), folosind un număr finit de stări interne. Gradul crescut de complexitate al automatului pushdown este dat însă de o memorie auxiliară de tip stivă<sup>1</sup>, teoretic infinită.

După ce a citit la intrare un caracter ' $a$ ', automatul stivă, aflat în starea  $q$  și cu ' $z$ ' caracter aflat în topul stivei, este capabil să efectueze următoarele acțiuni:

- Șterge ' $z$ ' din stivă și-l înlocuiește cu un cuvânt  $w$  (posibil vid);
- Trece în altă stare  $p$ ;
- Menține capul de citire pe același caracter ' $a$ ', sau trece la caracterul următor.

Automatul este în general nedeterminist: pentru fiecare triplet  $(q, a, z)$ , pot fi în general mai multe posibilități de continuare, alegerea unei variante fiind aleatoare teoretic.

Când un automat stivă începe citirea unui cuvânt de la intrare, el este într-o anumită stare (numită *stare inițială*) și are un anumit caracter de start în stivă. Cuvântul de

<sup>1</sup>Reamintim, caracteristica unei stive este aceea că informația deținută de ea este accesibilă și folosită numai în ordinea inversă intrării. Din acest motiv, o stivă este o memorie de tip *LIFO* (Last In, First Out).

intrare este acceptat dacă și numai dacă automatul ajunge la golirea stivei simultan cu terminarea parcurgerii cuvântului<sup>2</sup>.

**Definiția 5.6** *Se numește automat stivă (pushdown) structura  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$  unde*

- $Q, \Sigma, \Gamma$  sunt mulțimi finite nevide, numite respectiv mulțime de stări, alfabet de intrare, alfabet stivă;
- $q_0 \in Q$  este starea inițială;
- $Z_0 \in \Gamma$  este simbolul inițial din stivă;
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \longrightarrow 2^{Q \times \Gamma^*}$  este funcția de tranziție (în general, parțial definită).

Un automat stivă  $M$  este *determinist* dacă

$$|\delta(q, a, x)| \leq 1 \quad \forall q \in Q, a \in \Sigma \cup \{\epsilon\}, x \in \Gamma.$$

La un automat stivă se lucrează cu descrieri instantanee sub forma unor triplete  $(q, \alpha, w)$ , unde  $q \in Q$ ,  $\alpha \in \Sigma^*$ ,  $w \in \Gamma^*$ . O mișcare (deplasare) pe aceste triplete se definește astfel:

$$(q, a\alpha, xw) \vdash (p, \alpha, uw) \iff (p, u) \in \delta(q, a, x),$$

unde  $p, q \in Q$ ,  $a \in \Sigma \cup \{\epsilon\}$ ,  $x \in \Gamma$ ,  $u, w \in \Gamma^*$ .

**Observația 5.3** *Conform Definiției 5.6,  $\alpha \in \Sigma^*$  este un vector de tip stivă, cu vârful la stânga (la fiecare mișcare se șterge cel mult un caracter aflat la extremitatea stângă a lui  $\alpha$ ), iar  $w$  este memoria tip stivă a automatului, de asemenea cu vârful la stânga.*

Dacă notăm cu  $\vdash^*$  închiderea reflexivă și tranzitivă a relației  $\vdash$ , limbajul acceptat de un automat stivă va fi

$$\tau(M) = \{\alpha \mid \alpha \in \Sigma^*, (q_0, \alpha, Z_0) \vdash^* (q, \epsilon, \epsilon)\}$$

**Exemplul 5.6** *Fie limbajul  $L = \{a^n b^n \mid n \geq 1\}$ . Un automat stivă  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$  care acceptă  $L$  este:  $Q = \{q_0, q_1\}$ ,  $\Sigma = \{a, b\}$ ,  $\Gamma = \{Z_0, A\}$  și*

- |   |  |
|---|--|
| 1) $\delta(q_0, a, Z_0) = \{(q_0, A)\}$ | 3) $\delta(q_0, b, A) = \{(q_1, \epsilon)\}$ |
| 2) $\delta(q_0, a, A) = \{(q_0, AA)\}$  | 4) $\delta(q_1, b, A) = \{(q_1, \epsilon)\}$ |

---

<sup>2</sup>Există și posibilitatea de acceptare a cuvintelor de intrare prin ajungerea într-o stare finală – similar automatelor finite – indiferent de conținutul stivei. Se poate demonstra ușor că un astfel de automat poate fi transformat într-un automat stivă fără stări finale, care acceptă cuvintele prin golirea stivei.

Primul caracter 'a' citit la intrare ( $\epsilon \notin L$ , deci există cel puțin un 'a') schimbă pe  $Z_0$  din stivă în  $A$ . Apoi se acceptă toate 'a'-urile de la intrare, efectul fiind adăugarea de caractere  $A$  în stivă, cu rol de numărare: după citirea a  $p$  caractere 'a', în stivă se află  $p$  caractere  $A$ . Primul 'b' citit la intrare șterge un  $A$  din stivă și concomitent schimbă starea. În continuare, fiecare 'b' citit "bifează" câte un  $A$  prin ștergerea lui din stivă. Cuvântul este acceptat dacă au fost șterse toate  $A$ -urile din stivă, deci dacă numărul de 'b'-uri a fost egal cu numărul de 'a'-uri. Schimbarea stării a fost necesară pentru a nu mai permite citirea de caractere 'a' după 'b'-uri (cu formula 2).

Formal, egalitatea  $\tau(M) = L$  se arată prin dublă incluziune.

" $\supseteq$ ": Fie  $a^n b^n \in L$ ; atunci  $(q_0, a^n b^n, Z_0) \vdash^* (q_1, \epsilon, \epsilon)$  prin aplicarea succesivă a regulilor: (1) odată, (2) de  $n - 1$  ori, (3) odată, și (4) de  $n - 1$  ori.

" $\subseteq$ ": Fie  $\alpha \in \tau(M)$ ; deci  $(q_0, \alpha, Z_0) \vdash^* (q_1, \epsilon, \epsilon)$ . Singura regulă care se poate aplica la început este (1); deci cuvântul  $\alpha$  începe cu 'a'.

Să presupunem că numărul maxim de 'a'-uri de la începutul lui  $\alpha$  este  $n$ ; deci  $\alpha = a^n \alpha_1$ . Atunci

$$(q_0, a^n \alpha_1, Z_0) \vdash (q_0, a^{n-1} \alpha_1, A) \vdash^{n-1} (q_0, \alpha_1, A^n).$$

$\alpha_1$  începe cu 'b'. Să presupunem că sunt  $k$  'b'-uri succesive la începutul lui  $\alpha_1$ :  $\alpha_1 = b^k \alpha_2$ . Atunci

$$(q_0, b^k \alpha_2, A^n) \vdash (q_1, b^{k-1} \alpha_2, A^{n-1}) \vdash^{k-1} (q_1, \alpha_2, A^{n-k})$$

și  $\alpha_2$  nu începe cu 'b'. Sunt două posibilități:

1.  $\alpha_2 \neq \epsilon$ : Atunci  $\alpha_2$  începe cu 'a', ceea ce contrazice faptul că  $\alpha \in \tau(M)$ , derivarea blocându-se în acest moment (nu se mai poate aplica nici una din regulile 1. – 4.).
2.  $\alpha_2 = \epsilon$ : Atunci, pentru că  $\alpha \in \tau(M)$ , stiva trebuie să fie vidă, deci  $n = k$ . Rezultă  $\alpha = a^n b^n \in L$ .

**Exemplul 5.7** Automatele finite pot fi considerate cazuri particulare de automate stivă. Într-adevăr, un AFD  $M = (Q, V, \delta, q_0, F)$  poate fi scris sub forma  $M_1 = (Q, V, \{Z_0\}, \delta_1, q_0, Z_0)$  unde  $\delta_1(q, a, Z_0) = \{(\delta(q, a), Z_0)\} \cup A$  în care

$$A = \begin{cases} \emptyset & \text{dacă } \delta(q, a) \notin F \\ (q, \epsilon) & \text{dacă } \delta(q, a) \in F \end{cases}$$

Avem echivalența (evidentă)

$$(q, a_1 a_2 \dots a_n, Z_0) \vdash (q_1, a_2 \dots a_n, Z_0) \iff \delta(q, a) = q_1.$$

Aplicând această regulă de  $n$  ori, se ajunge la  $(q, a_1 a_2 \dots a_n, Z_0) \vdash^* (q_{n-1}, a_n, Z_0)$ .

Vom avea  $(q_{n-1}, a_n, Z_0) \vdash (q_{n-1}, \epsilon, \epsilon) \iff \delta(q_{n-1}, a_n) \in F$ ; deci

$$(q_0, \alpha, Z_0) \vdash^* (q, \epsilon, \epsilon) \iff \delta(q_0, \alpha) \in F$$

**Teorema 5.5** *Automatele stivă sunt metode de recunoaștere.*

*Demonstrație:* Fie automatul stivă  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ , unde – fără a micșora generalitatea – putem presupune că mulțimile  $Q, \Sigma$  și  $\Gamma$  sunt disjuncte (și nevide). Pe baza lui construim metoda de recunoaștere  $M_r$  astfel:

$V = Q \cup \Sigma \cup \Gamma$ ,  $AX = Q$ ,  $W = \Sigma$ ,

$P = \{(qx, p\alpha) \mid (p, \alpha) \in \delta(q, \epsilon, x)\} \cup \{(qax, p\alpha) \mid (p, \alpha) \in \delta(q, a, x)\} \cup \{(\epsilon, q_0 Z_0)\} \cup \{(xa, ax) \mid a \in \Sigma, x \in \Gamma\}$ .

Mai trebuie demonstrată egalitatea  $\tau(M) = L$ , unde  $\tau(M)$  este limbajul acceptat de automatul stivă  $M$ , iar  $L = \{w \mid w \in \Sigma^*, \exists q \in Q, w \xRightarrow{*} q\}$ .

Pentru aceasta, să arătăm întâi echivalența

$$(q, a\alpha, xw) \vdash (p, \alpha, uw) \iff qxwa\alpha \xRightarrow{*} puw\alpha$$

unde  $p, q \in Q$ ,  $a \in \Sigma \cup \{\epsilon\}$ ,  $\alpha \in \Sigma^*$ ,  $x \in \Gamma$ ,  $u, w \in \Gamma^*$ .

" $\implies$ ":  $(q, a\alpha, xw) \vdash (p, \alpha, uw) \implies (p, u) \in \delta(q, a, x)$ . Din construcția lui  $M_r$  rezultă că  $(qax, pu) \in P$ . Atunci

$$qxwa\alpha \xRightarrow{*} qaxw\alpha \implies puw\alpha \quad (2)$$

unde prima parte a derivării este posibilă datorită regulilor de rescriere  $\{(xa, ax) \mid \forall x \in \Gamma, a \in \Sigma\}$ .

De remarcat că dacă  $a = \epsilon$ , atunci această primă parte a derivării se aplică de 0 ori.

" $\impliedby$ ": Din analiza regulilor din  $P$  rezultă că (2) este singura posibilitate de generare a formei sentențiale  $puw\alpha$  plecând de la  $qxwa\alpha$ .

Deci, din existența ultimei generări directe, avem  $(p, u) \in \delta(q, a, x)$ , de unde rezultă – conform definiției – că  $(q, a\alpha, xw) \vdash (p, \alpha, uw)$ .

Folosind acum un procedeu de inducție după lungimea derivării, se obține echivalența

$$(q_0, \alpha, Z_0) \vdash^* (q, \epsilon, \epsilon) \iff q_0 Z_0 \alpha \xRightarrow{*} q.$$

Regula  $(\epsilon, q_0 Z_0)$  nu se poate aplica decât pe prima poziție (altfel, cuvântul aflat în fața lui  $q_0$  nu se mai poate reduce); deci

$$\alpha \implies q_0 Z_0 \alpha \xRightarrow{*} q.$$

q.e.d.

Prima teoremă fundamentală a acestei secțiuni este următoarea:

**Teorema 5.6** *Orice limbaj independent de context este acceptat de un automat stivă.*

*Demonstrație:* Fie  $L \in \mathcal{L}_2$  și  $G = (V_N, V_T, S, P)$  o gramatică independentă de context cu  $L(G) = L$ . Construim automatul stivă  $M = (\{q\}, V_T, V_N \cup V_T, \delta, q, S)$  unde funcția de tranziție  $\delta$  este definită astfel:

$$\begin{aligned} \delta(q, \epsilon, A) &= \{(q, \alpha) \mid A \longrightarrow \alpha \in P\} \text{ pentru toate neterminalele } A \in V_N; \\ \delta(q, x, x) &= \{(q, \epsilon)\}, \quad \forall x \in V_T; \\ \delta(q, a, b) &= \emptyset \quad \text{în rest.} \end{aligned}$$

Vom arăta că  $L(G) = \tau(M)$ .

" $\subseteq$ ": Fie  $\alpha \in L(G)$ ; deci există o derivare  $S \xRightarrow{*} \alpha$  în gramatica  $G$ . Fiind independentă de context, putem aranja această derivare în așa fel încât la fiecare pas, producția care se aplică este o rescriere a celui mai din stânga neterminat.

$$S \xRightarrow{*} \alpha_1 A_1 w_1 \xRightarrow{*} \alpha_1 \alpha_2 A_2 w_2 \xRightarrow{*} \alpha_1 \alpha_2 \dots \alpha_{k-1} A_{k-1} w_{k-1} \xRightarrow{*} \alpha_1 \alpha_2 \dots \alpha_k = \alpha$$

cu  $\alpha_i \in V_T^*$ ,  $A_i \in V_N$ ,  $w_i \in (V_N \cup V_T)^*$ .

Ținând cont acum de modul de construire al automatului  $M$ , rezultă

$$\begin{aligned} (q, \alpha_1 \alpha_2 \dots \alpha_k, S) &\vdash (q, \alpha_1 \alpha_2 \dots \alpha_k, \alpha_1 A_1 w_1) \vdash (q, \alpha_2 \dots \alpha_k, A_1 w_1) \vdash \\ &\vdash (q, \alpha_2 \dots \alpha_k, \alpha_2 A_2 w_2) \vdash (q, \alpha_k, A_{k-1} w_{k-1}) \vdash (q, \alpha_k, \alpha_k) \vdash (q, \epsilon, \epsilon). \end{aligned}$$

" $\supseteq$ ": Fie  $\alpha \in \tau(M)$ ; deci există o secvență

$$(q, \alpha, S) = (q, a_1 a_2 \dots a_n, u_1) \vdash (q, a_2 a_3 \dots a_n, u_2) \vdash \dots \vdash (q, a_n, u_n) \vdash (q, \epsilon, \epsilon)$$

cu  $u_i \in (V_N \cup V_T)^*$ ,  $a_i \in V_T \cup \{\epsilon\}$ .

Va trebui să arătăm că  $S \xRightarrow{*} \alpha$ . Pentru aceasta, vom demonstra prin inducție descrescătoare după  $i$  ( $1 \leq i \leq n$ ), că

$$u_i \xRightarrow{*} a_i a_{i+1} \dots a_n \quad (3)$$

Pentru cazul  $i = n$ ,  $(q, a_n, u_n) \vdash (q, \epsilon, \epsilon)$  numai în cazul  $u_n = a_n$ ; deci  $u_n \xRightarrow{*} a_n$ .

Să presupunem că derivarea (3) are loc pentru orice  $i$  ( $1 < i \leq n$ ) și să o arătăm

$$u_{i-1} \xRightarrow{*} a_{i-1} a_i \dots a_n.$$

Relația  $(q, a_{i-1} a_i \dots a_n, u_{i-1}) \vdash (q, a_i \dots a_n, u_i)$  conduce la două cazuri:

1.  $a_{i-1} \in V_T$ : atunci  $u_{i-1} = a_{i-1} u_i$  (altă posibilitate de derivare nu există) și deci

$$u_{i-1} = a_{i-1} u_i \xRightarrow{*} a_{i-1} a_i \dots a_n$$

conform ipotezei de inducție.

2.  $a_{i-1} = \epsilon$ : atunci derivarea a fost efectuată pe baza primei reguli. Deci  $u_{i-1} = A u'_{i-1}$  cu  $A \in V_N$ ,  $u_i = \beta u'_{i-1}$  și  $A \longrightarrow \beta \in P$ . Rezultă

$$u_{i-1} = A u'_{i-1} \xRightarrow{*} \beta u'_{i-1} = u_i \xRightarrow{*} a_i \dots a_n = a_{i-1} a_i \dots a_n.$$

Pentru  $i = 1$  relația (3) devine  $u_1 \xRightarrow{*} a_1 a_2 \dots a_n = \alpha$  și cum  $u_1 = S$ , rezultă  $S \xRightarrow{*} \alpha$ . q.e.d.



**Teorema 5.7** *Clasa limbajelor acceptate automatele stivă este  $\mathcal{L}_2$ .*

*Demonstrație:* Fie  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$  un automat stivă. Construim gramatica  $G = (V_N, V_T, S, P)$  astfel:

$$V_T = \Sigma,$$

$$V_N = \{[q, A, p] \mid p, q \in Q, A \in \Gamma\} \cup \{S\},$$

iar mulțimea  $P$  a producțiilor este formată din

$$1. S \longrightarrow [q_0, Z_0, q] \text{ pentru toate stările } q \in Q;$$

$$2. [q, A, q_{m+1}] \longrightarrow a[q_1, B_1, q_2][q_2, B_2, q_3] \dots [q_m, B_m, q_{m+1}]$$

$$\forall q, q_1, q_2, \dots, q_{m+1} \in Q, \quad a \in \Sigma \cup \{\epsilon\}, \quad A, B_1, B_2, \dots, B_m \in \Gamma \text{ cu proprietatea}$$

$$(q_1, B_1 B_2 \dots B_m) \in \delta(q, a, A).$$

Când  $m = 0$ , producția este  $[q, A, q_1] \longrightarrow a$ .

Pentru a arăta egalitatea  $L(G) = \tau(M)$ , vom demonstra în prima etapă – prin inducție după numărul de pași într-o derivare a lui  $G$ , respectiv după numărul de mișcări în automatul  $M$  – echivalența

$$[q, A, p] \xRightarrow{*} \alpha \quad \Longleftrightarrow \quad (q, \alpha, A) \vdash^* (p, \epsilon, \epsilon)$$

- "Dacă  $(q, \alpha, A) \vdash^i (p, \epsilon, \epsilon)$ , atunci  $[q, A, p] \xRightarrow{*} \alpha$ ."

Pentru  $i = 1$  :  $\delta(q, \alpha, A)$  conține elementul  $(p, \epsilon)$  (în particular,  $|\alpha| \leq 1$ ). Deci  $[q, A, p] \longrightarrow \alpha \in P$  și derivarea este imediată (conform definiției).

Să considerăm  $i > 1$ . Fie  $\alpha = a\beta$  și  $(q, a\beta, A) \vdash^{i-1} (q_1, \beta, B_1 B_2 \dots B_n) \vdash^* (p, \epsilon, \epsilon)$ .

Cuvântul  $\beta$  poate fi scris  $\beta = \beta_1 \beta_2 \dots \beta_n$ , unde fiecare  $\beta_j$  are ca efect ștergerea lui  $B_j$  din stivă (după un număr arbitrar de mișcări). Altfel spus, fie  $\beta_1$  prefixul lui  $\beta$  la sfârșitul căruia stiva ajunge la  $n - 1$  simboluri,  $\beta_2$  cuvântul care urmează după  $\beta_1$ , la sfârșitul căruia lungimea stivei mai scade cu o unitate etc. S-a considerat lungimea inițială a stivei ca fiind 1 (aici se află doar caracterul  $A$ ).

De remarcat că  $B_1$  nu se află în topul stivei tot timpul citirii lui  $\beta_1$ ; el poate fi schimbat (dacă este în top) cu zero, unu sau mai multe caractere. Important este faptul că  $B_2, B_3, \dots, B_n$  nu vor fi niciodată în topul stivei în timpul citirii lui  $\beta_1$ ; deci ele nu vor putea influența comportarea automatului. În general  $B_j$  stă nemodificat în stivă până la parcurgerea completă a lui  $\beta_1 \dots \beta_{j-1}$ .

Există stările  $q_2, q_3, \dots, q_{n+1}$  ( $q_{n+1} = p$ ) astfel încât

$$(q_j, \beta_j, B_j) \vdash^* (q_{j+1}, \epsilon, \epsilon)$$

în mai puțin de  $i$  mișcări ( $q_j$  este starea atinsă în momentul când stiva ajunge la  $n - j + 1$  elemente). Ipoteza de inducție asigură atunci derivările

$$[q_j, B_j, q_{j+1}] \xRightarrow{*} \beta_j, \quad (1 \leq j \leq n)$$

Primei mișcări,  $(q, a\beta, A) \vdash (q_1, \beta, B_1 B_2 \dots B_n)$  îi corespunde prin definiție producția  $[q, A, p] \longrightarrow a[q_1, B_1, q_2][q_2, B_2, q_3] \dots [q_n, B_n, q_{n+1}]$ .

Compunând derivările, se obține forma finală

$$[q, A, p] \xRightarrow{*} a\beta_1\beta_2 \dots \beta_n = \alpha$$

- ”Dacă  $[q, A, p] \xRightarrow{i} \alpha$  atunci  $(q, \alpha, A) \vdash^* (p, \epsilon, \epsilon)$ .”

$i = 1$ : Afirmatia este imediată, pentru că  $[q, A, p] \longrightarrow \alpha$  trebuie să fie o producție a gramaticii  $G$ , și deci  $(p, \epsilon) \in \delta(q, \alpha, A)$  (în subsidiar,  $|\alpha| \leq 1$ ).

Pentru a doua fază a inducției, să presupunem că primul pas al derivării este  $[q, A, p] \xRightarrow{*} a[q_1, B_1, q_2][q_2, B_2, q_3] \dots [q_n, B_n, q_{n+1}]$  unde  $q_{n+1} = p$ . Atunci, conform Lemei 4.1., putem scrie  $\alpha = a\alpha_1\alpha_2 \dots \alpha_n$ , unde

$$[q_j, B_j, q_{j+1}] \xRightarrow{*} \alpha_j \quad (1 \leq j \leq n).$$

Fiecare astfel de derivare are mai puțin de  $i$  pași. Conform ipotezei de inducție, va rezulta  $(q_j, \alpha_j, B_j) \vdash^* (q_{j+1}, \epsilon, \epsilon)$ . Dacă inserăm în stivă  $B_{j+1} \dots B_n$ , avem

$$(q_j, \alpha_j, B_j B_{j+1} \dots B_n) \vdash^* (q_{j+1}, \epsilon, B_{j+1} \dots B_n).$$

Din primul pas al derivării lui  $\alpha$  din  $[q, A, p]$  se obține

$$(q, \alpha, A) \vdash^* (q_1, \alpha_1\alpha_2 \dots \alpha_n, B_1 B_2 \dots B_n).$$

Va rezulta în final  $(q, \alpha, A) \vdash^* (p, \epsilon, \epsilon)$ .

Luând acum  $q = q_0$ ,  $A = Z_0$ , putem scrie

$$[q_0, Z_0, p] \xRightarrow{*} \alpha \quad \Longleftrightarrow \quad (q_0, \alpha, Z_0) \vdash^* (p, \epsilon, \epsilon)$$

sau – folosind și prima regulă din  $P$ :

$$S \xRightarrow{*} \alpha \quad \Longleftrightarrow \quad (q_0, \alpha, Z_0) \vdash^* (p, \epsilon, \epsilon)$$

adică  $\alpha \in L(G) \quad \Longleftrightarrow \quad \alpha \in \tau(M)$ .

q.e.d.

**Observația 5.4** Spre deosebire de automatele finite deterministe – care acceptă orice limbaj regulat – automatele stivă deterministe nu acoperă  $\mathcal{L}_2$ . Există limbaje independente de context care nu sunt acceptate de nici un automat stivă determinist. Un exemplu standard este

$$L = \{a^i b^j c^k \mid i = j \text{ sau } j = k, \quad i, j, k \geq 1\}.$$

**Exemplul 5.8** Fie gramatica  $G = (\{E, T, F\}, \{+, *, (, ), a\}, E, P)$  care generează expresiile aritmetice construite cu operatorii  $+$ ,  $*$  și operandul  $a$ . Producțiile acestei gramatici sunt

$$P = \{E \longrightarrow E + T \mid T, \quad T \longrightarrow T * F \mid F, \quad F \longrightarrow (E) \mid a\}.$$

Folosind construcția din demonstrația Teoremei 5.6, automatul stivă echivalent este  $M = (Q, \Sigma, \Gamma, \delta, q, Z_0)$  unde  $Q = \{q\}$ ,  $\Sigma = \{+, *, (, ), a\}$ ,  $\Gamma = \{E, T, F, +, *, (, ), a\}$ ,  $Z_0 = E$  și

$$\delta(q, \epsilon, E) = \{(q, E + T), (q, T)\}$$

$$\delta(q, \epsilon, T) = \{(q, T * F), (q, F)\}$$

$$\delta(q, \epsilon, F) = \{(q, (E)), (q, a)\}$$

$$\delta(q, x, x) = \{(q, \epsilon)\}, \quad x \in \{+, *, (, ), a\}$$

Să luăm de exemplu cuvântul  $(a + a) * a$ . O derivare a sa în gramatica  $G$  este

$$E \Longrightarrow T \Longrightarrow T * F \Longrightarrow F * F \Longrightarrow (E) * F \Longrightarrow (E + T) * F \Longrightarrow (T + T) * F \Longrightarrow (F + T) * F \Longrightarrow (a + T) * F \Longrightarrow (a + F) * F \Longrightarrow (a + a) * F \Longrightarrow (a + a) * a$$

În automatul  $M$  vom avea secvența de mișcări

$$\begin{aligned} (q, (a + a) * a, E) &\vdash (q, (a + a) * a, T) \vdash (q, (a + a) * a, T * F) \vdash (q, (a + a) * a, F * F) \vdash \\ &(q, (a + a) * a, (E) * F) \vdash (q, a + a) * a, E) * F \vdash (q, a + a) * a, E + T) * F \vdash (q, a + a) * a, T + \\ &T) * F \vdash (q, a + a) * a, F + T) * F \vdash (q, a + a) * a, a + T) * F \vdash (q, + a) * a, + T) * F \vdash \\ &(q, a) * a, T) * F \vdash (q, a) * a, F) * F \vdash (q, a) * a, a) * F \vdash (q, ) * a, ) * F \vdash (q, * a, * F) \vdash \\ &(q, a, F) \vdash (q, a, a) \vdash (q, \epsilon, \epsilon) \end{aligned}$$

De remarcat multitudinea de variante care apar la fiecare moment, și necesitatea de a alege drumul corect, evitând toate celelalte variante care nu conduc la rezultat, blocându-se mai devreme sau mai târziu.

**Exemplul 5.9** În cazul când gramatica  $G$  are numai producții de forma  $A \longrightarrow a\alpha$ , atunci automatul stivă asociat poate fi construit fără  $\epsilon$ -mișcări ( $\delta(q, a, x)$  este definit numai pentru  $a \neq \epsilon$ ). Această construcție se bazează pe faptul că aplicarea unei reguli  $\delta(q, \epsilon, A) = (q, a\alpha)$  este urmată obligatoriu de folosirea regulii  $\delta(q, a, a) = (q, \epsilon)$  – și numai de aceasta.

De aceea, cele două reguli se pot comasa în  $\delta(q, a, A) = (q, \alpha)$ .

Să luăm de exemplu gramatica de producții

$$S \longrightarrow aAA, \quad A \longrightarrow aS \mid bS \mid a$$

Automatul stivă echivalent va avea funcția de tranziție definită astfel:

$$\delta(q, a, S) = \{(q, AA)\}, \quad \delta(q, a, A) = \{(q, S), (q, \epsilon)\}, \quad \delta(q, b, A) = \{(q, S)\}$$

Această facilitate a condus la construirea unei forme speciale de gramatici independente de context pe care o vom prezenta în Capitolul următor – forma normală Greibach.

**Exemplul 5.10** Să construim o gramatică independentă de context pentru limbajul acceptat de automatul  $M = (\{q_0, q_1\}, \{0, 1\}, \{Z_0, X\}, \delta, q_0, Z_0)$  unde

- (1)  $\delta(q_0, 1, Z_0) = \{(q_0, XZ_0)\}$
- (2)  $\delta(q_0, 1, X) = \{(q_0, XX)\}$
- (3)  $\delta(q_0, 0, X) = \{(q_1, X)\}$
- (4)  $\delta(q_0, \epsilon, Z_0) = \{(q_1, \epsilon)\}$
- (5)  $\delta(q_1, 0, Z_0) = \{(q_0, Z_0)\}$
- (6)  $\delta(q_1, 1, X) = \{(q_1, \epsilon)\}$

Vom avea – conform construcției din demonstrația Teoremei 5.7:  $V_T = \{0, 1\}$ ,  $V_N = \{[q_0, Z_0, q_0], [q_0, Z_0, q_1], [q_1, Z_0, q_0], [q_1, Z_0, q_1], [q_0, X, q_0], [q_0, X, q_1], [q_1, X, q_0], [q_1, X, q_1], S\}$ ,

$$P = \left\{ \begin{array}{ll} (0) & S \longrightarrow [q_0, Z_0, q_0] \mid [q_0, Z_0, q_1] \\ (1) & [q_0, Z_0, q_0] \longrightarrow 1[q_0, X, q_0][q_0, Z_0, q_0] \mid 1[q_0, X, q_1][q_1, Z_0, q_0] \\ & [q_0, Z_0, q_1] \longrightarrow 1[q_0, X, q_0][q_0, Z_0, q_1] \mid 1[q_0, X, q_1][q_1, Z_0, q_1] \\ (2) & [q_0, X, q_0] \longrightarrow 1[q_0, X, q_0][q_0, X, q_0] \mid 1[q_0, X, q_1][q_1, X, q_0] \\ & [q_0, X, q_1] \longrightarrow 1[q_0, X, q_0][q_0, X, q_1] \mid 1[q_0, X, q_1][q_1, X, q_1] \\ (3) & [q_0, X, q_0] \longrightarrow 0[q_1, X, q_0] \\ & [q_0, X, q_1] \longrightarrow 0[q_1, X, q_1] \\ (4) & [q_0, Z_0, q_1] \longrightarrow \epsilon \\ (5) & [q_1, Z_0, q_0] \longrightarrow 0[q_0, Z_0, q_0] \\ & [q_1, Z_0, q_1] \longrightarrow 0[q_0, Z_0, q_1] \\ (6) & [q_1, X, q_1] \longrightarrow 1 \end{array} \right.$$

Dacă renotăm neterminalele:

$$\begin{aligned} [q_0, Z_0, q_0] &\equiv A_1, & [q_0, Z_0, q_1] &\equiv A_2, & [q_1, Z_0, q_0] &\equiv A_3, & [q_1, Z_0, q_1] &\equiv A_4, \\ [q_0, X, q_0] &\equiv B_1, & [q_0, X, q_1] &\equiv B_2, & [q_1, X, q_0] &\equiv B_3, & [q_1, X, q_1] &\equiv B_4, \end{aligned}$$

setul de producții poate fi rescris mai clar:

$$P = \left\{ \begin{array}{ll} S \longrightarrow A_1 \mid A_2 & B_2 \longrightarrow 1B_2B_4 \mid 1B_1B_2 \mid 0B_4 \\ A_1 \longrightarrow 1B_1A_1 \mid 1B_2A_3 \mid \epsilon & A_3 \longrightarrow 0A_1 \\ A_2 \longrightarrow 1B_1A_2 \mid 1B_2A_4 & A_4 \longrightarrow 0A_2 \\ B_1 \longrightarrow 1B_1B_2 \mid 1B_2B_3 \mid 0B_3 & B_4 \longrightarrow 1 \end{array} \right.$$

**Exemplul 5.11** Să considerăm limbajul  $L = \{a^mca^n \mid m \geq n \geq 0\}$ .

O gramatică independentă de context pentru  $L$  se poate defini imediat: vom lua de exemplu gramatica de producții  $S \longrightarrow aSa \mid aS \mid c$ .

Automatul stivă echivalent este  $M = (\{q\}, \{a, c\}, \{S, a, c\}, \delta, q, S)$ , unde funcția de tranziție  $\delta$  este definită

$$\delta(q, \epsilon, S) = \underbrace{\{(q, aSa)\}}_{1a}, \underbrace{\{(q, aS)\}}_{1b}, \underbrace{\{(q, c)\}}_{1c}, \quad \delta(q, a, a) = \underbrace{\{(q, \epsilon)\}}_2, \quad \delta(q, c, c) = \underbrace{\{(q, \epsilon)\}}_3$$

De exemplu, cuvântul  $a^2ca$  este acceptat pe baza secvenței de mișcări

$$(q, aaca, S) \stackrel{1a}{\vdash} (q, aaca, aSa) \stackrel{2}{\vdash} (q, aca, Sa) \stackrel{1b}{\vdash} (q, aca, aSa) \stackrel{2}{\vdash} (q, ca, Sa) \stackrel{1c}{\vdash} (q, ca, ca) \stackrel{3}{\vdash} (q, a, a) \stackrel{2}{\vdash} (q, \epsilon, \epsilon) \text{ (s-au aplicat regulile marcate)}.$$

Datorită formei particulare a producțiilor, automatul se poate construi și fără  $\epsilon$ -mișcări, conform Exercițiului 5.9. El este  $M' = (\{q\}, \{a, c\}, \{S, a\}, \delta', q, S)$ , unde funcția de tranziție  $\delta'$  se definește

$$\delta'(q, a, S) = \{\underbrace{(q, Sa)}_{1a}, \underbrace{(q, S)}_{1b}\}, \quad \delta'(q, c, S) = \{\underbrace{(q, \epsilon)}_2\}, \quad \delta'(q, a, a) = \{\underbrace{(q, \epsilon)}_3\}$$

Regula  $\delta'(q, c, c) = \{(q, \epsilon)\}$  nu este necesară, deoarece ' $c$ ' nu apare în stivă (deci o astfel de regulă nu se va aplica niciodată).

Pentru același cuvânt ( $a^2ca$ ) se va folosi secvența de mișcări (evident, mai scurtă):

$$(q, aaca, S) \stackrel{1a}{\vdash} (q, aca, Sa) \stackrel{1b}{\vdash} (q, ca, Sa) \stackrel{2}{\vdash} (q, a, a) \stackrel{3}{\vdash} (q, \epsilon, \epsilon)$$

Să încercăm acum procedeul invers: plecând de la acest automat stivă, să construim o gramatică independentă de context care să genereze  $L$ .

Vom avea  $V_T = \{a, c\}$ ,  $V_N = \{S, [q, S, q], [q, a, q]\}$  și

$$P = \begin{cases} (0) & S \longrightarrow [q, S, q] \mid [q, a, q] \\ (1a) & [q, S, q] \longrightarrow a[q, S, q][q, a, q] \\ (1b) & [q, S, q] \longrightarrow a[q, S, q] \\ (2) & [q, S, q] \longrightarrow c \\ (3) & [q, a, q] \longrightarrow a \end{cases}$$

Renotând  $[q, S, q] \equiv A$ ,  $[q, a, q] \equiv B$ , obținem gramatica de producții

$$S \longrightarrow A|B, \quad A \longrightarrow aAB|aA|c, \quad B \longrightarrow a$$

echivalentă (evident) cu gramatica inițială.

## 5.4 Automate pentru $\mathcal{L}_1$ și $\mathcal{L}_0$

Pentru clasele de limbaje Chomsky  $\mathcal{L}_1, \mathcal{L}_0$  se pot defini de asemenea metode de acceptare de tip automat. Astfel, pentru limbajele dependente de context există **automatele liniar mărginite**, iar pentru  $\mathcal{L}_0$  – **mașinile Turing**. Acestor două clase de automate le vom consacra Capitolele 7 și 8.

