

Limbaje formale

Mircea Drăgan

Ștefan Mărușter

February 23, 2005

Bibliografie

1. Octavian C. Dogaru, *Bazele informaticii. Limbaje formale*, Tipografia Universității din Timișoara, 1989.
2. Gheorghe Grigoraș, *Limbaje formale și tehnici de compilare*, Tipografia Universității "Alexandru Ioan Cuza", Iași, 1984.
3. J. E. Hopcroft și J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Reading Mass., 1979.
4. Solomon Marcus, *Gramatici și automate finite*, Editura Academiei, București, 1964.
5. Ștefan Mărușter, *Curs de Limbaje formale și tehnici de compilare*, Tipografia Universității din Timișoara, 1980.
6. Gheorghe Orman, *Limbaje formale*, Editura tehnică, București, 1982.
7. Gheorghe Păun, *Probleme actuale în teoria Limbajelor formale*, Editura Științifică și Enciclopedică, București, 1984.
8. Teodor Rus, *Mecanisme formale pentru specificarea limbajelor*, Editura Academiei, București, 1983.
9. Arto Salomaa, *Formal languages*, Academic Press, New York, 1973.
10. Dan Simovici, *Limbaje formale și tehnici de compilare*, Editura didactică și pedagogică, București, 1978.
11. Luca Dan Șerbănați, *Limbaje de programare și compilatoare*, Editura Academiei, București, 1987.

Capitolul 1

Limbaje formale

1.1 Limbaje, gramatici, proprietăți generale

Noțiunea generală de limbaj. Vom considera o mulțime finită și nevidă V , numită *alfabet* sau *vocabular*. Elementele mulțimii le vom numi *simboluri* (sau litere, caractere, variabile).

Definiție 1.1 *Un cuvânt peste un alfabet V este un șir $p = a_1a_2\dots a_n$, $a_i \in V, i = \overline{1, n}$.*

Numărul n , deci numărul simbolurilor cuvântului p , se numește *lungimea cuvântului* și va fi notat cu $|p|$ sau $l(p)$. Vom considera și *cuvântul vid* λ sau ϵ , care nu conține nici un simbol; evident $|\lambda| = 0$. Un rol deosebit îl are simbolul *blanc* notat cu b (bnu coincide cu λ) precum și simbolul marcaj notat $\#$ pentru delimitarea cuvintelor. Noțiunea de "cuvânt" este fundamentală în teoria limbajelor formale sau în alte domenii ale informaticii; termeni sinonimi utilizați în literatură sunt propoziție, frază sau șir. Să observăm că nu există o similitudine foarte bună între noțiunile de "alfabet", "cuvânt", etc. din teoria limbajelor formale și noțiunile corespunzătoare din lingvistică.

Mulțimea tuturor cuvintelor peste un alfabet V o notăm cu V^+ . Această mulțime împreună cu cuvântul vid va fi notată cu V^* .

În general, vom utiliza litere mari de la sfârșitul alfabetului pentru notarea diverselor alfabete, U, V, W , etc.; litere de la începutul alfabetului (mari sau mici) pentru notarea simbolurilor, $A, B, C, \dots, a, b, c, \dots, i, j, \dots$ (uneori cifre $0, 1, 2, \dots$); pentru notarea cuvintelor vom utiliza litere mici de la sfârșitul alfabetului, $p, q, r, s, t, u, v, w, x, y, z$, (această convenție de notare nu va fi absolută).

Fie $p = a_1 \dots a_n$, $q = b_1 \dots b_m$. Definim pe mulțimea V^* operația de concatenare sau juxtapunere prin $pq = a_1 \dots a_n b_1 \dots b_m$.

Se poate verifica ușor că această operație este asociativă. Prin urmare, mulțimea V^+ înzestrată cu această operație este un semigrup (*semigrupul liber peste V*). Mulțimea V^* cu aceeași operație este un semigrup cu unitate, deci un monoid, unitatea fiind cuvântul vid (*monoidul liber peste V*).

Fie din nou $p, q \in V^*$. Vom spune că q este un subcuvânt sau un *infix* (eventual propriu) al lui p dacă $p = uqv$, $u, v \in V^*$ (eventual $u, v \in V^+$); q este *prefix* (propriu) al lui p dacă $p = qv$, $v \in V^*$ ($v \in V^+$); q este *sufix* (propriu) al lui p dacă $p = uq$, $u \in V^*$ ($u \in V^+$).

Definiție 1.2 *Un limbaj L peste un alfabet V este o parte a monoidului liber V^* , deci $L \subseteq V^*$.*

Să observăm că V^* (sau V^+) este întotdeauna o mulțime infinită (evident numărabilă); în această accepțiune generală, un limbaj poate să fie o mulțime finită sau infinită, uneori chiar vidă.

Exemplu. Fie $V = \{0, 1\}$. Avem

$$V^+ = \{0, 1, 00, 01, 10, 000, \dots\},$$

$$V^* = \{\lambda, 0, 1, 00, 01, 10, 000, \dots\}.$$

Limbafe peste alfabetul V sunt de exemplu mulțimile

$$L_1 = \{\lambda, 00, 11\},$$

$$L_2 = \{1, 11, 111, \dots\} = \{1^n \mid n \geq 1\}.$$

Observație. Notăția a^n , unde a este un simbol al unui alfabet, înseamnă cuvântul constituit din n simboluri a , adică $\underbrace{aa \dots a}_n$. În particular $a^0 = \lambda$.

Operații cu limbaje. Limbajele fiind mulțimi se pot efectua cu limbaje operațiile obișnuite cu mulțimi: reuniune, intersecție, diferență, complementariere (față de V^*). Există și operații specifice limbajelor:

- *Produsul* (concatenarea) a două limbaje definit prin

$$L_1 L_2 = \{pq \mid p \in L_1, q \in L_2\}.$$

Dacă $L_1 = L_2 = L$ vom nota $LL = L^2$. Prin recurență, se definește L^n astfel

$$L^0 = \{\lambda\}, L^k = L^{k-1}L, k \geq 1.$$

- Închiderea (Kleene) a unui limbaj L este

$$L^* = \bigcup_{k=0}^{\infty} L^k.$$

În general, o operație de n -aritate oarecare (cel mai adesea binară sau unară) pe mulțimea V^* definește o operație corespunzătoare pe mulțimea limbajelor. Astfel, dacă

$$\alpha : V^* \longrightarrow \mathcal{P}(V^*) \text{ și } \beta : V^* \times V^* \longrightarrow \mathcal{P}(V^*)$$

sunt două operații pe V^* (unară și respectiv binară) și L_1, L_2 sunt două limbaaje peste V , putem defini limbaajele $\alpha(L_1)$ respectiv $\beta(L_1, L_2)$ prin

$$\alpha(L_1) = \bigcup_{x \in L_1} \alpha(x), \quad \beta(L_1, L_2) = \bigcup_{x \in L_1, y \in L_2} \beta(x, y).$$

Exemple:

1. Limbajul $Sub(L)$. Fie V^* și $Sub(x)$ mulțimea tuturor subcuvintelor lui x (evident Sub este o operație unară pe V^*). Dacă L este un limbaj peste V , putem defini limbajul

$$Sub(L) = \bigcup_{x \in L} Sub(x).$$

adică limbajul constituit din toate subcuvintele tuturor cuvintelor lui L . Semnificații analoage vor avea și limbaajele $Pref(L)$ și $Suf(L)$.

2. Limbajul $Mi(L)$. Fie $x = a_1 \dots a_n$ un cuvânt peste alfabetul V . Cuvântul $Mi(x) = a_n \dots a_1$ se numește răsturnatul lui x (Mi este prescurtarea cuvântului englez *mirror*). Se mai notează $Mi(x) = \tilde{x}$. Avem atunci și răsturnatul unui limbaj

$$Mi(L) = \bigcup_{x \in L} Mi(x).$$

3. Operația de substituție. Fie U și V două alfabete și fie aplicația $s : V \longrightarrow \mathcal{P}(U^*)$. Extindem (prelungim) această aplicație la V^* prin

$$s(\lambda) = \{\lambda\}, s(xy) = s(x)s(y), x, y \in V^*.$$

O astfel de prelungire se numește *canonică*; ea păstrează operația de concatenare, în sensul că dacă $p = xy$, atunci $s(p) = s(x)s(y)$ este concatenarea limbajelor $s(x), s(y)$. Operația de substituție a limbajelor este dată de

$$s(L) = \bigcup_{x \in L} s(x).$$

Să observăm că această operație transformă un limbaj peste un alfabet V într-un limbaj peste un alfabet U și că păstrează operația de concatenare. Dacă $\text{card } s(a) < \infty, \forall a \in V$, vom spune că substituția este *finită*, iar dacă $\text{card } s(a) = 1, \forall a \in V$ vom spune că s este un *homomorfism*.