## 5.5 Exerciții

**Exercițiul 5.1** Să se demonstreze echivalența (1), utilizând o inducție după lungimea cuvântului  $\alpha$ .

**Exercițiul 5.2** Să se construiască automate finite pentru gramaticile regulate, de producții

- a)  $P = \{S \longrightarrow aS|bA|a, \quad A \longrightarrow aA|bS|b\}$   
 b)  $P = \{S \longrightarrow aS|aA|a, \quad A \longrightarrow bA\epsilon\}$

**Exercițiul 5.3** Să se refacă demonstrația lemei de pompă în termeni de gramatici regulate.

**Exercițiul 5.4** Să se arate că limbajul din Exemplul 5.6 poate fi generat și de următoarele automate stivă

- a)  $Q = \{q_0\}, \quad \Sigma = \{a, b\}, \quad \Gamma = \{Z_0, a, b\},$   
 $\delta(q_0, \epsilon, Z_0) = \{(q_0, aZ_0b), (q_0, ab)\}, \quad \delta(q_0, a, a) = \{(q_0, \epsilon)\}, \quad \delta(q_0, b, b) = \{(q_0, \epsilon)\}.$   
 b)  $Q = \{q_0\}, \quad \Sigma = \{a, b\}, \quad \Gamma = \{Z_0, b\},$   
 $\delta(q_0, a, Z_0) = \{(q_0, Z_0b), (q_0, b)\}, \quad \delta(q_0, b, b) = \{(q_0, \epsilon)\}.$

**Exercițiul 5.5** Să se construiască automate stivă pentru limbajele

$$L_1 = \{a^n b^{n+1} \mid n \geq 0\}, \quad L_2 = \{a^n b^{2n} \mid n \geq 0\}, \quad L_3 = \{a^n b^m \mid n > m \geq 0\}.$$

**Exercițiul 5.6** Să se arate că un AF poate fi descris ca un automat stivă sub forma  $M_1 = (\{q_0\}, V, Q, \delta_1, q_0, q_0)$  unde

$$\begin{aligned} \delta_1(q_0, a, q) = (q_0, p) &\iff \delta(q, a) = p \\ \delta_1(q_0, \epsilon, q) = (q_0, \epsilon), &\quad \forall q \in F \end{aligned}$$

**Exercițiul 5.7** Să se construiască automate stivă care să accepte limbajele

$$L_1 = \{w \mid w \in \{a, b\}^*, |w|_a = |w|_b\}, \quad L_2 = \{a^n b^m a^n \mid n, m \geq 0\}, \quad L_3 = \{a^i b^j c^k \mid i, j, k \geq 0\}.$$

**Exercițiul 5.8** Să se construiască automate stivă pentru gramaticile de producții

1.  $S \longrightarrow ABC, \quad A \longrightarrow BB| \epsilon, \quad B \longrightarrow CC|a, \quad C \longrightarrow AA|b$
2.  $S \longrightarrow aSb|bSa|SS| \epsilon$

Să se verifice dacă aceste automate acceptă cuvintele:

$babb, abba$  (cazul 1.),  $aabbbab, ababbbbaa$  (cazul 2.)

**Exercițiul 5.9** Să se construiască automate stivă fără  $\epsilon$ -mişcări pentru gramaticile de producții

1.  $S \longrightarrow aSbS|bSaS|\epsilon$
2.  $S \longrightarrow aAA, \quad A \longrightarrow aS|bS|a$
3.  $S \longrightarrow aB|bA, \quad A \longrightarrow a|aS|bAA, \quad B \longrightarrow b|bS|aBB$

Pentru automatul stivă construit în cazul 1., să se cerceteze acceptarea cuvântului  $baab$  și să se compare cu aceeași problemă în cazul Exercițiului 5.7, limbajul  $L_1$ .

**Exercițiul 5.10** Să se definească gramatici independente de context echivalente cu automatele stivă având funcțiile de tranziție

1.  $\delta(a_0, a, Z_0) = \{(q_1, Z_0), (q_2, Z_0Z_0)\}, \quad \delta(q_1, b, Z_0) = \{(q_1, Z_0), (q_2, \epsilon)\},$   
 $\delta(q_2, a, Z_0) = \{(q_2, \epsilon), (q_1, Z_0)\}$
2.  $\delta(q_0, a, Z_0) = \{(q_0, XZ_0)\}, \quad \delta(q_0, a, X) = \{(q_0, XX)\}, \quad \delta(q_0, b, X) = \{(q_1, X)\}$   
 $\delta(q_0, \epsilon, Z_0) = \{(q_0, \epsilon)\}, \quad \delta(q_1, a, X) = \{(q_1, \epsilon)\}, \quad \delta(q_1, b, Z_0) = \{(q_0, Z_0)\}.$

**Exercițiul 5.11** Să se arate că dacă  $L$  este un limbaj independent de context, atunci există un automat stivă  $M$  cu  $L = \tau(M)$  astfel încât:

1. Dacă  $(p, u) \in \delta(q, a, x)$  atunci  $|u| \leq 2$ ;
2.  $(q, a\alpha, X\gamma) \vdash (0, \alpha, u\gamma) \implies \gamma = \epsilon \text{ sau } \gamma = X \text{ sau } \gamma = YX.$

# Capitolul 6

## Gramatici independente de context

### 6.1 Introducere

Importanța gramaticilor independente de context (*cfg*) este deosebită din cauza ariei largi de aplicabilitate. Modalitatea relativ ușoară de generare a cuvintelor a condus la un studiu aprofundat al lor. Rolul *cfg*-urilor a devenit predominant în probleme de limbaje de programare, compilatoare, prelucrarea secvențială a textelor (numită și *analiză sintactică*), studiul sistemelor, modele *ADN* etc.

Cea mai mare parte a structurii unui limbaj de programare poate fi specificată cu ajutorul unei astfel de gramatici. Un program (scris în *Pascal*, *C*, *C++*, dar nu numai) este un cuvânt generat de o astfel de gramatică. Primul lucru pe care-l face un compilator atunci când citește un program este să-l translateze într-o secvență de simboluri mai ușor de manipulat, iar apoi să-l "analizeze", adică să descopere dacă există o derivare de la simbolul de start la acest cuvânt. În caz afirmativ, programul merge mai departe spre execuție; altfel, este respins cu "erori sintactice".

Clasa gramaticilor independente de context a fost utilizată inițial pentru o descriere primară a limbajului natural. De exemplu, să considerăm regulile

$$\begin{aligned} < \text{propozitie} > &\longrightarrow < \text{subiect} > < \text{predicat} > \\ < \text{subiect} > &\longrightarrow < \text{substantiv} > < \text{adjectiv} > \\ < \text{substantiv} > &\longrightarrow < \text{nume} > \\ < \text{nume} > &\longrightarrow \text{apa} \\ < \text{adjectiv} > &\longrightarrow \text{rece} \mid \text{calda} \\ < \text{predicat} > &\longrightarrow \text{curge} \mid \text{citeste} \end{aligned}$$

unde  $V_N$  conține cuvintele ("atributele") cuprinse între parantezele unghiulare  $<, >$ , iar  $V_T = \{\text{apa}, \text{rece}, \text{calda}, \text{curge}, \text{citeste}\}$ .

O derivare conform acestor producții poate fi:

$$< \text{propozitie} > \Longrightarrow < \text{subiect} > < \text{predicat} > \Longrightarrow < \text{substantiv} > < \text{adjectiv} >$$



$\langle \text{predicat} \rangle \Rightarrow \langle \text{nume} \rangle \langle \text{adjectiv} \rangle \langle \text{predicat} \rangle \Rightarrow \text{apa} \langle \text{adjectiv} \rangle \langle \text{predicat} \rangle$

$\Rightarrow \text{apa rece} \langle \text{predicat} \rangle \Rightarrow \text{apa rece curge}$

Această modalitate de reprezentare a limbajelor naturale a fost însă parțial abandonată din mai multe motive; unul din cele mai importante a fost necesitatea de a folosi și informații semantice pentru generarea de propoziții corecte sintactic.

De exemplu, conform regulilor de mai sus, se poate obține și ”cuvântul” corect sintactic ”*apa caldă citește*”, care însă nu are nici un înțeles.

Utilizat cu discernământ, aparatul de lucru oferit de gramaticile independente de context joacă însă un rol destul de important în lingvistica pe calculator.

## 6.2 Arbori de derivare și derivări independente de context

Fiind dată o gramatică  $G = (V_N, V_T, S, P)$  și un cuvânt  $w \in L(G)$ , uneori este important să avem o privire de ansamblu asupra modului cum se obține cuvântul  $w$  conform regulilor gramaticii; adică dorim să avem o descriere structurală a derivării lui  $w$ . În cazul gramaticilor independente de context derivările sunt caracterizate natural prin *arborii de derivare*.

**Definiția 6.1** Fie  $Z$  o mulțime finită nevidă și  $U \subseteq Z \times Z$ . Se numește *arbore*, *graful*  $\Gamma = (Z, U)$  cu proprietățile:

1.  $\exists a_0 \in Z$  unic cu  $\{(a, a_0) \mid a \in Z\} \cap U = \emptyset$ ;
2.  $\forall a \in Z \setminus \{a_0\}, \exists a_1, a_2, \dots, a_k \in Z$  unici, cu  $a_k = a, (a_i, a_{i+1}) \in U$  ( $0 \leq i \leq k-1$ ).

**Observația 6.1** Un arbore este un graf fără cicluri și fără noduri izolate.

$a_0$  se numește *rădăcina arborelui*.

Pentru  $(a, a') \in U$ ,  $a'$  este *descendentul direct* al lui  $a$ , iar  $a$  – *ascendentul direct* al lui  $a'$  în arborele  $\Gamma$ .

$a'$  este *descendentul* lui  $a$  dacă există un drum în arbore de la  $a$  la  $a'$ .

Un nod  $a'$  cu proprietatea  $\{(a', a) \mid a \in Z\} \cap U = \emptyset$  se numește *nod final* sau *terminal*<sup>1</sup>. Altfel,  $a'$  este un *nod interior*.

Un nod  $a$  este pe *nivelul*  $k$  dacă drumul de la rădăcină la  $a$  este de lungime  $k-1$ ; în particular rădăcina este pe nivelul 1.  $k$  se numește *numărul de nivel* al nodului  $a$ .

<sup>1</sup>Este cunoscut și sub numele de *frunză*, din terminologia engleză a domeniului (*leaf*).

**Observația 6.2** Orice arc leagă două noduri aflate pe nivele vecine. Evident, dacă există un nod  $x$  pe nivelul  $k$ , atunci pe fiecare din nivelele  $1, 2, \dots, k-1$  se află câte un nod al cărui descendent este  $x$ . Aceste noduri sunt unice.

Se numește *subarbore* un arbore având rădăcina pe un nivel  $k > 1$ .

Fie acum  $G = (V_N, V_T, S, P)$  o gramatică independentă de context și

$$X = \alpha_0 \implies \alpha_1 \implies \dots \implies \alpha_r \quad (1)$$

o derivare oarecare în această gramatică. Să arătăm cum se poate atașa un arbore acestei derivări.

Nodurile arborelui vor fi notate cu elemente din  $V_N \cup V_T \cup \{\epsilon\}$ ; nodurile interioare sunt notate numai cu simboluri neterminale.

Punând în evidență producțiile utilizate în derivarea (1), fie

$$A_i \longrightarrow \beta_i \text{ astfel încât } \alpha_i = u_i A_i v_i, \quad \alpha_{i+1} = u_i \beta_i v_i, \quad (0 \leq i \leq r-1) \quad (2)$$

unde  $A_i \in V_N$ ,  $u_i, v_i, \beta_i \in (V_N \cup V_T)^*$ .

În particular  $A_0 = X$ ,  $\beta_0 = \alpha_1$ .

**Algoritm 6.1:**

**Intrare:** O derivare (1) în gramatica independentă de context  $G = (V_N, V_T, S, P)$ ;

**Ieșire:** Un arbore  $\Gamma = (Z, U)$  atașat derivării (1);

**Algoritm:**

1.  $i := 1$ ;  $Z := \{(1)\}$ ;
2. Presupunem că pentru  $\forall j$  ( $j \leq i$ ) s-a prelucrat forma sentențială  $\alpha_j = a_{1j} a_{2j} \dots a_{nj}$ ; atunci există  $(n_1, n_2, \dots, n_{jp}) \in Z$  cu  $a_{pj} = h((n_1, n_2, \dots, n_{jp}))$  ( $1 \leq p \leq n$ ).  
Fie acum nodul  $(n_1, n_2, \dots, n_t) \in Z$  de pe nivelul  $t$  cu  $h(n_1, n_2, \dots, n_t) = A_i$  ( $A_i$  dat de (2)). Apar două cazuri:
  - (a)  $\beta_i = \epsilon$ ; atunci  $Z := Z \cup \{(n_1, n_2, \dots, n_t, 1)\}$  și  $h((n_1, n_2, \dots, n_t, 1)) = \epsilon$ ;
  - (b)  $\beta_i = a_1 a_2 \dots a_n$ ; atunci  $Z := Z \cup \{(n_1, n_2, \dots, n_t, p) \mid 1 \leq p \leq n\}$  și  $h((n_1, n_2, \dots, n_t, p)) = a_p$ , ( $1 \leq p \leq n$ );
3.  $i := i + 1$ ; dacă  $i < r$  salt la Pasul 2;
4.  $U := \{((n_1, n_2, \dots, n_t), (n_1, n_2, \dots, n_t, n_{t+1})) \in Z \times Z \mid 1 \leq t \leq r\}$ .

Nodurile arborelui atașat derivării (1) vor fi  $k$ -tupluri ( $k \leq r+1$ ) de numere naturale. Notăm  $[N] = \bigcup_{i \geq 1} \mathcal{N}^i$ . Fie homomorfismul  $h$  parțial definit  $h : [N] \longrightarrow V_N \cup V_T \cup \{\epsilon\}$ ,  $h((n_1, n_2, \dots, n_t))$  reprezentând notarea nodului  $(n_1, n_2, \dots, n_t)$ .

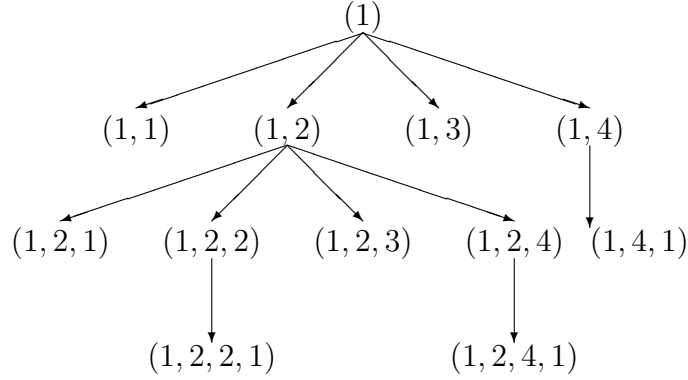
**Exemplul 6.1** Fie gramatica  $G = (V_N, V_T, S, P)$  cu

$$V_N = \{S\}, \quad V_T = \{a, b\}, \quad P = \{S \longrightarrow aSbS | bSaS | \epsilon\}$$

Pentru derivarea

$$S \Longrightarrow aSbS \Longrightarrow abSaSbS \Longrightarrow abaSbS \Longrightarrow abaSb \Longrightarrow abab$$

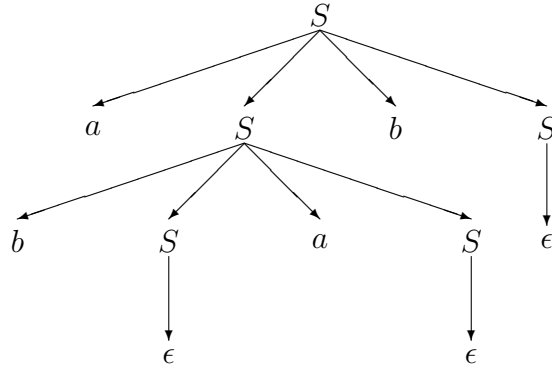
avem arborele de derivare:



Putem observa următoarele:

1.  $h(1) = S$ ;
2. Prima regulă aplicată este  $S \longrightarrow aSbS$ . Deci apar nodurile  $(1,1), (1,2), (1,3), (1,4)$  cu  $h(1,1) = a$ ,  $h(1,2) = S$ ,  $h(1,3) = b$ ,  $h(1,4) = S$ .
3. Primul neterminal  $S = h(1,2)$  din  $aSbS$  este rescris  $bSaS$ . Deci din nodul  $(1,2)$  apar nodurile  $(1,2,1), (1,2,2), (1,2,3), (1,2,4)$  având notațiile  $h(1,2,1) = b$ ,  $h(1,2,2) = S$ ,  $h(1,2,3) = a$ ,  $h(1,2,4) = S$ .
4. Neterminalul  $S = h(1,2,2)$  este rescris  $\epsilon$ ; deci din nodul  $(1,2,2)$  apare descendentul direct  $(1,2,2,1)$  cu  $h(1,2,2,1) = \epsilon$ .
5.  $S = h(1,2,4)$  este rescris  $\epsilon$ ; deci din nodul  $(1,2,4)$  apare descendentul direct  $(1,2,4,1)$  cu  $h(1,2,4,1) = \epsilon$ .
6. De asemenea,  $S = h(1,4)$  derivă în  $\epsilon$ , ceea ce înseamnă că și din nodul  $(1,4)$  apare un descendent direct  $(1,4,1)$  cu  $h(1,4,1) = \epsilon$ .

De obicei, pentru ușurința scrierii, atunci când vectorii numerici care definesc nodurile nu sunt necesari, se folosesc în reprezentarea arborelui numai notațiile nodurilor (valorile lui  $h$ ). Deci:



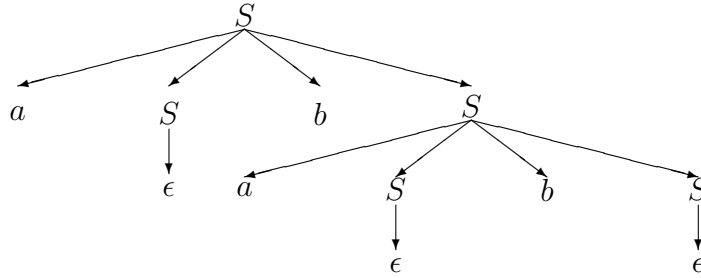
De remarcat că același arbore corespunde și derivării

$$S \Rightarrow aSbS \Rightarrow aSb \Rightarrow abSaSb \Rightarrow AbaSb \Rightarrow abab.$$

Mai este posibilă și situația când derivări diferite ale aceluiași cuvânt generează arbori diferiți. De exemplu, tot pentru cuvântul *abab* se poate construi și derivarea

$$S \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSbS \Rightarrow ababS \Rightarrow abab$$

al cărei arbore asociat este



Pe mulțimea  $Z$  a nodurilor unui arbore de derivare se definește următoarea relație de ordine parțială (ordonare lexicografică):

$$(n_1, n_2, \dots, n_p) < (m_1, m_2, \dots, m_q) \iff \exists k, n_k < m_k \text{ și } n_i = m_i \forall i, (1 \leq i \leq k-1).$$

**Propoziția 6.1** " $<$ " este o relație de ordine totală pe mulțimea  $Z_f \subset Z$  a nodurilor finale.

*Demonstrație:* Din construcția arborelui de derivare rezultă că dacă  $(n_1, \dots, n_p)$  și  $(n_1, \dots, n_p, n_{p+1}, \dots, n_q)$  sunt două noduri, atunci  $(n_1, \dots, n_q)$  este un descendent al lui  $(n_1, \dots, n_p)$ .

Evident,  $Z_f \neq \emptyset$ ; singurul lucru care mai trebuie arătat este

$\forall (n_1, \dots, n_p), (m_1, \dots, m_q) \in Z_f \implies \exists k \leq \min\{p, q\}$  cu  $m_k \neq n_k$  și  $m_i = n_i$ , ( $1 \leq i \leq k-1$ ).

Dacă  $p = q$  atunci afirmația este evidentă (altfel, cei doi vectori ar fi identici).

Să considerăm cazul  $p < q$  (variantea  $p > q$  se tratează similar). Dacă prin absurd  $n_i = m_i \forall i \leq p$ , atunci  $(m_1, \dots, m_q)$  este un descendent al lui  $(n_1, \dots, n_p)$ . Aceasta duce la concluzia că  $(n_1, \dots, n_p)$  – având descendenți – nu este nod final, contradicție. q.e.d.

Să considerăm un arbore de derivare  $\Gamma$  și  $Z_f$  mulțimea nodurilor finale, ordonată cu relația  $<$ ; fie  $v_1, v_2, \dots, v_n$  listarea ordonată a nodurilor din  $Z_f$ . Dacă  $h(v_1 v_2 \dots v_n) = \alpha$ , spunem că  $\Gamma$  generează  $\alpha$ , sau că este un arbore de derivare pentru  $\alpha$ , iar  $\alpha$  este *frontiera* lui  $\Gamma$ .

**Definiția 6.2** Un subarbore al arborelui de derivare  $\Gamma = (Z, U)$  este un nod al lui  $Z$  împreună cu toți descendenții lui și arcele corespunzătoare, toate nodurile păstrând notațiile din  $\Gamma$ .

### Observația 6.3

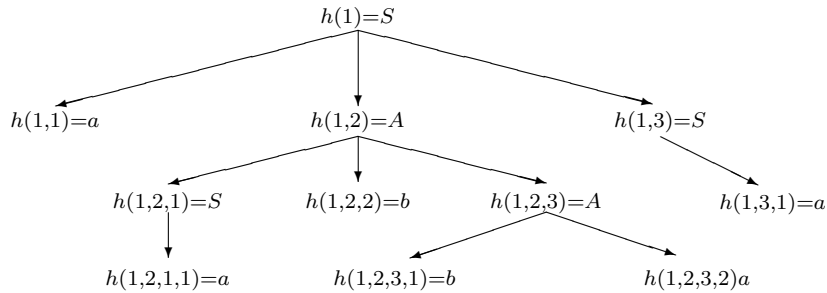
1. În cazul unui subarbore de derivare, rădăcina este un vector numeric cu  $k$  componente,  $k$  fiind numărul de nivel al nodului.
2. Dacă  $(n_1, \dots, n_t)$  este vectorul asociat rădăcinii subarborelui, și  $h(n_1 \dots, n_t) = A$ , vom numi acest subarbore un  $A$  - arbore.

În particular, arborii de derivare sunt  $S$  - arbori.

**Exemplul 6.2** Fie gramatica  $G = (\{S, A\}, \{a, b\}, S, P)$  unde

$$P = \{S \longrightarrow aAS|a, \quad A \longrightarrow SbA|SS|ba\}$$

Un arbore de derivare în această gramatică este (s-a păstrat și notația  $h$ ):



Nodurile finale, ordonate după relația  $<$ , sunt

$$(1, 1)(1, 2, 1, 1)(1, 2, 2)(1, 2, 3, 1)(1, 2, 3, 2)(1, 3, 1)$$

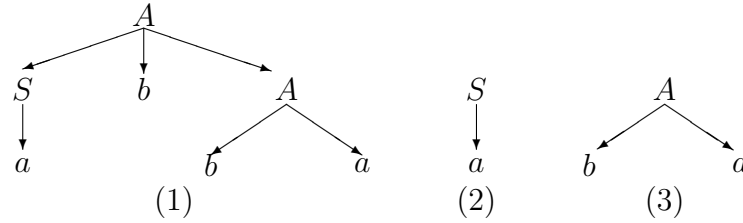
Acestei secvențe îi corespunde frontiera

$$h(1, 1)h(1, 2, 1, 1)h(1, 2, 2)h(1, 2, 3, 1)h(1, 2, 3, 2)h(1, 3, 1) = aabbaa$$

Pentru acest cuvânt ( $aabbaa$ ) există o derivare în gramatica  $G$ ; anume

$$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabAa \Rightarrow aabbaa$$

Exemple de subarbori ai acestui arbore:



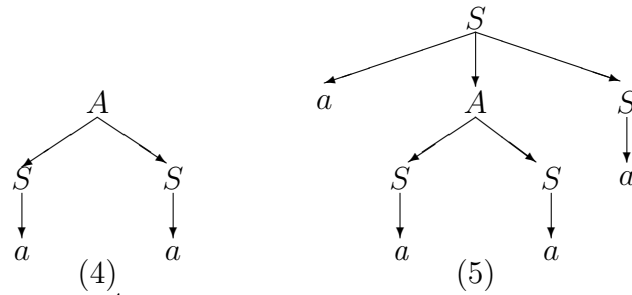
Ei corespund derivărilor

$$(1) \quad A \Rightarrow SbA \Rightarrow abA \Rightarrow abba$$

$$(2) \quad S \Rightarrow a$$

$$(3) \quad A \Rightarrow ba$$

În schimb, subarboarele (4) nu este subarbore al acestui arbore, dar este subarbore în (5):

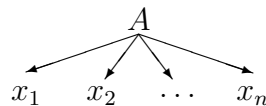


De asemenea,  $\begin{array}{c} A \\ \swarrow \searrow \\ S \quad A \end{array}$  nu este subarbore în nici un arbore atașat unei derivări în  $G$ , el neputând corespunde nici unei producții din  $P$ .

**Teorema 6.1** Fie  $G = (V_N, V_T, S, P)$  o gramatică independentă de context și  $A \in V_N$ . Atunci  $A \xRightarrow{*} \alpha$  dacă și numai dacă există un  $A$  - arbore de derivare atașat, cu frontiera  $\alpha$ .

*Demonstrație:* " $\Leftarrow$ ": Fie  $\alpha$  frontiera unui  $A$  - arbore. Vom demonstra prin inducție după numărul  $i$  de noduri interioare ale arborelui, că  $A \xRightarrow{*} \alpha$ .

$i = 1$ : singurul nod interior este rădăcina  $A$  și arborele este de forma:



Conform construcției,  $A \longrightarrow x_1x_2 \dots x_n \in P$  și  $\alpha = x_1x_2 \dots x_n$ , deci  $A \xRightarrow{*} \alpha$ .

Să presupunem afirmația adevărată pentru orice arbore cu  $i - 1$  noduri interioare și să considerăm frontiera  $\alpha$  a unui  $A$  - arbore cu  $i > 1$  noduri interioare. Fie  $x_1, x_2, \dots, x_n$  descendenții rădăcinii  $A$ , ordonați de la stânga la dreapta cu relația  $<$ ; atunci

1.  $A \longrightarrow x_1x_2 \dots x_n \in P$ ;
2. Simbolurile  $x_1, x_2, \dots, x_n$  nu sunt toate terminale sau  $\epsilon$  (altfel arborele ar avea un singur nod interior –  $A$ ). Atunci fiecare  $x_p \in V_N$  este rădăcina unui subarbore de frontieră  $\alpha_p$ . Conform ipotezei de inducție,  $x_p \xRightarrow{*} \alpha_p$ .

Pentru fiecare  $x_p \in V_T \cup \{\epsilon\}$  vom nota  $\alpha_p = x_p$ , deci  $x_p \xRightarrow{*} \alpha_p$ . Atunci:

- Pentru  $j < p$ , componentele lui  $\alpha_j$  sunt la "stânga" celor din  $\alpha_p$  (nodurile finale notate cu componentele lui  $\alpha_j$  au  $(1, j)$  pe primele două poziții, iar celelalte  $(1, p)$ ). Deci  $\alpha = \alpha_1\alpha_2 \dots \alpha_n$ ;
- $A \xRightarrow{*} x_1x_2 \dots x_n \xRightarrow{*} \alpha_1x_2 \dots x_n \xRightarrow{*} \alpha_1\alpha_2 \dots \alpha_n$ .

Deci  $A \xRightarrow{*} \alpha$ .

**Observația 6.4** *Derivarea de mai sus este doar una din derivările posibile asigurate de A-arborele atașat.*

" $\xRightarrow{*}$ ": Fie  $A \xRightarrow{k} \alpha$  o derivare în  $k$  pași.

Pentru  $k = 1$ ,  $A \longrightarrow \alpha \in P$  și atunci există un  $A$  - arbore de derivare cu frontiera  $\alpha$ , construit ca la implicația anterioară.

Să presupunem că, dacă  $X \xRightarrow{*} \beta$  printr-o derivare de lungime mai mică decât  $k$  (pentru orice  $X \in V_N$ ), atunci există un  $X$  - arbore de derivare de frontieră  $\beta$ .

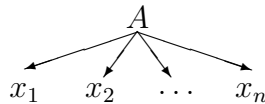
Să luăm acum o derivare  $A \xRightarrow{*} \alpha$  de lungime  $k$ ; fie  $A \longrightarrow x_1x_2 \dots x_n$  prima producție aplicată în derivare.

Conform Lemei 4.1, din  $A \xRightarrow{*} x_1x_2 \dots x_n \xRightarrow{*} \alpha$  rezultă  $\alpha = \alpha_1\alpha_2 \dots \alpha_n$  și  $x_i \xRightarrow{*} \alpha_i$ ,  $(1 \leq i \leq n)$ .

Dacă  $x_i \in V_T$ , atunci  $\alpha_i = x_i$  și lui  $x_i$  îi va corespunde un nod final.

Dacă  $x_i \in V_N$  atunci  $x_i \xRightarrow{*} \alpha_i$  în mai puțin de  $k$  pași; conform ipotezei de inducție, există un  $x_i$  - arbore de derivare de frontieră  $\alpha_i$ . Fie  $T_i$  acest arbore, având rădăcina  $(1, i)$  cu  $h((1, i)) = x_i$ .  $A$  - arborele asociat acestei derivări va fi construit astfel:

Inițial



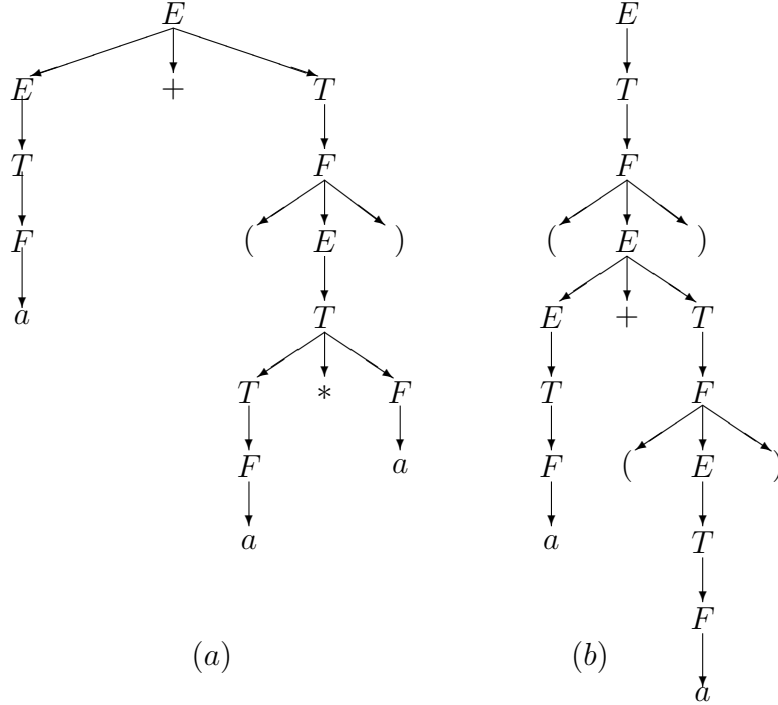
în care rădăcina este nodul  $(1)$ , cu notația  $h((1)) = A$ .

Apoi, pentru fiecare neterminal  $x_i$ , nodul  $x_i$  este înlocuit prin subarborele  $T_i$ .

Faptul că acest  $A$ -arbore are frontiera  $\alpha$  rezultă imediat prin construcție.

q.e.d.

**Exemplul 6.3** Fie gramatica  $G = (\{E, T, F\}, \{+, *, (, ), a\}, E, P)$  unde  
 $P = \{E \longrightarrow E + T \mid T, \quad T \longrightarrow T * F \mid F, \quad F \longrightarrow (E) \mid a\}$   
 și arborii de derivare:



Atunci arborelui de derivare (a) îi va corespunde o derivare

$$E \Longrightarrow E + T \Longrightarrow T + T \Longrightarrow F + T \Longrightarrow a + T \Longrightarrow a + F \Longrightarrow a + (E) \Longrightarrow a + (T) \Longrightarrow \\ \Longrightarrow a + (T * F) \Longrightarrow a + (F * F) \Longrightarrow a + (a * F) \Longrightarrow a + (a * a)$$

Invers, derivării

$$E \Longrightarrow T \Longrightarrow F \Longrightarrow (E) \Longrightarrow (E + T) \Longrightarrow (E + F) \Longrightarrow (E + (E)) \Longrightarrow (E + (T)) \Longrightarrow \\ \Longrightarrow (E + (F)) \Longrightarrow (E + (a)) \Longrightarrow (T + (a)) \Longrightarrow (F + (a)) \Longrightarrow (a + (a))$$

îi corespunde arborele (b).

Din cele arătate până acum rezultă că fiecărui arbore de derivare îi pot corespunde în general mai multe derivări. Două din acestea sunt deosebit de importante; anume:

**Definiția 6.3** O derivare  $A \xRightarrow{*} \alpha$  este o derivare stângă (sau "cea mai stângă derivare") dacă la fiecare pas variabila înlocuită nu are la stânga ei nici un alt neterminal.

Vom nota aceasta prin  $A \xRightarrow{*}_s \alpha$ .

În mod similar se definește derivarea dreaptă ("cea mai dreaptă derivare"),  $A \xRightarrow{*}_d \alpha$ .

În Exemplul 6.3, derivarea scrisă pentru arborele (a) este o derivare stângă, iar cea pentru arborele (b) - o derivare dreaptă.



**Propoziția 6.2** Pentru o gramatică independentă de context  $G = (V_N, V_T, S, P)$ , dacă  $A \xRightarrow{*} \alpha$ , atunci  $A \xRightarrow{*}_s \alpha$  (respectiv  $A \xRightarrow{*}_d \alpha$ ).

*Demonstrație:* Vom arată că  $A \xRightarrow{*}_s \alpha$  prin inducție după numărul  $n$  de pași ai derivării  $A \xRightarrow{n} \alpha$ .

Pentru  $n = 0$  sau  $n = 1$ , afirmația este banală.

Să presupunem Propoziția 6.2 adevărată pentru toate derivările de lungime  $k \leq n$  și fie  $A \xRightarrow{*} \alpha_1 \xRightarrow{*} \alpha$  o derivare în  $n + 1$  pași. Dacă  $\alpha_1 = x_1 x_2 \dots x_p$  cu  $x_i \in V_N \cup V_T$ , atunci – conform Lemei 4.1 – putem scrie  $\alpha = u_1 u_2 \dots u_p$  și  $x_i \xRightarrow{n_i} \alpha_i$  printr-o derivare de lungime  $n_i$  (când  $x_i \in V_T$ , atunci  $n_i = 0$ ). Cum  $n = \sum_{i=1}^p n_i$ , vom avea  $n_i \leq n$  deci, conform ipotezei de inducție,  $x_i \xRightarrow{*}_s u_i$ .

Putem construi acum derivarea stângă

$$A \xRightarrow{*} x_1 x_2 \dots x_p \xRightarrow{*}_s u_1 x_2 \dots x_p \xRightarrow{*}_s u_1 u_2 x_3 \dots x_p \xRightarrow{*}_s \dots \xRightarrow{*}_s u_1 u_2 \dots u_p = \alpha$$

Demonstrația pentru existența derivării drepte se face similar.

q.e.d.

**Propoziția 6.3** Pentru fiecare arbore de derivare  $\Gamma$  există o derivare stângă (dreaptă) unică.

*Demonstrație:* Se face similar cu demonstrația Teoremei 6.1, prin inducție după numărul de noduri interioare ale arborelui  $\Gamma$ .

q.e.d.

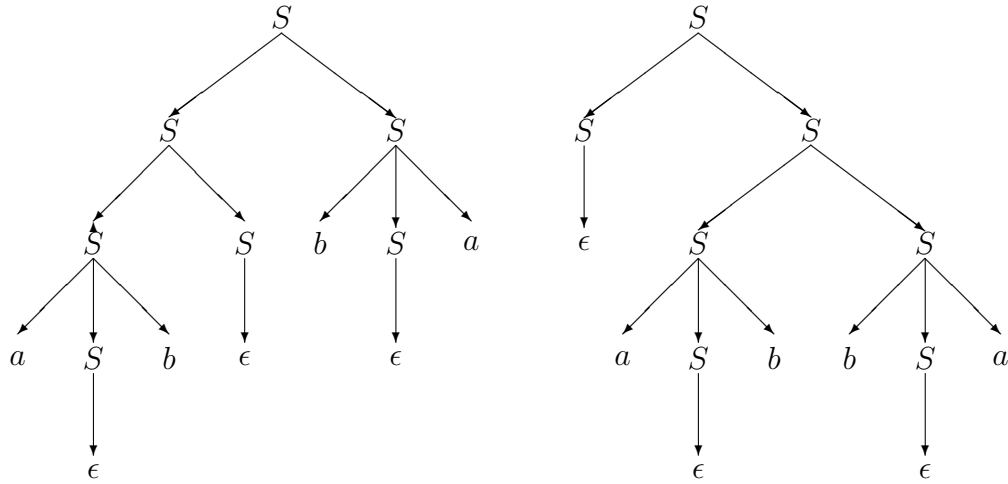
**Definiția 6.4** O gramatică independentă de context  $G$  este "ambiguă" dacă există  $w \in L(G)$  care are două derivări stângi (drepte) distincte. În caz contrar,  $G$  este o gramatică neambiguă.

Folosind Propoziția 6.3, putem spune că o gramatică este ambiguă dacă pentru un  $w \in L(G)$  se pot construi doi arbori de derivare distincți.

Un limbaj independent de context  $L$  este neambiguu dacă există o gramatică neambiguă  $G$  cu  $L(G) = L$ . Altfel,  $L$  este *inerent ambiguu*.

**Exemplul 6.4** Gramatica de producții  $S \longrightarrow aSb|bSa|SS|\epsilon$ , care generează limbajul  $L = \{w \mid |w|_a = |w|_b\}$  este o gramatică ambiguă.

Într-adevăr, pentru cuvântul  $w = abba$  se pot construi cel puțin doi arbori de derivare distincți:



Studiul ambiguității gramaticilor și limbajelor este dificil și nu se poate efectua decât pentru gramatici (limbaje) particulare; plecând de la un limbaj sau o gramatică independentă de context oarecare, nu se cunoaște nici un algoritm care să decidă proprietatea de ambiguitate sau neambiguitate.

### 6.3 Gramatici independente de context proprii

După cum s-a văzut, o gramatică independentă de context este caracterizată de forma generală a producțiilor sale:  $A \longrightarrow \alpha$  cu  $A \in V_N$ ,  $\alpha \in (V_N \cup V_T)^*$ .

Uneori însă este de dorit să impunem restricții asupra formei producțiilor, fără a micșora puterea generativă a gramaticii. Aceste restricții sunt necesare în special în aplicațiile limbajelor independente de context.

În cele ce urmează, fie  $G = (V_N, V_T, S, P)$  o gramatică independentă de context cu  $L(G) = L$ ,  $L \neq \emptyset$ .

#### Definiția 6.5

- $X \in V_N \cup V_T$  este un simbol "inaccesibil" dacă nu există nici o derivare  $S \xRightarrow{*} \alpha X \gamma$  cu  $\alpha, \gamma \in (V_N \cup V_T)^*$ ;
- $X \in V_N \cup V_T$  este un simbol "neutilizabil" dacă nu există nici o derivare  $S \xRightarrow{*} \alpha X \gamma \xRightarrow{*} \alpha \beta \gamma$  cu  $\alpha, \beta, \gamma \in V_T^*$ .

**Lema 6.1** Gramatica  $G$  este echivalentă cu o gramatică  $G'$  fără simboluri inaccesibile.

*Demonstrație:* Simbolurile inaccesibile din  $G$  pot fi eliminate folosind algoritmul liniar:

**Algoritm 6.2:**

**Intrare:**  $G = (V_N, V_T, S, P)$  independentă de context

**Ieșire:**  $G' = (V_N', V_T', S', P')$  cu proprietățile:

- i.  $L(G) = L(G')$
- ii.  $\forall A \in V_N' \cup V_T', \exists \alpha, \beta \in (V_N' \cup V_T')^*$  cu  $S' \xRightarrow{*} \alpha A \beta$ .

**Algoritm:**

- 1.  $V_0 := \{S\}, \quad i := 1;$
- 2.  $V_i = V_{i-1} \cup \{X \mid A \longrightarrow \alpha X \beta \in P, \quad A \in V_{i-1}\};$
- 3. Dacă  $V_i \neq V_{i-1}$  atunci  $i := i + 1$  și salt la Pasul 2;
- 4.  $V_N' := V_i \cap V_N, \quad V_T' := V_i \cap V_T, \quad S' := S, \quad P'$  conține producțiile lui  $P$  care au în ambii membri numai simboluri din  $V_i$ .

Deoarece  $P' \subseteq P$ , gramatica  $G'$  va fi tot independentă de context.

În plus, cum  $V_i \subseteq V_N \cup V_T$ , procesul de calcul se termină după un număr finit de pași (maxim  $|V_N \cup V_T|$ ), deci este un algoritm.

Pentru a încheia demonstrația, trebuie arătat că  $G'$  verifică condițiile (i) și (ii).

Proprietatea (ii) este îndeplinită prin construcția algoritmului. Pentru (i), vom arăta întâi prin inducție după  $n$  afirmația:

$$\text{Dacă } S \xRightarrow{n} \alpha X \beta \text{ atunci există } i \leq n \text{ cu } X \in V_i. \quad (3)$$

Când  $n = 1$ , avem banal  $i = 1$  ( $X \in V_1$ ).

Să presupunem (3) adevărată pentru  $n$  și să o arătăm pentru  $n + 1$ ; fie deci derivarea  $S \xRightarrow{n+1} \alpha X \beta$ . Apar două cazuri:

- 1.  $X \in V_n$ , și am terminat (există  $i \leq n$  cu  $X \in V_i \subseteq V_n$ );
- 2.  $X \notin V_n$ . Aceasta înseamnă că  $X$  nu a apărut pe parcursul derivării; punând în evidență ultimul pas, avem  $S \xRightarrow{n} X_1 X_2 \dots X_p \xRightarrow{} \alpha X \beta$ .  
Atunci  $\exists j \leq p$ , cu  $X_j \longrightarrow \gamma X \delta$  astfel încât  $\alpha = X_1 X_2 \dots X_{j-1} \gamma$ ,  $\beta = \delta X_{j+1} \dots X_p$ .  
Evident,  $X_j \in V_n$ , deci  $X \in V_{n+1}$  ( $i = n + 1$ ).

Fie acum  $w \in L(G)$ , deci  $S \xRightarrow{*} w$ , derivare în gramatica  $G$ . Există atunci (conform cu (3)) un  $i$  cu  $w \in V_i^*$ ; cum  $w \in V_T^*$ , va rezulta  $w \in (V_T \cap V_i)^*$ , deci  $w \in (V_T')^*$ , adică  $w \in L(G')$ . Altfel spus,  $L(G) \subseteq L(G')$ .

Incluziunea inversă este banală:  $S' \xRightarrow{*} w$  a folosit în derivare numai reguli din  $P$ , deci derivarea  $S \xRightarrow{*} w$  este o derivare atât în  $G$  cât și în  $G'$ . q.e.d.

**Lema 6.2** *Gramatica  $G$  este echivalentă cu o gramatică  $G'' = (V_N'', V_T, S, P'')$  cu proprietatea*

$$\forall A \in V_N'', \exists \alpha \in V_T^* \text{ cu } A \xRightarrow{*} \alpha \text{ (derivare în gramatica } G'')$$

*Demonstrație:* Construim noua gramatică folosind algoritmul recursiv:

**Algoritm 6.3:**

**Intrare:**  $G = (V_N, V_T, S, P)$  independentă de context.

**Ieșire:**  $G'' = (V_N'', V_T, S, P'')$  cu proprietățile:

- i.  $L(G) = L(G'')$ ;
- ii.  $\forall A \in V_N'', \exists \alpha \in V_T^* \text{ cu } A \xRightarrow{*} \alpha$ .

**Algoritm:**

1.  $V_0 := \emptyset, \quad i := 1$ ;
2.  $V_i = V_{i-1} \cup \{A \mid A \rightarrow \alpha \in P, \quad \alpha \in (V_{i-1} \cup V_T)^*\}$ ;
3. Dacă  $V_i \neq V_{i-1}$  atunci  $i := i + 1$  și salt la Pasul 2;
4.  $V_N'' := V_i, \quad P'$  conține producțiile lui  $P$  care au în membrul stâng simboluri din  $V_i$ .

Deoarece  $V_i \subseteq V_N$ , procedeul descris este un algoritm (numărul de treeri prin Pasul 2 este maxim  $|V_N|$ ).

Să demonstrăm întâi prin inducție după  $i$  afirmația

$$\forall A \in V_i \implies \exists w \in V_T^* \text{ cu } A \xRightarrow{*} w \text{ (în gramatica } G'') \quad (4)$$

$i = 1$ : evident, prin construcție.

Să presupunem proprietatea (4) adevărată pentru  $i$  și fie  $A \in V_{i+1}$ .

- Dacă  $A \in V_i$ , pasul inductiv asigură veridicitatea afirmației.

- Dacă  $A \in V_{i+1} \setminus V_i$ , atunci există o producție  $A \rightarrow X_1 X_2 \dots X_n$ , unde  $X_k \in V_T \cup V_i, (1 \leq k \leq n)$ . Dacă  $X_k \in V_T$ , se ia  $w_k = X_k$ ; dacă  $X_k \in V_i$ , atunci ipoteza de inducție asigură existența lui  $w_k \in V_T^*$  cu  $X_k \xRightarrow{*} w_k$ .

Acum,  $A \xRightarrow{*} X_1 X_2 \dots X_n \xRightarrow{*} w_1 X_2 \dots X_n \xRightarrow{*} \dots \xRightarrow{*} w_1 w_2 \dots w_n = w$ .

Aserțiunea (ii) rezultă luând  $i$  maxim care verifică (4) (atunci  $V_N'' = V_i$ ).

Pentru (i) este nevoie de încă un rezultat preliminar:

Dacă  $A \xRightarrow{n} w$ ,  $w \in V_T^*$ , atunci  $\exists i$  cu  $A \in V_i$ . (5)

Vom demonstra aserțiunea (5) prin inducție după  $n$ .

Pentru  $n = 1$  afirmația este banală ( $i = 1$ ).

Să presupunem (5) adevărată pentru  $n$  și fie  $A \xRightarrow{n+1} w$ . Atunci putem scrie (punând în evidență primul pas al derivării):  $A \Rightarrow X_1 X_2 \dots X_k \xRightarrow{n} w$ . Conform Lemei 4.1,  $w = w_1 w_2 \dots w_k$  și  $X_j \xRightarrow{n_j} w_j$  cu  $n_j \leq n$ .

Dacă  $X_j \in V_T$ , atunci  $n_j = 0$  ( $w_j = X_j$ ) și luăm  $i_j = 0$ . Dacă  $X_j \in V_N$ , atunci – conform ipotezei de inducție –  $X_j \in V_{i_j}$  pentru un anumit  $i_j$ .

Fie  $i = 1 + \max\{i_1, i_2, \dots, i_k\}$ . Atunci, conform definiției,  $A \in V_i$ ; și aserțiunea (5) este demonstrată.

Să luăm acum  $\alpha \in L(G)$  (prin ipoteză,  $L(G) \neq \emptyset$ ), deci  $S \xRightarrow{*} \alpha$ . Conform cu (5), există  $i$  cu  $S \in V_i$ , ceea ce înseamnă  $S \in V_N$  și  $S \xRightarrow{*} \alpha$  în gramatica  $G''$  (regulile folosite din  $P$  sunt și în  $P''$ ). Deci  $L(G) \subseteq L(G'')$ .

Incluziunea inversă este banală, folosind aserțiunea (4). q.e.d.

**Corolarul 6.1** *Este decidabil dacă limbajul generat de o gramatică independentă de context este vid sau nu.*

*Demonstrație:* Fie  $G = (V_N, V_T, S, P)$  o cfg. Se folosește **Algoritmul 3**, în care Pasul 4 este înlocuit cu

4'. Dacă  $S \in V_i$  atunci  $L(G) \neq \emptyset$ , altfel  $L(G) = \emptyset$ .

Corectitudinea Corolarului 6.1 se bazează pe (4), în care  $A$  se înlocuiește cu  $S$ . q.e.d.

**Lema 6.3** *Gramatica  $G$  este echivalentă cu o gramatică  $G'$  fără simboluri neutilizabile.*

*Demonstrație:* Pentru eliminarea simbolurilor neutilizabile vom folosi algoritmul următor:

**Algoritm 6.4:**

**Intrare:**  $G = (V_N, V_T, S, P)$  independentă de context (cu  $L(G) \neq \emptyset$ );

**Ieșire:**  $G' = (V_N', V_T', S, P')$  cu proprietățile:

- i.  $L(G) = L(G')$ ;
- ii. În  $V_N' \cup V_T'$  nu sunt simboluri neutilizabile.

**Algoritm:**

1. Se aplică Algoritmul 6.3 gramaticii  $G$  și se obține gramatica  $G''$ ;
2. Se aplică Algoritmul 6.2 gramaticii  $G''$  și se obține gramatica  $G'$ .

Cu Pasul 1 se elimină din gramatica  $G$  toate neterminalele care nu pot genera secvențe de terminale. Ulterior, Pasul 2 va elimina toate simbolurile care nu sunt accesibile din  $S$ .

Egalitatea  $L(G) = L(G')$  se poate demonstra folosind Lemele 4.1 și 4.2.

Să presupunem că simbolul  $A \in V_N'$  este neutilizabil. Conform Definiției 6.5, aceasta se poate întâmpla numai în două cazuri:

1. În  $G'$  nu este posibilă nici o derivare  $S \xRightarrow{*} \alpha A \beta$ ,  $\alpha, \beta \in (V_T')^*$ .

Cum  $A$  este accesibil (altfel ar fi fost eliminat cu Algoritmul 2, la Pasul 2), rezultă că există  $u, v \in (V_N' \cup V_T')^*$  cu  $S \xRightarrow{*} uAv$ .

Neterminalele din  $u$  și  $v$  au proprietatea că generează secvențe de terminale. Deci  $\exists \alpha, \beta \in (V_T')^*$  astfel încât  $u \xRightarrow{*} \alpha$ ,  $v \xRightarrow{*} \beta$ . Rezultă că în  $G'$  există derivarea  $S \xRightarrow{*} \alpha A \beta$ , contradicție.

2.  $S \xRightarrow{*} \alpha A \beta$  este adevărată, dar  $\alpha A \beta \xRightarrow{*} \alpha \gamma \beta$  este falsă  $\forall \gamma \in (V_T')^*$ . Conform definiției derivărilor într-o gramatică independentă de context, rezultă că  $A \xRightarrow{*} \gamma$  este falsă  $\forall \gamma \in (V_T')^*$ , deci  $A$  trebuia eliminat de la Pasul 1 al Algoritmului 4.

În mod similar se arată că nici un terminal din  $V'$  nu este neutilizat.

q.e.d.

**Exemplul 6.5** Fie gramatica  $G = (\{S, A, B, C\}, \{a, b\}, S, P)$  unde

$$P = \{S \longrightarrow A|B, \quad A \longrightarrow aB|bS|b, \quad B \longrightarrow AB|Ba, \quad C \longrightarrow AS|b\}.$$

Să eliminăm simbolurile neutilizabile din această gramatică.

La primul pas se aplică Algoritmul 6.3. Construim secvența succesivă de mulțimi:

$$V_0 = \emptyset,$$

$$V_1 = V_0 \cup \{X \mid X \longrightarrow \alpha \in P, \alpha \in (V_0 \cup V_T)^*\} = \{X \mid X \longrightarrow \alpha \in P, \alpha \in V_T^*\} = \{A, C\}, \text{ neterminale introduse pe baza producțiilor } A \longrightarrow b \text{ respectiv } C \longrightarrow b.$$

$$V_2 = V_1 \cup \{X \mid X \longrightarrow \alpha \in P, \alpha \in (V_1 \cup V_T)^*\} = \{A, C\} \cup \{X \mid X \longrightarrow \alpha \in P, \alpha \in \{A, C, a, b\}^*\} = \{A, C\} \cup \{S, A, C\} = \{S, A, C\} \text{ (s-a putut folosi numai producția } S \longrightarrow A).$$

$$V_3 = V_2 \cup \{X \mid X \longrightarrow \alpha \in P, \alpha \in (V_2 \cup V_T)^*\} = \{S, A, C\} \cup \{X \mid X \longrightarrow \alpha \in P, \alpha \in \{S, A, C, a, b\}^*\} = \{S, A, C\} \cup \{S, A, C\} = \{S, A, C\} = V_2.$$

În acest moment Algoritmul 3 se termină și se obține gramatica  $G''$  cu

$$V_N'' = \{S, A, C\}, \quad V_T'' = \{a, b\}, \quad P'' = \{S \longrightarrow A, \quad A \longrightarrow bS|b, \quad C \longrightarrow AS|b\}$$

(toate producțiile care utilizează  $B$  se elimină deoarece  $B \notin V_N''$ ).

Acestei gramatici  $i$  se aplică Algoritmul 6.2; vom obține pas cu pas:

$$V_0 = \{S\};$$

$$V_1 = V_0 \cup \{X \mid A \longrightarrow \alpha X \beta \in P, A \in V_0\} = \{S\} \cup \{X \mid S \longrightarrow \alpha X \beta \in P\} = \{S\} \cup \{A\} = \{S, A\}, \text{ deoarece există o singură } S\text{-producție: } S \longrightarrow A.$$

$$V_2 = V_1 \cup \{X \mid Y \longrightarrow \alpha X \beta \in P, Y \in V_1\} = \{S, A\} \cup \{X \mid A \longrightarrow \alpha X \beta \in P \text{ sau } S \longrightarrow \alpha X \beta \in P\} = \{S, A\} \cup \{S, A, b\} = \{S, A, b\}, \text{ regulile utilizate fiind } S \longrightarrow A \text{ și } A \longrightarrow bS|b, \text{ care au în membrul drept pe } S, A \text{ și } b.$$

$V_3 = V_2 \cup \{X \mid Y \longrightarrow \alpha X \beta \in P, Y \in \{S, A\}\} = \{S, A, b\} \cup \{S, A, b\} = \{S, A, b\} = V_2$ .  
 Gramatica finală este  $G' = (V_N', V_T', S, P')$  unde  
 $V_N' = V_N'' \cap V_3 = \{S, A\}$ ,  $V_T' = V_T'' \cap V_3 = \{b\}$ ,  $P' = \{S \longrightarrow A, A \longrightarrow bS|b\}$ .  
 Deci din gramatica inițială au fost eliminate pe rând  $B$  (cu Algoritmul 6.3),  $a, C$  (cu Algoritmul 6.2), noua gramatică având și avantajul unui număr substanțial redus de producții. De remarcat că terminalul ' $a$ ' a dispărut din producții încă de la primul pas.

**Exemplul 6.6** Fie gramatica  $G = (\{S, A, B\}, \{a, b\}, S, P)$  unde  
 $P = \{S \longrightarrow A|a, A \longrightarrow AB, B \longrightarrow b\}$ .

Aplicând Algoritmul 6.3 (primul pas al Algoritmului 6.4) se obțin  $V_N'' = V_1 = \{S, B\}$ , deci  $P'' = \{S \longrightarrow a, B \longrightarrow b\}$ .

Apoi, cu Algoritmul 6.2 (pasul 2 al Algoritmului 6.4) avem  $V_2 = V_1 = \{S, a\}$ .

Deci  $G' = (\{S\}, \{a\}, S, \{S \longrightarrow a\})$ .

Dacă am aplica întâi Algoritmul 6.2, am obține ca rezultat că toate simbolurile sunt accesibile, deci gramatica nu se modifică. Utilizând în etapa a doua Algoritmul 6.3 se ajunge la gramatica  $G''$  drept gramatică finală, în care  $B$  este de fapt un simbol neutilizabil.

Deci ordinea de aplicare celor doi algoritmi (Algoritmul 6.3 urmat de Algoritmul 6.2) este importantă; utilizarea lor în ordine inversă nu elimină întotdeauna simbolurile neutilizabile.

**Definiția 6.6** Fie  $G = (V_N, V_T, S, P)$  o gramatică independentă de context. Atunci:

- O regulă de forma  $A \longrightarrow \epsilon$  se numește  $\epsilon$  - producție;
- O regulă de forma  $A \longrightarrow B$  se numește redenumire.

Ne vom îndrepta atenția în continuare asupra eliminării din  $P$  a  $\epsilon$  - producțiilor și a redenumirilor. Evident, dacă  $\epsilon \in L(G)$ , atunci nu se pot elimina toate  $\epsilon$  - producțiile; așa că în prima fază vom considera numai cazul  $\epsilon \notin L(G)$ .

Adăugarea lui  $\epsilon$  la  $L(G)$  se face ulterior, în mod similar cu condiția de la gramaticile dependente de context: introducem un simbol de start nou  $S'$  și producțiile  $S' \longrightarrow \epsilon|S$ . Redenumirea  $S' \longrightarrow S$  va fi eliminată în etapa următoare.

**Lema 6.4** Gramatica independentă de context  $G$  este echivalentă cu o gramatică  $G'$  în care:

1. Dacă  $\epsilon \notin L(G)$  atunci  $G'$  nu are  $\epsilon$  - producții;
2. Dacă  $\epsilon \in L(G)$  atunci singura  $\epsilon$  - producție este  $S' \longrightarrow \epsilon$  și simbolul de start  $S'$  nu apare în membrul drept al nici unei producții.

*Demonstrație:* Construcția gramaticii  $G'$  se face cu următorul algoritm:

**Algoritm 6.5:**

**Intrare:**  $G = (V_N, V_T, S, P)$  gramatică independentă de context;

**Ieșire:**  $G' = (V_N', V_T, S', P')$  cu proprietățile:

- i.  $L(G) = L(G')$ ;
- ii.  $G'$  nu are  $\epsilon$  - producții.

**Algoritm:**

1. Se construiește (cu un algoritm similar Algoritmului 6.2 sau 6.3) mulțimea  $N_\epsilon = \{A \mid A \in V_N, A \xRightarrow{*} \epsilon\}$ ; în plus,  $P' := \emptyset$ ;
2. Fie  $A \longrightarrow u_0 B_1 u_1 B_2 \dots u_{n-1} B_n u_n$ , ( $n \geq 0$ ), unde  $B_i \in N_\epsilon$ ,  $u_i \in (V_T \cup (V_N \setminus N_\epsilon))^*$ , ( $0 \leq i \leq n$ ). Dacă  $n = 0$ , atunci  $P' := P' \cup \{A \longrightarrow u_0\} \setminus \{A \longrightarrow \epsilon\}$  și salt la Pasul 4;
3.  $P' := P' \cup \{A \longrightarrow u_0 X_1 u_1 X_2 \dots u_{n-1} X_n u_n \mid X_i \in \{B_i, \epsilon\}, i = \overline{1, n}\} \setminus \{A \longrightarrow \epsilon\}$ ;
4.  $P := P \setminus \{A \longrightarrow u_0 B_1 u_1 B_2 \dots u_{n-1} B_n u_n\}$ ; dacă  $P \neq \emptyset$ , atunci salt la Pasul 2;
5. Dacă  $S \in N_\epsilon$ , atunci  $P' := P' \cup \{S' \longrightarrow \epsilon \mid S\}$ ,  $V_N' := V_N \cup \{S'\}$  și  $G' = (V_N', V_T, S', P')$ ; altfel  $G' = (V_N, V_T, S, P')$ .

Prin construcția asigurată de Algoritm 6.5, gramatica  $G'$  nu are  $\epsilon$ -producții (decât eventual regula  $S' \longrightarrow \epsilon$ ). Singura verificare care rămâne de făcut este echivalența gramaticilor  $G$  și  $G'$ . Pentru aceasta vom demonstra aserțiunea

$$A \xRightarrow{*} w \text{ (în } G) \iff w \neq \epsilon \text{ și } A \xRightarrow{*} w \text{ (în } G') \quad (6)$$

" $\implies$ ": Fie  $A \xRightarrow{i} w$  o derivare în gramatica  $G$  și  $w \neq \epsilon$ ; vom arăta prin inducție după  $i$  că  $A \xRightarrow{*} w$  în gramatica  $G'$ .

$i = 1$ : Atunci  $A \implies w$ ,  $w \neq \epsilon$ , deci  $A \longrightarrow w \in P$  nu este o  $\epsilon$ -producție. Atunci  $A \longrightarrow w \in P'$ , adică  $A \implies w$  în  $G'$ .

Să presupunem proprietatea (6) adevărată pentru orice derivare în mai puțin de  $i$  pași, și fie  $A \xRightarrow{i} w$  o derivare în  $G$ .

Punând în evidență primul pas al derivării,  $A \implies X_1 X_2 \dots X_n \xRightarrow{i-1} w$ . Conform Lemei 4.1,  $w = w_1 w_2 \dots w_n$  și  $X_k \xRightarrow{*} w_k$  în mai puțin de  $i$  pași ( $1 \leq k \leq n$ ).

Dacă  $X_k \in V_N$  și  $w_k \neq \epsilon$ , ipoteza de inducție asigură derivarea în  $G'$ :  $X_k \xRightarrow{*} w_k$ .

Dacă  $X_k \in V_T$ , atunci  $w_k = X_k$  și deci evident  $X_k \xRightarrow{*} w_k$  (în  $G'$ ).



Dacă  $w_k = \epsilon$ , atunci  $X_k \in N_\epsilon$ . Conform Algoritmului 5, pasul 3, în  $P'$  există producția  $A \longrightarrow \alpha_1 \alpha_2 \dots \alpha_n$ , în care

$$\alpha_k = \begin{cases} X_k & \text{dacă } w_k \neq \epsilon \\ \epsilon & \text{dacă } w_k = \epsilon \end{cases}$$

Deoarece  $w \neq \epsilon$ , rezultă  $\alpha_1 \alpha_2 \dots \alpha_n \neq \epsilon$ . Avem atunci derivarea în  $G'$ :

$$A \Longrightarrow \alpha_1 \alpha_2 \dots \alpha_n \xRightarrow{*} w_1 \alpha_2 \dots \alpha_n \xRightarrow{*} \dots \xRightarrow{*} w_1 w_2 \dots w_n = w.$$

” $\Leftarrow$ ”: Fie  $A \xRightarrow{i} w$  o derivare în  $i$  pași în  $G'$ . În mod cert  $w \neq \epsilon$  (deoarece  $G'$  nu are  $\epsilon$ -producții). Vom arăta prin inducție după  $i$  că  $A \xRightarrow{*} w$  în gramatica  $G$ .

$i = 1$ : Avem  $A \longrightarrow w \in P'$ , deci există o regulă  $A \longrightarrow \alpha_0 B_1 \alpha_1 B_2 \dots B_k \alpha_k \in P$  cu  $\alpha_0 \alpha_1 \dots, \alpha_k = w$ ,  $B_0, B_1, \dots, B_k \in N_\epsilon$ . În  $G$  se poate construi derivarea

$$A \Longrightarrow \alpha_0 B_1 \alpha_1 B_2 \dots B_k \alpha_k \xRightarrow{*} \alpha_0 \alpha_1 B_2 \dots B_k \alpha_k \xRightarrow{*} \dots \xRightarrow{*} \alpha_0 \alpha_1 \dots \alpha_k = w.$$

Să presupunem afirmația adevărată până la  $i$  și fie  $A \xRightarrow{i} w$  o derivare în  $G'$ . Punând în evidență primul pas, avem  $A \Longrightarrow X_1 X_2 \dots X_n \xRightarrow{i-1} w$ . Atunci

- $A \xRightarrow{*} X_1 X_2 \dots X_n$  în  $G$  (similar cazului  $i = 1$ );
- Conform Lemei 4.1,  $w = w_1 w_2 \dots w_n$  și  $X_k \xRightarrow{*} w_k$  (în  $G'$ ) în mai puțin de  $i$  pași ( $1 \leq k \leq n$ ).

Dacă  $X_k \in V_{N'}' = V_N$  atunci, conform ipotezei de inducție,  $X_k \xRightarrow{*} w_k$  (în  $G$ ).

Dacă  $X_k \in V_T$  atunci  $w_k = X_k$  și putem scrie  $X_k \xRightarrow{*} w_k$  (în  $G$ ).

Rezumând, avem derivarea în gramatica  $G$ :  $A \Longrightarrow X_1 X_2 \dots X_n \xRightarrow{*} w_1 w_2 \dots w_n = w$ .

În final, dacă se alege  $S$  în loc de  $A$ , avem

$$S \xRightarrow{*} w \text{ în } G' \iff w \neq \epsilon \text{ și } S \xRightarrow{*} w \text{ în } G$$

adică  $L(G') = L(G) \setminus \{\epsilon\}$ .

q.e.d.

**Exemplul 6.7** Fie gramatica de producții  $S \longrightarrow aSbS|bSaS|\epsilon$ , căreia îi aplicăm Algoritmul 6.5 de eliminare a  $\epsilon$ -producțiilor.

Inițial avem  $N_\epsilon = \{S\}$ .

Producția  $S \longrightarrow aSbS$  introduce în  $P'$  regulile  $S \longrightarrow aSbS|aSb|abS|ab$ .

Similar,  $S \longrightarrow bSaS$  introduce în  $P'$  pe  $S \longrightarrow bSaS|bSa|baS|ba$ .

Producția  $S \longrightarrow \epsilon$  se elimină.

În final, deoarece  $S \in N_\epsilon$ , se introduce un simbol de start nou  $S'$  și producțiile  $S' \longrightarrow \epsilon|S$ .

Se obține gramatica  $G' = (V_{N'}', V_T, S', P')$  unde  $V_{N'}' = \{S, S'\}$ ,  $V_T = \{a, b\}$ ,  $P' = \{S' \longrightarrow S|\epsilon, S \longrightarrow aSbS|aSb|abS|ab|bSaS|bSa|baS|ba\}$ .

**Lema 6.5** Gramatica  $G$  este echivalentă cu o gramatică  $G'$  fără redenumiri.

*Demonstrație:* Algoritmul de eliminare a redenumirilor este:

**Algoritm 6.6:**

**Intrare:**  $G = (V_N, V_T, S, P)$  gramatică independentă de context;

**Ieșire:**  $G' = (V_N, V_T, S, P')$  cu proprietățile:

i.  $L(G) = L(G')$ ;

ii.  $G'$  nu are redenumiri.

**Algoritm:**

1. Pentru fiecare  $A \in V_N$  se construiește mulțimea  $R_A = \{B | A \xRightarrow{*} B\}$  în mod iterativ, astfel:

(a)  $R_0 := \{A\}$ ,  $i := 1$ ;

(b)  $R_i := R_{i-1} \cup \{C | B \rightarrow C \in P, B \in R_{i-1}\}$ ;

(c) Dacă  $R_i \neq R_{i-1}$  atunci  $i := i + 1$  și salt la (b);

(d)  $R_A := R_i$ .

În plus,  $P' := \emptyset$ .

2. Fie  $B \rightarrow \alpha \in P$ ; dacă  $\alpha \in V_N$  atunci salt la Pasul 4;

3.  $P' := P' \cup \{A \rightarrow \alpha | B \in R_A\}$ ;

4.  $P := P \setminus \{B \rightarrow \alpha\}$ ; dacă  $P \neq \emptyset$  atunci salt la Pasul 2;

5.  $G' = (V_N, V_T, S, P')$ .

Faptul că în  $P'$  nu sunt redenumiri rezultă din construcția dată de algoritm: pentru fiecare redenumire se sare direct de la Pasul 2 la Pasul 4 fără a se mai adăuga nimic la mulțimea  $P'$ .

Mai trebuie arătată egalitatea  $L(G) = L(G')$

" $\subseteq$ ": Fie  $w \in L(G)$ . Deci în gramatica  $G$  există derivarea stângă

$$S = \alpha_0 \Rightarrow_s \alpha_1 \Rightarrow_s \alpha_2 \Rightarrow_s \dots \Rightarrow_s \alpha_n = w.$$

Dacă producția folosită pentru  $\alpha_i \Rightarrow_s \alpha_{i+1}$  nu este o redenumire, atunci aceeași generare directă  $\alpha_i \Rightarrow_s \alpha_{i+1}$  este valabilă și în  $G'$  ( $0 \leq i \leq n$ ).

Să presupunem că  $\alpha_i \Rightarrow_s \alpha_{i+1}$  printr-o redenumire și  $\alpha_{i-1} \Rightarrow_s \alpha_i$  nu a folosit redenumire, sau  $i = 0$ . Fie  $\alpha_{i+1} \Rightarrow_s \dots \Rightarrow_s \alpha_{j-1} \Rightarrow_s \alpha_j$  derivarea care folosește numai redenumiri și  $\alpha_j \Rightarrow_s \alpha_{j+1}$  prima generare directă fără redenumiri. Atunci  $\alpha_i, \alpha_{i+1}, \dots, \alpha_j$  au toate aceeași lungime și – pentru că am lucrat cu o derivare stângă – simbolul înlocuit

în fiecare din ele se află pe aceeași poziție. Dar atunci  $\alpha_i \Rightarrow_s \alpha_{j+1}$  este o generare directă în  $G'$  folosind o producție din  $P' \setminus P$ .

Folosind acum o procedură de inducție după  $n$ , rezultă  $S \xRightarrow{*}_s w$  în  $G'$ .

" $\supseteq$ ": Fie  $w \in L(G')$ ; deci există o derivare în gramatica  $G'$ :

$$S = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_n = w.$$

Dacă  $A \rightarrow \beta$  este producția folosită la generarea directă  $\alpha_i \Rightarrow \alpha_{i+1}$ , atunci există  $B \in V_N$  (posibil  $A = B$ ) astfel ca  $A \xRightarrow{*} B$  și  $B \xRightarrow{*} \beta$  (derivări în  $G$ ). Deci  $A \xRightarrow{*} \beta$  și rezultă  $\alpha_i \xRightarrow{*} \alpha_{i+1}$  (în  $G$ ). Folosind acest raționament pentru  $i = \overline{1, n-1}$  se obține derivarea în gramatica  $G$ :  $S \xRightarrow{*} w$ . q.e.d.

**Exemplul 6.8** Să eliminăm redenumirile din gramatica  $G = (\{E, T, F\}, \{+, *, (, ), a\}, E, P)$  de producții

$$P = \{E \rightarrow E + T | T, \quad T \rightarrow T * F | F, \quad F \rightarrow (E) | a\}.$$

Cu subalgoritmul de la Pasul 1 se obține  $R_E = \{E, T, F\}$ ,  $R_T = \{T, F\}$ ,  $R_F = \{F\}$ .

Regulile  $E \rightarrow T$  și  $T \rightarrow F$  sunt eliminate (fiind redenumiri).

Fie  $E \rightarrow E + T$ . Neterminalul  $E$  se află într-o singură mulțime  $R$ : în  $R_E$ . Deci  $E \rightarrow E + T \in P'$ .

Pentru  $T \rightarrow T * F$  avem  $T \in R_E$  și  $T \in R_T$ , deci  $E \rightarrow T * F$  și  $T \rightarrow T * F$  se includ în  $P'$ .

$F \rightarrow (E)$ ; cum  $F$  se află în toate cele trei mulțimi  $R$ , în  $P'$  se includ producțiile  $E \rightarrow (E)$ ,  $T \rightarrow (E)$ ,  $F \rightarrow (E)$ . La fel pentru regula  $F \rightarrow a$ .

În final, mulțimea  $P'$  a producțiilor este

$$P' = \{E \rightarrow E + T | T * F | (E) | a, \quad T \rightarrow T * F | (E) | a, \quad F \rightarrow (E) | a\}$$

**Exemplul 6.9** În Exemplul 6.5, după eliminarea simbolurilor neutilizabile s-a ajuns la gramatica  $G'$  de producții  $P' = \{S \rightarrow A, \quad A \rightarrow bS | b\}$  care conține o redenumire ( $S \rightarrow A$ ).

Pentru eliminarea ei avem  $R_S = \{S, A\}$ ,  $R_A = \{A\}$ .

Producția  $S \rightarrow A$  se elimină, iar pentru  $A \rightarrow bS$  se introduc în  $P_1$  regulile  $S \rightarrow bS$ ,  $A \rightarrow bS$ ; similar pentru  $A \rightarrow b$ . Noua gramatică  $G_1$  va avea

$$V_N^1 = \{S, A\}, \quad V_T^1 = \{b\}, \quad P_1 = \{S \rightarrow bS | b, \quad A \rightarrow bS | b\}.$$

Aplicând acum din nou Algoritmul 6.2 de eliminare a simbolurilor inaccesibile, neterminalul  $A$  se elimină și se ajunge la o gramatică de producții  $S \rightarrow bS | b$ ; simplitatea ei față de gramatica inițială din Exemplul 6.5 este evidentă.

**Definiția 6.7** O gramatică independentă de context fără simboluri neutilizabile,  $\epsilon$ -producții și redenumiri se numește gramatică proprie.

**Teorema 6.2** Orice gramatică independentă de context  $G$  cu  $L(G) \neq \emptyset$  este echivalentă cu o gramatică independentă de context proprie.

*Demonstrație:* Se aplică succesiv Algoritmii 6.3, 6.2, 6.5 și 6.6.