Gramatici generative de tip Chomsky. Un limbaj peste un alfabet poate să fie o mulțime finită sau infinită. Dacă este o mulțime finită, el poate fi definit prin scrierea efectivă a cuvintelor limbajului. În cazul în care este o mulțime infinită, el poate fi definit în anumite cazuri punând în evidență structura cuvintelor lui. De exemplu

$$L_2 = \{01, 0011, 000111, \dots\} = \{0^n 1^n \mid n \geq 1\}.$$

Există două procedee mai generale pentru definirea limbajelor:

1. Procedee *generative*, care permit generarea tuturor cuvintelor limbajului. Există mai multe tipuri de mecanisme de generare a limbajelor, între care gramaticile Chomsky, sisteme Lindenmayer, etc.
2. Procedee *analitice*, care determina dacă un cuvânt dat aparține sau nu limbajului. Sunt așa-numitele *automate*, *automate finite*, *automate push-down*, etc.

Un rol deosebit în teoria limbajelor formale îl au gramaticile Chomsky. Fie V_N și V_T două alfabete disjuncte, numite respectiv alfabetul simbolurilor neterminale (V_N) și alfabetul simbolurilor terminale (V_T). Notăm $V_G = V_N \cup V_T$ (alfabetul general) și fie $P \subseteq V_G^* V_N V_G^* \times V_G^*$. Mulțimea P va fi deci formată din perechi de forma (u, v) , unde $u = u' A u''$, $u', u'' \in V_G^*$, $A \in V_N$ iar $v \in V_G^*$, deci u și v sunt cuvinte peste V_G , cu observația că u trebuie să conțină cel puțin un simbol neterminale. Vom spune că o astfel de pereche este o **regulă** (producție, regulă de generare, regulă de rescriere) și o vom nota $u \rightarrow v$ (vom spune: u se transformă în v). Apartenența unei reguli la P o vom nota în mod obișnuit $(u \rightarrow v) \in P$, sau mai simplu, $u \rightarrow v \in P$ (nu va exista confuzia cu faptul că v este un element al lui P).

Definiție 1.3 O gramatică (Chomsky) G este un sistem $G = (V_N, V_T, S, P)$, unde V_N este alfabetul simbolurilor neterminale, V_T este alfabetul simbolurilor terminale, $S \in V_N$ și se numește simbolul de start al gramaticii, iar P este mulțimea de reguli.

Observație. Simbolurile alfabetului V_N le vom nota în general cu litere mari A, B, C, \dots, X, Y, Z (mai puțin U, V, W) iar cele ale alfabetului V_T cu litere mici de la început a, b, c, \dots sau cu cifre $0, 1, 2, \dots$.

Fie G o gramatică și $p, q \in V_G^*$. Vom spune că p se transformă direct în q și vom scrie $p \xrightarrow{G} q$ (sau $p \Rightarrow q$ dacă nu există nici o confuzie) dacă există $r, s, u, v \in V_G^*$ astfel încât $p = rus, q = rvs$ iar $u \rightarrow v \in P$. Vom spune că p' se transformă în p'' (fără specificația direct) dacă există $p_1, p_2, \dots, p_n, n \geq 1$ astfel încât

$$p' = p_1 \xrightarrow{G} p_2 \xrightarrow{G} \dots \xrightarrow{G} p_n = p''.$$

Vom scrie $p' \xrightarrow{+} p''$ (sau $p' \xrightarrow{\pm} p''$ când nu există nici o confuzie) dacă $n > 1$ și $p' \xrightarrow{*} p''$ (sau $p' \xrightarrow{\pm} p''$) dacă $n \geq 1$. Șirul de mai sus va fi numit **derivare** iar numărul de derivări directe din șir îl vom numi **lungimea** derivării; se mai spune că p' derivă în p'' .

Să observăm că transformările astfel definite $\Rightarrow, \xrightarrow{\pm}, \xrightarrow{*}$ sunt relații pe V_G^* . Este clar că relația $\xrightarrow{\pm}$ este închiderea tranzitivă a relației \Rightarrow , iar relația $\xrightarrow{*}$ este închiderea tranzitivă și reflexivă a relației de transformare directă.

Definiție 1.4 . Limbajul generat de gramatica G este prin definiție limbajul

$$L(G) = \{p \mid p \in V_T^*, S \xrightarrow{*}_G p\}.$$

Observație. Dacă $p \in V_G^*$ și $S \xrightarrow{*}_G p$ se spune că p este o *formă propozițională* în gramatica G .

Exemple:

1. Fie $G = (V_N, V_T, S, P)$, unde $V_N = \{A\}$, $V_T = \{0, 1\}$, $S = A$ (evident) și $P = \{A \rightarrow 0A1, A \rightarrow 01\}$. O derivare în această gramatică este, de exemplu

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000111 = 0^31^3.$$

Este evident că $L(G) = \{0^n1^n \mid n \geq 1\}$.

Observație. În cazul în care mai multe reguli au aceeași parte stângă, le vom scrie compact astfel $u \rightarrow v_1|v_2|\dots|v_n$, simbolul $|$ având sensul de sau; în cazul nostru, $A \rightarrow 0A1|01$.

2. $G = (V_N, V_T, S, P)$, unde
- $$V_N = \{ \langle \text{propoziție} \rangle, \langle \text{subiect} \rangle, \langle \text{atribut} \rangle, \langle \text{predicat} \rangle, \langle \text{complement} \rangle, \langle \text{substantiv} \rangle, \langle \text{adjectiv} \rangle, \langle \text{verb} \rangle, \langle \text{articol} \rangle \},$$
- $$V_T = \{ \text{o, orice, matrice, funcție, derivabilă, continuă, este} \},$$
- $$S = \langle \text{propoziție} \rangle,$$
- $$P = \{ \langle \text{propoziție} \rangle \rightarrow \langle \text{subiect} \rangle \langle \text{atribut} \rangle \langle \text{predicat} \rangle \langle \text{complement} \rangle, \langle \text{subiect} \rangle \rightarrow \langle \text{articol} \rangle \langle \text{substantiv} \rangle, \langle \text{atribut} \rangle \rightarrow \langle \text{adjectiv} \rangle, \langle \text{predicat} \rangle \rightarrow \langle \text{verb} \rangle, \langle \text{complement} \rangle \rightarrow \langle \text{adjectiv} \rangle, \langle \text{articol} \rangle \rightarrow \text{o} | \text{orice}, \langle \text{substantiv} \rangle \rightarrow \text{matrice} | \text{funcție}, \langle \text{adjectiv} \rangle \rightarrow \text{derivabilă} | \text{continuă}, \langle \text{verb} \rangle \rightarrow \text{este}.$$

Observație. În acest exemplu, " $\langle \text{propoziție} \rangle$ ", " $\langle \text{subiect} \rangle$ ", etc., reprezintă fiecare câte un simbol neterminale; de asemenea, "o", "orice", "matrice", etc., reprezintă simboluri terminale.

Se poate ușor observa că această gramatică generează propoziții simple de forma subiect-atribut-predicat-complement care exprimă judecăți asupra conceptelor de "matrice" și "funcție". De exemplu, se poate forma propoziția: "orice funcție derivabilă este continuă", care este din punct de vedere semantic corectă, precum și propoziția "orice funcție continuă este derivabilă", care, după cum se știe, este falsă. Evident, se pot genera și numeroase propoziții care nu au sens. Ceea ce ne interesează în acest moment este aspectul formal, deci din punct de vedere sintactic toate aceste propoziții sunt corecte; semantic, unele propoziții pot să fie incorecte sau chiar să nu aibe sens.

Să mai observăm că o gramatică Chomsky este în măsură să constituie un model matematic pentru sintaxa unei limbi, fără să intereseze aspectele semantice. Este ceea ce a încercat să întreprindă Naom Chomsky pentru limba engleză în lucrările sale din anii 1957.

3. $G = (V_N, V_T, A_0, P)$, unde
- $$V_N = \{ \langle \text{program} \rangle, \langle \text{instrucție} \rangle, \langle \text{atribuire} \rangle, \langle \text{if} \rangle, \langle \text{expresie} \rangle, \langle \text{termen} \rangle, \langle \text{factor} \rangle, \langle \text{variabilă} \rangle, \langle \text{index} \rangle \},$$
- $$V_T = \{ \text{begin, end, if, then, stop, t, i, +, *, (,), =, ,, ;} \},$$
- $$A_0 = \langle \text{program} \rangle$$
- $$P = \{ \langle \text{program} \rangle \rightarrow \text{begin} \langle \text{linie} \rangle \text{end} \}$$
- $$\langle \text{linie} \rangle \rightarrow \langle \text{linie} \rangle ; \langle \text{instrucție} \rangle \mid \langle \text{instrucție} \rangle$$
- $$\langle \text{instrucție} \rangle \rightarrow \langle \text{atribuire} \rangle \mid \langle \text{if} \rangle \mid \text{stop}$$
- $$\langle \text{atribuire} \rangle \rightarrow \langle \text{variabilă} \rangle = \langle \text{expresie} \rangle$$

$\langle \text{if} \rangle \rightarrow \text{if}(\langle \text{expresie} \rangle) \text{ then } \langle \text{atribuire} \rangle$
 $\langle \text{expresie} \rangle \rightarrow \langle \text{expresie} \rangle + \langle \text{termen} \rangle \mid \langle \text{termen} \rangle$
 $\langle \text{termen} \rangle \rightarrow \langle \text{termen} \rangle^* \langle \text{factor} \rangle \mid \langle \text{factor} \rangle$
 $\langle \text{factor} \rangle \rightarrow (\langle \text{expresie} \rangle) \mid \langle \text{variabilă} \rangle$
 $\langle \text{variabilă} \rangle \rightarrow t(\langle \text{index} \rangle) \mid i$
 $\langle \text{index} \rangle \rightarrow \langle \text{index} \rangle, \langle \text{expresie} \rangle \mid \langle \text{expresie} \rangle$

Gramatica din acest exemplu definește un limbaj de programare simplu cu trei tipuri de instrucții: atribuirii, if-then, stop. Expresiile aritmetice au numai operatorii + și * iar variabilele pot fi simple sau indexate (tablouri). Menționăm că definirea în acest mod a unui limbaj, inclusiv utilizarea croșetelor pentru desemnarea simbolurilor neterminale, poartă adesea denumirea de notație Backus Naur; în acest mod s-a definit limbajul ALGOL.