## 6.4 Forme normale

Înafara gramaticilor proprii – forme reduse ale gramaticilor independente de context – un rol deosebit de important îl joacă gramaticile în care producțiile au anumite forme standard, forme de mare ajutor în aplicațiile practice. Cunoscute sub o formă generalizată de *forme normale*, atenția noastră va fi îndreptată în special a două din ele:

- forma normală Chomsky (*FNC*),
- forma normală Greibach (*FNG*).

### 6.4.1 Forma normală Chomsky

**Teorema 6.3** *Orice limbaj independent de context  $L$  cu  $\epsilon \notin L$  este generat de o gramatică care are toate producțiile de forma*

$$A \longrightarrow BC \quad \text{sau} \quad A \longrightarrow a$$

*Demonstrație:* Deoarece  $L$  este independent de context, fie  $G = (V_N, V_T, S, P)$  o cfg cu  $L = L(G)$ . Folosind rezultatele din paragraful precedent, putem presupune că  $G$  este proprie (este important în special ca  $G$  să nu aibă redenumiri sau  $\epsilon$ -producții). Transformarea lui  $G$  într-o gramatică cu producțiile de tipul specificat în Teoremă se va face în două etape:

1. Fie  $A \longrightarrow \alpha \in P$  o producție a gramaticii  $G$ ;  $\alpha \neq \epsilon$  deoarece  $G$  nu are  $\epsilon$ -producții. Dacă  $|\alpha| = 1$ , atunci  $\alpha \in V_T$  (cum  $G$  nu are redenumiri,  $\alpha \notin V_N$ ) și producția rămâne nemodificată.

Fie  $\alpha = x_1x_2 \dots x_m$  ( $m \geq 2$ ). Pentru fiecare  $i$  ( $1 \leq i \leq m$ ), dacă  $x_i \in V_N$ , el va rămâne neschimbat. Dacă  $x_i = a \in V_T$ , introducem un neterminal nou  $C_a$  și

- În producția  $A \longrightarrow x_1x_2 \dots x_m$  înlocuim  $x_i$  cu  $C_a$ ;
- Adăugăm la  $P$  regula  $C_a \longrightarrow a$ .

Procedăm în acest fel cu toate producțiile lui  $P$  și obținem un set nou de reguli de forma  $A \longrightarrow a$  sau  $A \longrightarrow x_1x_2 \dots x_m$ ,  $x_i \in V_N'$ , unde  $V_N'$  este noua mulțime de neterminale; gramatica  $G' = (V_N', V_T, S, P')$  astfel obținută este echivalentă cu  $G$ .

2. Plecând de la gramatica  $G'$ , modificăm în continuare forma producțiilor pentru a ajunge la gramatica cerută în enunțul teoremei.

Producțiile de forma  $A \longrightarrow a$  rămân nemodificate.

Fie  $A \longrightarrow x_1x_2 \dots x_m$  o regulă din  $P'$  ( $m \geq 2$ ). Dacă  $m = 2$ , atunci ea nu se modifică. Dacă  $m \geq 3$ , adăugăm neterminalele noi  $D_1, D_2, \dots, D_{m-2}$  și înlocuim producția cercetată cu mulțimea de reguli

$$A \longrightarrow x_1D_1, \quad D_1 \longrightarrow x_2D_2, \quad \dots \quad D_{m-3} \longrightarrow x_{m-2}D_{m-2}, \quad D_{m-2} \longrightarrow x_{m-1}x_m.$$

Fie  $P''$  mulțimea producțiilor obținute prin prelucrarea tuturor regulilor din  $P'$  și  $V_N''$  noua mulțime de neterminale. Atunci gramatica  $G'' = (V_N'', V_T, S, P'')$  este de forma cerută de Teoremă și – în plus – verifică evident egalitatea  $L(G) = L(G'')$ . q.e.d.

Această formă a unei gramatici independente de context poartă numele de *Formă Normală Chomsky*.

**Exemplul 6.10** Să se aducă la forma normală Chomsky gramatica de producții

$$S \longrightarrow bA|aB, \quad A \longrightarrow bAA|aS|a, \quad B \longrightarrow aBB|bS|b.$$

În prima etapă se observă că singurele producții care nu se vor modifica sunt  $A \longrightarrow a$  și  $B \longrightarrow b$ . Penru celelalte reguli se introduc două neterminale noi  $C_a, C_b$  și două reguli noi  $C_a \longrightarrow a$ ,  $C_b \longrightarrow b$ , care se adaugă la  $P$ .

Celelalte producții din  $P$  se transformă în

$$S \longrightarrow C_bA|C_aB, \quad A \longrightarrow C_bAA|C_aS, \quad B \longrightarrow C_aBB|C_bS.$$

În etapa a doua se vor modifica numai producțiile care au lungimea membrului drept cel puțin 3: anume  $A \longrightarrow C_bAA$  și  $B \longrightarrow C_aBB$ . Pentru ele introducem neterminalele noi  $D_1, D_2$  și regulile în discuție se înlocuiesc cu

$$A \longrightarrow C_bD_1, \quad B \longrightarrow C_aD_2, \quad D_1 \longrightarrow AA, \quad D_2 \longrightarrow BB.$$

Gramatica în FNC rezultată va avea mulțimea de neterminale  $V_N'' = \{S, A, B, C_a, C_b, D_1, D_2\}$  și setul de producții

$$P'' = \begin{cases} S \longrightarrow C_bA|C_aB, & D_1 \longrightarrow AA, & C_a \longrightarrow a, \\ A \longrightarrow C_bD_1|C_aS|a, & D_2 \longrightarrow BB, & C_b \longrightarrow b. \\ B \longrightarrow C_aD_2|C_bS|b, \end{cases}$$

Să considerăm o derivare oarecare în  $G$ :

$$S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabSB \Rightarrow aabAB \Rightarrow aabbaB \Rightarrow aabbab.$$

Același cuvânt va avea în  $G''$  derivarea

$$S \Rightarrow C_aB \Rightarrow aB \Rightarrow aC_aD_2 \Rightarrow aaD_2 \Rightarrow aaBB \Rightarrow aaC_bSB \Rightarrow aabSB \Rightarrow aabC_bAB \Rightarrow aabbAB \Rightarrow aabbaB \Rightarrow aabbab.$$

### 6.4.2 Forma normală Greibach

O altă formă normală a gramaticilor independente de context  $G$  cu  $\epsilon \notin L(G)$ , deosebit de utilă în trecerile cfg – automat stivă – cfg, este *Forma Normală Greibach (FNG)*.

**Definiția 6.8** O gramatică  $G = (V_N, V_T, S, P)$  este în FNG dacă  $P$  nu are decât producții de forma  $A \longrightarrow a\alpha$  cu  $A \in V_N$ ,  $a \in V_T$ ,  $\alpha \in (V_N \cup V_T)^*$ .

Pentru început sunt necesare câteva rezultate preliminare.

**Definiția 6.9** O gramatică independentă de context proprie  $G = (V_N, V_T, S, P)$  este recursivă la stânga dacă ea conține cel puțin o producție de forma  $A \longrightarrow A\alpha \in P$ ,  $\alpha \in (V_N \cup V_T)^+$ ; neterminalul  $A$  folosit în această producție se numește recursiv la stânga.

Știm că o  $A$  - producție este o regulă de forma  $A \longrightarrow \alpha$ ; o producție  $A \longrightarrow A\alpha$  se numește  $A$  - producție recursivă la stânga.

**Lema 6.6** Fie  $G = (V_N, V_T, S, P)$  o gramatică independentă de context proprie recursivă la stânga. Atunci există o gramatică  $G' = (V_N', V_T, S, P')$  echivalentă, nerecursivă la stânga.

*Demonstrație:* Fie  $A \longrightarrow A\alpha_1 | A\alpha_2 | \dots | A\alpha_r$  mulțimea  $A$  - producțiilor recursive la stânga și  $A \longrightarrow \beta_1 | \beta_2 | \dots | \beta_s$  celelalte  $A$  - producții (pentru un neterminal  $A$  recursiv la stânga). Deoarece gramatica  $G$  este proprie, rezultă  $r > 0, s > 0$ , iar  $\alpha_i \neq \epsilon, \beta_i \neq \epsilon, \forall i \geq 1$ . Introducem un neterminal nou  $B$  și înlocuim toate  $A$  - producțiile din  $P$  cu setul de reguli:

$$\begin{aligned} A &\longrightarrow \beta_i, & A &\longrightarrow \beta_i B, & (1 \leq i \leq s), \\ B &\longrightarrow \alpha_i, & B &\longrightarrow \alpha_i B, & (1 \leq i \leq r). \end{aligned}$$

În acest mod, toate  $A$  - producțiile recursive la stânga au fost eliminate (nici un  $\beta_i$  nu începe cu  $A$ ). Deoarece  $B$  este un neterminal nou,  $B$  - producțiile introduse nu sunt recursive la stânga.

Se procedează similar pentru toate neterminalele recursive la stânga. Fie  $G'$  noua gramatică astfel obținută. Mai rămâne de arătat că  $L(G) = L(G')$ .

Cum  $G$  este o gramatică proprie, toate  $A$ -producțiile sunt utilizate în derivări efective. Mai mult, orice secvență de producții de forma  $A \longrightarrow A\alpha_i$  se încheie cu o regulă  $A \longrightarrow \beta_j$  (altfel nu se poate elimina neterminalul  $A$ ). Lucrând într-o gramatică independentă de context, toate aceste reguli se pot rearanja pentru a fi aplicate succesiv (eventual se lucrează cu o derivare stângă). Atunci secvența

$$A \Longrightarrow A\alpha_{i_1} \Longrightarrow A\alpha_{i_2}\alpha_{i_1} \Longrightarrow \dots \Longrightarrow A\alpha_{i_p}\alpha_{i_{p-1}}\dots\alpha_{i_1} \Longrightarrow \beta_j\alpha_{i_p}\dots\alpha_{i_1}$$

va fi înlocuită în  $G'$  prin derivarea

$$A \Longrightarrow \beta_j B \Longrightarrow \beta_j\alpha_{i_p} B \Longrightarrow \dots \Longrightarrow \beta_j\alpha_{i_p}\alpha_{i_{p-1}}\dots\alpha_{i_2} \Longrightarrow \beta_j\alpha_{i_p}\alpha_{i_2}\alpha_{i_1}.$$

Această substituție de derivări poate fi făcută pentru ambele incluziuni. q.e.d.

**Exemplul 6.11** Să eliminăm recursivitatea la stânga din gramatica de producții  

$$S \longrightarrow SS|a.$$

Există un singur neterminal recursiv la stânga –  $S$ . Aici  $\alpha_1 = S, \beta_1 = a, r = s = 1$ .

Introducând un nou simbol  $B$  și aplicând construcția din demonstrația Lemei 6.6, se obțin producțiile:

$$S \longrightarrow a, \quad B \longrightarrow S, \quad S \longrightarrow aB, \quad B \longrightarrow SB.$$

Aceasta nu este însă o gramatică proprie deoarece conține o redenumire ( $B \longrightarrow S$ ). Folosind Algoritmul de eliminare a redenumirilor, se obține gramatica echivalentă proprie  $G' = (\{S, B\}, \{a\}, S, \{S \longrightarrow a|aB, B \longrightarrow a|aB|SB\})$ .

**Lema 6.7** Fie  $A \longrightarrow \alpha_1 B \alpha_2$  o  $A$ -producție și  $B \longrightarrow \beta_1 | \beta_2 | \dots | \beta_r$  toate  $B$ -producțiile lui  $P$ . Notăm cu  $G_1 = (V_N, V_T, S, P_1)$  gramatica obținută din  $G$ , în care

$$P_1 = (P \setminus \{A \longrightarrow \alpha_1 B \alpha_2\}) \cup \{A \longrightarrow \alpha_1 \beta_1 \alpha_2 | \alpha_1 \beta_2 \alpha_2 | \dots | \alpha_1 \beta_r \alpha_2\}.$$

Atunci  $L(G) = L(G_1)$ .

*Demonstrație:* Incluziunea  $L(G_1) \subseteq L(G)$  este evidentă: dacă producția  $A \longrightarrow \alpha_1 \beta_i \alpha_2$  este folosită într-o derivare în gramatica  $G_1$ , ea poate fi înlocuită în  $G$  cu secvența

$$A \Longrightarrow \alpha_1 B \alpha_2 \Longrightarrow \alpha_1 \beta_i \alpha_2.$$

Pentru incluziunea  $L(G) \subseteq L(G_1)$ :  $A \longrightarrow \alpha_1 B \alpha_2$  este singura regulă din  $P \setminus P_1$ . Dacă într-o derivare din  $G$  se folosește producția  $A \longrightarrow \alpha_1 B \alpha_2$ , neterminalul  $B$  va fi rescris ulterior cu o regulă de forma  $B \longrightarrow \beta_i$ . Gramatica fiind independentă de context, cele două producții se pot aplica succesiv, și în  $G_1$  ele vor fi înlocuite cu  $A \longrightarrow \alpha_1 \beta_i \alpha_2$ . q.e.d.

**Exemplul 6.12** În gramatica de producții

$$P = \{E \longrightarrow E + T | T, \quad T \longrightarrow T * F | F, \quad F \longrightarrow (E) | a\}$$

să eliminăm regula  $T \longrightarrow T * F$ . Sunt două posibilități de aplica Lema 6.7.

- Se consideră  $B = F$ ,  $\alpha_1 = T*$ ,  $\alpha_2 = \epsilon$ . Atunci  $T \longrightarrow T * F$  se înlocuiește cu  $T \longrightarrow T * (E) | T * a$  și  $G_1$  va avea setul de producții

$$P_1 = \{E \longrightarrow E + T | T, \quad T \longrightarrow T * (E) | T * a | F, \quad F \longrightarrow (E) | a\}.$$

Dacă vrem să eliminăm și  $T \longrightarrow F$ , atunci în gramatica care se va obține, neterminalul  $F$  devine inaccesibil și se pot elimina toate  $F$ -producțiile, ajungând la setul de reguli

$$P_1' = \{E \longrightarrow E + T | T, \quad T \longrightarrow T * (E) | T * a | (E) | a\}.$$

- Dacă  $B = T$ ,  $\alpha_1 = \epsilon$ ,  $\alpha_2 = *F$ , producția  $T \longrightarrow T * F$  se înlocuiește cu  $T \longrightarrow T * F * F | F * F$  și gramatica  $G_1$  va avea setul de reguli

$$P = \{E \longrightarrow E + T | T, \quad T \longrightarrow T * F * F | F * F, \quad F \longrightarrow (E) | a\}.$$

**Teorema 6.4** Orice limbaj independent de context  $L$  cu  $\epsilon \notin L$  poate fi generat de o gramatică în FNG.

*Demonstrație:* Fie  $G = (V_N, V_T, S, P)$  o gramatică independentă de context cu  $L = L(G)$ . Conform Teoremei 6.3, putem presupune că  $G$  este în forma normală Chomsky. Fie

$V_N = \{A_1, A_2, \dots, A_n\}$  mulțimea neterminalelor, unde s-a notat  $A_1 = S$ . Algoritmul de aducere a gramaticii  $G$  la forma normală Greibach cuprinde două etape:

**Faza 1:** Aplicând Lemele 6.6, 6.7, se obține o gramatică echivalentă cu  $G$  care are numai reguli de forma  $A_i \longrightarrow a\gamma$  sau  $A_i \longrightarrow A_j\gamma$  cu  $i < j$ . Această fază este realizată de Algoritmul 6.7:

**Algoritmul 6.7:**

**Intrare:**  $G = (\{A_1, A_2, \dots, A_m\}, V_T, A_1, P)$  proprie.

**Ieșire:**  $G' = (V_N', V_T, A_1, P')$  cu

$$\begin{aligned} i. \quad & L(G) = L(G'); \\ ii. \quad & \forall A_i \longrightarrow \alpha \in P' \implies \begin{aligned} & \alpha = a\gamma \quad (a \in V_T) \text{ sau} \\ & \alpha = A_j\gamma \text{ cu } i < j. \end{aligned} \end{aligned}$$

1.  $V_N' := V_N, \quad P' := P;$
2.  $k := 1, \quad n := 0;$
3. Dacă  $A_k \longrightarrow A_k\alpha \in P$  atunci  $\begin{cases} V_N' := V_N' \cup \{B_k\}, \\ P' := (P' \cup \{B_k \longrightarrow \alpha | \alpha B_k\}) \setminus \{A_k \longrightarrow A_k\alpha\}, \\ n := 1. \end{cases}$
4. Dacă  $n = 0$  atunci salt la pasul 5.  
Altfel,  $\forall A_k \longrightarrow \beta \in P' \implies P' := P' \cup \{A_k \longrightarrow \beta B_k\}.$
5.  $k := k + 1.$  Dacă  $m < k$  atunci salt la Pasul 8; altfel  $j := 1.$
6. Pentru orice regulă  $A_k \longrightarrow A_j\alpha \in P' \implies$   
 $P' := (P' \cup \{A_k \longrightarrow \beta\alpha \mid A_j \longrightarrow \beta \in P'\}) \setminus \{A_k \longrightarrow A_j\alpha\}.$
7.  $j := j + 1.$  Dacă  $j < k$  atunci salt la Pasul 6; altfel  $n := 0$  și salt la Pasul 3.
8. Dacă  $P'$  nu are proprietatea (b), atunci salt la Pasul 2;  
altfel  $G' = (V_N', V_T, A_1, P'),$  STOP.

Se observă că Pașii 3 și 4 reprezintă de fapt aplicarea Lemei 6.6 (gramatica fiind proprie, aplicarea cel puțin odată a Pasului 3 duce la o aplicare a Pasului 4), iar Pasul 6 reprezintă aplicarea Lemei 6.7 cu  $\alpha_1 = \epsilon, \alpha_2 = \alpha, A = A_k, B = A_j.$

Cum acești pași constituie toate modificările de producții din algoritm (și – în plus – orice regulă din  $P$  este prelucrată cu unul din pașii 3, 4 sau 6), rezultă că  $L(G) = L(G').$

Pentru aserțiunea *ii.* vom raționa inductiv după  $k$ :



Pentru  $k = 1$  se aplică numai Pașii 3 și 4 care introduc în  $P'$  doar  $A_1$ -producții de forma  $A_1 \rightarrow \gamma$ , iar  $\gamma$  nu începe cu  $A_1$ .

Presupunem că pentru toți  $i$  ( $1 \leq i \leq k$ ),  $A_i \rightarrow A_j \gamma \in P'$  numai dacă  $i < j$ ; vom prelucra acum  $A_k$  - producțiile.

Dacă  $A_k \rightarrow A_j \gamma \in P'$  și  $k < j$  sau dacă  $A_k \rightarrow a \gamma$  cu  $a \in V_T$ , atunci se poate aplica numai Pasul 4, care păstrează aceste reguli în  $P'$ .

Dacă  $A_k \rightarrow A_j \gamma \in P'$  și  $j < k$ , atunci – cu Pasul 6 – această regulă se înlocuiește cu mulțimea de producții  $\{A_k \rightarrow \beta \gamma \mid A_j \rightarrow \beta \in P'\}$ . Dar cum  $j < k$ , în  $P'$  au fost introduse (conform ipotezei de inducție) la un pas anterior, numai reguli în care  $\beta$  are una din formele  $a \gamma$  (cu  $a \in V_T$ ) sau  $A_s \gamma$  cu  $j < s$ .

Deci o trecere prin algoritm ridică indicele  $j$  din  $A_k \rightarrow A_j \gamma$  cu cel puțin o unitate. Rezultă că după cel mult  $k - 1$  treceri (asigurate de Pasul 7) obținem numai  $A_k$  - reguli de forma  $A_k \rightarrow A_s \gamma$  cu  $s \geq k$  sau  $A_k \rightarrow a \gamma$ . Producțiile cu  $s = k$  sunt apoi eliminate la trecerea următoare prin Pasul 3.

Deci numărul de treceri asigurat de Pasul 7 este finit; rezultă că procedura propusă este un algoritm.

**Observația 6.5** Rolul lui  $n$  este de a nu introduce simboluri  $B_k$  neutilizabile (eliminate apoi cu Algoritmul 4).

**Faza 2:** În urma aplicării Algoritmului 6.7, gramatica  $G'$  are numai producții de forma  $A_i \rightarrow A_j \gamma$  ( $i < j$ ),  $A_i \rightarrow a \gamma$ ,  $B_i \rightarrow \gamma$ ,  $\gamma \in (V_T \cup \{B_1, B_2, \dots, B_m\})^*$  (7)

Pentru a ajunge la forma cerută de Teorema 6.4, aplicăm următorul algoritm:

**Algoritmul 6.8:**

**Intrare:**  $G' = (V_N', V_T, A_1, P')$  cu producțiile de forma (7);

**Ieșire:**  $G'' = (V_N'', V_T, A_1, P'')$  cu proprietățile

a)  $L(G'') = L(G')$ ;

b)  $P'' = \{A \rightarrow a \gamma \mid a \in V_T, \gamma \in (V_N'' \cup V_T)^*\}$ .

1.  $P'' := P$ ,  $k := m - 1$ ;

2. Pentru fiecare  $A_k \rightarrow A_j \alpha \in P' \implies$   
 $P'' := (P'' \cup \{A_k \rightarrow \beta \alpha \mid A_j \rightarrow \beta \in P'\}) \setminus \{A_k \rightarrow A_j \alpha\}$ .

3.  $k := k - 1$ ; dacă  $1 \leq k$  atunci salt la Pasul 2;

4. Pentru fiecare  $B_i \rightarrow A_j \alpha \in P''$ , ( $1 \leq i \leq m$ ,  $1 \leq j \leq m$ )  $\implies$   
 $P'' := (P'' \cup \{B_i \rightarrow \beta \alpha \mid A_j \rightarrow \beta \in P''\}) \setminus \{B_i \rightarrow A_j \alpha\}$ .

5.  $G'' = (V_N', V_T, A_1, P'')$ .

Aserțiunea (a) rezultă din faptul că singurele modificări de producții – cele date de Pași 2 și 4 – s-au efectuat conform cu Lema 7, care păstrează echivalența gramaticilor.

De asemenea, numărul de pași este finit (cel mult  $|P'|$ ), deci avem de-a face cu un algoritm. Mai rămâne de verificat aserțiunea (b).

Din Algoritmul 6.7 rezultă că toate  $A_m$ -producțiile sunt de forma  $A_m \rightarrow a\gamma$  ( $m$  este indicele maxim al neterminalelor din  $V_N$ ). Pentru  $m-1$ ,  $A_{m-1}$ -producțiile sunt de una din formele  $A_{m-1} \rightarrow a\gamma$  (și nu se mai modifică nimic) sau  $A_{m-1} \rightarrow A_m\gamma$ . Trecerea prin Pasul 2 înlocuiește regula  $A_{m-1} \rightarrow A_m\gamma$  cu producțiile  $A_{m-1} \rightarrow \beta\gamma$  pentru toate  $A_m$ -producțiile  $A_m \rightarrow \beta$ . Dar – conform construcției –  $\beta = a\gamma'$ , deci  $A_{m-1} \rightarrow a\gamma'\gamma$ .

Algoritmul 6.8 prelucrează în mod analog, în mod descrescător, toate regulile având succesiv ca membru stâng  $A_{m-1}, A_{m-2}, \dots, A_1$ .

În final, să examinăm regulile care au  $B$ -uri ca membru stâng. Orice  $B_k$ -producție este introdusă de Pasul 2 al Algoritmului 6.7, și nu mai este modificată de nici o trecere ulterioară prin algoritm. Ea apare în urma înlocuirii unei reguli  $A_k \rightarrow A_k\alpha$ .

Cum gramatica inițială a fost în  $FNC$ , folosind o inducție simplă după numărul de aplicări ale Lemelor 6.6 și 6.7, obținem că orice  $A_i$ -producție ( $1 \leq i \leq m$ ) începe în dreapta cu un terminal sau cu un prefix de forma  $A_j A_k$ . Deci cuvântul  $\alpha$  din Pasul 2 al Algoritmului 6.8 nu este niciodată  $\epsilon$  sau cu un  $B$  pe prima poziție. Rezultă că orice  $B_i$ -producție începe în dreapta sau cu un terminal, sau cu un  $A_j$ . Aplicând încă odată Lema 6.7 (Pasul 4), obținem gramatica  $G''$  sub forma normală cerută. q.e.d.

**Observația 6.6** Gramatica  $G''$  obținută în urma aplicării Algoritmilor 6.7 și 6.8 nu este în general proprie (pot exista simboluri  $B_k$  neutilizabile). Se verifică ușor că aplicarea în continuare a Algoritmilor 6.2 – 6.6 pentru a transforma  $G''$  într-o gramatică proprie conduce tot la o gramatică în  $FNG$ .

**Exemplul 6.13** Să aducem la  $FNG$  gramatica de producții

$$P = \{E \rightarrow E + T | T, \quad T \rightarrow T * F | F, \quad F \rightarrow (E) | a\}.$$

Pentru a aplica mai ușor Algoritmul 6.7, vom renota neterminalele, astfel încât producțiile vor avea forma

$$A_1 \rightarrow A_1 + A_2 | A_2, \quad A_2 \rightarrow A_2 * A_3 | A_3, \quad A_3 \rightarrow (A_1) | a$$

unde  $m = 3$ .

**Pas 1:**  $P' := P$ ;

**Pas 2:**  $k := 1, \quad n := 0$ ;

**Pas 3:** Regula  $A_1 \rightarrow A_1 + A_2$  se înlocuiește în  $P'$  cu  $B_1 \rightarrow +A_2 | +A_2 B_1$ , unde  $B_1$  este un neterminal nou. În plus,  $n := 1$ ;

**Pas 4:** Cum  $n \neq 0$ , regula  $A_1 \rightarrow A_2$  introduce în  $P'$  producția  $A_1 \rightarrow A_2 B_1$ ;

În acest moment,  $P'$  este formată din producțiile

$$P' = \{A_1 \rightarrow A_2 B_1 | A_2, \quad A_2 \rightarrow A_2 * A_3 | A_3, \quad A_3 \rightarrow (A_1) | a, \quad B_1 \rightarrow +A_2 | +A_2 B_1\}.$$

**Pas 5:**  $k := 2$ . Cum  $k \leq 3$ , se ia  $j := 1$  și se trece la Pasul 6;

**Pas 6:** Pentru că nu există producții de forma  $A_2 \rightarrow A_1\alpha$ ,  $P'$  nu se modifică;

**Pas 7:**  $j := 2$ . Condiția  $j \leq k - 1$  nu este verificată, deci  $n := 0$  și salt la pasul 3;

**Pas 3:** Regula  $A_2 \rightarrow A_2 * A_3$  este scoasă din  $P'$  și înlocuită cu  $B_2 \rightarrow *A_3 | *A_3B_2$ , unde  $B_2$  este un neterminal nou. În plus,  $n := 1$ .

**Pas 4:** Deoarece  $n \neq 0$ , producția  $A_2 \rightarrow A_3$  introduce în  $P'$  producția  $A_2 \rightarrow A_3B_2$ .

În acest moment,  $P' = \{A_1 \rightarrow A_2B_1 | A_2, A_2 \rightarrow A_3B_2 | A_3, A_3 \rightarrow (A_1)|a, B_1 \rightarrow +A_2 | +A_2B_1, B_2 \rightarrow *A_3 | *A_3B_2\}$ ;

**Pas 5:**  $k := 3$ . Condiția  $k > 3$  nu este verificată, deci  $j := 1$  și salt la Pasul 6;

**Pas 6:**  $P'$  nu se modifică (deoarece nu are reguli de forma  $A_3 \rightarrow A_1\alpha$ );

**Pas 7:**  $j := 2$ . Condiția  $j \leq 2$  se verifică și se reia Pasul 6;

**Pas 6:**  $P'$  nu conține reguli de forma  $A_3 \rightarrow A_2\alpha$ ;

**Pas 7:**  $j := 3$ . Condiția  $j \leq 2$  nu este îndeplinită, așa că  $n := 0$  și salt la Pasul 3;

**Pas 3:** Nu există reguli de forma  $A_3 \rightarrow A_3\alpha$ ;

**Pas 4:** Cum  $n = 0$ , se trece la Pasul 5;

**Pas 5:**  $k := 4$ . Acum  $k > 3$  este falsă, deci urmează Pasul 8;

**Pas 8:**  $P'$  are proprietatea ii.;

**Pas 9:** Se obține gramatica  $G' = (V_N', V_T, A_1, P')$  unde  $V_N' = \{A_1, A_2, A_3, B_1.B_2, B_3\}$ ,  $P' = \{A_1 \rightarrow A_2B_1 | A_2, A_2 \rightarrow A_3B_2 | A_3, A_3 \rightarrow (A_1)|a, B_1 \rightarrow +A_2 | +A_2B_1, B_2 \rightarrow *A_3 | *A_3B_2\}$ .

Cu această gramatică se începe Algoritmul 6.8:

**Pas 1:**  $P'' := P'$ ,  $k := 2$ ;

**Pas 2:** Producția  $A_2 \rightarrow A_3B_2$  se înlocuiește în  $P''$  cu  $A_2 \rightarrow (A_1)B_2|aB_2$ ; similar, în locul regulei  $A_2 \rightarrow A_3$  se introduc producțiile  $A_2 \rightarrow (A_1)|a$ .  $P''$  va fi acum

$P'' = \{A_1 \rightarrow A_2B_1, A_2 \rightarrow (A_1)B_2|aB_2|(A_1)|a, A_3 \rightarrow (A_1)|a, B_1 \rightarrow +A_2 | +A_2B_1, B_2 \rightarrow *A_3 | *A_3B_2\}$ ;

**Pas 3:**  $k := 1$ ; cum  $k \geq 1$  este adevărată, se reia pasul 2;

**Pas 2:**  $A_1 \rightarrow A_2B_1$  se înlocuiește cu  $A_1 \rightarrow (A_1)B_2B_1|aB_2B_1|(A_1)B_1|aB_1$ , iar producția  $A_1 \rightarrow A_2$  cu  $A_1 \rightarrow (A_1)B_2|aB_2|(A_1)|a$ . După acest pas,  $P''$  este format din

$P'' = \{A_1 \rightarrow (A_1)B_2B_1|(A_1)B_1|(A_1)B_2|(A_1)|aB_2B_1|aB_1|aB_2|a, A_2 \rightarrow (A_1)B_2|(A_1)|aB_2|a, A_3 \rightarrow (A_1)|a, B_1 \rightarrow +A_2 | +A_2B_1, B_2 \rightarrow *A_3 | *A_3B_2\}$ ;

**Pas 3:**  $k := 0$ . Condiția  $k \geq 1$  nu mai este îndeplinită;

**Pas 4:** Nu există producții de forma  $B_i \rightarrow A_j\alpha$ , deci  $P''$  rămâne nemodificat și gramatica în FNG este cea cu producțiile date mai sus, iar  $V_N'' = V_N' \setminus \{B_3\}$ .

Să luăm o derivare arbitrară în gramatica  $G$ :

$A_1 \Rightarrow A_2 \Rightarrow A_2 * A_3 \Rightarrow (A_1) * A_3 \Rightarrow (A_1 + A_2) * A_3 \Rightarrow (A_2 + A_2) * A_3 \Rightarrow (A_3 + A_2) * A_3 \Rightarrow (a + A_2) * A_3 \Rightarrow (a + A_3) * A_3 \Rightarrow (a + 1) * A_3 \Rightarrow (a + a) * a$ .

Corespunzător, în  $G''$  are loc derivarea:

$A_1 \Rightarrow (A_1)B_2 \Rightarrow (aB_1)B_2 \Rightarrow (a + A_2)B_2 \Rightarrow (a + a)B_2 \Rightarrow (a + a) * A_3 \Rightarrow (a + a) * a$ .

**Observația 6.7** Algoritmii 6.7 și 6.8 aduc la Forma Normală Greibach și gramatici care nu sunt proprii sau în FNC. Singura condiție necesară este să nu existe  $\epsilon$ -producții (condiție asigurată de Algoritmul 6.5 și de ipoteza  $\epsilon \notin L$ ).

## 6.5 Teorema $uvwxy$ și aplicații

Lema de pompare pentru limbajele regulate – a cărei importanță am văzut la momentul cuvenit – poate fi tratată ca un caz particular al unei teoreme valabile pentru limbajele independente de context.

Cunoscută mai ales sub numele de *Teorema  $uvwxy$*  (după forma sub care se pot descompune cuvintele unui limbaj independent de context), lema de pompare pentru limbajele independente de context are o serie de consecințe importante în dezvoltarea studiului limbajelor din această clasă.

### 6.5.1 Enunț și demonstrație. Variante

**Teorema 6.5** Fie  $L$  un limbaj independent de context. Există atunci o constantă  $n$  astfel ca pentru orice cuvânt  $\alpha \in L$ ,  $|\alpha| \geq n$ , este posibilă descompunerea  $\alpha = uvwxy$  cu

1.  $|vx| \geq 1$ ;
2.  $|vwx| \leq n$ ;
3.  $uv^iwx^iy \in L, \quad \forall i \geq 0$ .

*Demonstrație:* Fie  $G = (V_N, V_T, S, P)$  o cfg în FNC cu  $L(G) = L \setminus \{\epsilon\}$ . Vom arăta prin inducție după  $i$  următoarea aserțiune:

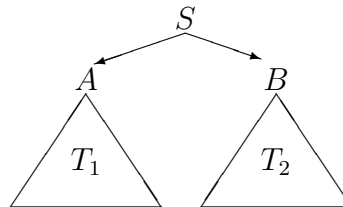
$\forall \alpha \in L(G)$ , dacă arborele de derivare pentru  $\alpha$  nu are drumuri de lungime mai mare decât  $i$ , atunci  $|\alpha| \leq 2^{i-1}$ . (8)

$i := 1$ : afirmație banală, deoarece – gramatica fiind în Forma Normală Chomsky –

singurul arbore posibil cu drum de lungime 1 este



Fie acum  $i > 1$  și arborele pentru  $\alpha$  de forma:



Dacă subarborii  $T_1$  și  $T_2$  au frontierele  $\alpha_1$  respectiv  $\alpha_2$ , atunci  $\alpha = \alpha_1\alpha_2$ . Cum  $T_1$  și  $T_2$  au drumuri de lungimi maxim  $i - 1$ , rezultă (conform ipotezei de inducție)  $|\alpha_1| \leq 2^{i-2}$ ,  $|\alpha_2| \leq 2^{i-2}$ . Deci  $|\alpha| = |\alpha_1| + |\alpha_2| \leq 2^{i-2} + 2^{i-2} = 2^{i-1}$ .

Revenind la demonstrația Teoremei 6.5, fie  $k = |V_N|$  și  $n = 2^k$ . Să considerăm acum  $\alpha \in L(G)$ ,  $|\alpha| \geq n = 2^k$ . Conform cu (8), orice arbore de derivare pentru  $\alpha$  trebuie să aibă cel puțin un drum de lungime minimum  $k + 1$ . Un astfel de drum are cel puțin  $k + 2$  vârfuri, dintre care doar ultimul nu este notat cu neterminale.

Cum în  $V_N$  sunt  $k$  neterminale și drumul are minim  $k + 1$  noduri interioare, rezultă că pe acest drum există cel puțin două noduri distincte notate cu același neterminal.

Formal, există două vârfuri  $v_1$ ,  $v_2$  cu proprietățile:

- i.  $h(v_1) = h(v_2) = A$ ,  $A \in V_N$ ;
- ii.  $v_2$  este un descendent al lui  $v_1$ ;
- iii. Drumul de la  $v_1$  la nodul terminal este de lungime cel mult  $k + 1$ .

Nodurile  $v_1$  și  $v_2$  pot fi găsite efectiv, parcurgând drumul de la nodul terminal spre rădăcină, reținând primele două noduri cu aceeași notație.

Notăm cu  $w$  frontiera subarborelui de rădăcină  $v_2$  (deci  $A \xRightarrow{*} w$ ) și  $\beta$  frontiera subarborelui de rădăcină  $v_1$  ( $A \xRightarrow{*} \beta$ ). Cum  $v_1$  și  $v_2$  sunt distincte și  $v_2$  este descendent al lui  $v_1$ , rezultă că este posibilă derivarea

$$A \xRightarrow{*} vAx \xRightarrow{*} \beta \quad \text{cu} \quad v, x \in V_T^*. \quad (9)$$

Deci

- $\beta = vwx$ ;
- Cel puțin unul din cuvintele  $v$  sau  $x$  este nevid, deoarece – gramatica  $G$  fiind în FNC – prima regulă aplicată în derivarea (9) este de forma  $A \rightarrow BC$ ; așa că neterminalul  $A$  din forma sentențială  $vAx$  provine sau din  $B$  sau din  $C$ , celălalt neterminal generând un cuvânt nevid.
- Subarboarele de rădăcină  $v_1$  are drumul de lungime maxim  $k + 1$  deci, conform cu (8),  $|\beta| \leq 2^k$ , adică  $|vwx| \leq n$ .