Tipuri de gramatici. După forma regulilor de generare, gramaticile Chomsky se împart în mai multe tipuri; clasificarea obișnuită este următoarea:

- *Gramatici de tipul 0*; sunt gramatici fără restricții asupra regulilor;
- *Gramatici de tipul 1 (dependente de context)*; sunt gramatici care au reguli de forma

$$uAv \rightarrow upv, \quad u, p, v \in V_G^*, \quad p \neq \lambda, \quad A \in V_N$$

sau $A \rightarrow \lambda$ și în acest caz $A \in V_N$ nu apare în dreapta vreunei reguli.
Observație. Evident, regulile de forma a doua au sens numai dacă A este simbolul de start.

- *Gramaticile de tipul 2 (independente de context)*; sunt gramatici care au reguli de forma

$$A \rightarrow p, \quad A \in V_N, \quad p \in V_G^*.$$

- *Gramaticile de tipul 3 (regulate)*; sunt gramatici care au reguli de forma

$$\begin{cases} A \rightarrow Bp \\ C \rightarrow q \end{cases} \quad \text{sau} \quad \begin{cases} A \rightarrow pB \\ C \rightarrow q \end{cases}$$

cu $A, B, C \in V_N$ și $p, q \in V_T^*$.

Vom nota cu L_j , $j = 0, 1, 2, 3$ familiile de limbaie generate de gramaticile de tipurile $j = 0, 1, 2, 3$; vom avea astfel limbaie de tipul 0, limbaie de tipul 1 (sau *dependente de context*), limbaie de tipul 2 (sau *independente de context*) și limbaie de tipul 3 (sau *regulate*). Să observăm că este importantă

structura cuvintelor unui limbaj și nu modul în care sunt notate simbolurile terminale. De exemplu, limbajele

$$L'_2 = \{0^n 1^n \mid n \geq 1\}, L''_2 = \{a^n b^n \mid n \geq 1\}$$

sunt în mod practic identice. Putem utiliza o unică notație pentru alfabetul simbolurilor terminale, de exemplu, $V_T = \{i_1, \dots, i_n\}$. Clasificarea de mai sus este fundamentală în teoria limbajelor formale, ea a fost introdusă de Naom Chomsky în 1958 și prin tradiție noile clase de limbaje sunt raportate la această clasificare. O altă clasificare este următoarea

- *Gramatici de tipul 0; fără restricții;*
- *Gramatici monotone:*

$$u \rightarrow v, |u| \leq |v|, u, v \in V_G^*;$$

- *Gramatici dependente de context (de tipul 1):*

$$uAv \rightarrow upv, u, p, v \in V_G^*, p \neq \lambda, A \in V_N;$$

- *Gramatici independente de context (de tipul 2):*

$$A \rightarrow p, A \in V_N, p \in V_G^*;$$

- *Gramatici liniare:*

$$\begin{cases} A \rightarrow uBv \\ C \rightarrow p \end{cases} \quad A, B, C \in V_N, u, v, p \in V_T^*;$$

- *Gramatici stâng (drept) liniare:*

$$\begin{cases} A \rightarrow uB (A \rightarrow Bv) \\ C \rightarrow p \end{cases} \quad A, B, C \in V_N, u, v, p \in V_T^*;$$

- *Gramatici regulate (de tipul 3); gramatici stâng liniare sau gramatici drept liniare.*

Gramaticile monotone ca și cele dependente de context nu pot avea reguli cu partea dreaptă vidă. Se introduce următoarea convenție de completare: *într-o gramatică monotonă sau dependentă de context se admite o regula de forma $A \rightarrow \lambda$ cu condiția ca A să nu apară în partea dreaptă a vreunei reguli.* După cum vom vedea, existența sau inexistența regulilor de forma $A \rightarrow \lambda$, reguli numite de ștergere, poate modifica esențial puterea generativă a unei

gramatici. O gramatică în care nu avem astfel de reguli o vom numi uneori λ -liberă; de asemenea, un limbaj care nu conține cuvântul vid, îl vom numi λ -liber. Să mai observăm că existența regulilor de ștergere într-o gramatică nu implică în mod necesar existența cuvântului vid în limbajul generat.

Două gramatici care generează același limbaj se numesc **echivalente**.

Gramaticile monotone și gramaticile dependente de context sunt echivalente; de asemenea, gramaticile drept și stâng liniare sunt echivalente, justificându-se astfel clasa gramaticilor regulate.

1.2 Ierarhia Chomsky

Lemele care urmează vor avea o utilizare frecventă în cele ce urmează.

Lema de localizare a gramaticilor independente de context.

Lema 1.1 *Fie G o gramatică independentă de context și fie derivarea*

$$X_1 \dots X_m \xrightarrow{*} p, X_j \in V_G, j = \overline{1, m}, p \in V_G^*.$$

Atunci există $p_1, p_2, \dots, p_m \in V_G^$ astfel încât $p = p_1 \dots p_m$ și $X_j \xrightarrow{*} p_j, j = \overline{1, m}$.*

Demonstrație. Procedăm prin inducție asupra lungimii derivării l .

Dacă $l = 0$ atunci $p = X_1 \dots X_m$ și luăm $p_j = X_j$.

Presupunem că proprietatea este adevărată pentru derivări de lungime l și fie o derivare de lungime $l + 1, X_1 \dots X_m \xrightarrow{*} p$. Punem în evidență ultima derivare directă $X_1 \dots X_m \xrightarrow{*} q \Rightarrow p$. Conform ipotezei inductive, $q = q_1 \dots q_m$ și $X_j \xrightarrow{*} q_j, j = \overline{1, m}$.

Fie apoi $A \rightarrow u$ regula care se aplică în derivarea directă $q \Rightarrow p$ și să presupunem că A intră în subcuvântul q_k , deci $q_k = q'_k A q''_k$. Vom lua

$$p_j = \begin{cases} q_j & , j \neq k \\ q'_k u q''_k & , j = k \end{cases}$$

Este evident că $X_j \xrightarrow{*} p_j, j \neq k$, iar pentru $j = k$ avem

$$X_k \xrightarrow{*} q_k = q'_k A q''_k \Rightarrow q'_k u q''_k = p_k. \square$$

Vom pune în evidență în continuare o proprietate asupra structurii regulilor gramaticilor Chomsky. Partea dreaptă a unei reguli, pentru toate tipurile de gramatici, este un cuvânt format din terminale sau neterminale. Este convenabil de multe ori ca partea dreaptă a regulilor să conțină un singur tip de simboluri, terminale sau neterminale. Acest lucru este posibil fără modificarea tipului gramaticii.

Lema 1.2 Fie $G = (V_N, V_T, S, P)$ o gramatică de tipul 2. Există o gramatică G' echivalentă, de același tip, cu proprietatea că dacă o regulă are în partea dreaptă un terminal, atunci ea este de forma $A \rightarrow i, A \in V_N, i \in V_T$.

Demonstrație. Luăm gramatica G' de forma $G' = (V_N', V_T', S, P')$ unde V_N' și P' se construiesc astfel: $V_N \subseteq V_N'$ și includem în P' toate regulile din P care convin. Fie acum o regulă din P care nu convine

$$u \rightarrow v_1 i_1 v_2 i_2 \dots i_n v_{n+1}, i_k \in V_T, v_k \in V_N^*.$$

Vom introduce în P' următoarele reguli:

$$u \rightarrow v_1 X_{i_1} v_2 X_{i_2} \dots X_{i_n} v_{n+1}, X_{i_k} \rightarrow i_k, k = \overline{1, n},$$

unde X_{i_k} sunt neterminale noi pe care le adăugăm la V_N' . Este evident că G' păstrează tipul lui G și că $L(G') = L(G)$. \square

Ierarhia Chomsky. Este evident că $\mathcal{L}_3 \subseteq \mathcal{L}_2$ și că $\mathcal{L}_1 \subseteq \mathcal{L}_0$, deoarece orice regulă a unei gramatici de tipul 3 respectă prescripțiile unei gramatici de tipul 2; analog pentru familiile \mathcal{L}_1 și \mathcal{L}_2 . Aparent, o regulă de forma $A \rightarrow p$ (de tipul 2) este un caz particular a unei reguli de forma $uAv \rightarrow upv$ (de tipul 1), pentru $u = v = \lambda$; totuși, realitatea nu este aceasta, deoarece la tipul 2 de gramatici sunt permise reguli de ștergere, pe când la tipul 1 se impune condiția $p \neq \lambda$. Vom arăta că avem $\mathcal{L}_2 \subseteq \mathcal{L}_1$.