Completând derivarea pentru cuvântul  $\alpha$  dat de acest arbore, există  $u, y \in V_T^*$  cu

$$S \xRightarrow{*} uAy \xRightarrow{*} uvAxy \xRightarrow{*} uvwxy.$$

Mai rămâne de arătat proprietatea (3) din Teoremă. Ea rezultă folosind derivarea (9):

$$\text{Pentru } i := 0 : \quad S \xRightarrow{*} uAy \xRightarrow{*} uwy;$$

Pentru  $i \geq 1$  :  $S \xRightarrow{*} uAy \xRightarrow{*} uvAxy \xRightarrow{*} \dots \xRightarrow{*} uv^iAx^iy \xRightarrow{*} uv^iwx^iy$ .

În derivare s-a folosit de  $i$  ori secvența  $A \xRightarrow{*} vAx$  ( $i \geq 0$ ) și apoi – la sfârșit – secvența  $A \xRightarrow{*} w$ . q.e.d.

### Observația 6.8

1. Lema de pompare este un caz particular al Teoremei 6.5 în care  $x = y = \epsilon$ . La această concluzie se ajunge scriind o gramatică regulată ca o gramatică independentă de context în FNC și refăcând demonstrația.

2. Aserțiunea (8) se poate rescrie: " $\forall \alpha \in L(G)$ , dacă orice arbore de derivare pentru  $\alpha$  are toate drumurile de lungime cel mult  $i$ , atunci  $|\alpha| \leq 2^{i-1}$ ."

Din această variantă se deduce forma utilizată în demonstrația Teoremei: " $\forall \alpha \in L(G)$ , dacă  $|\alpha| \geq 2^{i-1}$ , atunci în orice arbore de derivare pentru  $\alpha$  există cel puțin un drum de lungime minim  $i$ ."

3. Teorema 6.5 nu este valabilă pentru **orice** drum de lungime minim  $k + 1$ ; important este faptul că **există** cel puțin un drum de lungime minim  $k + 1$  pentru care descompunerea este posibilă. De obicei se alege un drum de lungime maximă în arbore.

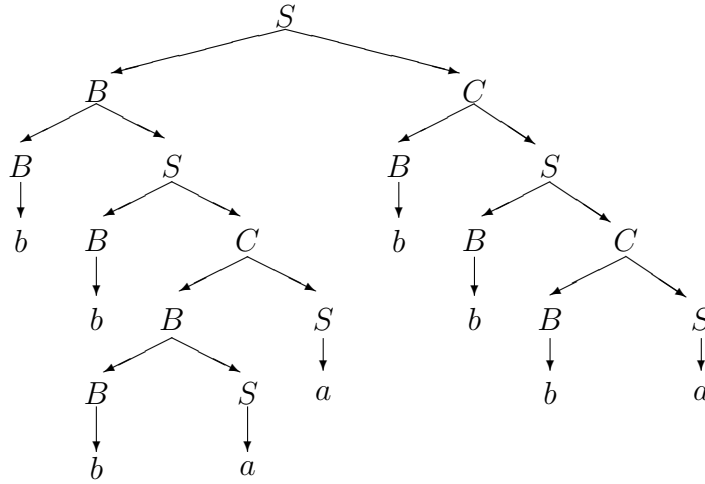
Să vedem aceasta pe un exemplu.

**Exemplul 6.14** Fie gramatica de producții

$$S \longrightarrow BC|a, \quad B \longrightarrow BS|b, \quad C \longrightarrow BS.$$

Aici  $k = 3$  și  $n = 2^3 = 8$ . Să considerăm  $\alpha = bbbaabbba$ .

Un arbore de derivare pentru  $\alpha$  este



Pe baza acestui arbore să încercăm să găsim toate descompunerile posibile ale cuvântului  $\alpha$  (descompuneri date de Teorema 6.5).

1. Pentru drumul  $S - B - \underline{S} - C - B - \underline{S} - a$  (cel mai lung) descompunerea este  $u = b$ ,  $v = bb$ ,  $w = a$ ,  $x = a$  și  $y = bbba$ .
2. Pentru  $S - \underline{B} - S - \underline{B} - b$  avem  $u = \epsilon$ ,  $v = b$ ,  $w = b$ ,  $x = baa$ ,  $y = bbba$ .
3. Pentru  $S - B - S - C - \underline{B} - \underline{B} - b$  avem  $u = bb$ ,  $v = \epsilon$ ,  $w = b$ ,  $x = a$ ,  $y = abbba$ .
4. Pentru  $S - B - \underline{S} - C - \underline{S} - a$  avem  $u = b$ ,  $v = bba$ ,  $w = a$ ,  $x = \epsilon$ ,  $y = bbba$ .
5. Pentru  $S - \underline{C} - S - \underline{C} - B - b$  avem  $u = bbbaa$ ,  $v = bb$ ,  $w = b$ ,  $x = a$ ,  $y = \epsilon$ .
6. Pentru  $S - A - \underline{S} - C - \underline{S} - a$  avem  $u = bbbaab$ ,  $v = bb$ ,  $w = a$ ,  $x = \epsilon$ ,  $y = \epsilon$ .
7. Pentru  $\underline{S} - C - \underline{S} - B - b$  avem  $u = \epsilon$ ,  $v = bbbaab$ ,  $w = bba$ ,  $x = \epsilon$ ,  $y = \epsilon$ .

În acest caz însă  $|vwx| = 9 \geq n$ , deci nu este o descompunere conformă cu Teorema 6.5. Cauza este aceea că în construcția lui  $v$  există drumuri cu lungimea mai mare decât cel ales în construcția variantei 7.

Înafara acestora mai sunt și alte descompuneri posibile, ca de exemplu cea corespunzătoare drumului  $S - \underline{B} - \underline{B} - b$  în care  $u = v = \epsilon$ ,  $w = b$ ,  $x = bbaa$ ,  $y = bbba$ . Ea verifică toate ipotezele Teoremei, dar nu a fost construită pe baza demonstrației, deoarece drumul  $S - B - B - b$  este de lungime prea mică.

Dacă pe drumul  $S - B - S - B - b$  am alege drept noduri pe cele notate cu  $S$ , s-ar obține descompunerea  $u = \epsilon$ ,  $v = b$ ,  $w = bbaa$ ,  $x = bbba$ ,  $y = \epsilon$  care nu verifică condiția  $|vwx| \leq 8$ . De aceea se alege pe fiecare drum **ultimele** două noduri notate identic.

Folosind concluzia (3) din Teoremă rezultă că în  $L(G)$  sunt incluse și limbajele:

$$L_1 = \{b^{2i+1}a^{i+1}b^3a \mid i \geq 0\}, \quad L_2 = \{b^{i+1}(baa)^ib^3a \mid i \geq 0\}, \quad L_3 = \{b^3a^{i+1}b^3a \mid i \geq 0\}, \\ L_4 = \{b(bba)^iab^3a \mid i \geq 0\}, \quad L_5 = L_6 = \{b^3a^2b^{2i+1}a \mid i \geq 0\}.$$

O variantă a Teoremei  $uvwxy$  este Lema Bar-Hillel. Aceasta se bazează pe faptul că Teorema 6.5 este valabilă și pentru cuvinte de lungime mai mică decât  $2^k$ , anume pentru orice  $\alpha \in L(G)$  cu  $|\alpha| \geq 2^{k-1}$ .

**Teorema 6.6** (Lema Bar-Hillel): Fie  $L$  un limbaj independent de context. Atunci există două constante  $p$  și  $q$  astfel încât  $\forall \alpha \in L$ ,  $|\alpha| > p$ , are loc descompunerea  $\alpha = uvwxy$  cu proprietățile

1.  $|vx| \geq 1$ ;
2.  $|vwx| \leq q$ ;
3.  $uv^iwx^iy \in L$ ,  $\forall i \geq 0$ .

Demonstrația este identică cu cea de la Teorema 6.5, unde se alege  $p = 2^k$ ,  $q = 2^{k-1}$ . q.e.d.

**Exemplul 6.15** Fie  $L = \{a^kb^kc^k \mid k \geq 1\}$ . Să presupunem că  $L$  este un limbaj independent de context și fie  $n$  constanta dată de Teorema 6.5. Să luăm cuvântul  $a^n b^n c^n$ . Cum

$|\alpha| = 3n > n$ , rezultă că  $\alpha = uvwxy$  cu  $|vx| \geq 1$ ,  $|vwx| \leq n$  și  $uv^iwx^iy \in L \quad \forall i \geq 0$ . Din  $|vwx| \leq n$  rezultă că  $vx$  nu poate avea simultan 'a'-uri și 'c'-uri pentru că între ele nu pot încapa  $n$  caractere 'b'.

Dacă  $vx \in a^+$ , atunci  $uwv$  (cazul  $i = 0$ ) are  $n$  'b'-uri și  $n$  'c'-uri dar mai puține 'a'-uri (pentru că  $vx \neq \epsilon$ ). Deci  $uwv \notin L$ , contradicție.

Similar dacă  $vx \in c^+$  sau  $vx \in b^+$ .

Dacă  $vx \in a^+b^+$  atunci  $uwv$  are mai multe 'c'-uri decât 'a'-uri sau 'b'-uri, deci din nou contradicție. Similar dacă  $vx \in b^+c^+$ .

Din toată această analiză se poate trage o singură concluzie: Teorema 6.5 nu se poate aplica, deci  $L$  nu este un limbaj independent de context.

**Exemplul 6.16** Să ne punem aceeași problemă pentru limbajul  $L = \{a^p \mid p \text{ prim}\}$ . Dacă  $L$  este independent de context, atunci există  $n$  prim pentru care Teorema 6.5 este adevărată. Luând  $\alpha = a^n$ , avem egalitatea  $a^n = uvwxy$ . Să considerăm  $v = a^k$ ,  $x = a^s$ . Atunci concluzia (1) asigură  $k + s > 0$ , iar concluzia (3), că  $uv^iwx^iy \in L$  pentru orice  $i \geq 0$ ; deci  $\forall i \geq 0$ ,  $|uv^iwx^iy|$  este număr prim. Adică  $n + i(k + s)$  este prim pentru orice număr natural  $i$ . Luând  $i = n$  se obține  $n(k + s + 1)$  prim, imposibil pentru că  $n > 1$ ,  $k + s + 1 > 1$ . Deci  $L$  nu este limbaj independent de context.

Din aceste exemple rezultă că incluziunea  $\mathcal{L}_2 \subseteq \mathcal{L}_1$  este strictă (pentru limbajul din Exemplul 6.15 s-a construit în Capitolul 3 o gramatică dependentă de context).

Cu ajutorul Teoremei  $uvwxy$  se poate deci demonstra faptul că unele limbaje nu sunt independente de context.

Acest lucru nu este întotdeauna posibil.

De exemplu, limbajul  $L = \{a^ib^jc^kd^s \mid i = 0 \text{ sau } j = k = s\}$  nu este independent de context. Dar dacă luăm  $\alpha = b^jc^kd^s$  ( $\alpha \in L$  pentru  $i = 0$ ) și scriem  $\alpha = uvwxy$ , va fi totdeauna posibil să alegem secvențele  $u, v, w, x, y$  astfel ca  $uv^iwx^iy \in L$ ,  $\forall i \geq 0$  (de exemplu se poate lua  $vwx \in b^+$ ).

Pentru  $\alpha = a^ib^jc^jd^j$ , dacă alegem  $vx \in A^+$ , avem de asemenea  $uv^iwx^iy \in L \quad \forall i \geq 0$ .

Deci Teorema 6.5 poate fi verificată de anumite limbaje dependente de context. Această observație a dus la necesitatea de a transpune Teorema 6.5 într-o formă care să corespundă și unor astfel de cazuri. Cea mai cunoscută variantă este Lema lui Ogden.

**Teorema 6.7** (Lema lui Ogden): Fie  $L$  un limbaj independent de context. Atunci există o constantă  $n$  astfel încât  $\forall \alpha \in L$  în care "marcăm" cel puțin  $n$  poziții oarecare, putem scrie  $\alpha = uvwxy$ , unde

1.  $vx$  are cel puțin o poziție marcată;
2.  $vwx$  are cel mult  $n$  poziții marcate;
3.  $uv^iwx^iy \in L$ ,  $\forall i \geq 0$ .



Teorema 6.5 este un caz particular al Lemei lui Ogden, în care se marchează toate pozițiile lui  $\alpha$ .

*Demonstrație:* Fie  $G = (V_N, V_T, S, P)$  o gramatică în FNC cu  $L(G) = L \setminus \{\epsilon\}$  și  $k = |V_N|$ . Alegem  $n = 2^k + 1$  și luăm  $\alpha \in L(G)$  cu  $|\alpha| \geq n$ , în care marcăm  $n$  poziții.

În arborele de derivare cu frontiera  $\alpha$  construim recursiv un drum  $P$ , în felul următor:

Introducem rădăcina în drumul  $P$ .

Fie  $v$  ultimul nod introdus; dacă  $v$  este un nod terminal, drumul se încheie. Dacă nu,  $v$  are doi descendenți (gramatica  $G$  este în FNC). Se adaugă la drum acel descendent direct pe a cărui frontieră sunt cele mai multe poziții marcate; dacă numărul pozițiilor marcate pe frontierele celor doi descendenți este egal, se alege arbitrar unul din ei.

Prin această construcție, fiecare nod al drumului are pe frontieră cel puțin jumătate din pozițiile marcate, față de ascendentul său direct. Pentru că numărul pozițiilor marcate este cel puțin  $n = 2^k + 1$ , rezultă că drumul astfel construit are cel puțin  $k + 1$  noduri distincte. Există deci două noduri  $v_1$  și  $v_2$  notate identic.

Mai departe, demonstrația decurge la fel cu demonstrația Teoremei 6.5. q.e.d.

În Exemplul anterior, luăm  $\alpha = a^i b^j c^j d^j$ ; și marcăm  $n$   $'b'$ -uri și  $'c'$ -uri (condiția este să fie marcate cel puțin un  $'b'$  și un  $'c'$ ). Aplicând Lema lui Ogden și raționând similar ca în Exemplul 6.15, se ajunge la contradicție pentru fiecare caz studiat. Deci limbajul nu este independent de context.

### 6.5.2 Aplicații ale Teoremei $uvwxy$

Din Teorema  $uvwxy$  și din faptul că există limbaje în  $\mathcal{L}_1 \setminus \mathcal{L}_2$  (afirmație care a putut fi probată cu ajutorul acestei teoreme), pot fi demonstrate o serie de proprietăți interesante legate de limbajele independente de context.

**Propoziția 6.4**  $\mathcal{L}_2$  nu este închisă la intersecție și complementară.

*Demonstrație:* Fie limbajele  $L_1 = \{a^i b^j c^j \mid i, j \geq 1\}$  și  $L_2 = \{a^i b^i c^j \mid i, j \geq 1\}$ .

Ele pot fi generate de gramaticile  $G_k = (V_N^k, V_T, S_k, P_k)$  unde  $V_N^k = \{S_k, A_k, B_k\}$  ( $k = 1, 2$ ),  $V_T = \{a, b, c\}$  și

$$\begin{aligned} P_1 &= \{S_1 \longrightarrow A_1 B_1, \quad A_1 \longrightarrow a A_1 | a, \quad B_1 \longrightarrow b B_1 c | bc\}, \\ P_2 &= \{S_2 \longrightarrow A_2 B_2, \quad A_2 \longrightarrow a A_2 b | ab, \quad B_2 \longrightarrow c B_2 | c\}. \end{aligned}$$

Deci  $L_1$  și  $L_2$  sunt independente de context; intersecția lor însă,  $L_1 \cap L_2 = \{a^i b^i c^i \mid i \geq 1\}$ , nu este un limbaj independent de context (Exemplul 6.15).

Pentru a doua afirmație folosim faptul că  $\mathcal{L}_2$  este închisă la reuniune (Capitolul 4). Dacă ar fi închisă și față de complementară, atunci – conform egalității de mulțimi

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

ar rezulta că  $\mathcal{L}_2$  este închisă la intersecție, contradicție. q.e.d.

**Teorema 6.8**  $\mathcal{L}_2$  este închisă la intersecția cu limbaje regulate.

*Demonstrație:* Fie  $L \in \mathcal{L}_2$  și  $R \in \mathcal{L}_3$  două limbaje pe care le presupunem că sunt definite peste același alfabet  $V_T$  (lucru ușor de îndeplinit printr-o operație de reuniune).

În plus, se poate lucra și în ipoteza  $\epsilon \notin L \cup R$  (dacă  $\epsilon \in L \cup R$ , se va construi întâi o gramatică pentru limbajul  $(L - \{\epsilon\}) \cap (R - \{\epsilon\})$  după modelul de mai jos,, apoi, dacă  $\epsilon \in L \cap R$ , se adaugă un simbol de start nou  $S'$  și producțiile  $S' \rightarrow S|\epsilon$ ).

Fie automatul finit determinist  $M = (Q, V_T, \delta, q_0, F)$  care acceptă limbajul  $R$ , și gramatica independentă de context  $G = (V_N, V_T, S, P)$  în FNC, care generează  $L$ . Dacă  $F = \{s_1, s_2, \dots, s_h\}$ , vom nota  $R_i = \tau(M_i)$  unde  $M_i = (Q, V_T, \delta, q_0, \{s_i\})$ . Evident

$$L \cap R = (L \cap R_1) \cup (L \cap R_2) \cup \dots \cup (L \cap R_h),$$

deci este suficient să arătăm că  $L \cap R_i$  este un limbaj independent de context  $\forall i = 1, 2, \dots, h$  (cum  $\mathcal{L}_2$  este închisă la reuniune, va rezulta imediat că  $L \cap R$  este tot independent de context).

Construim gramatica  $G_1 = (V_N^1, V_T, S_1, P_1)$  unde  $V_N^1 = (V_N \cup V_T) \times Q \times Q$ ,  $S_1 = (S, q_0, q_0)$  și  $P_1$  format din toate producțiile de forma

1.  $(A, q, q_1) \rightarrow (B, q, q_2)(C, q_2, q_1)$  unde  $q, q_1, q_2 \in Q$ ,  $A \rightarrow BC \in P$ ;
2.  $(A, q, q_1) \rightarrow (a, q, q_1)$  unde  $q, q_1 \in Q$ ,  $A \rightarrow a \in P$ ;
3.  $(a, q, q_1) \rightarrow a$ , unde  $a \in V_T$ ,  $\delta(q, a) = q_1$ .

$G_1$  este independentă de context. Mai trebuie arătat doar că  $L(G_1) = L \cap R_i$ .

• **Aserțiunea 1:** Dacă se aplică numai producții de tipul (1) atunci

$$\left[ (A, q_i, q_j) \xRightarrow{n} (X_1, q_i, q_1)(X_2, q_1, q_2) \dots (X_{n+1}, q_n, q_j) \right] \iff \left[ A \xRightarrow{n} X_1 X_2 \dots X_{n+1} \right] \\ \forall A, X_1, X_2, \dots, X_{n+1} \in V_N, \quad q \in Q.$$

Demonstrația acestei aserțiuni se face prin inducție după  $n$ :

$n = 1$ : evident, prin definiție.

Să presupunem Aserțiunea 1 adevărată pentru  $n$  pași, și să o demonstrăm pentru  $n + 1$ .

" $\implies$ ": Fie derivarea (în care s-a detaliat primul pas):

$$(A, q_i, q_j) \implies (B, q_i, q')(C, q', q_j) \xRightarrow{n} (X_1, q_i, q_1) \dots (X_{n+1}, q_n, q_j).$$

Conform Lemei 4.1., există  $k$  ( $1 \leq k \leq n$ ) cu  $q' = q_k$  și

$$(B, q_i, q') \xRightarrow{n_1} (X_1, q_i, q_1) \dots (X_k, q_{k-1}, q_k),$$

$$(C, q_k, q_j) \xRightarrow{n_2} (X_{k+1}, q_k, q_{k+1}) \dots (X_{n+1}, q_n, q_j).$$

Evident,  $n = n_1 + n_2$ ,  $n_1 > 0$ ,  $n_2 > 0$ . Conform ipotezei de inducție, avem  $B \xRightarrow{n_1} X_1 X_2 \dots X_k$ ,  $C \xRightarrow{n_2} X_{k+1} \dots X_{n+1}$ . Din primul pas al derivării în  $G_1$ ,  $A \rightarrow BC \in P$ . Deci se poate construi derivarea în  $G$ :

$$A \Rightarrow BC \xRightarrow{n_1} X_1 X_2 \dots X_k C \xRightarrow{n_2} X_1 X_2 \dots X_{n+1}, \text{ deci } A \xRightarrow{n+1} X_1 X_2 \dots X_{n+1}.$$

" $\Leftarrow$ ": Similar.

• **Aserțiunea 2:**

$$\left[ (a_1, q_1, q_2)(a_2, q_2, q_3) \dots (a_n, q_n, q_{n+1}) \xRightarrow{n} a_1 a_2 \dots a_n \right] \iff [\delta(q_1, a_1 a_2 \dots a_n) = q_{n+1}]$$

Afirmația este evidentă datorită faptului că în derivare se aplică numai producții de tipul 3 și – conform definiției – are loc echivalența

$$(a_i, q_i, q_{i+1}) \rightarrow a_i \in P_1 \iff \delta(q_i, a_i) = q_{i+1}, \quad \forall i = 1, 2, \dots, n.$$

Cu aceste aserțiuni, putem demonstra egalitatea  $L(G_1) = L \cap R_i$ .

" $\subseteq$ ":  $w = a_1 a_2 \dots a_n \in L(G_1)$ ; deci  $S_1 \xRightarrow{n} a_1 a_2 \dots a_n$ , derivare care – ținând cont că  $G_1$  este în FNC – poate fi detaliată astfel:

$$(S, q_0, s_i) \xRightarrow{*} (A_1, q_0, q_1) \dots (A_n, q_{n-1}, s_i) \xRightarrow{*} (a_1, q_0, q_1) \dots (a_n, q_{n-1}, s_i) \xRightarrow{*} a_1 \dots a_n$$

unde s-au aplicat la început producțiile de forma (1), apoi cele de tipul (2), și – în final – de tipul (3).

Conform Aserțiunii 1, rezultă că  $S \xRightarrow{*} A_1 A_2 \dots A_n$ , iar cu Aserțiunea 2,  $\delta(q_0, a_1 a_2 \dots a_n) = s_i$ . Cum  $s_i \in F$ , va rezulta  $w \in R_i$ .

Din a doua parte a derivării rezultă că în  $P$  există producțiile  $A_i \rightarrow a_i$ , ( $1 \leq i \leq n$ ). Deci  $S \xRightarrow{*} A_1 A_2 \dots A_n \xRightarrow{*} a_1 a_2 \dots a_n$  adică  $w \in L$ . În concluzie,  $w \in L \cap R_i$ .

" $\supseteq$ ": Fie  $w = a_1 a_2 \dots a_n \in L \cap R_i$ .

Din  $a_1 a_2 \dots a_n \in R_i$ , fie  $q_i = \delta(q_0, a_1 a_2 \dots a_j)$ , ( $1 \leq j \leq n-1$ ). În plus,  $s_i = \delta(q_0, a_1 a_2 \dots a_n)$ . Cum  $G$  este o gramatică independentă de context în FNC, aranjăm derivarea  $S \xRightarrow{*} a_1 a_2 \dots a_n$  sub forma

$$S \xRightarrow{*} A_1 A_2 \dots A_n \xRightarrow{*} a_1 a_2 \dots a_n$$

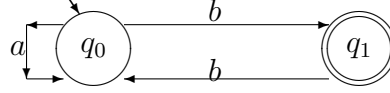
Din această derivare și Aserțiunile 1 și 2 se poate deduce derivarea în gramatica  $G_1$ :

$$\begin{aligned} S_1 = (S, q_0, s_i) &\xRightarrow{*} (A_1, q_0, q_1)(A_2, q_1, q_2) \dots (A_n, q_{n-1}, s_i) \xRightarrow{*} \\ &\xRightarrow{*} (a_1, q_0, q_1)(a_2, q_1, q_2) \dots (a_n, q_{n-1}, s_i) \xRightarrow{*} a_1 a_2 \dots a_n = w. \end{aligned} \quad \text{q.e.d.}$$

**Propoziția 6.5** Dacă  $L \in \mathcal{L}_2$  și  $R \in \mathcal{L}_3$  atunci  $L \setminus R \in \mathcal{L}_2$ .

*Demonstrație:* Folosim egalitatea (de mulțimi)  $L \setminus R = L \cap \overline{R}$ . Cum  $R$  este limbaj regulat, în Capitolul 4 am demonstrat că și  $\overline{R}$  este regulat. Cu Teorema 6.8 se obține deci că  $L \setminus R$  este independent de context. q.e.d.

**Exemplul 6.17** Să considerăm limbajul  $L = \{a^k b^k \mid k \geq 1\}$  generat de gramatica de producții  $P = \{S \rightarrow AC|AB, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow SB\}$ , și automatul cu diagrama de tranziție



având  $\tau(M) = R$ .

Să construim o gramatică pentru limbajul  $L \cap R$ .

Pentru ușurința scrierii, vom nota

$$\begin{aligned} S_{ij} &\equiv (S, q_i, q_j), & A_{ij} &\equiv (A, q_i, q_j), & B_{ij} &\equiv (B, q_i, q_j) \\ C_{ij} &\equiv (C, q_i, q_j), & X_{ij} &\equiv (a, q_i, q_j), & Y_{ij} &\equiv (b, q_i, q_j) \end{aligned}$$

Folosind construcția din demonstrația Teoremei 6.8:

$$S \rightarrow AC \text{ induce producțiile } \begin{cases} S_{00} \rightarrow A_{00}C_{00}|A_{01}C_{01} \\ S_{01} \rightarrow A_{00}C_{01}|A_{01}C_{11} \\ S_{10} \rightarrow A_{10}C_{00}|A_{11}C_{10} \\ S_{11} \rightarrow A_{10}C_{01}|A_{11}C_{11} \end{cases}$$

$$S \rightarrow AB \text{ induce producțiile } \begin{cases} S_{00} \rightarrow A_{00}B_{00}|A_{01}B_{10} \\ S_{01} \rightarrow A_{00}B_{01}|A_{01}B_{11} \\ S_{10} \rightarrow A_{10}B_{00}|A_{11}B_{10} \\ S_{11} \rightarrow A_{10}B_{01}|A_{11}B_{11} \end{cases}$$

$$C \rightarrow SB \text{ induce producțiile } \begin{cases} C_{00} \rightarrow S_{00}B_{00}|S_{01}B_{01} \\ C_{01} \rightarrow S_{00}B_{01}|S_{01}B_{11} \\ C_{10} \rightarrow S_{10}B_{00}|S_{11}B_{10} \\ C_{11} \rightarrow S_{10}B_{01}|S_{11}B_{11} \end{cases}$$

$$\text{Lui } A \rightarrow a \text{ îi corespund } \begin{cases} A_{00} \rightarrow X_{00}, & A_{01} \rightarrow X_{01} \\ A_{10} \rightarrow X_{10}, & A_{11} \rightarrow X_{11} \end{cases}$$

$$\text{iar pentru } B \rightarrow b \text{ avem } \begin{cases} B_{00} \rightarrow Y_{00}, & B_{01} \rightarrow Y_{01} \\ B_{10} \rightarrow Y_{10}, & B_{11} \rightarrow Y_{11} \end{cases}$$

$$\text{Ultimele reguli din } P_1 \text{ sunt } \begin{cases} X_{00} \rightarrow a & (\text{dată de } \delta(q_0, a) = q_0), \\ Y_{01} \rightarrow b & (\text{dată de } \delta(q_0, b) = q_1) \\ Y_{10} \rightarrow b & (\text{pentru } \delta(q_1, b) = q_0) \end{cases}$$

Gramatica  $G_1$  astfel formată are 24 neterminale și 35 producții; este evident, foarte dificil de folosit sub această formă. Aplicând algoritmul de eliminare a simbolurilor neutilizabile se obține o gramatică mai simplă, cu 10 neterminale,  $S_{01}$  simbol de start, producții  $S_{01} \rightarrow A_{00}B_{01}|A_{00}C_{01}$ ,  $S_{00} \rightarrow A_{00}C_{00}$ ,  $A_{00} \rightarrow X_{00}$ ,  $B_{01} \rightarrow Y_{01}$ ,  $B_{10} \rightarrow Y_{10}$ ,  $C_{00} \rightarrow S_{01}B_{10}$ ,  $C_{01} \rightarrow S_{00}B_{01}$ ,  $X_{00} \rightarrow a$ ,  $Y_{01} \rightarrow b$ ,  $Y_{10} \rightarrow b$ .

Eliminând redenumirile și rebotezând din nou neterminalele, se ajunge la o gramatică în FNC, cu producțiile

$$\{S \rightarrow AB|AC, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow S'B \quad S' \rightarrow AD, \quad D \rightarrow SB\}$$

sau – după eliminarea unor producții cu Lema 6.7, rămâne un singur neterminal  $S$  și regulile  $\{S \longrightarrow ab|aaSbb\}$ .

**Teorema 6.9**  $\mathcal{L}_2$  este închisă la aplicațiile gsm și translatările finite.

*Demonstrație:* Fie  $M = (Q, V_i, V_e, \delta, q_0, F)$  un gsm și  $L \subseteq V_i^*$  un limbaj independent de context. Există atunci o gramatică  $G = (V_N, V_T, S, P)$  în FNC cu  $L(G) = L$ .

Definim  $V_N' = (Q \times V_N \times Q) \cup \{S'\}$ ,  $V_T' = V_T$  și  $P'$  construit astfel:

- i.  $\forall q \in F, S' \longrightarrow [q_0, S, q] \in P'$ ;
- ii.  $\forall A \longrightarrow BC \in P, \forall p, q, r \in Q, [p, A, q] \longrightarrow [p, B, r][r, C, q] \in P'$ ;
- iii.  $\forall A \longrightarrow a \in P, \forall (p, w) \in \delta(q, a), [q, A, p] \longrightarrow w \in P'$ .

Gramatica  $G' = (V_N', V_T', S', P')$  este independentă de context; egalitatea  $L(G') = M(L)$  se demonstrează similar ca la Teorema 6.8.

Dacă  $M$  este un translator finit, atunci se adaugă regula:

- iv.  $\forall (p, w) \in \delta(q, \epsilon), \forall A \in V_N, \forall r \in Q, [q, A, r] \longrightarrow w[p, A, r] \in P'$ . q.e.d.

În afară de aceste proprietăți de închidere ale limbajelor independente de context – care completează rezultatele din Capitolele 4 și 5, tot din Teorema 6.5 pot fi deduse unele proprietăți de decidabilitate, cum ar fi:

- Verificarea dacă un limbaj nu este independent de context, bazată pe construcții exemplificate de Exemplele 6.15 și 6.16.
- Verificarea dacă un limbaj independent de context este finit sau nu. Se poate construi un algoritm similar celui dat în Capitolul 2 pentru limbaje regulate, bazat de o consecință a Teoremei Bar-Hillel (care dă limite mai stricte decât Teorema  $uvwxy$ )

**Propoziția 6.6** Un limbaj independent de context  $L$  este infinit dacă și numai dacă

$$\exists \alpha \in L \quad \text{cu} \quad p < |\alpha| \leq p + q$$

*Demonstrația* este analogă cu cea de la Propoziția 2.1 și o lăsăm ca exercițiu. q.e.d.

## 6.6 Exerciții

**Exercițiul 6.1** Câți arbori diferiți se pot construi cu nodurile  $a, b, c$  ?

**Exercițiul 6.2** În gramatica de producții

$$A \longrightarrow a|aBB, \quad B \longrightarrow b|bCA, \quad C \longrightarrow c|cAB$$

având simbol de start  $A$ , să se construiască un arbore de derivare pentru cuvântul  $(abc)^3$ .

**Exercițiul 6.3** În gramatica de producții  $S \longrightarrow SS|a$ , câți arbori de derivare distincți se pot construi pentru cuvântul  $a^{30}$  ?

**Exercițiul 6.4** Pentru gramatica din Exemplul 6.4, care este numărul total de arbori de derivare care se pot construi pentru cuvântul  $abba$  ?

**Exercițiul 6.5** Să se arate că orice gramatică care conține printre producțiile sale regula  $S \longrightarrow SS$ , este o gramatică ambiguă.

**Exercițiul 6.6** Fie gramatica de producții  $S \longrightarrow aSa|aS|c$ . Câte derivări distincte există pentru cuvântul  $a^m c a^n$  ?

**Exercițiul 6.7** Să se arate că gramatica de producții  $S \longrightarrow S + S|a$  este ambiguă. Să se construiască o gramatică neambiguă echivalentă.

**Exercițiul 6.8** Să se construiască o gramatică pentru limbajul

$$L = \{a^i b^j c^k \mid i = j \text{ sau } j = k, \quad i, j, k \geq 1\}$$

și să se arate că această gramatică este ambiguă.

**Exercițiul 6.9** Gramatica din Exemplul 6.3 nu este ambiguă. Găsiți o justificare a acestei afirmații.

**Exercițiul 6.10** Fie gramatica de producții

$$P = \{S \longrightarrow aB|bA, \quad A \longrightarrow a|aS|bAA, \quad B \longrightarrow b|bS|aBB\}$$

Pentru cuvântul  $w = a^3 b^2 a b^3 a$  să se construiască toate derivările stângi (drepte).

**Exercițiul 6.11** Să se elimine simbolurile neutilizabile din gramaticile de producții

- $P_1 = \{S \longrightarrow AA|CA, \quad A \longrightarrow a, \quad B \longrightarrow BC|AB, \quad C \longrightarrow aB|b\};$
- $P_2 = \{S \longrightarrow AB|a, \quad A \longrightarrow bB|cC, \quad B \longrightarrow SA|A, \quad C \longrightarrow aA|bB|S\};$
- $P_3 = \{S \longrightarrow SS|AAa, \quad A \longrightarrow aAB|aS|b\}.$

**Exercițiul 6.12** Să se decidă dacă limbajul generat de gramaticile următoare este nevid:

- $P_1 = \{S \longrightarrow AS|A, \quad A \longrightarrow aB|bA\};$
- $P_2 = \{S \longrightarrow ABC, \quad A \longrightarrow BB|\epsilon, \quad B \longrightarrow CC|a, \quad C \longrightarrow AA|b\}.$

**Exercițiul 6.13** Să se elimine  $\epsilon$ -producțiile din gramaticile de producții

$$P_1 = \{S \longrightarrow ABC, \quad A \longrightarrow BB|\epsilon, \quad B \longrightarrow CC|a, \quad C \longrightarrow AA|b\}$$

$$P_2 = \{S \longrightarrow aSb|bSa|SS|\epsilon\}$$

**Exercițiul 6.14** Să se construiască o gramatică independentă de context proprie echivalentă cu gramatica de producții

$$P = \{S \longrightarrow A|B, \quad A \longrightarrow C|D, \quad B \longrightarrow D|E, \quad C \longrightarrow S|a|\epsilon, \quad D \longrightarrow S|b, \quad E \longrightarrow S|c|\epsilon\}.$$

**Exercițiul 6.15** Să se construiască gramatici în FNC pentru limbajele:

$$L_1 = \{w \mid w \in \{a, b\}^*, |w|_a = 2|w|_b\}, \quad L_2 = \{a^i b^{i+j} c^j \mid i, j \geq 1\}.$$

**Exercițiul 6.16** Fie  $G$  o gramatică în FNC și  $S \xRightarrow{n} \alpha$  o derivare în această gramatică. Cât este  $|\alpha|$  ?

**Exercițiul 6.17** Să se elimine recursivitatea la stânga în gramaticile de producții:

$$P_1 = \{E \longrightarrow E + T \mid T, \quad T \longrightarrow T * F \mid F, \quad F \longrightarrow (E) \mid a\};$$

$$P_2 = \{S \longrightarrow aSb \mid bSa \mid SS \mid ab \mid ba\}.$$

**Exercițiul 6.18** Să se aplice Lema 6.7 pentru eliminarea primei producții din gramatica de reguli  $P = \{S \longrightarrow SS \mid a\}$ .