Șirul de incluziuni

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

poartă denumirea de ierarhia Chomsky (vom arăta pe parcursul acestui curs că incluziunile sunt stricte). Această ierarhie caracterizează puterea generativă a celor patru tipuri de gramatici, această putere fiind crescătoare de la 3 la 0. Orice alte mecanisme generative se raportează la această ierarhie fundamentală. Vom demonstra mai întâi următoarea lemă.

Lema 1.3 (Lema eliminării regulilor de ștergere). *Orice limbaj independent de context λ -liber poate fi generat de o gramatică de tipul 2 fără reguli de ștergere.*

Demonstrație. Fie $G = (V_N, V_T, S, P)$ o gramatică independentă de context și $L(G)$ limbajul generat. Prin ipoteză $\lambda \notin L(G)$. Definim prin recurență șirul de mulțimi $\{U_k\}_{k \in \mathbb{N}}$ astfel:

$$\begin{aligned} U_1 &= \{X \mid X \in V_N, X \rightarrow \lambda \in P\} \\ U_{k+1} &= \{X \mid X \in V_N, X \rightarrow p \in P, p \in U_k^*\}. \end{aligned}$$

Să observăm că $U_1 \subseteq U_2 \subseteq \dots \subseteq U_k \dots$. Într-adevăr, $U_1 \subseteq U_2$ deoarece dacă $X \in U_1$ rezultă că $X \rightarrow \lambda \in P$ și cum $\lambda \in U_1^*$ rezultă că $X \in U_2$. Dacă acum $U_{k-1} \subseteq U_k$, atunci evident $U_{k-1}^* \subseteq U_k^*$ și dacă $X \in U_k$, deci $X \rightarrow p$, $p \in U_{k-1}^* \subseteq U_k^*$, ceea ce înseamnă că $X \in U_{k+1}$, adică $U_k \subseteq U_{k+1}$.

Cum toate aceste mulțimi sunt incluse în V_N și V_N este finită, rezultă că există o mulțime finală U_f astfel astfel încât

$$U_1 \subseteq U_2 \subseteq \dots \subseteq U_f = U_{f+1} = \dots$$

Are loc de asemenea și implicația $X \in U_f \Leftrightarrow X \xrightarrow{*} \lambda$.

Vom ilustra această implicație cu un exemplu. Să presupunem că $U_f = U_3$ și fie $X \in U_3$. Atunci în mod necesar trebuie să avem $X \rightarrow p \in P$, $p \in U_2^*$; de exemplu $p = X_1 X_2$ și $X_1, X_2 \in U_2$. În mod analog

$$\begin{cases} X_1 \rightarrow Y_1 Y_2 Y_3 \\ X_2 \rightarrow Z_1 Z_2 \end{cases}, Y_1, Y_2, Y_3, Z_1, Z_2 \in U_1,$$

prin urmare $Y_1 \rightarrow \lambda$, $Y_2 \rightarrow \lambda$, $Y_3 \rightarrow \lambda$, $Z_1 \rightarrow \lambda$, $Z_2 \rightarrow \lambda$. Putem scrie derivarea

$$X \Rightarrow X_1 X_2 \xrightarrow{*} Y_1 Y_2 Y_3 Z_1 Z_2 \xrightarrow{*} \lambda.$$

Definim acum următoarea gramatică independentă de context fără reguli de ștergere $G' = (V_N, V_T, S, P')$ unde V_N, V_T, S sunt ca în gramatica dată, iar P' se construiește pornind de la P astfel. Fie $X \rightarrow p \in P, p \neq \lambda$. Includem atunci în P' această regulă precum și toate regulile de forma $X \rightarrow p_j$, unde p_j se obține din p lăsând la o parte, în toate modurile posibile, simbolurile din U_f . De exemplu dacă $X \rightarrow ABC \in P$ și $A, B \in U_f$, vom induce în P' regulile

$$X \rightarrow ABC, X \rightarrow BC, X \rightarrow AC, X \rightarrow C.$$

Să observăm că în acest fel mulțimea P a fost pe de o parte micșorată (au fost excluse regulile de ștergere), iar pe de altă parte îmbogățită cu eventualele noi reguli. Să mai observăm că G' este independentă de context și că nu conține reguli de ștergere.

Vom arăta că $L(G) = L(G')$.

Mai întâi, să arătăm că $L(G) \subseteq L(G')$.

Fie $p \in L(G)$, deci $S \xrightarrow{*}_G p$; vom arăta că $S \xrightarrow{*}_{G'} p$.

Vom arăta o implicație ceva mai generală, $X \xrightarrow{*}_G p$ implică $X \xrightarrow{*}_{G'} p$,

pentru orice $X \in V_N$ (relația cerută se obține pentru $X = S$). Procedăm prin inducție asupra lungimii derivării l . Dacă $l = 1$ avem

$$X \xrightarrow[G]{\Rightarrow} p, X \rightarrow p \in P, p \neq \lambda \Rightarrow X \rightarrow p \in P'$$

și deci $X \xrightarrow[G']{*} p$.

Presupunem că afirmația este adevărată pentru $l = n$ și luăm o derivare cu lungimea $l = n + 1$. Punem în evidență prima derivare directă

$$X \xrightarrow[G]{\Rightarrow} X_1 \dots X_m \xrightarrow[G]{*} p$$

Conform lemei de localizare avem $p = p_1 \dots p_m$ și $X_j \xrightarrow[G]{*} p_j$, $j = \overline{1, m}$. Unele din cuvintele p_j pot să fie vide; pentru precizarea ideilor să presupunem că $p_2, p_3, p_5 = \lambda$. Atunci pentru derivările $X_j \xrightarrow[G]{*} p_j$, $j \neq 2, 3, 5$, care au lungimea de cel mult n , conform ipotezei inductive avem $X_j \xrightarrow[G']{*} p_j$, $j = 2, 3, 5$.

Pe de altă parte, pentru $j = 2, 3, 5$ avem $X_2 \xrightarrow[G]{*} \lambda$, $X_3 \xrightarrow[G]{*} \lambda$, $X_5 \xrightarrow[G]{*} \lambda$, deci $X_2, X_3, X_5 \in U_f$. Rezultă că

$$X \rightarrow X_1 X_4 X_6 \dots X_m \in P'$$

așa încât putem scrie

$$X \xrightarrow[G']{\Rightarrow} X_1 X_4 X_6 \dots X_m \xrightarrow[G']{*} p_1 p_4 p_6 \dots p_m = p.$$

Deci $X \xrightarrow[G']{*} p$ și luând $X = S$ obținem $p \in L(G')$. Prin urmare $L(G) \subseteq L(G')$.

Să arătăm acum incluziunea inversă, $L(G'') \subseteq L(G)$.

Fie $p \in L(G')$, $S \xrightarrow[G']{*} p$. Punem în evidență o derivare directă oarecare

$$S \xrightarrow[G']{*} u \xrightarrow[G']{\Rightarrow} v \xrightarrow[G']{*} p.$$

Dacă în derivarea directă $u \xrightarrow[G']{\Rightarrow} v$ se aplică o regulă care există și în G , atunci evident pasul respectiv poate fi făcut și în G . Să presupunem că se

aplică o regulă nou introdusă de exemplu $X \rightarrow BC$, deci pasul respectiv va avea forma

$$u = u'Xu'' \xrightarrow{G'} u'BCu'' = v$$

dar regula $X \rightarrow BC$ din P' a provenit dintr-o regulă din P , lăsând la o parte simboluri din U_f , în cazul nostru din $X \rightarrow ABC$, lăsându-l la o parte pe $A \in U_f$. Deoarece $A \xrightarrow[G]{*} \lambda$, avem

$$u = u'Xu'' \xrightarrow[G]{*} u'ABCu'' \xrightarrow[G]{*} u'ABCu'' = v.$$

Prin urmare orice pas al derivării considerate se poate obține și în gramatica G , deci $A_0 \xrightarrow[G]{*} p$ și $p \in L(G)$, adică $L(G') \subseteq L(G)$.

Să considerăm acum o gramatică G de tipul 2 și să presupunem că $\lambda \in L(G)$. Construim gramatica G' ca și mai sus; orice cuvânt $p \neq \lambda$ din $L(G)$ se poate obține în G' și invers, deci $L(G') = L(G) \setminus \{\lambda\}$. Considerăm atunci o gramatică $G'' = (V_N \cup \{S''\}, V_T, S'', P' \cup \{S'' \rightarrow \lambda, S'' \rightarrow S\})$. Evident $L(G'') = L(G)$. Toate regulile lui G'' respectă tipul 1 (cu $u = v = \lambda$) și conține o singură regulă de ștergere $S'' \rightarrow \lambda$ iar S'' nu apare în partea dreaptă a vreunei reguli. Deci G'' este de tipul 1 și orice limbaj independent de context este inclus în \mathcal{L}_1 , adică $\mathcal{L}_2 \subseteq \mathcal{L}_1$. \square

1.3 Proprietăți de închidere a familiilor Chomsky

Fiind dată o operație binară notată cu "•" pe o familie de limbaje L , vom spune că familia L este închisă la operația "•" dacă $L_1, L_2 \in L$ implică $L_1 \bullet L_2 \in L$. Definiția noțiunii de închidere pentru operații unare sau cu aritate oarecare este analoagă.

Vom numi familiile din clasificarea Chomsky, mai scurt, familiile Chomsky.

Închiderea familiilor Chomsky la reuniune.

Teorema 1.1 *Familiile de limbaje $L_j, j = 0, 1, 2, 3$ sunt închise față de reuniune.*

Demonstrație. Fie $G_k = (V_{N_k}, V_{T_k}, S_k, P_k), k = 1, 2$ două gramatici de același tip $j, j = 0, 1, 2, 3$. Putem presupune că $V_{N_1} \cap V_{N_2} = \emptyset$. Trebuie să arătăm că limbajul $L(G_1) \cup L(G_2)$ este de același tip j . În acest scop vom construi o gramatică de tipul j care să genereze limbajul $L(G_1) \cup L(G_2)$. Considerăm următoarea gramatică:

$$G = (V_{N_1} \cup V_{N_2} \cup \{S\}, V_{T_1} \cup V_{T_2}, P_1 \cup P_2 \cup \{S \rightarrow S_1 | S_2\}).$$

Este evident că G are același tip j ca și cele două gramatici date.

Vom arăta că $L(G) = L(G_1) \cup L(G_2)$.

Fie $p \in L(G)$, adică $S \xrightarrow[G]{*} p$. În prima derivare directă trebuie cu necesitate să se utilizeze una din regulile $S \rightarrow S_1$ sau $S \rightarrow S_2$. Prin urmare avem una din următoarele două variante:

$$\begin{aligned} S \xrightarrow[G]{*} S_1 \xrightarrow[G]{*} p, \text{ deci } S_1 \xrightarrow[G_1]{*} p, p \in L(G_1) \subseteq L(G_1) \cup L(G_2) \\ S \xrightarrow[G]{*} S_2 \xrightarrow[G]{*} p, \text{ deci } S_2 \xrightarrow[G_2]{*} p, p \in L(G_2) \subseteq L(G_1) \cup L(G_2). \end{aligned}$$

Deci $p \in L(G_1) \cup L(G_2)$, adică $L(G) \subseteq L(G_1) \cup L(G_2)$.

Observație. În implicația " $S_1 \xrightarrow[G_1]{*} p$ în gramatica G_1 rezultă $S_1 \xrightarrow[G]{*} p$ "

se utilizează faptul că neterminalele celor două gramatici sunt disjuncte. Astfel, derivarea $S_1 \Rightarrow p$ se va scrie detaliat astfel

$$G : S_1 \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \dots \Rightarrow u_n = p.$$

Întrucât $S_1 \in V_{N_1}$ și $S_1 \notin V_{N_2}$ în derivarea directă $S_1 \Rightarrow u_1$ se utilizează cu necesitate o regulă din P_1 ; prin urmare $u_1 \in V_{G_1}^*$, deci toate neterminalele din u_1 sunt din V_{N_1} . Mai departe, în derivarea directă $u_1 \Rightarrow u_2$ se utilizează cu necesitate o regulă din P_1 , etc. În consecință, în derivarea $S_1 \xrightarrow[G]{*} p$ toate simbolurile sunt din V_{G_1} și se utilizează numai reguli din P_1 , de unde rezultă implicația.

Invers, fie $p \in L(G_1) \cup L(G_2)$. Dacă, de exemplu, $p \in L(G_1)$, avem $S_1 \xrightarrow[G_1]{*} p$, rezultă $S_1 \xrightarrow[G]{*} p$, deci putem construi derivarea $S \xrightarrow[G]{*} S_1 \xrightarrow[G]{*} p$.

Prin urmare, $p \in L(G)$ și $L(G_1) \cup L(G_2) \subseteq L(G)$. \square

Exemplu. Fie limbajele

$$L_1 = \{1, 11, 111, \dots\}, \quad L_2 = \{01, 0011, 000111, \dots\}.$$

Se poate verifica că aceste limbaje sunt generate de gramaticile

$$\begin{aligned} G_1 : A &\rightarrow 1A|1, \\ G_2 : B &\rightarrow 0B1|01. \end{aligned}$$

Se vede că $L_1 \in \mathcal{L}_3 \subseteq \mathcal{L}_2$, $L_2 \in \mathcal{L}_2$ deci ambele limbaje sunt de tipul 2. Problema închiderii familiei \mathcal{L}_2 la reuniune constă în: *este limbajul*

$$L_1 \cup L_2 = \{1, 11, \dots, 01, 0011, \dots\}$$

de asemenea de tipul 2 (adică există o gramatică de tipul 2 care să îl genereze)? Răspunsul este afirmativ; gramatica de tipul 2 care generează limbajul $L_1 \cup L_2$ are regulile

$$S \rightarrow A|B, A \rightarrow 1A|1, B \rightarrow 0B1|01.$$

Închiderea familiilor Chomsky la produs.

Teorema 1.2 *Famiiliile de limbaje $\mathcal{L}_j, j = 0, 1, 2, 3$ sunt închise față de produs.*

Demonstrație. Fie $G_k = (V_{N_k}, V_{T_k}, S_k, P_k), k = 1, 2$ două gramatici de același tip $j, j = 0, 1, 2, 3$. Putem presupune că $V_{N_1} \cap V_{N_2} = \emptyset$. Vom construi o gramatică de tipul j care să genereze limbajul $L(G_1)L(G_2)$.

Famiiliile $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2$. Luăm gramatica G astfel

$$G = (V_{N_1} \cup V_{N_2} \cup \{S\}, V_{T_1} \cup V_{T_2}, P_1 \cup P_2 \cup \{S \rightarrow S_1S_2\}).$$

Fie $p \in L(G), S \xrightarrow[G]{*} p$. Această derivare va avea forma

$$S \xrightarrow[G]{*} S_1S_2 \xrightarrow[G]{*} p.$$

Vom aplica în continuare o implicație asemănătoare cu lema de localizare de la limbajele independente de context: "dacă $S_1S_2 \xrightarrow{*} p$ atunci $p = p_1p_2$ și $S_1 \xrightarrow{*} p_1, S_2 \xrightarrow{*} p_2$ ". Demonstrație prin inducție asupra lungimii derivării l .

Dacă $l = 2$ (cea mai mică lungime!) atunci se vor aplica reguli de forma $S_1 \rightarrow i_1 \in P_1, S_2 \rightarrow i_2 \in P_2$ unde $i_1 \in V_{T_1} \cup \{\lambda\}, i_2 \in V_{T_2} \cup \{\lambda\}$. Astfel că $p = i_1i_2$ și putem lua $p_1 = i_1, p_2 = i_2$.

Presupunem că implicația este adevărată pentru $l = n$ și luăm $l = n + 1$. În derivarea $S_1S_2 \Rightarrow p$ punem în evidență ultima derivare directă

$$S_1S_2 \xrightarrow{*} q \Rightarrow p.$$

Conform ipotezei inductive, $q = q_1q_2$ și $S_1 \Rightarrow q_1, S_2 \Rightarrow q_2$. Fie $u \rightarrow v$ regula care se aplică în $q \Rightarrow p$. Deoarece $u \rightarrow v$ este sau din P_1 sau din P_2 , rezultă că u este subcuvânt al lui q_1 sau q_2 . Să presupunem că $u \in \text{Sub}(q_1)$; atunci $q_1 = q'_1uq''_1$ și luăm $p_1 = q'_1vq''_1, p_2 = q_2$.

Este clar că $p = p_1p_2$ și că $S_1 \xrightarrow[G_1]{*} p_1, S_2 \xrightarrow[G_2]{*} p_2$.

Mai departe, avem că $p_1 \in L(G_1), p_2 \in L(G_2)$ și deci $p \in L(G_1)L(G_2)$.

Invers, fie $p \in L(G_1)L(G_2)$. Atunci $p = p_1p_2$ și $S_1 \xrightarrow[G_1]^* p_1, S_2 \xrightarrow[G_2]^* p_2$.

Aceste derivări sunt valabile și în gramatica G . Prin urmare putem scrie:

$$G : S \xrightarrow[G]^* S_1S_2 \xrightarrow[G]^* p_1p_2 = p,$$

adică $p \in L(G)$ și $L(G_1)L(G_2) \subseteq L(G)$. \square

Observație. În cazul familiei \mathcal{L}_2 demonstrația primei părți se poate simplifica aplicând direct lema de localizare. Deci, dacă $S \xrightarrow[G]^* p$, atunci $p = p_1p_2, S_1 \xrightarrow[G_1]^* p_1, S_2 \xrightarrow[G_2]^* p_2$. Ținând cont că $V_{N_1} \cap V_{N_2} = \emptyset$ avem $S_1 \xrightarrow[G_1]^* p_1, S_2 \xrightarrow[G_2]^* p_2$

și deci $p_1 \in L(G_1), p_2 \in L(G_2)$.