**Exercițiul 6.19** Să se aducă la Forma Normală Greibach gramaticile de producții:

$$P_1 = \{S \longrightarrow AA \mid a, \quad A \longrightarrow SS \mid b\}, \quad P_2 = \{S \longrightarrow AaA \mid b, \quad A \longrightarrow SbS \mid a\}.$$

**Exercițiul 6.20** Fie  $G$  o gramatică în FNG și  $w \in L(G)$ ,  $|w| = n$ . Cât este lungimea derivării  $S \xRightarrow{*} w$  ?

**Exercițiul 6.21** Să se arate că orice gramatică independentă de context  $G$  cu  $\epsilon \notin L(G)$  este echivalentă cu o gramatică având numai reguli de forma

$$A \longrightarrow aBC, \quad A \longrightarrow aB, \quad A \longrightarrow a \quad (A, B, C \in V_N, \quad a \in V_T)$$

**Exercițiul 6.22** Să se aplice Lema lui Ogden pentru gramatica de producții  $P_1 = \{S \longrightarrow AA, \quad A \longrightarrow SS \mid b\}$  și cuvântul  $\alpha = \underline{abbb}a\underline{aaab}$  ( $s$ -au subliniat pozițiile marcate).

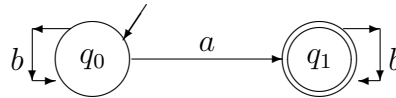
**Exercițiul 6.23** Să se arate că limbajele

$$L_1 = \{a^{n^2} \mid n \geq 1\}, \quad L_2 = \{a^{2^n} \mid n \geq 0\}, \quad L_3 = \{a^n b^m a^n b^m \mid n, m \geq 1\}$$

nu sunt independente de context.

**Exercițiul 6.24** Să se construiască un algoritm care să testeze dacă limbajul generat de o gramatică independentă de context este vid sau nu.

**Exercițiul 6.25** Fie limbajul  $L = \{w \mid w \in \{a, b\}^*, |w|_a = |w|_b\}$  și  $R$  acceptat de automatul cu diagrama de tranziție



Să se construiască o gramatică pentru limbajul  $L \cap R$ .

**Exercițiul 6.26** Să se cerceteze dacă limbajul generat de gramatica de producții

$$P = \{S \longrightarrow SB \mid A, \quad A \longrightarrow a \mid aB, \quad B \longrightarrow SB \mid BB\}$$

este finit sau nu.

# Capitolul 7

## $\mathcal{L}_1$ și automate liniar mărginite

### 7.1 Gramatici ”monotone” și gramatici ”dependente de context”

După cum am văzut în clasificarea Chomsky (Capitolul 3), un limbaj dependent de context este generat de o gramatică ale cărei producții sunt de forma  $\alpha \rightarrow \beta$  cu  $|\alpha| \leq |\beta|$ . Singura excepție o constituie eventual producția  $S \rightarrow \epsilon$ ; în acest caz  $S$  nu apare în membrul drept al nici unei producții.

Să considerăm o astfel de gramatică, dar fără producția  $S \rightarrow \epsilon$ ; datorită restricției  $|\alpha| \leq |\beta|$  verificată de toate producțiile  $\alpha \rightarrow \beta$ , ea se numește *gramatică monotona*. Cum regula  $S \rightarrow \epsilon$  nu se aplică decât în eventualitatea  $\epsilon \in L(G)$  – și atunci, numai în derivarea  $S \xRightarrow{*} \epsilon$ , în continuare vom lucra numai cu gramatici monotone.

**Definiția 7.1** Se numește ponderea unei gramatici  $G = (V_N, V_T, S, P)$  valoarea

$$\text{pond}(G) = \max\{|\beta| \mid \alpha \rightarrow \beta \in P\}.$$

**Teorema 7.1** Pentru orice gramatică monotona  $G = (V_N, V_T, S, P)$  există o gramatică monotona echivalentă, de pondere cel mult 2.

*Demonstrație:* Fie  $G = (V_N, V_T, S, P)$  o gramatică monotona. Putem considera (Lema 4.1) că singurele producții care conțin terminale sunt de forma  $A \rightarrow a$ . Aceste producții se păstrează și în noua gramatică.

Să presupunem că  $P$  conține reguli de forma  $\alpha \rightarrow \beta$  cu  $|\beta| > 2$  (altfel  $G$  este chiar gramatica căutată).

Fie  $A_1 A_2 \dots A_m \rightarrow B_1 B_2 \dots B_n \in P$  o astfel de regulă, cu  $m \leq n$ ,  $n > 2$ ,  $A_i, B_j \in V_N$  ( $1 \leq i \leq m$ ,  $1 \leq j \leq n$ ).

(1)



Introducem  $n - 1$  neterminale noi  $C_1, C_2, \dots, C_{n-1}$  și înlocuim producția considerată cu setul de reguli

$$\left\{ \begin{array}{l} A_1 \longrightarrow C_1, \\ C_1 A_2 \longrightarrow B_1 C_2, \\ \dots\dots\dots \\ C_{m-1} A_m \longrightarrow B_{m-1} C_m, \\ C_m \longrightarrow B_m C_{m+1}, \\ \dots\dots\dots \\ C_{n-1} \longrightarrow B_{n-1} B_n \end{array} \right. \quad (2)$$

Generăm câte un set de neterminale  $C_i$  și de producții de tipul (2) pentru fiecare regulă de tipul (1); fie  $V_N'$  noua mulțime de neterminale (formată din  $V_N$  la care se adaugă neterminalele nou introduse) și  $P'$  mulțimea de reguli din  $P$  în care fiecare regulă de tip (1) este înlocuită cu setul de producții (2).

Rămâne de arătat că noua gramatică  $G' = (V_N', V_T, S, P')$  este echivalentă cu gramatica  $G$ .

$L(G) \subseteq L(G') :$  Fie  $w \in L(G)$ ; deci în gramatica  $G$  există derivarea  $S \xRightarrow{*} w$ . Considerăm un pas  $\alpha_i \Rightarrow \alpha_{i+1}$  al acestei derivări. Dacă s-a aplicat o producție din  $P \cap P'$ , atunci acest pas rămâne nemodificat și în  $G'$ . Dacă s-a aplicat o producție de tipul (1), atunci  $\alpha_i = \beta_1 A_1 A_2 \dots A_m \beta_2$  și  $\alpha_{i+1} = \beta_1 B_1 B_2 \dots B_n \beta_2$ , iar  $\alpha_i \xRightarrow{*} \alpha_{i+1}$  (derivare în  $G'$ ), aplicând producțiile din (2) (care se află în  $P' \setminus P$ ).

$L(G') \subseteq L(G) :$  Fie  $w \in L(G')$ ; deci în gramatica  $G'$  există derivarea  $S \xRightarrow{*} w$ . În această derivare lăsăm nemodificați toți pașii care folosesc producții din  $P$ . Fie  $\alpha_i \Rightarrow \alpha_{i+1}$  prima generare directă care nu folosește o producție din  $P$ ; atunci acea producție utilizată nu poate fi decât de forma  $A_1 \longrightarrow C_1$ , unde  $A_1 \in V_N$ ,  $C_1 \in V_N' \setminus V_N$  ( $\alpha_i$  conține doar elemente din  $V_N \cup V_T$ , iar  $\alpha_{i+1}$  are și elemente din  $V_N' \setminus V_N$ ; singura regulă care poate realiza acest lucru este de această formă). Deci  $\alpha_i = \beta_1 A_1 \beta_2$ ,  $\alpha_{i+1} = \beta_1 C_1 \beta_2$ . Primele caractere din  $\beta_2$  sunt neterminale (altfel derivarea se blochează). În plus – pentru că se ajunge la  $w \in V_T^*$  – toate neterminalele trebuie să dispară, deci în derivare se folosesc toate producțiile din (2). Aceste producții pot fi grupate astfel încât să fie aplicate consecutiv. Rezultă că se poate scrie  $\beta_2 = A_2 \dots A_m \beta_3$  și deci

$$\alpha_i = \beta_1 A_1 \beta_2 \Rightarrow \beta_1 C_1 A_2 \dots A_m \beta_3 \xRightarrow{*} \beta_1 B_1 B_2 \dots B_n \beta_3$$

în gramatica  $G'$ . Această derivare poate fi comprimată în  $G$  într-un singur pas:

$$\beta_1 A_1 A_2 \dots \beta_3 \Rightarrow \beta_1 B_1 B_2 \dots B_n \beta_3.$$

Aplicăm acest raționament în derivarea  $S \xRightarrow{*} w$  din  $G'$  și obținem o derivare în  $G : S \xRightarrow{*} w$ . Deci  $w \in L(G)$ . q.e.d.

În noua gramatică  $G'$  pot fi eliminate regulile de forma  $A \longrightarrow B$  printr-un algoritm de eliminare a redenumirilor similar celui din Capitolul 6. Deci putem considera că orice

limbaj  $L \in \mathcal{L}_1$  cu  $\epsilon \notin L$  poate fi generat de o gramatică având numai producții de forma

$$A \longrightarrow a, \quad A \longrightarrow BC, \quad AB \longrightarrow CD.$$

Aceasta este *Forma Normală Kuroda* a gramaticilor dependente de context<sup>1</sup>.

**Definiția 7.2** O producție  $\alpha_1 A \alpha_2 \longrightarrow \alpha_1 \beta \alpha_2$  se numește "senzitivă de context  $(\alpha_1, \alpha_2)$ ". Detaliind, "A se rescrie  $\beta$  în contextul  $(\alpha_1, \alpha_2)$ ".

**Observația 7.1** Producțiile de forma  $A \longrightarrow \alpha$  sunt senzitive de context  $(\epsilon, \epsilon)$ ; de aceea, aceste reguli se numesc "independente de context".

**Teorema 7.2** Un limbaj  $L \in \mathcal{L}_1$  poate fi generat de o gramatică care are toate producțiile senzitive de context.

*Demonstrație:* Fie  $G = (V_N, V_T, S, P)$  o gramatică de tip 1 cu  $L(G) = L \setminus \{\epsilon\}$ . Fără a micșora generalitatea, putem presupune că  $G$  este în forma normală Kuroda. Deoarece toate producțiile de tipul  $A \longrightarrow a$ ,  $A \longrightarrow BC$  sunt senzitive de context  $(\epsilon, \epsilon)$ , ele vor rămâne nemodificate. Ne interesează numai regulile din  $P$  de forma  $AB \longrightarrow CD$ . Acestea pot fi clasificate în:

1.  $AB \longrightarrow AD$ ; sunt reguli senzitive de context  $(A, \epsilon)$ ;
2.  $AB \longrightarrow CB$ ; sunt reguli senzitive de context  $(\epsilon, B)$ .
3.  $AB \longrightarrow CD$  cu  $A \neq C$ ,  $B \neq D$ . Aceste reguli nu sunt senzitive de context.

Pentru a demonstra teorema, este suficient să arătăm o modalitate de a înlocui producțiile de tipul (3) cu producții senzitive de context.

Fie  $AB \longrightarrow CD \in P$  o producție de tipul (3). Introducem două neterminale noi  $E, F \notin V_N$  și înlocuim producția cu

$$(i) AB \longrightarrow EB, \quad (ii) EB \longrightarrow EF, \quad (iii) EF \longrightarrow CF, \quad (iv) CF \longrightarrow CD$$

Fie  $G' = (V_N', V_T, S, P')$  noua gramatică obținută din  $G$  prin înlocuirea tuturor producțiilor de tipul (3) cu seturi de reguli de această formă. Demonstrarea egalității  $L(G) = L(G')$  se face similar cu demonstrația Teoremei 7.1.

Dacă  $\epsilon \in L$ , atunci în gramatica  $G'$  se introduce un simbol nou de start  $S'$ , iar în  $P'$  se adaugă producțiile  $S' \longrightarrow S|\epsilon$ , ambele de context  $(\epsilon, \epsilon)$ . q.e.d.

Ca o consecință a Teoremei 7.2, clasa  $\mathcal{L}_1$  este numită *clasa limbajelor senzitive (dependente) de context*.

---

<sup>1</sup>Forma normală Chomsky este – evident – un caz particular al Formei Normale Kuroda.

**Exemplul 7.1** Să reluăm limbajul  $L = \{a^n b^n c^n \mid n \geq 1\}$  din Exemplul 3.11 (Capitolul 3). El a fost generat de gramatica de producții

1.  $S \longrightarrow aSBC$ ,
2.  $S \longrightarrow aBC$ ,
3.  $CB \longrightarrow BC$ ,
4.  $aB \longrightarrow ab$ ,
5.  $bB \longrightarrow bb$ ,
6.  $bC \longrightarrow bc$ ,
7.  $cC \longrightarrow cc$

Gramatica este monotonă. Dintre producțiile sale, numai a treia producție nu este senzitivă de context. Conform construcției din demonstrația Deoremei 7.2, o vom înlocui cu setul de reguli  $\{CB \longrightarrow CD, CD \longrightarrow ED, ED \longrightarrow BD, BD \longrightarrow BC\}$ , unde  $D, E$  sunt neterminale noi. Celelalte reguli, deși nu sunt în forma normală Kuroda, pot fi păstrate.

Deci  $L$  poate fi generat de gramatica  $G' = (V_N', V_T, S, P')$  cu  $V_N' = \{S, B, C, D, E\}$ ,  $V_T = \{a, b, c\}$  și producțiile

- |                          |                                   |                         |                                   |
|--------------------------|-----------------------------------|-------------------------|-----------------------------------|
| $S \longrightarrow aSBC$ | de context $(\epsilon, \epsilon)$ | $S \longrightarrow aBC$ | de context $(\epsilon, \epsilon)$ |
| $aB \longrightarrow ab$  | de context $(a, \epsilon)$        | $bB \longrightarrow bb$ | de context $(b, \epsilon)$        |
| $bC \longrightarrow bc$  | de context $(b, \epsilon)$        | $cC \longrightarrow cc$ | de context $(c, \epsilon)$        |
| $CB \longrightarrow CD$  | de context $(C, \epsilon)$        | $CD \longrightarrow ED$ | de context $(\epsilon, D)$        |
| $ED \longrightarrow BD$  | de context $(\epsilon, D)$        | $BD \longrightarrow BC$ | de context $(B, \epsilon)$        |

Dacă dorim să transformăm gramatica inițială într-o gramatică în forma normală Kuroda, vom proceda în felul următor:

- În prima fază introducem trei neterminale noi  $X, Y, Z$  și aducem producțiile la forma

1.  $S \longrightarrow XSBC$ ,
2.  $S \longrightarrow XBC$ ,
3.  $CB \longrightarrow BC$ ,
4.  $XB \longrightarrow XY$ ,
5.  $YB \longrightarrow YY$ ,
6.  $YC \longrightarrow YZ$ ,
7.  $ZC \longrightarrow ZZ$ ,
8.  $X \longrightarrow a$ ,
9.  $Y \longrightarrow b$ ,
10.  $Z \longrightarrow c$

unde ultimele trei producții (8, 9, 10) sunt noi.

- Mai trebuie modificate primele două reguli (care nu satisfac condițiile de formă normală). Pentru ele sunt suficiente două neterminale noi  $D, E$ ; vom înlocui
  - prima regulă  $S \longrightarrow XSBC$  cu  $S \longrightarrow XD$ ,  $D \longrightarrow SE$ ,  $E \longrightarrow BC$ ;
  - a doua regulă  $S \longrightarrow XBC$  cu  $S \longrightarrow XE$  (de remarcat că este suficientă o singură producție).

Pe baza faptului că orice gramatică monotonă generează un limbaj senzitiv de context putem demonstra următorul rezultat:

**Teorema 7.3** *Orice limbaj din  $\mathcal{L}_1$  este recursiv.*

*Demonstrație:* Un limbaj  $L \subseteq V^*$  este recursiv dacă există un algoritm care să decidă – pentru orice  $w \in V^*$  – dacă  $w \in L$  sau nu.

**Algoritmul 7.1.:**

**Intrare:** O gramatică  $G = (V_N, V_T, S, P)$  monotonă,  $w \in V_T^*$ ,  $|w| = n$ ;

**Ieșire:** DA - dacă  $w \in L(G)$ , NU - altfel.

**Algoritm:**

1.  $T_0 = \{S\}$ ,  $i := 1$ ;
2.  $T_i := T_{i-1} \cup \{u \mid |u| \leq n, \exists v \in T_{i-1}, v \Rightarrow u\}$ ;
3. Dacă  $T_i \neq T_{i-1}$  atunci  $i := i + 1$  și salt la P2;
4. Dacă  $w \in T_i \cap V_T^*$  atunci DA altfel NU.

Se poate demonstra ușor – similar demonstrațiilor Algoritmilor 6.2 și 6.3 din Capitolul 6, secțiunea 3:

$$[w \in T_i \cap V_T^*] \iff \exists n (n \leq i) [S \xRightarrow{n} w].$$

Algoritmul 7.1. funcționează și pentru cazul  $\epsilon \in L(G)$ ; în această situație  $\epsilon \in V_1$ . q.e.d.

**Observația 7.2** *Este esențial faptul că  $G$  este gramatică monotonă. De exemplu, dacă se ia gramatica independentă de context de producții*

$$S \longrightarrow aSbS|bSaS|\epsilon$$

*care generează limbajul  $L = \{w \mid w \in \{a,b\}^*, |w|_a = |w|_b\}$ , cuvântul  $ab \in L(G)$  nu va fi recunoscut: pentru  $|ab| = 2 = n$  Algoritmul 7.1. dă  $T_0 = \{S\}$ ,  $T_1 = \{S, \epsilon\}$ ,  $T_2 = T_1$  și  $ab \notin T_2 \cap V_T^* = \{\epsilon\}$ .*

*Pentru o aplicare corectă a Algoritmului 7.1. va trebui întâi să eliminăm  $\epsilon$ -producțiile din gramatica  $G$  (cu Algoritmul 6.5), obținând o gramatică  $G'$  cu regulile*

$$S' \longrightarrow \epsilon|S, \quad S \longrightarrow aSbS|bSaS|aSb|bSa|abS|baS|ab|ba.$$

*În acest caz,  $T_0 = \{S'\}$ ,  $T_1 = \{S', S, \epsilon\}$ ,  $T_2 = \{S', S, \epsilon, ab, ba\}$ ,  $T_3 = T_2$ . La pasul 4 al Algoritmului 7.1. rezultă mulțimea  $T_3 \cap V_T^* = \{\epsilon, ab, ba\}$ , care conține cuvântul  $ab$ .*

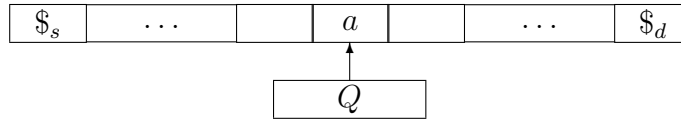
Teorema 7.3 arată că orice limbaj senzitiv de context este recursiv. Reciproca nu este însă adevărată: există limbaje recursive care nu sunt în  $\mathcal{L}_1$ . O demonstrație a acestei afirmații va fi dată în Capitolul următor.

## 7.2 Automate liniar mărginite

O metodă de recunoaștere a limbajelor senzitive de context este *automatul liniar mărginit* (*LBA* - Linear Bounded Automaton).

**Definiția 7.3** *Un automat liniar mărginit (LBA) este o structură  $M = (Q, \Sigma, \Gamma, \delta, q_0, \$_s, \$_d, F)$  unde:*

- $Q$  este o mulțime finită nevidă de stări;
- $\Gamma$  este o mulțime finită de simboluri numită alfabet de lucru;
- $\Sigma$  ( $\Sigma \subset \Gamma$ ) este alfabetul de intrare;
- $q_0 \in Q$  este starea inițială a automatului;
- ${}_s, {}_d \in \Sigma$  sunt două caractere speciale numite marcatori stânga/dreapta;
- $F \subseteq Q$  ( $F \neq \emptyset$ ) este mulțimea stărilor finale;
- $\delta : Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{s, d\}}$  este funcția de tranziție, funcție parțial definită, cu proprietatea ( $\forall q \in Q$ )  $[(\delta(q, {}_s) = \{(q, {}_s, d)\}, \delta(q, {}_d) = \{(q, {}_d, s)\})]$ .



Un automat liniar mărginit poate fi privit ca o memorie de tipul unei benzi finite, pe care sunt scrise elemente din  $\Gamma$ . Un cap de citire este poziționat în dreptul unui caracter scris pe bandă. La o mișcare (descrisă de funcția  $\delta$ ), acest cap rescrie un caracter peste caracterul curent, după care se deplasează cu o poziție spre dreapta sau stânga. Capul de citire nu poate depăși marcatorii de margine ai benzii ( ${}_s, {}_d$ ).

Pentru a detalia funcționarea unui *LBA* folosim descrieri instantanee (*DI*). O *DI* este o configurație de forma  $(q, \alpha, X\beta)$  unde  $q \in Q$ ,  $\alpha, \beta \in \Gamma^*$ ,  $X \in \Gamma$ .

$q$  este starea curentă a automatului,  $\alpha X \beta$  este conținutul benzii, iar  $X$  este caracterul aflat în dreptul capului de citire. Pe mulțimea configurațiilor definim o relație binară  $\vdash$  (relație de tranziție), în modul următor:

1.  $(q, \alpha a, X\beta) \vdash (p, \alpha, ab\beta)$     dacă     $(p, b, s) \in \delta(q, X)$ ,  $a \neq {}_s$ ;
2.  $(q, \alpha, X\beta) \vdash (p, \alpha b, \beta)$     dacă     $(p, b, d) \in \delta(q, X)$ ,  $X \neq {}_d$ .

De remarcat că pentru  $a = \$_s$  capul de citire nu se poate deplasa spre stânga, iar pentru  $X = \$_d$ , capul de citire nu se poate deplasa spre dreapta.

Fie  $\vdash^*$  închiderea reflexivă și tranzitivă a relației  $\vdash$ . Putem defini acum limbajul acceptat de un automat liniar mărginit  $M$  prin:

$$\tau(M) = \{w \mid w \in (\Sigma \setminus \{\$, \$_d\})^*, (q_0, \$_s, w\$_d) \vdash^* (q, \alpha, \beta), q \in F\}$$

**Teorema 7.4** *LBA este o metodă de recunoaștere.*

*Demonstrație:* Plecând de la un automat liniar mărginit  $M = (Q, \Sigma, \Gamma, \delta, q_0, \$_s, \$_d, F)$  cu  $\tau(M) = L$ , construim o metodă de recunoaștere  $M_r(L) = (SR, AX, X)$  bazată pe sistemul de rescriere  $SR = (W, P)$ . Componentele lui  $M_r$  sunt definite astfel:

1.  $W = \Gamma \cup Q$ ; presupunem – fără a micșora generalitatea – că  $Q \cap \Gamma = \emptyset$ ;
2.  $X = \Sigma$ ;
3.  $AX = \{\$, q\$_d \mid q \in F\}$ ;
4.  $P = \{(Xqa, pXb) \mid (p, b, s) \in \delta(q, a), X \in \Gamma \setminus \{\$, \$_s\}\} \cup \{(qa, pb) \mid (p, b, d) \in \delta(q, a)\} \cup \{(aq, q), (qa, q) \mid q \in F, a \in \Sigma \setminus \{\$, \$_d\}\} \cup \{(\$, \$_sq_0)\}$ .

Se arată ușor (prin argumente similare teoremelor analoge referitoare la automatele finite și automatele stivă) că

$$M_r(L) = \{w \mid w \in \Sigma^*, \exists q \in F, w \xRightarrow{*} \$_s \alpha q \beta \$_d, \alpha \beta \in \Sigma^*, |\alpha \beta| = |w|\} = \tau(M)$$

**Exemplul 7.2** *Să construim un automat liniar mărginit care să accepte limbajul*

$$L = \{a^{n^2} \mid n \geq 1\}$$

*Construcția se bazează pe formula  $n^2 = 1 + 3 + 5 + \dots + (2n - 1)$ .*

*Automatul  $M$  va avea:*

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_f\}, \quad \Sigma = \{a, \$, \$_d\}, \quad \Gamma = \{a, b, c, x, \$, \$_d\},$$

*iar funcția de tranziție  $\delta$  definită prin tabelul*

$\delta$	$a$	$b$	$c$	$x$	$\$$	$\$_d$
$q_0$	$(q_1, b, d)$	—	—	—	—	—
$q_1$	$(q_2, c, d)$	$(q_1, b, d)$	—	—	—	—
$q_2$	$(q_3, c, s)$	—	—	—	—	—
$q_3$	—	$(q_3, b, s)$	$(q_3, c, s)$	$(q_4, x, d)$	$(q_4, \$, d)$	—
$q_4$	$(q_6, a, s)$	$(q_5, x, d)$	$(q_4, c, d)$	—	—	$(q_f, \$_d, s)$
$q_5$	$(q_3, c, s)$	$(q_5, b, d)$	$(q_5, c, d)$	—	—	—
$q_6$	—	$(q_6, x, s)$	$(q_6, b, s)$	$(q_1, x, d)$	—	—
$q_0$	—	—	—	—	—	—

Definiția lui  $\delta$  fiind deterministă, s-a simplificat notația prin înlocuirea mulțimii cu elementul pe care îl conține. De asemenea este posibilă introducerea unei stări de eroare  $q_e$  care să completeze spațiile libere din tabel.

Rolurile celor opt stări sunt:

- $q_0$  este starea inițială; automatul merge spre dreapta și transformă primul 'a' în 'b'.
- În starea  $q_1$  automatul se deplasează spre dreapta și găsește primul 'a' pe care îl transformă în 'c'.
- $q_2$  are rolul de a transforma al doilea 'a' în 'c' și să treacă în starea  $q_3$ . În acest moment au fost marcate două caractere care fac trecerea de la  $2i - 1$  la  $2i + 1$  din suma lui  $n^2$ ; mai departe trebuie găsite și înlocuite următoarele  $2i - 1$  caractere 'a'.
- În starea  $q_3$  automatul parcurge banda de intrare spre stânga, până la primul caracter diferit de 'b' sau 'c'; atunci trece în starea  $q_4$ .
- Suntem în starea  $q_4$ . Automatul merge spre dreapta și înlocuiește primul 'b' cu 'x' (trecând în starea  $q_5$ ). Dacă întâlnește un caracter 'a' înseamnă că nu mai sunt 'b'-uri și se trece în starea  $q_6$ . Dacă se ajunge la delimitator, cuvântul este acceptat.
- În starea  $q_5$  automatul merge spre dreapta până la primul 'a' pe care îl înlocuiește cu 'c' (și revine la starea  $q_3$ ).
- Starea  $q_6$  este destinată renotării caracterelor pentru a reîncepe un nou ciclu. Automatul se deplasează spre stânga și înlocuiește 'b'-urile cu 'x' (pentru a le neutraliza) și 'c'-urile cu 'b' (pentru a "bifa" ulterior același număr de caractere 'a', în trecerea de la  $2i - 1$  la  $2i + 1$ ). La întâlnirea primului caracter 'x', automatul reia ciclul trecând în starea  $q_1$ .

Să luăm de exemplu cuvântul de intrare  $a^9$ . Automatul  $M$  va funcționa în felul următor:

$$\begin{aligned}
& (q_0, \$, a^9\$_d) \vdash (q_1, \$, b, a^8\$_d) \vdash (q_2, \$, bc, a^7\$_d) \vdash (q_3, \$, b, cca^6\$_d) \vdash (q_3, \$, bc^2a^6\$_d) \vdash \\
& \vdash (q_3, \epsilon, \$, c^2a^6\$_d) \vdash (q_4, \$, bc^2a^6\$_d) \vdash (q_5, \$, x, c^2a^6\$_d) \vdash (q_5, \$, xc, ca^6\$_d) \vdash \\
& \vdash (q_5, \$, xc^2, a^6\$_d) \vdash (q_3, \$, xc, c^2a^5\$_d) \vdash (q_3, \$, x, c^3a^5\$_d) \vdash (q_3, \$, xc^3a^5\$_d) \vdash \\
& \vdash (q_4, \$, x, c^3a^5\$_d) \vdash (q_4, \$, xc, c^2a^5\$_d) \vdash (q_4, \$, xc^2, ca^5\$_d) \vdash (q_4, \$, xc^3, a^5\$_d) \vdash \\
& \vdash (q_6, \$, xc^2, ca^5\$_d) \vdash (q_6, \$, xc, cba^5\$_d) \vdash (q_6, \$, x, cb^2a^5\$_d) \vdash (q_4, \$, xb^3a^5\$_d) \vdash \\
& \vdash (q_1, \$, x, b^3a^5\$_d) \vdash (q_1, \$, xb, b^2a^5\$_d) \vdash (q_1, \$, xb^2, ba^5\$_d) \vdash (q_1, \$, xb^3, a^5\$_d) \vdash \\
& \vdash (q_2, \$, xb^3c, a^4\$_d) \vdash (q_3, \$, xb^3, c^2a^3\$_d) \vdash (q_3, \$, xb^2, bc^2a^3\$_d) \vdash (q_3, \$, xb, b^2c^2a^3\$_d) \vdash \\
& \vdash (q_2, \$, x, b^3c^2a^3\$_d) \vdash (q_2, \$, xb^3c^2a^3\$_d) \vdash (q_4, \$, x, b^3c^2a^3\$_d) \vdash (q_5, \$, x^2, b^2c^2a^3\$_d) \vdash^* \\
& \vdash^* (q_5, \$, x^2b^2c^2, a^3\$_d) \vdash (q_3, \$, x^2b^2c, c^2a^2\$_d) \vdash^* (q_3, \$, x^2, b^2c^3a^2\$_d) \vdash (q_3, \$, x, xb^2c^3a^2\$_d)
\end{aligned}$$

$$\begin{aligned}
& \vdash (q_4, \$_s x^2, b^2 c^3 a^2 \$_d) \vdash (q_5, \$_s x^3, b c^3 a^2 \$_d) \vdash^* (q_5, \$_s x^3 b c^3, a^2 \$_d) \vdash (q_3, \$_s x^3 b c^2, c^2 a \$_d) \vdash^* \\
& \vdash^* (q_3, \$_s x^2, x b c^4 a \$_d) \vdash (q_3, \$_s x^3, b c^4 a \$_d) \vdash (q_5, \$_s x^4, c^4 a \$_d) \vdash^* (q_5, \$_s x^4 c^4, a \$_d) \vdash \\
& \vdash (q_3, \$_s x^4 c^3, c^2 \$_d) \vdash^* (q_3, \$_s x^3, x c^5 \$_d) \vdash (q_4, \$_s x^4, c^5 \$_d) \vdash^* (q_4, \$_s x^4 c^5, \$_d) \vdash \\
& \vdash (q_f, \$_s x^4 c^4, c \$_d)
\end{aligned}$$

deci cuvântul  $a^9$  este acceptat.

Pe baza formulei  $n^2 = 1 + 3 + \dots + (2n - 1)$  se poate arăta echivalența

$$a^k \in \tau(M) \quad \Longleftrightarrow \quad \exists n [k = n^2].$$

Următoarele două teoreme demonstrează faptul că automatele liniar mărginite sunt metodele duale gramaticilor senzitive de context, ambele sisteme de rescriere definind în mod complet clasa de limbaje  $\mathcal{L}_1$ .

**Teorema 7.5** Dacă  $L \in \mathcal{L}_1$  și  $\epsilon \notin L$ , atunci există un automat liniar mărginit  $M$  cu  $\tau(M) = L$ .

*Demonstrație:* Dacă  $L \in \mathcal{L}_1$  atunci există o gramatică în forma Normală Kuroda  $G = (V_N, V_T, S, P)$  cu  $L = L(G)$ . Vom presupune  $\epsilon \notin L$  (datorită modului de construcție a gramaticilor senzitive de context, modificările necesitate de posibila existență a cuvântului vid în limbaj sunt minore și nu afectează clasa  $\mathcal{L}_1$ ). Un cuvânt  $w = a_1 a_2 \dots a_n \in V_T^+$  este în  $L$  dacă și numai dacă  $S \xRightarrow{*} w$ .

Vom construi un  $LBA$  în care banda este segmentată în două componente paralele: în prima parte se află cuvântul  ${}_s w {}_d$ , iar pe a doua parte se simulează derivări în  $G$ , plecând de la simbolul de start  $S$ .

Detaliind: fie  $i : \alpha_i \longrightarrow \beta_i$ , ( $1 \leq i \leq |P|$ ) producțiile gramaticii  $G$ . Avem

1.  $Q = \{q_i, q'_i \mid 1 \leq i \leq |P|\} \cup \{q''_i \mid 1 \leq i \leq |P|, |\alpha_i| < |\beta_i|\} \cup \{q_X \mid X \in V_N \cup V_T\} \cup \{q_o, q_f, q_r, q_v\}$ , unde  $q_o$  este starea inițială,  $q_f$  este starea finală,  $q_r$  este o stare de revenire iar  $q_v$  este o stare de verificare;
2.  $\Gamma = \{[a, X] \mid a \in V_T, X \in V_N \cup V_T \cup \{Z\}\} \cup \{[{}_s, {}_s], [{}_c, {}_c]\}$ , unde  $Z \notin V_N \cup V_T$  este un simbol nou, iar  ${}_s, {}_d$  sunt cei doi delimitatori;
3.  $\Sigma = \{[a, a] \mid a \in V_T\}$ ;
4.  $F = \{q_f\}$ ;
5. Funcția de transfer  $\delta$  este definită:

$$(a) \quad \delta(q_o, [a, X]) = \{(q_i, [a, X], d) \mid \alpha_i = X \alpha'_i\} \cup \{(q_o, [a, X], d) \mid a \neq {}_d\};$$

Automatul este în starea inițială și capul de citire se deplasează spre dreapta. Dacă există o producție  $i : \alpha_i \longrightarrow \beta_i$  în care  $\alpha_i$  începe cu  $X$ , automatul poate trece în starea  $q_i$ .



- (b) Automatul este în starea  $q_i$  și  $|\alpha_i| = |\beta_i| = 1$ .  
 $\delta(q_i, [a, X]) = \{(q'_i, [a, X], s)\}, \quad \forall a \in V_T \cup \{\$d\}, X \in V_N \cup V_T \cup \{\$d\};$   
 $\delta(q'_i, [a, \alpha_i]) = \{(q_r, [a, \beta_i], s)\}, \quad \forall a \in V_T.$
- (c) Automatul este în starea  $q_i$  și  $|\alpha_i| = |\beta_i| = 2$ . Fie  $\alpha_i = AB$ ,  $\beta_i = CD$ .  
 $\delta(q_i, [a, B]) = \{(q'_i, [a, D], s)\}, \quad \forall a \in V_T;$   
 $\delta(q'_i, [a, A]) = \{(q_r, [a, C], s)\}, \quad \forall a \in V_T.$
- (d) Automatul este în starea  $q_i$  și  $|\alpha_i| < |\beta_i|$ . Fie  $\alpha_i = A$ ,  $\beta_i = BC$ .  
 i. Dacă elementul curent pe bandă este  $[a, Z]$ :  
 $\delta(q_i, [a, Z]) = \{(q'_i, [a, C], s)\}, \quad \forall a \in V_T;$   
 $\delta(q'_i, [a, A]) = \{(q_r, [a, B], s)\}, \quad \forall a \in V_T.$   
 ii. Dacă elementul curent pe bandă este  $[a, X]$  cu  $X \in V_N \cup V_T$ :  
 $\delta(q_i, [a, X]) = \{(q''_i, [a, X], s)\}, \quad a \in V_T, X \in V_N \cup V_T;$   
 $\delta(q''_i, [a, A]) = \{(q''_i, [a, B], d)\}, \quad \forall a \in V_T;$   
 $\delta(q''_i, [a, X]) = \{(q_X, [a, B], d)\}, \quad \forall a \in V_T, X \in V_N \cup V_T;$   
 $\delta(q_X, [a, Y]) = \{(q_Y, [a, X], d)\}, \quad \forall a \in V_T, X, Y \in V_N \cup V_T;$   
 $\delta(q_X, [a, Z]) = \{(q_r, [a, X], s)\}, \quad \forall a \in V_T, X \in V_N \cup V_T.$
- (e)  $\delta(q_r, [a, X]) = \{(q_r, [a, X], s)\}, \quad [a, X] \neq [\$s, \$s];$   
 Capul de citire se deplasează spre stânga până la primul element de pe bandă.
- (f)  $\delta(q_r, [\$s, \$s]) = \{(q_0, [\$s, \$s], d), (q_v, [\$s, \$s], d)\};$   
 Automatul trece în starea inițială și reia căutarea unei noi aplicări a unei producții, sau trece în starea  $q_v$  și se deplasează spre dreapta.
- (g)  $\delta(q_v, [a, a]) = (q_v, [a, a], d), \quad \forall a \in V_T;$   
 Se verifică dacă elementele de pe cele două componente ale benzii coincid.
- (h)  $\delta(q_v, [\$d, \$d]) = (q_f, [\$d, \$d], s).$   
 Se ajunge în starea finală; cuvântul aflat pe prima componentă a benzii este din  $L$ .