Să mai observăm că construcția de mai sus nu se poate aplica la familia \mathcal{L}_3 , deoarece regula nou introdusă $S \rightarrow S_1S_2$ nu este de tipul 3.

Familia \mathcal{L}_3 . Luăm gramatica G de forma

$$G = (V_{N_1} \cup V_{N_2}, V_{T_1} \cup V_{T_2}, S_1, P'_1 \cup P_2),$$

unde P'_1 se obține din P_1 prin înlocuirea regulilor de forma $A \rightarrow p$ cu $A \rightarrow pS_2$. Evident, G astfel construită este de tipul 3.

Vom numi regulile de forma $A \rightarrow pB$ de categoria I iar cele de forma $C \rightarrow q$ de categoria a II-a. Evident că într-o derivare care transformă simbolul de start într-un cuvânt al limbajului generat de gramatică, în toate derivările directe se aplică reguli de categoria I iar în ultima o regulă de categoria II. Să mai observăm că P'_1 are reguli numai de categoria I.

Fie $p \in L(G)$, deci $S_1 \xrightarrow[G]^* p$. În această derivare, primele reguli care se aplică sunt din P_1 , apoi o regulă nou introdusă (cu necesitate, căci altfel nu putem elimina neterminalele) iar ultimele reguli sunt din P_2 . Prin urmare, derivarea va avea forma

$$S_1 \xrightarrow[G]^* qA \xrightarrow[G]^* qrS_2 = p_1S_2 \xrightarrow[G]^* p_1p_2 = p.$$

Este clar că $S_1 \xrightarrow[G_1]^* qr = p_1$ (la ultima derivare s-a reutilizat regula de categoria II din P_1 din care a provenit regula nouă) și $S_2 \xrightarrow[G_2]^* p_2$, adică $p_1 \in L(G_1), p_2 \in L(G_2)$. Prin urmare $p = p_1p_2 \in L(G_1)L(G_2)$ și deci $L(G) \subseteq L(G_1)L(G_2)$.

Fie acum $p \in L(G_1)L(G_2)$. Rezultă că $p = p_1p_2, p_1 \in L(G_1), p_2 \in L(G_2)$ și deci $S_1 \xrightarrow[G_1]^* p_1, S_2 \xrightarrow[G_2]^* p_2$. În derivarea $S_1 \xrightarrow[G_1]^* p_1$ ultima regulă

1.3. PROPRIETĂȚI DE ÎNCHIDERE A FAMILIILOR CHOMSKY 19

utilizată este de categoria a II-a, să presupunem $A \rightarrow r$, deci această derivare va avea forma

$$S_1 \xrightarrow[G_1]{*} qA \Rightarrow qr = p_1.$$

Ținând cont că în G avem regula $A \rightarrow rS_2$, putem scrie în G derivarea

$$S_1 \xrightarrow[G]{*} qA \xrightarrow[G]{*} qrS_2 = p_1S_2 \xrightarrow[G]{*} p_1p_2 = p,$$

adică $p \in L(G)$ și $L(G_1)L(G_2) \subseteq L(G)$. \square

Închiderea familiilor Chomsky la operația închidere Kleene.

Teorema 1.3 *Familiile $\mathcal{L}_j, j = 0, 1, 2, 3$ sunt închise față de operația de închidere Kleene.*

Demonstrație. *Familiile $\mathcal{L}_0, \mathcal{L}_1$.* Fie G o gramatică de tipul 0 sau 1 care satisface condiția din lema 1.2. Construim o gramatică G^* astfel

$$G^* = (V_N \cup \{S^*, X\}, V_T, S^*, P \cup \{S^* \rightarrow \lambda | S | XS, Xi \rightarrow Si | XSi, i \in V_T\})$$

Regulile nou introduse sunt de tipul 1, deci G^* nu modifică tipul lui G .

Fie $p \in L(G^*)$, $S^* \xrightarrow{*} p$. Deosebim 3 cazuri după prima regulă care se aplică:

1. Prima regulă este $S^* \rightarrow \lambda$; atunci derivarea noastră va avea forma $S^* \Rightarrow \lambda$ și evident $p = \lambda \in L(G)^*$.
2. Prima regulă este $S^* \rightarrow S$; atunci în G avem $S^* \Rightarrow S^* p$ și în mod evident $S \xrightarrow[G]{*} p, p \in L(G) \subseteq L(G)^*$.
3. Prima regulă care se aplică este $S^* \rightarrow XS$; derivarea va avea forma (în G^*):

$$(A) \quad \begin{aligned} S^* \Rightarrow XS &\Rightarrow Xp_n \Rightarrow \\ &\Rightarrow XSp_n \xrightarrow{*} Xp_{n-1}p_n \Rightarrow \\ &\dots \\ &\Rightarrow XSp_3 \dots p_n \xrightarrow{*} Xp_2p_3 \dots p_{n-1}p_n \Rightarrow \\ &\Rightarrow XSp_2p_3 \dots p_n \xrightarrow{*} p_1p_2p_3 \dots p_n. \end{aligned}$$

Pentru fiecare subcuvânt p_j avem $S^* \xrightarrow{*} p_j$, deci $S \xrightarrow[G]{*} p_j$ și $p_j \in L(G)$

așa încât $p = p_1 \dots p_n \in L(G)^n \subseteq L(G)^*$.

Observație. Faptul că în G^* $S^* \xrightarrow{*} p_j$ rezultă astfel: ținând cont că nu putem elimina neterminul X decât cu o regulă de forma $Xi \rightarrow Si$, în

XS trebuie cu necesitate să îl transformăm pe S până când al doilea simbol va fi un terminal, deci

$$XS \xrightarrow{*} Xiq, \quad q \in V_G^* \text{ și } S \xrightarrow{*} iq.$$

În acest moment putem să îl rescriem pe Xi cu una din regulile $Xi \rightarrow Si|XSi$, etc.

Invers, dacă $p \in L(G)^*$ atunci $p = p_1 \dots p_n$ și $S \xrightarrow{*}_G p_j, j = \overline{1, n}$ deci și în G^* avem $S \xrightarrow{*} p_j$. Putem atunci scrie derivarea (A) și $p \in L(G)^*$. \square

Familia \mathcal{L}_2 . Construim gramatica G^* astfel

$$G^* = (V_N \cup \{S^*\}, V_T, S^*, P \cup P' \cup \{S^* \rightarrow S^*S|\lambda\})$$

Fie $p \in L(G^*)$, deci $S \xrightarrow{*} p$. Această derivare va avea forma (în G^*)

$$(B) \quad S^* \xrightarrow{*} S^*S \dots S^*p.$$

Conform lemei de localizare, $p = p_1 \dots p_n$, $S \xrightarrow{*} p_j, j = \overline{1, n}$ și deci $S \xrightarrow{*} p_j$, ceea ce înseamnă că $p \in L(G)^n \subseteq L(G)^*$, adică $L(G)^* \subseteq L(G^*)$.

Invers, fie $p \in L(G)^*$; atunci $p = p_1 \dots p_n$, $S \xrightarrow{*} p_j, j = \overline{1, n}$. Putem scrie atunci derivarea (B) și $p \in L(G^*)$, adică $L(G)^* \subseteq L(G^*)$. \square

Familia \mathcal{L}_3 . Construim gramatica G^* astfel

$$G^* = (V_N \cup \{S^*\}, V_T, S^*, P \cup P' \cup \{S^* \rightarrow S|\lambda\}),$$

unde P' se obține din regulile de categoria II din P și anume, dacă $A \rightarrow p \in P$ atunci $A \rightarrow pS \in P'$.

Fie $p \in L(G^*)$, adică $S^* \xrightarrow{*} p$. Punând în evidență prima derivare directă, avem

$$G^* : S^* \Rightarrow S^* \xrightarrow{*} p.$$

Dacă în derivarea $S^* \xrightarrow{*} p$ se aplică numai reguli din P , atunci $S \xrightarrow{*}_G p, p \in L(G) \subseteq L(G^*)$.

Să presupunem că se aplică și reguli din P' ; derivarea va avea forma

$$(C) \quad S^* \Rightarrow S^* \xrightarrow{*} p_1 S^* \xrightarrow{*} p_1 p_2 S^* \xrightarrow{*} \dots \xrightarrow{*} p_1 \dots p_n.$$

Evident, $S \xrightarrow{*} p_j$, deci $p_j \in L(G)$, adică $p \in L(G)^*$, ceea ce înseamnă că $L(G^*) \subseteq L(G)^*$.

Invers, fie $p \in L(G)^*$; atunci $p = p_1 \dots p_n$, $S \xrightarrow{*} p_j, j = \overline{1, n}$, sau $S \xrightarrow{*} p_j$ și putem scrie derivarea (C). Aceasta înseamnă că $p \in L(G^*)$ sau $L(G)^* \subseteq L(G^*)$. \square

1.3. PROPRIETĂȚI DE ÎNCHIDERE A FAMILIILOR CHOMSKY 21

Observație. Operațiile reuniune, produs și închidere Kleene se numesc *operații regulate*. Proprietățile de închidere demonstrate până acum pot fi sintetizate astfel:

Teorema 1.4 *Famiiliile de limbaje din clasificarea Chomsky sunt închise la operațiile regulate.*

Recapitulare asupra construcțiilor gramaticilor:

Reuniune

$$j = 0, 1, 2, 3 : \\ G = (V_{N_1} \cup V_{N_2} \cup \{S\}, V_{T_1} \cup V_{T_2}, P_1 \cup P_2 \cup \{S \rightarrow S_1|S_2\}).$$

Produs

$$j = 0, 1, 2 : \\ G = (V_{N_1} \cup V_{N_2} \cup \{S\}, V_{T_1} \cup V_{T_2}, P_1 \cup P_2 \cup \{S \rightarrow S_1S_2\}). \\ j = 3 : \\ G = (V_{N_1} \cup V_{N_2}, V_{T_1} \cup V_{T_2}, S_1, P'_1 \cup P_2),$$

unde P'_1 se obține din P_1 prin înlocuirea regulilor de forma $A \rightarrow p$ cu $A \rightarrow pS_2$.

Închidere Kleene

$$j = 0, 1 : \\ G^* = (V_N \cup \{S^*, X\}, V_T, S^*, P \cup \{S^* \rightarrow \lambda|S|XS, Xi \rightarrow Si|XS_i, i \in V_T\}) \\ j = 2 : \\ G^* = (V_N \cup \{S^*\}, V_T, S^*, P \cup \{S^* \rightarrow S^*S|\lambda\}) \\ j = 3 : \\ G^* = (V_N \cup \{S^*\}, V_T, S^*, P \cup P' \cup \{S^* \rightarrow S|\lambda\}).$$

1. Specificați limbajul generat de gramatica $G = (V_N, V_T, S, P)$, precizând tipul gramaticii (cf. clasificării Chomsky):

- (a) $V_N = \{A\}, V_T = \{a, b\}, S = A, P = \{A \rightarrow aA|b\}$;
- (b) $V_N = \{x_0\}, V_T = \{A, B, \dots, Z\}, S = x_0, P = \{x_0 \rightarrow x_1D, x_1 \rightarrow x_2N, x_2 \rightarrow E\}$;
- (c) $V_N = \{A\}, V_T = \{0, 1, 2\}, S = A, P = \{A \rightarrow 0A0|1A1|2A2|\lambda\}$;
- (d) $V_N = \{S, A\}, V_T = \{0, 1, \dots, 9, \cdot\}, P = \{S \rightarrow A.A, A \rightarrow 0A|1A|\dots|9A|0|1|\dots|9\}$;
- (e) $V_N = \{S\}, V_T = \{PCR, PDAR, UDMR\}, P = \{S \rightarrow PCR|PDAR|UDMR\}$;

- (f) $V_N = \{A, B, C\}$, $V_T = \{0, 1\}$, $S = A$, $P = \{A \rightarrow 0A|1B|1, B \rightarrow 0C|1A, C \rightarrow 0B|1C|0\}$;
- (g) $V_N = \{S, A, B, C\}$, $V_T = \{0, 1, \dots, 9, +, -\}$,
 $P = \{S \rightarrow +A| - A|A, A \rightarrow 0A|1A| \dots |9A|0|1| \dots |9\}$;
- (h) $V_N = \{S\}$, $V_T = \{(\cdot)\}$, $P = \{S \rightarrow S(S)S|\lambda\}$;
- (i) $V_N = \{E, T, F\}$, $V_T = \{(\cdot), i, +, *\}$, $S = E$,
 $P = \{E \rightarrow E + T|T, T \rightarrow T * F|F, F \rightarrow (E)|i\}$;
- (j) $V_N = \{S, A, B\}$, $V_T = \{a, b, c\}$, $P = \{S \rightarrow abc|aAbc, Ab \rightarrow bA, Ac \rightarrow Bbcc, bB \rightarrow Bb, aB \rightarrow aaA|aa\}$;
- (k) $V_N = \{S, A, B, C, D, E\}$, $V_T = \{a\}$, $P = \{S \rightarrow ACaB, Ca \rightarrow aaC, CB \rightarrow DB|E, aD \rightarrow Da, AD \rightarrow AC, aE \rightarrow Ea, AE \rightarrow \lambda\}$;
- (l) $V_N = \{S, A, B, C, D, E\}$, $V_T = \{a, b\}$, $P = \{S \rightarrow ABC, AB \rightarrow aAD|bAE, DC \rightarrow BaC, EC \rightarrow BbC, Da \rightarrow aD, Db \rightarrow bD, Ea \rightarrow aE, Eb \rightarrow bE, AB \rightarrow \lambda, C \rightarrow \lambda, aB \rightarrow Ba, bB \rightarrow Bb\}$;
2. Folosind teorema să se construiască o gramatică independentă de context, fără reguli de ștergere care generează limbajul de la problema 1-h.
3. Găsiți gramatici care generează limbajele următoare:
- (a) $L = \{\lambda\}$;
- (b) $L = \emptyset$;
- (c) $L = \{0^n | n \in \mathbb{N}\}$;
- (d) $L = \{a^i b^j a^i b^j\}$;
- (e) $L = \{awbbw' | w, w' \in \{0, 1\}^*\}$;
- (f) $L = \{w | w \text{ constantă reală în limbajul PASCAL}\}$;
- (g) $L = \{w \in \{0, 1\}^* | w \text{ conține maxim 2 de } 0\}$;
- (h) $L = \{wa\tilde{w} | w \in \{0, 1\}^*\}$;
- (i) $L = \{w | w \text{ octet ce reprezintă un număr par}\}$;
- (j) $L = \{A, B, C, \dots, Z\}$;
4. Se consideră limbajele generate de gramaticile de la problema 1 punctul a,b,c,d. Notăm aceste limbaje cu L_a, L_b, L_c, L_d . Folosind demonstrația teoremelor referitoare la închiderile familiilor de limbaje din clasificarea Chomsky să se găsească gramatici care generează limbajele $L_a \cup L_b, L_c \cup L_d, L_a L_b, L_c L_d, L_a^*, L_c^*$.

Capitolul 2

Limbaje regulate

2.1 Automate finite și limbaje regulate

Automate finite. Automatele finite sunt mecanisme pentru recunoașterea limbajelor de tipul 3 (regulate). Un automat finit (AF) se compune dintr-o bandă de intrare și un dispozitiv de comandă.

Pe banda de intrare sunt înregistrate simboluri ale unui alfabet de intrare, constituind pe bandă un cuvânt p . La fiecare pas de funcționare banda se deplasează cu o poziție spre stânga.

Dispozitivul de comandă posedă un dispozitiv de citire de pe bandă; dispozitivul se află permanent într-o anumită stare internă, element al unei mulțimi finite de stări. Schema unui automat finit este redată în figura 2.1.

Automatul finit funcționează în pași discreți. Un pas de funcționare constă din: dispozitivul de comandă citește de pe banda de intrare simbolul aflat în dreptul dispozitivului de citire; în funcție de starea internă și de simbolul citit, automatul trece într-o nouă stare și mută banda cu o poziție spre stânga. Automatul își încetează funcționarea după ce s-a citit ultimul simbol înregistrat pe bandă; în acest moment el se va afla într-o anumită stare, care, după cum vom vedea, va juca un rol important în recunoașterea cuvintelor.

Din punct de vedere matematic, un automat finit este un sistem

$$AF = (\Sigma, I, f, s_0, \Sigma_f),$$

unde

Σ este mulțimea de stări;

I este alfabetul de intrare;

$f : \Sigma \times I \longrightarrow \mathcal{P}(\Sigma)$ este funcția de evoluție;

$s_0 \in \Sigma$ este starea inițială;

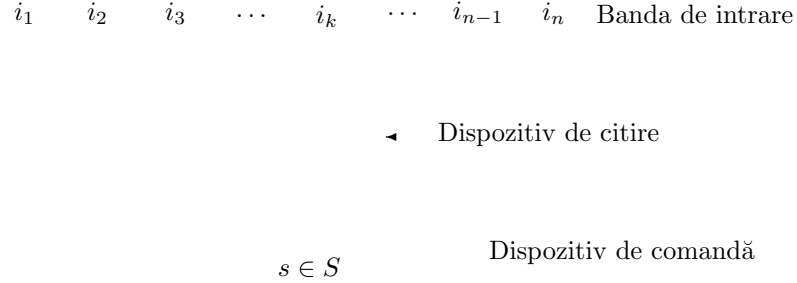


Figura 2.1: Reprezentarea schematică a unui automat finit

$\Sigma_f \subseteq \Sigma$ este mulțimea de stări finale.

Dacă pentru orice $(s, i) \in \Sigma \times I$, avem $|f(s, i)| \leq 1$ automatul se numește *determinist*; în caz contrar se numește *nedeterminist*.