Se pornește cu configurația inițială  $(q_0, [\$s, \$s], [a_1, S][a_2, Z] \dots [a_n, Z][\$d, \$d])$ .

Automatul va opera în modul următor:

- Se alege nedeterminist o poziție  $k$  ( $1 \leq k \leq n$ ) astfel ca elementul aflat pe poziția  $k$  în a doua componentă a benzii să fie primul element al membrului stâng al unei producții. Fie  $\alpha_i \rightarrow \beta_i$  această producție. Atunci automatul trece în starea  $q_i$  (definiția (a)).
- Se înlocuiește  $\alpha_i$  cu  $\beta_i$  (definițiile (b),(c),(d)), eventual deplasând spre dreapta conținutul componentei a doua a benzii (definiția (d.ii)).

- Procedeu se reia pentru o altă producție (definițiile (e),(f)).
- Se compară cuvântul rezultat pe a doua componentă cu cel de pe prima (definiția (g)). Dacă cele două cuvinte coincid (definiția (h)), procesul se încheie.

Pentru orice altă situație (nu se poate aplica nici o regulă, a doua componentă este mai lungă decât prima, cele două cuvinte nu coincid) automatul se blochează.

Din acest algoritm reiese imediat că  $w \in L(M)$  dacă și numai dacă există o derivare  $S \xRightarrow{*} w$  în gramatica  $G$ , deci  $w \in L(G)$ . q.e.d.

**Teorema 7.6** Dacă  $M = (Q, \Sigma, \Gamma, \delta, q_0, \$_s, \$_d, F)$  este un LBA atunci  $\tau(M) \in \mathcal{L}_1$ .

*Demonstrație:* Fie  $L = \tau(M)$ . Vom construi o gramatică monotonă  $G = (V_N, V_T, S, P)$  care să genereze  $L \setminus \{\epsilon\}$ . Vom avea:

- $V_N = \{[a, q_0 \$_s X], [a, q_0 \$_s X \$_d], [a, X], [a, qX], [a, \$_s qX], [a, q \$_s X], [a, \$_s qX \$_d], [a, q \$_s X \$_d], [a, \$_s X q \$_d] \mid a \in \Sigma \cup \{\epsilon\}, X \in \Gamma, q_0, q \in Q\} \cup \{S, S_1\}$ ;
- $V_T = \Sigma$ ;
- $P$  conține regulile de producție (s-a notat  $a, b \in \Sigma \setminus \{\$_s, \$_d\}$ ,  $X, Y, Z \in \Gamma$ ,  $p, q \in Q$ ):
  1.  $S \longrightarrow [a, q_0 \$_s a] S_1 \mid [a, \$_s a \$_d]$ ;
  2.  $S_1 \longrightarrow [a, a] S_1 \mid [a, a \$_d]$ ;
  3. Dacă  $(p, \$_s, d) \in \delta(q, \$_s)$  atunci  
 $[a, q \$_s X] \longrightarrow [a, \$_s pX], \quad [a, q \$_s a \$_d] \longrightarrow [a, \$_s p a \$_d]$ ;
  4. Dacă  $(p, Y, d) \in \delta(q, X)$  atunci  
 $[a, qX][b, Z] \longrightarrow [a, Y][b, pZ], \quad [a, \$_s qX][b, Z] \longrightarrow [a, \$_s Y][b, pZ],$   
 $[a, \$_s qX \$_d] \longrightarrow [a, \$_s Y p \$_d]$ ;
  5. Dacă  $(p, Y, s) \in \delta(q, X)$  atunci  
 $[a, \$_s qX \$_d] \longrightarrow [a, p \$_s Y \$_d], \quad [a, \$_s qX] \longrightarrow [a, p \$_s Y],$   
 $[b, Z][a, qX] \longrightarrow [b, pZ][a, Y], \quad [b, Z][a, qX \$_d] \longrightarrow [b, pZ][a, Y \$_d]$ ;
  6. Dacă  $(p, \$_d, s) \in \delta(q, \$_d)$  atunci  
 $[a, b q \$_d] \longrightarrow [a, p b \$_d], \quad [a, \$_s X q \$_d] \longrightarrow [a, \$_s p X \$_d]$ ;
  7. Dacă  $q \in F$ ,  $\alpha, \beta \in \Gamma \cup \{\epsilon\}$  atunci  
 $[a, \alpha q \beta] \longrightarrow a, \quad b[a, \alpha] \longrightarrow ba,$   
 $[a, \alpha] b \longrightarrow ab, \quad \alpha \in \Gamma \cup \{\$_s\} \Gamma \{\$_d\} \cup \{\$_s\} \Gamma \cup \Gamma \{\$_d\}.$

În prima fază, gramatica generează o secvență finită de perechi  $[a, X]$ , unde prima componentă formează prin concatenare un cuvânt din  $V_T^+$ . În continuare, pe a doua componentă se simulează mișcările automatului. La sfârșit se ajunge la un cuvânt din  $L(G)$  numai dacă pe a doua componentă rezultă un cuvânt identic cu cel de pe prima componentă.

De remarcat că  $\epsilon \notin L(G)$ , deoarece  $\$_s \$_d$  nu este o intrare pentru automatul liniar mărginit  $M$ . q.e.d.

## 7.3 Exerciții

**Exercițiul 7.1** *Construiți gramatici dependente de context pentru limbajele*

$$\begin{aligned} L_1 &= \{a^{n+1}b^nc^{n-1} \mid n \geq 1\}, \\ L_2 &= \{a^n b^n c^{2n} \mid n \geq 1\} \\ L_3 &= \{a^n b^m a^n b^m \mid m, n \geq 1\} \\ L_4 &= \{ww \mid w \in \{a, b\}^+\} \end{aligned}$$

**Exercițiul 7.2** *Construiți gramatici dependente de context pentru limbajele*

$$\begin{aligned} L_1 &= \{w \mid w \in \{a, b, c\}^+, |w|_a = |w|_b = |w|_c\}, \\ L_2 &= \{w \mid w \in \{a, b, c\}^+, |w|_a = |w|_b < |w|_c\} \\ L_3 &= \{a^{2^n} \mid n \geq 0\} \\ L_4 &= \{a^{n^2} \mid n \geq 1\} \end{aligned}$$

**Exercițiul 7.3** *Să se arate că  $\mathcal{L}_1$  este închisă la operația de oglindire.*

**Exercițiul 7.4** *Construiți automate liniar mărginite pentru limbajele*

$$\begin{aligned} L_1 &= \{a^p \mid p \text{ număr prim}\}; \\ L_2 &= \{a^p \mid p \text{ nu este număr prim}\}; \\ L_3 &= \{ww \mid w \in \{a, b\}^+\}; \end{aligned}$$

**Exemplul 7.3** *Completați demonstrația din Exemplul 7.2*

**Exercițiul 7.5** *Construiți automate liniar mărginite pentru limbajele din Exercițiile 7.1 și 7.2.*

**Exercițiul 7.6** *Să se arate că orice LBA este echivalent cu un LBA în care  $|F| = 1$ .*

# Capitolul 8

## Limbaje de tip 0 și mașini Turing

Mașinile Turing constituie un model matematic simplu al calculatorului. În ciuda simplității lor însă, ele modelează capacitatea generală de calcul a acestuia. Studiul unei mașini Turing poate fi făcut atât din punctul de vedere al clasei de limbaje pe care le definește (mulțimile recursiv numărabile, sau pe scurt –  $\mathcal{L}_0$ ) cât și din cel al clasei de funcții întregi pe care le calculează (funcțiile parțial recursive).

Noțiunea intuitivă de *algoritm* sau *procedură efectivă* a apărut în matematică de multă vreme. Știm că există algoritmi efectivi de determinare dacă mulțimea acceptată de un  $AF$  este vidă sau finită (Propoziția 2.1. Capitolul 2). Există însă și altfel de întrebări, la care răspunsul este negativ (de exemplu, nu există nici un algoritm care să determine dacă și complementara unui limbaj independent de context este vidă, deși pentru  $\mathcal{L}_2$  o asemenea procedură efectivă există).

La începutul sec.XX, matematicianul *David Hilbert* a pus problema găsirii unui algoritm care să determine valoarea de adevăr sau fals a unei propoziții matematice<sup>1</sup>. În particular, el a cercetat dacă există o procedură pentru a determina dacă o formulă arbitrară din calculul predicatelor de ordinul  $I$ , aplicată la  $\mathcal{Z}$ , este adevărată sau nu.

În 1931 *Kurt Gödel* publică faimoasa sa teoremă de incompletitudine, care demonstrează că o astfel de procedură efectivă nu există. El a construit o formulă în calculul predicatelor aplicat lui  $\mathcal{Z}$ , a cărei valoare de adevăr nu poate nici afirmat nici negat în interiorul sistemului său logic. Formalizarea acestui argument și deci formalizarea și clarificarea noțiunii intuitive de procedură efectivă de calcul este una din cele mai mari realizări teoretice secolului XX.

Odată formalizată noțiunea de procedură efectivă, s-a arătat că nu există proceduri efective de calcul pentru multe funcții. Deși existența unor funcții necalculabile nu surprinde, ceea ce a surprins a fost existența de funcții necalculabile cu o mare pondere în

---

<sup>1</sup>Problema este cunoscută sub numele de *SAT*.

matematică, computer science sau în alte discipline<sup>2</sup>

Astăzi, mașinile Turing au devenit o formalizare acceptabilă a procedurilor efective. Evident, nu se poate demonstra că un model de mașină Turing este echivalent cu noțiunea intuitivă de calculator, dar există argumente în sprijinul unei asemenea aserțiuni, cum ar fi *teza lui Church*. În particular, mașina Turing este echivalentă în putere de calcul cu un calculator așa cum îl știm astăzi precum și cu toate noțiunile cele mai generale de calcul matematic cunoscute.

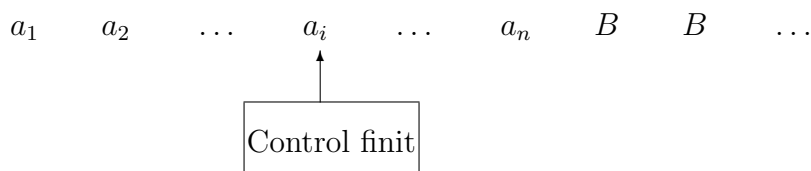
## 8.1 Definiția mașinii Turing

Conform lui Turing (1936), un model formal pentru o procedură efectivă trebuie să posede anumite proprietăți:

- Descrierea să poată fi făcută finit (în timp și spațiu);
- Procedura este formată din pași consecutivi care se pot realiza mecanic.

Modelul prezentat de Turing constă din următoarele componente (similare tuturor automatelor):

1. un control finit (număr finit de stări);
2. o bandă de intrare teoretic infinită, divizată în celule (unități de memorie) secvențiale, fiecare celulă putând conține un caracter (de intrare) sau blank (notat  $B$ ); se cunoaște poziția primei celule;
3. un cap de citire pe bandă, care prelucrează o celulă pe unitatea de timp; modalitatea de prelucrare diferențiază automatele între ele.



La mașinile Turing, în funcție de starea controlului finit și de simbolul curent citit pe bandă, automatul efectuează următoarele acțiuni:

- schimbă starea;

---

<sup>2</sup>Securitatea informației și în special Criptografia cu chei publice se bazează în mare parte pe studiul unor astfel de funcții.

- înlocuiește simbolul citit pe bandă cu altul;
- mișcă capul de citire cu o poziție spre dreapta sau stânga.

**Definiția 8.1** Se numește mașină Turing (MT) structura  $M = (Q, V, \Gamma, \delta, q_0, B, F)$  cu:

1.  $Q$  – mulțime finită nevidă de elemente numite "stări";
2.  $V$  – mulțime finită numită "alfabet de intrare";
3.  $\Gamma$  – mulțime finită nevidă de "simboluri de bandă";  $Q \cap \Gamma = \emptyset$ ,  $V \subseteq \Gamma$ ;
4.  $q_0$  – un element din  $Q$  numit "stare inițială";
5.  $B$  – un element din  $\Gamma$  reprezentând blankul (spațiul liber);  $B \in \Gamma \setminus V$ ;
6.  $F \subseteq Q$  – mulțime nevidă de elemente numite "stări finale";
7.  $\delta : Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{s, d\}}$  – "funcția de tranziție" (poate fi și parțial definită).

În cazul când  $|\delta(q, x)| \leq 1$ ,  $\forall q \in Q$ ,  $x \in \Gamma$ , mașina Turing este deterministă.

**Observația 8.1** Un automat liniar mărginit este un caz particular de MT, în care nu există simbolul  $B$  (blank) – deci simbolurile de pe bandă nu pot fi șterse ci numai înlocuite (eventual). În plus, banda oferă un spațiu de lucru finit (delimitat de marcatorii  $\$s, \$d$  care mărginesc cuvântul de intrare), ceea ce nu este cazul la o MT, unde banda este ipotetic infinită; orice celulă de pe bandă conține un element din  $\Gamma$  (eventual  $B$ ).

Multe elemente definite pentru LBA sunt valabile și în cazul MT. Astfel, o descriere instantanee (DI) a unei mașini Turing este tripletul  $(q, \alpha, \beta)$  unde:

- $q$  este starea curentă;
- $\alpha$  este o stivă cu vârful la dreapta; reprezintă ceea ce se află pe bandă până la capul de citire;
- $\beta$  este o stivă de lucru, cu vârful la stânga; reprezintă conținutul benzii, de la capul de bandă până la ultimul caracter diferit de  $B$ ; primul caracter din  $\beta$  este pe poziția capului de citire și-l vom numi "caracter curent".

Dacă  $\beta = \epsilon$ , atunci pe poziția capului de citire este  $B$ .

**Observația 8.2** O descriere instantanee  $(q, \alpha, \beta)$  poate fi notată – fără a crea ambiguități – și printr-o secvență  $\alpha q \beta$ .

*Mișcări:*

$$1. (q, \alpha a, b\beta) \stackrel{s}{\vdash} (p, \alpha, ax\beta) \text{ dacă } \delta(q, b) = (p, x, s).$$

O astfel de mișcare este posibilă numai dacă  $\alpha a \neq \epsilon$ .

$$2. (q, \alpha a, b\beta) \stackrel{d}{\vdash} (p, \alpha ax, \beta) \text{ dacă } \delta(q, b) = (p, x, d).$$

Putem nota mișcările în general și prin  $\stackrel{M}{\vdash}$  sau – dacă nu apare nici o ambiguitate de notație – pur și simplu cu  $\vdash$ .

Notăm cu  $\vdash^*$  închiderea reflexiv-tranzitivă a acestei relații; putem atunci defini *limbajul acceptat* de mașina Turing  $M$  ca fiind

$$\tau(M) = \{w \mid w \in V^*, (q_0, \epsilon, w) \vdash^* (p, \alpha, \beta) \text{ unde } p \in F, \alpha, \beta \in \Gamma^*\}.$$

De remarcat că pentru orice cuvânt  $w \in \tau(M)$ , mașina Turing se oprește după un număr finit de mișcări.

Dacă  $w \notin \tau(M)$ , este posibil ca mașina Turing să nu se oprească niciodată.

**Exemplul 8.1** *Să construim o MT  $M$  care să accepte limbajul  $L = \{a^n b^n \mid n \geq 1\}$ .*

*Inițial, banda lui  $M$  va conține cuvântul  $a^n b^n$ , urmat de o infinitate de blancuri. Operațiunile efectuate vor fi următoarele:*

- *Cel mai din stânga caracter 'a' se înlocuiește cu  $X$ ;*
- *$M$  se deplasează spre dreapta până la primul 'b' pe care-l înlocuiește cu  $Y$ ;*
- *$M$  se deplasează la stânga până găsește cel mai din dreapta  $X$ ;*
- *Dacă nu mai găsește 'a' - uri,  $M$  caută dacă mai sunt  $b$  - uri; dacă DA, se oprește fără acceptare; dacă NU, cuvântul este acceptat;*
- *Dacă nu mai găsește 'b' - uri,  $M$  nu acceptă cuvântul.*

*Funcția de tranziție  $\delta$  este definită astfel:*

$\delta$	$a$	$b$	$X$	$Y$	$B$
$q_0$	$(q_1, X, d)$	—	—	$(q_3, Y, d)$	—
$q_1$	$(q_1, a, d)$	$(q_2, Y, s)$	—	$(q_1, Y, d)$	—
$q_2$	$(q_2, a, s)$	—	$(q_0, X, d)$	$(q_2, Y, s)$	—
$q_3$	—	—	—	$(q_3, Y, d)$	$(q_4, B, d)$
$q_4$	—	—	—	—	—

$q_4$  este singura stare finală.

De exemplu, pentru  $a^2b^2$ ,  $M$  va lucra astfel:

$$\begin{aligned} (q_0, \epsilon, aabb) &\stackrel{s}{\vdash} (q_1, X, abb) \stackrel{d}{\vdash} (q_1, Xa, bb) \stackrel{s}{\vdash} (q_2, X, aYb) \stackrel{s}{\vdash} (q_2, \epsilon, XaYb) \stackrel{d}{\vdash} (q_0, X, aYb) \stackrel{d}{\vdash} \\ (q_1, XX, Yb) &\stackrel{d}{\vdash} (q_1, XXY, b) \stackrel{s}{\vdash} (q_2, XX, YY) \stackrel{s}{\vdash} (q_2, X, XYY) \stackrel{d}{\vdash} (q_0, XX, YY) \stackrel{d}{\vdash} \\ (q_3, XXY, Y) &\stackrel{d}{\vdash} (q_3, XXY, \epsilon) \stackrel{d}{\vdash} (q_4, XXYB, \epsilon). \end{aligned}$$

## 8.2 Proprietăți ale mașinilor Turing

**Teorema 8.1** *Mașinile Turing sunt metode de recunoaștere.*

*Demonstrație:* Este similară demonstrației de la Teorema 7.4. (Capitolul 7), relativă la automatele liniar mărginite. q.e.d.

**Teorema 8.2** *Orice limbaj de tip 0 este acceptat de o mașină Turing.*

*Demonstrație:* Deoarece  $L \in \mathcal{L}_0$ , există o gramatică  $G = (V_N, V_T, S, P)$  cu  $L = L(G)$ . Se construiește o mașină Turing  $M$  care să funcționeze în felul următor: Pentru un cuvânt de intrare  $w \in V_T^*$  (scris inițial pe bandă),  $M$  alege în mod nedeterminist o poziție  $i$  în  $w$  și o producție  $u \rightarrow v \in P$ . Dacă începând cu poziția  $i$ , pe bandă apare  $v$  ca subcuvânt al lui  $w$  (adică  $w = \alpha v \beta$ ), atunci  $v$  este înlocuit cu  $u$ . De remarcat că prin această înlocuire este posibil ca  $\beta$  să fie deplasat spre stânga (dacă  $|v| > |u|$ ) sau spre dreapta (dacă  $|v| < |u|$ ).

Dacă după un număr finit de astfel de operații conținutul benzii este  $S$  (de fapt  $B^*SB^*$ ), atunci  $w$  este acceptat; altfel este respins. Evident,  $L(G) = \tau(M)$ .

Lăsăm ca exercițiu detaliile de construcție ale lui  $M$ .

q.e.d.

**Teorema 8.3** *Limbajul acceptat de o mașină Turing este de tip 0.*

*Demonstrație:* Fie mașina Turing  $M = (Q, V, \Gamma, \delta, q_0, B, F)$  și  $L = \tau(M)$ . Definim gramatica de tip 0  $G = (V_N, V_T, S, P)$  astfel:

$V_N = ((\Sigma \cup \{\epsilon\}) \times \Gamma) \cup Q \cup \{S, A_1, A_2\}$ ,  $V_T = \Sigma$ , iar mulțimea  $P$  de producții este formată din:

1.  $S \rightarrow q_0 A_1$ ;
2.  $A_1 \rightarrow [a, a] A_1, \quad \forall a \in \Sigma$ ;
3.  $A_1 \rightarrow A_2$ ;
4.  $A_2 \rightarrow [\epsilon, B] A_2$ ;



5.  $A_2 \longrightarrow \epsilon$ ;
6.  $q[a, x] \longrightarrow [a, y]p$  dacă  $(p, y, d) \in \delta(q, x)$ ,  $\forall a \in \Sigma \cup \{\epsilon\}$ ,  $p, q \in Q$ ,  $x, y \in \Gamma$ ;
7.  $[b, z]q[a, x] \longrightarrow p[b, z][a, y]$  dacă  $(p, y, s) \in \delta(q, x)$   $\forall a, b \in \Sigma \cup \{\epsilon\}$ ,  
 $p, q \in Q$ ,  $x, y, z \in \Gamma$ ;
8.  $[a, x]q \longrightarrow qaq$ ,  $q[a, x] \longrightarrow qaq$ ,  $q \longrightarrow \epsilon$   $\forall a \in \Sigma \cup \{\epsilon\}$ ,  $q \in Q$ ,  $x \in \Gamma$ .

Să demonstrăm prin dublă incluziune egalitatea  $L(G) = \tau(M)$ .

" $\tau(M) \subseteq L(G)$ ": Fie  $w = a_1 a_2 \dots a_n \in \tau(M)$ ; deci

$$(q_0, \epsilon, a_1 a_2 \dots a_n) \stackrel{*}{\vdash} (q, x_1 \dots x_{r-1}, x_r \dots x_p) \text{ și } q \in F.$$

Va trebui să găsim o derivare  $S \stackrel{*}{\Rightarrow} w$  (care conduce la concluzia  $w \in L(G)$ ). Utilizând regulile (1) și (2), prima parte a acestei derivări trebuie să fie

$$S \stackrel{*}{\Rightarrow} q_0[a_1, a_1] \dots [a_n, a_n]A_1 \text{ cu } a_i \in \Sigma \text{ (} 1 \leq i \leq n \text{)}.$$

Cum  $M$  acceptă cuvântul  $w$ , fie  $m$  numărul de locații la dreapta ultimului caracter, cercetat de capul de citire al  $MT$  atunci când acceptă această secvență. Utilizând producția (3), apoi (4) de  $m$  ori și în final (5), se obține derivarea

$$S \stackrel{*}{\Rightarrow} q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m.$$

Vom demonstra prin inducție după numărul de pași în  $M$  că

$$(q_0, \epsilon, a_1 \dots a_n) \stackrel{s}{\vdash} (q, x_1 \dots x_{r-1}, x_r \dots x_p) \quad (i)$$

implică

$$q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m \stackrel{*}{\Rightarrow} [a_1, x_1] \dots [a_{r-1}, x_{r-1}]q[a_r, x_r] \dots [a_{n+m}, a_{n+m}] \quad (ii)$$

unde

$$a_1, \dots, a_n \in \Sigma, a_{n+1} = \dots = a_{n+m} = \epsilon, x_1, \dots, x_{n+m} \in \Gamma, x_{p+1} = \dots = x_{n+m} = B.$$

**s = 0:** În acest caz (1) devine  $(q_0, \epsilon, a_1 \dots a_n) \stackrel{0}{\vdash} (q_0, \epsilon, a_1 \dots a_n)$ , deci  $r = 1$  și  $p = n$ . În gramatica  $G$  avem

$$q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m \stackrel{0}{\Rightarrow} q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m$$

și  $a_1, \dots, a_n \in \Sigma$ ,  $a_{n+1} = \dots = a_{n+m} = \epsilon$ ,  $x_1, \dots, x_{n+m} \in \Gamma$ ,  $x_{p+1} = \dots = x_{n+m} = B$ .

Să presupunem implicația adevărată pentru  $s \leq k-1$  și să o demonstrăm pentru  $s = k$ . Fie

$$(q_0, \epsilon, a_1 \dots a_n) \stackrel{k-1}{\vdash} (q, x_1 \dots x_{r-1}, x_r \dots x_p) \vdash (p, y_1 \dots y_{t-1}, y_t \dots y_n) \quad (iii)$$

Conform ipotezei de inducție, avem

$$q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m \stackrel{*}{\Rightarrow} [a_1, x_1] \dots [a_{r-1}, x_{r-1}]q[a_r, x_r] \dots [a_{n+m}, a_{n+m}]$$

unde toate componentele verifică condițiile din (ii).

Ultimul pas din (3) se poate face printr-o mișcare a capului de citire spre dreapta sau spre stânga.

- Dacă mișcarea se face spre dreapta, atunci  $t - 1 = r$  și  $(p, y_r, d) \in \delta(q, x_r)$ . Deci – conform cu (6) – în  $P$  există producția  $q[a_r, x_r] \longrightarrow [a_r, y_r]p$  care – folosită în derivarea din gramatica  $G$  – conduce la

$$q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m \xRightarrow{*} [a_1, x_1] \dots [a_{r-1}, x_{r-1}]q[a_r, x_r] \dots [a_{n+m}, a_{n+m}] \Rightarrow \\ \Rightarrow [a_1, x_1] \dots [a_{r-1}, x_{r-1}][a_r, y_r]p[a_{r+1}, x_{r+1}] \dots [a_{n+m}, a_{n+m}].$$

- Dacă are loc o mișcare spre dreapta a capului de citire, atunci  $t = r - 1$  și  $(p, y_r, s) \in \delta(q, x_r)$  ( $r > 2$ ). Va rezulta – conform cu (7) – că în  $P$  există producția  $[a_{r-1}, x_{r-1}]q[a_r, x_r] \longrightarrow p[a_{r-1}, x_{r-1}][a_r, y_r]$ . Putem atunci scrie în gramatica  $G$ :

$$q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m \xRightarrow{*} [a_1, x_1] \dots [a_{r-1}, x_{r-1}]q[a_r, x_r] \dots [a_{n+m}, a_{n+m}] \Rightarrow \\ \Rightarrow [a_1, x_1] \dots p[a_{r-1}, x_{r-1}][a_r, y_r] \dots [a_{n+m}, a_{n+m}], \text{ deci de forma (iii).}$$

Deoarece  $w = a_1 \dots a_n \in \tau(M)$  rezultă că există secvența de mișcări

$$(q_0, \epsilon, a_1 \dots a_n) \vdash^* (q, x_1 \dots x_{r-1}, x_r \dots x_p), \quad q \in F.$$

Din faptul că (i) implică (ii) se obține derivarea

$$q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m \xRightarrow{*} [a_1, x_1] \dots [a_{r-1}, x_{r-1}]q[a_r, x_r] \dots [a_{n+m}, x_{n+m}] \text{ cu } q \in F.$$

Deci  $S \xRightarrow{*} q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m \xRightarrow{*} [a_1, x_1] \dots [a_{r-1}, x_{r-1}]q[a_r, x_r] \dots [a_{n+m}, x_{n+m}]$  cu  $q \in F$ .

Ținând acum cont de producțiile (8), această derivare se completează cu

$$[a_1, x_1] \dots [a_{r-1}, x_{r-1}]q[a_r, x_r] \dots [a_{n+m}, x_{n+m}] \xRightarrow{*} a_1 \dots a_n,$$

deci  $S \xRightarrow{*} w$ .

" $L(G) \subseteq \tau(M)$ ": Fie  $w = a_1 \dots a_n \in L(G)$ ; deci există derivarea  $S \xRightarrow{*} w$ . Din forma producțiilor (1) – (8) rezultă că ultimele reguli aplicate sunt de tipul (8) (sunt singurele producții terminale). Dar pentru aplicarea lor trebuie să existe o formă sentențială de tipul membrului drept din (ii), cu  $q \in F$ . Deci derivarea trebuie să fie de forma

$$S \xRightarrow{*} q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m \xRightarrow{*} [a_1, x_1] \dots [a_{r-1}, x_{r-1}]q[a_r, x_r] \dots [a_{n+m}, x_{n+m}] \xRightarrow{*} w.$$

Considerăm derivarea

$$q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m \xRightarrow{s} [a_1, x_1] \dots [a_{r-1}, x_{r-1}]q[a_r, x_r] \dots [a_{n+m}, x_{n+m}]$$

unde  $q \in F$ ,  $a_1, \dots, a_n \in \Sigma$ ,  $a_{n+1} = \dots = a_{n+m} = \epsilon$  și  $x_1, \dots, x_{n+m} \in \Gamma$ .

Aceasta este de forma (ii). Vom arăta că (ii) implică (i) prin inducție după lungimea  $s$  a derivării.

**s = 0:** Atunci (ii) devine  $q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m \xRightarrow{*} q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m$  și deci în  $M$  există secvența de 0 mișcări  $(q_0, \epsilon, a_1 \dots a_n) \vdash^0 (q_0, \epsilon, a_1 \dots a_n)$ .

Presupunem implicația adevărată pentru derivări de lungime  $s \leq k - 1$  și o demonstrăm pentru  $s = k$ .

Fie derivarea de lungime  $k$ :

$$\begin{aligned} q_0[a_1, a_1] \dots [a_n, a_n][\epsilon, B]^m &\xRightarrow{k-1} [a_1, x_1] \dots [a_{r-1}, x_{r-1}]q[a_r, x_r] \dots [a_{n+m}, x_{n+m}] \implies \\ &\implies [a_1, y_1] \dots [a_{t-1}, y_{t-1}]p[a_t, y_t] \dots [a_{n+m}, y_{n+m}]. \end{aligned}$$

Ultimul pas al derivării folosește o producție de tipul (6) sau (7).

- Dacă este folosită o producție de tipul (6), atunci  $(p, y_r, d) \in \delta(q, x_r)$  și  $t = r + 1$ .  
În acest caz avem  $(q, x_1 \dots x_{r-1}, x_r \dots x_p) \vdash (p, x_1 \dots x_{r-1}y_r, x_{r+1} \dots x_p)$ .
- Dacă este folosită o producție de tipul (7), atunci  $(p, y_r, s) \in \delta(q, x_r)$  și  $t = r - 1$ ;  
deci în  $M$  avem mișcarea  $(q, x_1 \dots x_{r-1}, x_r \dots x_p) \vdash (p, x_1 \dots x_{r-2}, x_{r-1}y_rx_{r+1} \dots x_p)$ .

Deoarece în derivarea existentă în gramatica  $G$  avem o secvența de tipul (ii), iar (ii) implică (i), în  $MT$   $M$  vom avea

$$(q_0, \epsilon, a_1 \dots a_n) \vdash^* (q, x_1, \dots x_r, x_{r+1} \dots x_p)$$

cu  $q \in F$ , deci  $w \in \tau(M)$ .

q.e.d.

### 8.3 Limbaje recursive și recursiv numărabile

**Definiția 8.2** *Limbaajul acceptat de o mașină Turing se numește recursiv numărabil.*

Termenul de *numărabil* derivă din faptul că acestea sunt limbajele ale căror cuvinte pot fi numărate (listate) de o mașină Turing. Pentru un limbaj  $L$  și o mașină Turing  $M$  care acceptă  $L$ , după ce mașina primește la intrare un cuvânt  $w$ , se pot întâmpla următoarele evenimente:

- $M$  se oprește după un număr finit de mișcări într-o stare finală, după ce a parcurs complet  $w$ ; spunem că  $w$  este acceptat de  $M$  (sau  $w \in L$ );
- $M$  se oprește după un număr finit de mișcări într-o stare nefinală;
- $M$  nu se oprește niciodată (intră într-o buclă infinită);

În aceste două ultime cazuri, spunem că  $w$  nu este acceptat de  $M$  (sau  $w \notin L$ ).

Există o subclasă de limbaje, numite *limbaje recursive*, care sunt acceptate de mașini Turing cu proprietatea că se opresc pentru orice cuvânt de intrare.

**Teorema 8.4** *Orice limbaj  $L \in \mathcal{L}_1$  este acceptat de o MT care oprește pentru toate intrările.*

*Demonstrație:* Deoarece  $L$  este un limbaj senzitiv de context, există o gramatică monotonă  $G = (V_N, V_T, S, P)$  cu  $L = L(G)$ . Pe de altă parte, cum  $\mathcal{L}_1 \subseteq \mathcal{L}_0$ , există o mașină Turing (nedeterministă)  $M$  cu  $L = \tau(M)$ .

Fie  $w \in V_T^*$  cu  $|w| = n$ . Folosind Algoritmul 7.1. (dat de Teorema 7.3, Capitolul 7), putem decide dacă  $w \in L(G)$ . Vom construi  $M$  în așa fel încât să simuleze acest algoritm: pe bandă să apară numai elemente din  $T_i$  obținute din  $T_{i-1}$  prin aplicarea de producții din  $P$ . Cum o MT poate simula derivări din  $G$ , acest lucru este posibil. În plus, orice secvență  $\alpha \in V_i \cap V_T^*$  obținută pe bandă va trece mașina într-o stare finală. Evident, după un număr finit de pași, mașina Turing se va opri. Cuvântul de pe bandă este în  $L(G)$  dacă și numai dacă starea în care s-a oprit automatul este finală. q.e.d.

Pe baza Teoremei 8.4 putem demonstra că incluziunea  $\mathcal{L}_1 \subseteq \mathcal{L}_0$  este strictă. Aceasta va rezulta din următoarele două propoziții.

**Propoziția 8.1** *Fie  $M_1, M_2, \dots$  o enumerare a mașinilor Turing care opresc pentru toate intrările. Există un limbaj recursiv care nu este acceptat de nici o mașină Turing.*

*Demonstrație:* Fie  $\Sigma = \{0, 1\}$  și  $L \subset \Sigma^*$  definit astfel:  $w \in L$  dacă și numai dacă mașina Turing  $M_i$  nu acceptă  $w$ , unde reprezentarea binară a lui  $i$  este  $w$ .

Limbajul  $L$  este recursiv: fiind dat un cuvânt  $w \in \Sigma^*$ , calculăm  $i$  și vedem dacă  $w \in \tau(M_i)$  sau nu (lucru posibil, deoarece  $M_i$  se oprește pentru toate intrările). Vom arăta că în șirul din enunț nu se află nici o mașină Turing  $M_j$  cu  $L = \tau(M_j)$ .

Presupunem prin absurd că ar exista o astfel de mașină  $M_j$  și fie  $x$  reprezentarea în binar a lui  $j$ . Din construcția limbajului  $L$  rezultă că dacă  $x \in L$  atunci  $x \notin \tau(M_j)$  și dacă  $x \notin L$  atunci  $x \in \tau(M_j)$ ; deci  $L \neq \tau(M_j)$ . q.e.d.

Din Propoziția 8.1 rezultă că  $\mathcal{L}_1$  este inclusă strict în clasa limbajelor recursive.

**Propoziția 8.2** *Există un limbaj recursiv enumerabil care nu este senzitiv de context.*

*Demonstrație:* Considerăm o gramatică  $G = (V_N, \{0, 1\}, S, P)$  de tip 1. Vom codifica caracterele gramaticii  $0, 1, ,, \longrightarrow, \{, \}, (, )$  cu  $10, 1^2, \dots, 10^8$ . Pentru neterminale, codificarea se face în felul următor: al  $i$ -lea neterminal este codificat cu  $10^{8+i}$ . Deci putem codifica fiecare gramatică  $G$  cu un cod binar, cod care permite să enumerăm toate gramaticile senzitive de context. Fiecărei astfel de gramatici îi corespunde o MT echivalentă care se oprește pentru toate intrările (Teorema 8.4). Din Propoziția 8.1 am văzut că există un limbaj  $L$  care nu poate fi recunoscut de o MT care oprește pentru toate intrările. Deci  $L \notin \mathcal{L}_1$ . q.e.d.

## 8.4 Mașina Turing folosită pentru calculul funcțiilor întregi

Înafara rolului de a accepta limbaje recursiv numărabile, o mașină Turing poate fi privită și ca un mod de a calcula funcții de variabilă întreagă cu valori întregi.

Ideea este de a reprezenta întregii în baza 1: un număr  $i \geq 0$  poate fi reprezentat prin șirul  $0^i$ . Dacă o funcție are  $n$  argumente  $i_1, i_2, \dots, i_n$ , aceste numere întregi scrise în baza 1 sunt plasate pe bandă și separate prin câte un 1:

$$0^{i_1} 1 0^{i_2} 1 \dots 1 0^{i_n}$$

Dacă mașina Turing se oprește (într-o stare finală sau nu) cu  $0^m$  pe bandă, spunem că

$$f(i_1, i_2, \dots, i_n) = m.$$

De remarcat că  $f$  poate să nu fie definită pentru orice set de valori  $(i_1, i_2, \dots, i_n)$ .

Dacă  $f(i_1, i_2, \dots, i_n)$  este definită pentru toate  $n$ -tuplurile  $(i_1, i_2, \dots, i_n)$ , atunci spunem că  $f$  este *total recursivă*. O funcție calculată de o mașină Turing este numită *funcție parțial recursivă*.

Prin analogie, funcțiile total recursive corespund limbajelor recursive iar funcțiile parțial recursive corespund limbajelor recursiv numărabile.

Toate funcțiile matematice uzuale sunt funcții total recursive.

**Exemplul 8.2** *Operația de scădere întreagă este definită*

$$m \dot{-} n = \begin{cases} m - n & \text{dacă } m \geq n \\ 0 & \text{altfel} \end{cases}$$

Mașina Turing  $M = (\{q_0, \dots, q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_6\})$  definită mai jos, pleacă din configurația de bandă  $0^m 1 0^n$  și se oprește când pe bandă rămâne  $0^{m-n}$ .

Ca idee de lucru,  $M$  șterge sistematic primul 0, apoi caută spre dreapta până la ultimul 1; primul 0 de după acesta îl înlocuiește cu 1.

Funcția  $\delta$  este definită:

$\delta(q_0, 0) = (q_1, B, d)$ ; Se începe ciclul; se înlocuiește primul 0 cu B.

$\delta(q_1, 0) = (q_1, 0, d)$ ;

$\delta(q_1, 1) = (q_2, 1, d)$ ; Se caută spre dreapta primul 1.

$\delta(q_2, 1) = (q_2, 1, d)$ ;

$\delta(q_2, 0) = (q_3, 1, s)$ ; Deplasare dreapta până la primul 0, care se transformă în 1.

$\delta(q_3, 0) = (q_3, 0, s)$ ;

$\delta(q_3, 1) = (q_3, 1, s)$ ;

$\delta(q_3, B) = (q_0, B, d);$       *Deplasare stânga până la primul blank; se repetă ciclul.*  
 $\delta(q_2, B) = (q_4, B, s);$   
 $\delta(q_4, 1) = (q_4, B, s);$   
 $\delta(q_4, 0) = (q_4, 0, s);$   
 $\delta(q_4, B) = (q_6, 0, d);$       *Configurație finală de acceptare; se trece în starea  $q_6$  și **Stop**.*  
 $\delta(q_0, 1) = (q_5, B, d);$   
 $\delta(q_5, 0) = (q_5, B, d);$   
 $\delta(q_5, 1) = (q_5, B, d);$   
 $\delta(q_5, B) = (q_6, B, d);$       *Configurație finală de acceptare; se trece în starea  $q_6$  și **Stop**.*

*De exemplu, pentru intrarea 0010, mașina  $M$  efectuează calculele:*

$(q_0, \epsilon, 0010) \vdash (q_1, B, 010) \vdash (q_1, B0, 10) \vdash (q_2, B01, 0) \vdash (q_3, B0, 11) \vdash (q_3, B, 011) \vdash$   
 $(q_3, \epsilon ps, B011) \vdash (q_0, B, 011) \vdash (q_1, BB, 11) \vdash (q_2, BB1, 1) \vdash (q_2, 11, \epsilon) \vdash (q_4, 1, 1) \vdash$   
 $(q_4, BB, 1) \vdash (q_4, B, \epsilon) \vdash (q_6, B0, \epsilon).$

*iar pentru intrarea 0100,  $M$  procedează astfel:*

$(q_0, \epsilon, 0100) \vdash (q_1, B, 100) \vdash (q_2, B1, 00) \vdash (q_3, B, 110) \vdash (q_3, \epsilon, B110) \vdash (q_0, B, 110) \vdash$   
 $(q_5, BB, 10) \vdash (q_5, BBB, 0) \vdash (q_5, BBBB, ) \vdash (q_6, BBBBB, \epsilon).$

## 8.5 Tehnici de construcție a mașinilor Turing

Metoda de a defini o mașină Turing scriind lista completă a stărilor precum și toată tabela de tranziție, nu este totdeauna un lucru ușor de realizat. În cazul unor  $MT$  complicate, este necesar să folosim anumite concepte mai generale; vom face în continuare o trecere în revistă a unor astfel de construcții, folosind exemplificări pentru fiecare caz.

### Stocarea în control finit

Controlul finit poate fi o informație utilă în descrierea unei  $MT$ . Pentru aceasta, o stare este scrisă ca o pereche de elemente: una pentru controlul iar a doua pentru stocarea unui simbol. În rest, definiția mașinii Turing rămâne neschimbată.

**Exemplul 8.3** *Să considerăm o mașină Turing care ia primul simbol de pe bandă, îl înregistrează și apoi cercetează dacă acesta mai apare. Cuvântul este acceptat în cazul în care primul caracter nu se repetă.*

*Problema este foarte simplă (de fapt  $\tau(M)$  este un limbaj regulat) dar o folosim pentru a reliefa utilizarea acestui tip de definire pentru o mașinile Turing.*

$$M = (Q, \{0, 1\}, \{0, 1, B\}, \delta, [q_0, B], B, F)$$

unde  $Q = \{q_0, q_1\} \times \{0, 1, B\}$ ,  $F = \{[q_1, B]\}$ , iar  $\delta$  este definită astfel:

$$\delta([q_0, B], 0) = ([q_1, 0], 0, d)$$

$$\delta([q_0, B], 1) = ([q_1, 1], 1, d)$$

Inițial,  $q_0$  este componenta de control a stării, iar  $M$  se deplasează la dreapta; prima componentă a stării lui  $M$  devine  $q_1$  și primul simbol este stocat pe a doua poziție.

$$\delta([q_1, 0], 1) = ([q_1, 0], 1, d)$$

$$\delta([q_1, 1], 0) = ([q_1, 1], 0, d)$$

Dacă simbolul stocat nu coincide cu cel curent,  $M$  continuă deplasarea spre dreapta.

$$\delta([q_1, 0], B) = ([q_1, B], 0, s)$$

$$\delta([q_1, 1], B) = ([q_1, B], 0, s).$$

$MT$  găsește pe bandă un blank fără să fi aflat un simbol identic cu cel căutat; se trece în starea finală.

Dacă  $M$  găsește simbolul căutat, se oprește fără acceptare (funcția  $\delta$  nu mai este definită).

### Stocare multiplă

Putem imagina banda unei mașini Turing ca fiind suficient de lată pentru a putea fi împărțită în  $n$  benzi paralele ( $n \geq 1$ ,  $n$  finit). Simbolurile de pe bandă vor fi atunci vectori de dimensiune  $n$ .

**Exemplul 8.4** Să construim o mașină Turing care să determine dacă un număr natural (mai mare decât 2) este prim. Ea va fi formată din 3 benzi paralele: pe prima, delimitat între  $\$s$  și  $\$d$ , scriem numărul în binar.

Deci, simbolurile de intrare: 0, 1,  $\$s$ ,  $\$d$  se vor scrie  $[0, B, B]$ ,  $[1, B, B]$ ,  $[\$s, B, B]$ ,  $[\$d, B, B]$ .


În particular, blankul se reprezintă prin  $[B, B, B]$ .

Pentru a testa dacă numărul este prim,  $MT$  va scrie numărul 2 în binar pe a doua bandă (aliniat la dreapta față de primul număr) și copiază prima bandă pe a treia. Apoi, banda a doua este scăzută din a treia cât timp este posibil.

Dacă ceea ce rămâne (restul împărțirii celor două numere) este zero, atunci numărul de pe prima bandă nu este prim. În caz contrar numărul de pe banda a doua crește cu o unitate; dacă el este egal cu cel de prima bandă, numărul este prim. Altfel, toată operația se reia.

Situația de pe banda din figură:

$\$s$	1	0	1	1	1	1	$\$d$	$B$	$B$	...
$B$	$B$	$B$	$B$	1	0	1	$B$	$B$	$B$	...
$B$	1	0	0	1	0	1	$B$	$B$	$B$	...



Control finit

corespunde testării dacă 47 este număr prim; momentul este acela când se încearcă împărțirea cu 5, 5 fiind scăzut de două ori (deci pe banda 3 se află numărul 37).

De remarcat că MT din acest exemplu simulează un LBA, lucru normal dacă știm că limbajul  $L = \{a^p \mid p \text{ prim}\}$  este un limbaj dependent de context.

### Controlul exterior al simbolurilor

Această modalitate face posibilă o vizualizare a modului cum o mașină Turing acceptă limbaje definite prin șiruri repetate, cum ar fi de exemplu

$$\{ww \mid w \in V^*\} \quad \{wcy \mid w, y \in V^*, w \neq y\} \quad \{w\tilde{w} \mid w \in V^*\}$$

sau când în definirea limbajelor intervine – în operații de comparare – și lungimea cuvintelor, cum ar fi

$$\{a^n b^n \mid n \geq 1\} \quad \{a^i b^j c^k \mid i \neq j \text{ sau } j \neq k\}.$$

Pentru aceasta, împărțim banda în două: prima parte (superioară) este folosită normal, iar pe a doua parte vor fi numai simbolurile  $B$  și  $\$$ .

Simbolul  $\$$  va apare pentru marcare.

**Exemplul 8.5** Să construim o MT  $M = (Q, V, \Gamma, \delta, q_0, B, F)$  care să recunoască limbajul  $\{wcw \mid w \in \{a, b\}^*\}$ .

Pentru aceasta, definim:

$Q = \{[q_i, x] \mid 1 \leq i \leq 9, x \in \{a, b, B\}\}$ , a doua componentă a stării fiind folosită la stocarea unui caracter de intrare.

$V = \{[B, x] \mid x \in \{a, b, c\}\}$ . Simbolul  $[B, x]$  este o reprezentare internă (de bandă) pentru  $x$ .

$\Gamma = \{[X, x] \mid X \in \{B, \$\}, x \in \{a, b, c, B\}\}$ . Reprezentarea internă a lui  $B$  este  $[B, B]$ .

Pentru orice  $x, y \in \{a, b\}$  funcția  $\delta$  este definită:

$$\delta([q_1, x], [B, y]) = ([q_2, y], [ \$, x], d);$$

Simbolul găsit la intrare este stocat de stare; apoi MT se deplasează la dreapta.

$$\delta([q_2, y], [B, x]) = ([q_2, y], [B, x], d);$$

$M$  continuă să se deplaseze spre dreapta, căutând simbolul  $c$ .

$$\delta([q_2, y], [B, c]) = ([q_3, y], [B, c], d);$$

S-a găsit  $c$ .  $M$  trece într-o nouă stare.

$$\delta([q_3, y], [ \$, x]) = ([q_3, y], [ \$, x], d);$$

$M$  se deplasează la dreapta, peste simbolurile controlate.

$$\delta([q_3, y], [B, y]) = ([q_4, B], [ \$, y], s);$$

$M$  a ajuns la un simbol necontrolat încă. Dacă acesta coincide cu simbolul păstrat pe a doua componentă a stării, MT îl bifează (cu  $\$$ ) și începe deplasarea spre stânga. Dacă simbolurile nu coincid,  $M$  nu are definită altă mișcare și se oprește fără acceptare.

$$\delta([q_4, B], [ \$, y]) = ([q_4, B], [ \$, y], s);$$



$$\begin{aligned}\delta([q_4, B], [B, c]) &= ([q_5, B], [B, c], s); \\ \delta([q_5, B], [B, y]) &= ([q_6, B], [B, y], s); \\ \delta([q_6, B], [B, y]) &= ([q_6, B], [B, y], s); \\ \delta([q_6, B], [\$, y]) &= ([q_1, B], [\$, y], d);\end{aligned}$$

*M s-a deplasat până la cel mai din stânga simbol necontrolat; îl marchează, îl stochează în starea curentă și reia totul de la început.*

$$\delta([q_5, B], [\$, y]) = ([q_7, B], [\$, y], d);$$

*M se află în situația când simbolul aflat la stânga lui c a fost controlat; deci s-au terminat de verificat toate caracterele. Acum MT trebuie să verifice dacă au fost controlate și toate caracterele aflate la dreapta lui c.*

$$\begin{aligned}\delta([q_7, B], [B, c]) &= ([q_8, B], [B, c], d); \\ \delta([q_8, B], [\$, y]) &= ([q_8, B], [\$, y], d); \\ \delta([q_8, B], [B, B]) &= ([q_9, B], [\$, B], s).\end{aligned}$$

*M găsește blankul și se oprește cu acceptare.*

### Mărirea spațiului de lucru

O mașină Turing poate să-și crească spațiul de lucru (lucru imposibil pentru un automat liniar mărginit), deplasând spre dreapta cu un număr finit de poziții toate caracterele diferite de blank.

**Exemplul 8.6** *Să presupunem că MT  $M = (Q, V, \Gamma, \delta, q_0, B, F)$  trebuie să-și deplaseze simbolurile cu două poziții spre dreapta; fără a micșora generalitatea, vom considera că pe bandă simbolurile diferite de blank sunt scrise compact (nu conțin blankuri între ele).*

*$Q$  va conține stări de tipul  $[q_i, A_1, A_2]$  unde  $i = 1, 2, \dots$ ,  $A_1, A_2 \in \Gamma$ .*

*Fie  $X$  un simbol special, nefolosit în altă parte de  $M$ .*

*$M$  începe deplasarea din starea  $[q_1, B, B]$ . Elementele interesante ale funcției  $\delta$  sunt:*

$$\delta([q_1, B, B], A_1) = ([q_1, B, A_1], X, d), \quad A_1 \in \Gamma \setminus \{B, X\};$$

*Primul simbol citit este stocat pe a treia componentă a stării; se tipărește  $X$  pe celula scanată și  $M$  se deplasează la dreapta.*

$$\delta([q_1, B, A_1], A_2) = ([q_1, A_1, A_2], X, d), \quad A_1, A_2 \in \Gamma \setminus \{B, X\};$$

*$M$  mută simbolul de pe a treia componentă pe a doua, stochează simbolul nou găsit în a treia componentă, scrie în locul lui un  $X$  și continuă deplasarea la dreapta.*

$$\delta([q_1, A_1, A_2], A_3) = ([q_1, A_2, A_3], A_1, d), \quad A_1, A_2, A_3 \in \Gamma \setminus \{B, X\};$$

*$M$  continuă deplasarea, păstrând pe pozițiile 2 și 3 ultimele elemente citite pe bandă.*

$$\delta([q_1, A_1, A_2], B) = ([q_1, A_2, B], A_1, d), \quad A_1, A_2 \in \Gamma \setminus \{B, X\};$$

*Când se găsește un blank, ultimele simboluri depozitate sunt scrise pe bandă.*

$$\delta([q_1, A_1, B], B) = ([q_2, B, B], A_1, s);$$

*$M$  a scris cele două simboluri; trece cu prima componentă într-o stare nouă și începe deplasarea spre stânga pentru a găsi prima poziție, marcată cu  $X$ .*

$$\delta([q_2, B, B], A) = ([q_2, B, B], A, s), \quad A \in \Gamma \setminus \{B, X\};$$

Când se ajunge la simbolul  $X$ , automatul va face transferul la o stare care există în  $Q$  și continuă alte operații proprii lui  $M$ .

### Subrutine

Lucrul cu mașinile Turing este mult ușurat dacă – similar programării – vom folosi diverse module pentru a efectua unele operații elementare.

O  $MT$  poate simula orice tip de subrutine din limbajele de programare, inclusiv procedurile recursive sau mecanismele de transfer de parametri.

Ideea generală este de a scrie o parte a  $MT$  care să servească drept subrutină; ea va avea propria ei stare inițială precum și o stare de revenire (din care momentan nu se face nici o mișcare) la rutina apelantă. Când scriem o  $MT$  care "apelează" subrutina, este asigurat un set de stări noi pentru subrutină, precum și transferul din starea de revenire.

Apelul este efectuat intrând în starea inițială a subrutinei iar întoarcerea – prin mișcarea din starea de revenire.

**Exemplul 8.7** Să presupunem că  $M$  are nevoie de funcția total recursivă de "înmulțire". Ea va pleca cu  $0^m 10^n$  pe bandă și se va încheia cu  $0^{mn}$  aflat între blăncuri.

Ideea este de a plasa 1 după  $0^m 10^n$  și apoi de a copia secvența de  $n$  zerouri la dreapta de  $m$  ori, la fiecare trecere ștergând câte un zero din cele  $m$  aflate la început.

Se obține astfel  $10^n 10^{mn}$ .

În final se șterge prefixul  $10^n 1$ .

$MT$  va folosi subrutina **COPY** care începe cu o descriere instantanee  $(q_1, 0^m 1, 0^n 10^i)$  și se termină cu  $(q_5, 0^m 1, 0^n 10^{i+n})$ .

Această subrutină are funcția de tranziție dată de tabela:

	0	1	2	$B$
$q_1$	$(q_2, 2, d)$	$(q_4, 1, s)$	—	—
$q_2$	$(q_2, 0, d)$	$(q_2, 1, d)$	—	$(q_3, 0, s)$
$q_3$	$(q_3, 0, s)$	$(q_3, 1, s)$	$(q_1, 2, d)$	—
$q_4$	—	$(q_5, 1, d)$	$(q_4, 0, s)$	—

Pentru a completa, este necesar ca programul principal să aducă descrierea instantanee inițială  $(q_0, 0^m 10^n)$  la  $(q_1, B 0^{m-1} 1, 0^n 1)$ . Aceasta se face definind regulile:

$$\begin{aligned} \delta(q_0, 0) &= (q_6, B, d) & \delta(q_6, 0) &= (q_6, 0, d) \\ \delta(q_6, 1) &= (q', 1, d) & \delta(q', 0) &= (q', 0, d) \\ \delta(q', B) &= (q'', 1, s) & \delta(q'', 0) &= (q'', 0, s) \\ \delta(q'', 1) &= (q_1, 1, d) \end{aligned}$$

Alte stări suplimentare sunt necesare pentru a transforma o descriere instantanee  $(q_5, B^i 0^{m-i} 1, 0^n 10^{ni})$  în  $(q_1, B^{i+1} 0^{m-i-1} 1, 0^n 10^{ni})$  (după care MT acționează **COPY**), și de a verifica dacă  $i = m$ . În această ultimă situație, primii  $10^n 1$  sunt șterși și mașina se oprește în starea  $q_{12}$ .

Aceste mișcări sunt descrise în tabelul:

	0	1	2	B
$q_5$	$(q_7, 0, s)$	—	—	—
$q_7$	—	$(q_8, 1, s)$	—	—
$q_8$	$(q_9, 0, s)$	—	—	$(q_{10}, B, d)$
$q_9$	$(q_9, 0, s)$	—	—	$(q_0, B, d)$
$q_{10}$	—	$(q_{11}, B, d)$	—	—
$q_{11}$	$(q_{11}, B, d)$	$(q_{12}, B, d)$	—	—

## 8.6 Modificări ale mașinilor Turing

Una din rațiunile care au condus la acceptarea mașinii Turing ca model general de calcul este echivalența ei cu multe alte versiuni care creează diverse facilități de lucru. Vom trece în revistă câteva din astfel de modalități de definire ale mașinilor Turing.

### Mașină Turing cu bandă dublu infinită

O mașină Turing  $M = (Q, V, \Gamma, \delta, q_0, B, F)$  cu bandă dublu infinită are proprietatea că banda de citire este infinită și la stânga (nu numai la dreapta). Se consideră că prelungirile porțiunii nevide ale benzii, în ambele direcții, sunt formate din două infinități de blancuri.

Relația  $\stackrel{M}{\vdash}$  se definește la fel ca la mașinile Turing clasice, cu două completări:

1. dacă  $\delta(q, X) = (p, Y, s)$  atunci  $(q, \epsilon, X\alpha) \stackrel{M}{\vdash} (p, \epsilon, BY\alpha)$   
(în modelul inițial nu exista nici o mișcare în această situație);
2. dacă  $\delta(q, X) = (p, B, d)$  atunci  $(q, \epsilon, X\alpha) \stackrel{M}{\vdash} (p, B, \alpha)$ .

Configurația inițială este  $(q_0, \epsilon, w)$ .

**Teorema 8.5** *Limbaajul  $L$  este acceptat de o MT cu bandă dublu infinită dacă și numai dacă este acceptat de o MT cu bandă infinită la dreapta.*

*Demonstrație:* Faptul că o mașină Turing cu bandă dublu infinită poate simula o MT obișnuită, este simplu de arătat. La început se marchează celula aflată la stânga poziției



stare și de simbolul analizat de fiecare cap de citire, mașina poate:

- să schimbe starea;
- să tipărească un simbol nou pe fiecare din cele  $n$  benzi;
- să deplaseze independent fiecare cap de bandă (dreapta/stânga sau să-l lase pe loc).

Inițial, intrarea se află pe prima bandă, celelalte fiind goale (numai  $B$ ).

**Teorema 8.6** *Dacă un limbaj  $L$  este acceptat de o  $MT$  multibandă, atunci el este acceptat de o  $MT$  cu o bandă.*

*Demonstrație:* Teorema prezintă numai o implicație, reciproca fiind banală (orice  $MT$  poate fi privită ca o  $MT$  cu  $n = 1$  benzi).

Fie limbajul  $L$  acceptat de  $MT$   $M_1$  cu  $n$  benzi. Vom construi o mașină Turing  $M_2$  cu stocare multiplă pe  $2n$  benzi paralele, câte două pentru fiecare bandă a lui  $M_1$ : una din componente va reține conținuturile benzii corespunzătoare a lui  $M_1$  iar cealaltă are numai blancuri, înafara unui marcator în celula care conține simbolul scanat de capul de citire corespunzător din  $M_1$ . Controlul finit al lui  $M_2$  stochează starea lui  $M_1$  împreună cu numărul de marcatori aflați la dreapta capului de citire din  $M_2$ .

Fiecare mișcare a lui  $M_1$  este simulată de o parcurgere dreapta - stânga - dreapta a capului de citire a lui  $M_2$ .

Inițial, acest cap este poziționat la cea mai din stânga celulă care conține un marcator.

Pentru simularea unei mișcări a lui  $M_1$ ,  $M_2$  merge spre dreapta, vizitează fiecare celulă semnalată de un marcator și înregistrează fiecare simbol prelucrat de capul de citire respectiv din  $M_1$ . Când  $M_2$  întâlnește un marcator, contorul care înregistrează numărul de marcatori aflați la dreapta capului de citire, scade cu o unitate.

Dacă acest contor ajunge la 0,  $M_2$  începe mișcarea spre stânga, către primul marcator. La această revenire, la fiecare marcator,  $M_2$  efectuează pe banda paralelă operațiile date de  $M_1$  pe banda respectivă (inclusiv eventuala mișcare a marcatorului).

În final,  $M_2$  schimbă starea în starea lui  $M_1$ . Dacă aceasta este de acceptare, atunci și  $M_2$  se va opri cu acceptare. q.e.d

Justificarea utilizării unei  $MT$  multibandă constă în simplitatea ei: pentru  $n$  mișcări pe benzile lui  $M_1$  sunt necesare aproximativ  $2n^2$  mișcări pe  $MT$   $M_2$  care o simulează.

**Exemplul 8.8** *Limbajul  $L = \{w\tilde{w} \mid w \in \{a,b\}^*\}$  poate fi acceptat de o  $MT$  cu o bandă care mișcă capul de citire înainte și înapoi pe banda de intrare, controlând simbolurile de la ambele capete. Pentru o  $MT$  cu două benzi, cuvântul de intrare este copiat pe a doua bandă; apoi, la fiecare mișcare, cele două capete – aflate la cele două extremități – încep să se miște unul spre altul comparând intrările.*

*Deci, pentru  $\alpha \in L$ ,  $|\alpha| = k$ , numărul de mișcări pentru prima mașină Turing este aproximativ  $k^2$ , iar pentru a doua – numai  $k$ .*

### Mașini Turing nedeterministe și deterministe

Conform definiției, în general o mașină Turing este nedeterministă: pentru o stare dată și un simbol citit pe bandă, mașina are la dispoziție un număr finit de posibilități de continuare. Fiecare alegere constă dintr-o stare, un simbol de bandă (pentru tipărit) și o direcție de mișcare a capului de bandă. Cuvântul de intrare este acceptat dacă există o secvență de mișcări prin care se ajunge la o stare de acceptare.

Ca și la automatele finite, folosirea *MT* nedeterministe nu duce la o capacitate de acceptare sporită față de *MT*-urile deterministe.

**Teorema 8.7** *Dacă  $L$  este un limbaj acceptat de o MT nedeterministă  $M_1$ , atunci  $L$  este acceptat de o MT deterministă  $M_2$ .*

*Demonstrație:* Pentru orice stare și simbol de intrare al lui  $M_1$ , există un număr finit de alegeri pentru mișcarea următoare; să numerotăm aceste alegeri cu  $1, 2, \dots$ . Fie  $r$  numărul maxim de alegeri pentru orice variantă (stare, simbol de intrare). Atunci orice secvență finită de alegeri se poate reprezenta ca un cuvânt peste alfabetul  $V = \{1, 2, \dots, r\}$ .

Bineînțeles, nu orice cuvânt va reprezenta o secvență de mișcări (unele situații pot permite mai puține variante).

$M_2$  va avea 3 benzi.

Pe prima bandă se introduce cuvântul de intrare.

Pe a doua bandă,  $M_2$  generează – în ordine crescătoare – cuvinte peste  $V$ . Pentru fiecare astfel de cuvânt generat,  $M_2$  va copia pe banda 3 cuvântul de intrare și va simula  $M_1$  pe această bandă folosind secvența de pe banda 2 pentru a dicta mișcările lui  $M_1$ ; dacă  $M_1$  ajunge la o stare de acceptare,  $M_2$  va accepta de asemenea.

Dacă există o secvență de mișcări care duce la acceptarea cuvântului, ea este un cuvânt din  $L$ , deci va fi generată pe banda 2.

Dacă nici o secvență de mișcări ale lui  $M_1$  nu duce la acceptare, nici  $M_2$  nu va accepta cuvântul de intrare. q.e.d

### Mașini Turing multidimensionale

Ca o altă variantă, să considerăm o *MT* a cărei bandă este (pentru un  $n$  fixat), un tablou  $n$  - dimensional de celule, infinit în toate cele  $2n$  direcții. În funcție de stare și de simbolul scanat, se schimbă starea, se tipărește un nou simbol și se mișcă capul de citire în una din cele  $2n$  direcții, pe una din cele  $n$  axe. Intrarea este inițială pe una din axe, cu capul de citire poziționat la stânga.

La orice moment, numai un număr finit de linii din orice dimensiune conțin elemente (diferite de blanc) iar numărul acestor elemente este finit.

**Teorema 8.8** *Dacă  $L$  este acceptat de o MT 2 - dimensională  $M_2$ , atunci  $L$  este acceptat de o MT  $M_1$  de dimensiune 1.*

*Demonstrație:* Generalizarea teoremei la mai multe dimensiuni este imediată.

$M_1$  va simula banda lui  $M_2$  ca în exemplul de mai jos:

$$\begin{array}{ccccccc}
 B & B & B & a_1 & B & B & B \\
 B & B & a_2 & a_3 & a_4 & a_5 & B \\
 a_6 & a_7 & a_8 & a_9 & B & a_{10} & B \\
 B & a_{11} & a_{12} & a_{13} & B & a_{14} & a_{15} \\
 B & B & a_{16} & a_{17} & B & B & B
 \end{array}$$

$M_2$

$$**BBBa_1BBB*BBa_2a_3a_4a_5B*a_6a_7a_8a_9Ba_{10}B*Ba_{11}a_{12}a_{13}Ba_{14}a_{15}*BBa_{16}a_{17}BBB**$$

$$M_1$$

Înafara unei benzi dublu infinite,  $M_1$  dispune și de o a doua bandă.

Să presupunem că  $M_2$  face o mișcare în care capul de citire nu părăsește dreptunghiul reprezentat pe bandă.

Dacă mișcarea este orizontală, atunci  $M_1$  își mișcă pur și simplu capul de citire cu o poziție spre stânga sau dreapta, schimbă starea și tipărește (eventual) un simbol; totul similar lui  $M_2$ .

Dacă mișcarea este verticală,  $M_1$  folosește a doua bandă pentru a reține poziția capului de citire față de cel mai din stânga caracter \* anterior; apoi el traversează spre dreapta (dacă mișcarea este în jos) sau stânga (dacă trebuie mers în sus) un caracter \*, după care re poziționează capul de citire conform informației reținute pe banda a doua.

În cazul în care capul lui  $M_2$  se deplasează înafara dreptunghiului reprezentat pentru  $M_1$ : dacă mișcarea este verticală, se adaugă un nou bloc de blancuri la dreapta sau stânga, folosind banda a doua pentru a număra lungimea curentă a blocurilor. Dacă mișcarea este orizontală,  $M_1$  folosește tehnica de mărire a spațiului de lucru pentru a adăuga un blank la capul din stânga sau dreapta al fiecărui bloc.

Sfârșitul blocurilor este detectat prin ajungerea la \*\*.

q.e.d.

### Restricții

După cum am văzut, toate aceste încercări de generalizare ale mașinilor Turing nu au reușit să mărească capacitatea lor de acceptare; se poate raționa și invers: este posibil să construim modele restrictive de MT care să păstreze de asemenea clasa  $\mathcal{L}_0$  de limbaje acceptate. Dintre diversele modalități de restricție avute în vedere în literatură, ne vom

referi cu precădere la acelea care limitează numărul de stări sau de simboluri de bandă. Un exemplu de rezultat interesant în această direcție este următorul: *dacă nu există nici o restricție asupra alfabetului de bandă, atunci 3 stări și o bandă sunt suficiente pentru a accepta orice limbaj recursiv numărabil*. (demonstrația acestei aserțiuni este lăsată ca exercițiu).

Relativ la restricțiile alfabetelor de bandă, putem da următoarea teoremă:

**Teorema 8.9** *Dacă  $L \subseteq \{0, 1\}^*$  este un limbaj recursiv numărabil, atunci  $L$  este acceptat de o MT cu o bandă, având alfabetul de bandă  $\{0, 1, B\}$ .*

*Demonstrație:* Fie  $L = L(M_1)$  unde  $M_1 = (Q, \{0, 1\}, \Gamma, \delta, q_0, B, F)$ . Să presupunem  $2^{n-1} < |\Gamma| \leq 2^n$ ; în acest fel orice simbol de bandă poate fi codificat folosind  $n$  biți.

Vom construi MT  $M_2$  cu alfabetul de bandă  $\{0, 1, B\}$  care să simuleze  $M_1$ . Banda lui  $M_2$  conține o secvență de coduri pentru simbolurile lui  $M_1$ ; controlul lui  $M_2$  reține stările lui  $M_1$  precum și poziția capului de citire al lui  $M_2$  modulo  $n$ , așa că  $M_2$  poate ști când se află la începutul unui simbol de bandă codificat.

Comportarea lui  $M_2$  este simplu de detaliat.

Trebuie clarificată o singură situație; anume aceea că un cuvânt de intrare  $w$  este o secvență binară care reprezintă exact  $w$  – și nu o codificare a sa. Deci, înainte de a simula  $M_1$ ,  $M_2$  trebuie să înlocuiască  $w$  cu codificarea sa, ceea ce se poate realiza prin mărirea spațiului de lucru folosind simbolul  $B$ ; anume, pentru fiecare simbol de intrare – începând cu cel mai din stânga – secvența aflată la dreapta simbolului este deplasată spre dreapta cu  $n - 1$  poziții, în locul lor fiind introdus  $B$  (deci fiecare simbol  $a \in \{0, 1\}$  este înlocuit cu secvența  $aB^{n-1}$ ). q.e.d.

Putem aplica aceeași tehnică de codificare binară și în cazul în care alfabetul de intrare nu este  $\{0, 1\}$ ; se obține:

**Corolarul 8.1** *Dacă  $L$  este un limbaj recursiv numărabil, atunci el este acceptat de o mașină Turing liniară cu o bandă și alfabetul  $V = \{0, 1, B\}$ .*

**Teorema 8.10** *Orice mașină Turing poate fi simulată de o mașină liniară cu o bandă de stocare având simbolurile 0 (pentru blank) și 1. MT poate tipări 0 sau 1 peste 0 dar nu poate tipări 0 peste 1.*

*Demonstrație:* Ideea este de a crea descrierile instantanee succesive al mașinii Turing originale pe banda unei alte MT. Simbolurile de bandă sunt codificate. Fiecare descriere instantanee este copiată peste cea veche. q.e.d



## 8.7 Exerciții

**Exercițiul 8.1** Fie mașina Turing având funcția de tranziție definită

$\delta$	$B$	$a$	$b$	$c$
$q_0$	$(q_1, B, d)$	—	—	—
$q_1$	$(q_2, B, s)$	$(q_1, a, d)$	$(q_1, c, d)$	$(q_1, c, d)$
$q_2$	—	$(q_2, c, s)$	—	$(q_2, b, s)$

Descrieți funcționarea mașinii pentru secvențele de intrare aacba și bcbc.

**Exercițiul 8.2** Fie mașina Turing având funcția de tranziție definită

$\delta$	$B$	$a$	$b$	$c$
$q_0$	$(q_1, B, d)$	—	—	—
$q_1$	$(q_2, B, d)$	$(q_1, a, d)$	$(q_1, b, d)$	$(q_1, c, s)$
$q_2$	—	$(q_2, b, s)$	—	$(q_2, a, s)$

Descrieți funcționarea mașinii pentru secvențele de intrare abcab și abab.

**Exercițiul 8.3** Construiți o MT cu alfabetul  $\Sigma = \{a, b, c\}$  care acceptă șiruri din  $\Sigma^+$  cu cel puțin un 'c', în care primul caracter 'c' este precedat de șirul aaa.

**Exercițiul 8.4** Construiți o MT peste  $\Sigma = \{a, b\}$  care acceptă limbajul  
 $L = \{ab^{n_1}ab^{n_2} \dots ab^{n_k} \mid k > 0, n_1 < n_2 < \dots < n_k\}.$

**Exercițiul 8.5** Detaliați construcția MT din Teorema 8.2.

**Exercițiul 8.6** Demonstrați că orice limbaj recursiv numărabil este acceptat de o mașină Turing cu o singură stare finală.

**Exercițiul 8.7** Generați mașini Turing pentru următoarele limbaje:

$$L_1 = \{1, 2, \dots, n\}, \quad L_2 = \{a^i \mid i \text{ divizibil cu } 3\}, \quad L_3 = \{ww \mid w \in \{a, b\}^+\}$$

**Exercițiul 8.8** Fie  $L$  un limbaj. Dacă  $L$  și  $\bar{L}$  sunt recursiv numărabile, atunci  $L$  este recursiv.

**Exercițiul 8.9** Demonstrați că orice limbaj independent de context este recursiv.

# Bibliografie

- [1] Aho, A.V., Ullman, J.D. - *The Theory of Parsing, Translation and Compiling*, Prentice Hall, Englewood Cliffs, N.J. 1973;
- [2] Aho, A.V., Ulmann, J.D. - *Principles of Compiler Design*, Addison - Wesley, Reading, Mass., 1977;
- [3] Aho, A.V., Sethi R., Ullman, J.D. - *Compilers; Principles, Techniques and Tools*, Addison - Wesley, Reading, Mass., 1986;
- [4] Atanasiu, A. - *Bazele Informaticii pentru anul II seral*, Tipografia Universității București, 1987;
- [5] Atanasiu, A. - *Automate și translatori*, fascicula I, 1995;
- [6] Atanasiu, A. - *Arhitectura calculatoarelor*, InfoData, Cluj, 2006
- [7] Hopcroft, J.E., Ullman, J.D. - *Formal Languages and their relation to Automata*, Addison - Wesley, Reading, Mass. 1969
- [8] Hopcroft, J.E., Ullman, J.D. - *Introduction to Automata Theory, Languages and Computation*, Addison - Wesley, Reading, Mass. 1979
- [9] Jucan, T. - *Limbaje formale și automate*, Ed. Matrix Rom, București 1999
- [10] Linz, P. - *An Introduction to Formal languages and Automata*, Jones and Bartlett Publ., 2001
- [11] Păun, Gh. - *Mecanisme generative ale proceselor economice*, Ed. Tehnică, 1980;
- [12] Păun, Gh. - *Probleme actuale în teoria limbajelor formale*, Ed. Științifică și Enciclopedică, București 1984;
- [13] Rozenberg, G., Salomaa, A - *Handbook of Formal Languages*, vol 1, Springer Verlag, 1997

- [14] Salomaa, A - *Formal Languages*, Academic Press, N.Y. 1973
- [15] Sipser, M. - *Introduction to the Theory of Computation*, Intern. Thomson Publ. 1997
- [16] Sudkamp, Th. - *Languages and Machines*, Addison Wesley, 1997