Observații:

(1) $\mathcal{P}(\Sigma)$ este mulțimea părților lui Σ ; deoarece $\emptyset \in \mathcal{P}(\Sigma)$ este posibil ca pentru un anumit $s \in S$ și $i \in I$ să avem $f(s, i) = \emptyset$.

(2) În cazul unui automat determinist vom scrie $f(s, i) = s'$ (în loc de $f(s, i) \in \{s'\}$).

(3) Definiția dată este specifică teoriei limbajelor formale. O altă definiție (mai generală), întâlnită în teoria automatelor este următoarea: un automat finit este un sistem $AF = (\Sigma, I, O, f, g, s_0, F)$ unde, Σ, I, f, s_0, F au semnificația de mai sus, O este *alfabetul de ieșire* iar $g : \Sigma \times I \rightarrow \mathcal{P}(O)$ este *funcție de ieșire*. Funcționalitatea unui astfel de automat finit este analoagă cu cea descrisă mai sus, cu deosebirea că la fiecare pas automatul furnizează o ieșire $o \in g(s, i)$.

Funcționarea unui automat finit se poate bloca în situația în care el se află în starea s , citește de pe bandă simbolul i și $f(s, i) = \emptyset$; evident că în acest caz funcționarea în continuare nu mai este posibilă.

Prin diagrama de stări a unui automat finit înțelegem un graf orientat care are nodurile etichetate cu stările $s \in \Sigma$ iar arcele se construiesc astfel: nodurile s, s' se unesc cu un arc orientat de la s la s' dacă există $i \in I$ astfel încât $s' \in f(s, i)$; arcul respectiv va fi notat cu i .

Exemplu $AF = (\Sigma, I, f, s_0, \Sigma_f)$ unde $\Sigma = \{s_0, s_1, s_2\}$, $I = \{i_1, i_2\}$, $\Sigma_f = \{s_2\}$, iar f este dată de tabelul

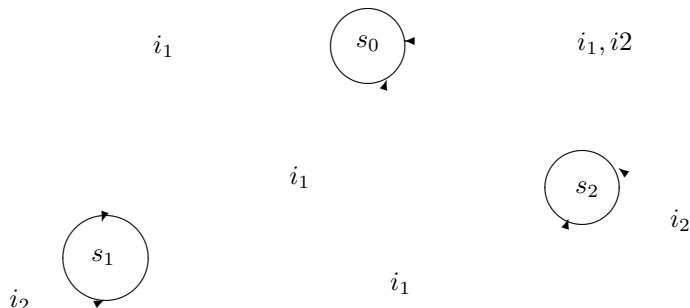


Figura 2.2: Diagrama de stări

	s_0	s_1	s_2
i_1	$\{s_1\}$	$\{s_2\}$	$\{s_0\}$
i_2	\emptyset	$\{s_0, s_1\}$	$\{s_0, s_1\}$

Diagrama de stări corespunzătoare este prezentată în figura 2.2.

Funcția de evoluție Funcția f se prelungește de la $\Sigma \times I$ la $\mathcal{P}(\Sigma) \times I^*$ deci $f : \mathcal{P}(\Sigma) \times I^* \rightarrow \mathcal{P}(\Sigma)$, astfel:

- $f(s, \lambda) = \{s\}, \forall s \in \Sigma$,
- $f(\emptyset, i) = \emptyset, \forall i \in I$,
- $f(Z, i) = \bigcup_{s \in Z} f(s, i), Z \in \mathcal{P}(\Sigma), Z \neq \emptyset$,
- $f(Z, pi) = f(f(Z, p), i)$.

Să observăm că relațiile de mai sus constituie o definiție prin recurență, corectă; fiind dat f , putem defini mai întâi toate valorile $f(Z, i)$ (un număr finit, deoarece I este o mulțime finită), apoi $f(Z, p)$ pentru $|p| = 2$, în continuare $f(Z, p)$ pentru $|p| = 3$, etc.

Proprietățile funcției f :

- Dacă Z_k este o familie de părți a lui Σ , avem

$$f\left(\bigcup_k Z_k, i\right) = \bigcup_k f(Z_k, i)$$

Demonstrație. Utilizând (c) putem scrie

$$\bigcup_k f(Z_k, i) = \bigcup_k \left(\bigcup_{s \in Z_k} f(s, i) \right) = \bigcup_{s \in \bigcup_k Z_k} f(s, i) = f\left(\bigcup_k Z_k, i\right). \square$$

- $f(Z, p) = \bigcup_{s \in Z} (f(s, p)), p \in I^*$.

Demonstrație. Prin inducție asupra lungimii lui p .

Pentru $|p| = 1$ avem $f(Z, i) = \bigcup_{s \in Z} f(s, i)$, adică (c).

Presupunem că relația este adevărată pentru $|p| = m$ și considerăm un p astfel încât $|p| = m + 1$, deci $p = p'i, |p'| = m$. Putem scrie

$$\begin{aligned} f(Z, p) &= f(Z, p'i) = f(f(Z, p'), i) = f(\bigcup_{s \in Z} f(s, p'), i) = \square \\ &= \bigcup_{s \in Z} f(f(s, p'), i) = \bigcup_{s \in Z} f(s, p'i) = \bigcup_{s \in Z} f(s, p). \end{aligned}$$

3. $f(s, pq) = f(f(s, p), q), p, q \in I^*$.

Demonstrație. Inducție asupra lungimii lui q .

Dacă $|q| = 1$, atunci $q = i$ și relația se reduce la (d).

Presupunem că proprietatea este adevărată pentru r și considerăm $|q| = r + 1$. Deci $q = q'i$. Avem

$$\begin{aligned} f(s, pq) &= f(s, pq'i) = f(f(s, pq'), i) = f(f(f(s, p), q'), i) = \square \\ &= f(f(s, p), q'i) = f(f(s, p), q). \end{aligned}$$

Limbaje regulate.

Definiție 2.1 *Limbajul recunoscut de automatul finit $AF = (\Sigma, I, f, s_0, \Sigma_f)$ este*

$$L(AF) = \{p | p \in I^*, f(s_0, p) \cap \Sigma_f \neq \emptyset\}$$

Deci $p \in L(AF)$ dacă automatul aflându-se în starea inițială s_0 , după $|p|$ pași de funcționare poate să ajungă într-o stare finală.

În cazul unui automat finit determinat limbajul recunoscut poate fi definit în modul următor. Pentru fiecare $s \in \Sigma$ definim funcția $f_s : I^* \rightarrow \mathcal{P}(\Sigma)$ prin $f_s(p) = f(s, p)$.

Atunci

$$f(s_0, p) \cap \Sigma_f \neq \emptyset \Leftrightarrow f(s_0, p) = f_{s_0}(p) \in \Sigma_f,$$

deci

$$L(AF) = \{p | f(s_0, p) \cap \Sigma_f \neq \emptyset\} = f_{s_0}^{-1}(\Sigma_f)$$

Limbajele recunoscute de automate finite le vom numi *limbaje regulate*; familia acestor limbaje o vom nota cu \mathcal{R} . Evident, familia limbajelor recunoscute de automate finite deterministe, \mathcal{R}_d , este o parte a lui \mathcal{R} , $\mathcal{R}_d \subseteq \mathcal{R}$. Vom arăta că cele două familii coincid.

Teorema 2.1 $\mathcal{R}_d = \mathcal{R}$.

Demonstrație. Fie $AF = (\Sigma, I, f, s_0, \Sigma_f)$ un automat finit (în general nedeterminist). Construim următorul automat finit determinist $AF' = (\Sigma', I, f', \{s_0\}, \Sigma'_f)$ unde $\Sigma' = \mathcal{P}(\Sigma)$, $f' = f$ (prelungirea la $\mathcal{P}(\Sigma) \times I^*$), $\Sigma'_f = \{Z | Z \in \mathcal{P}(\Sigma), Z \cap \Sigma_f \neq \emptyset\}$.

Evident, automatul AF' este determinist.

Fie $p \in L(AF)$. Atunci $f(s_0, p) \cap \Sigma_f \neq \emptyset$ și $f(s_0, p) \in \Sigma'_f$. Pe de altă parte, conform cu proprietatea 2 a funcției de evoluție, avem

$$f'(\{s_0\}, p) = f(s_0, p)$$

și deci $f(s_0, p) \in \Sigma'_f$, adică $p \in L(AF')$ și $L(AF) \subseteq L(AF')$.

Pe o cale analoagă se arată că $L(AF') \subseteq L(AF)$. \square

Observație. Faptul că un cuvânt este recunoscut de un automat finit se poate verifica prin calcul direct sau pe diagrama de stări.

Exemplu. Considerăm automatul din exemplul anterior și fie $p = i_1 i_2 i_1$. Prin calcul direct:

$$\begin{aligned} f(s_0, i_1 i_2 i_1) &= f(f(f(s_0, i_1), i_2), i_1) = f(f(\{s_1\}, i_2), i_1) = \\ &= f(\{s_0, s_1\}, i_1) = f(\{s_0\}, i_1) \cup f(\{s_1\}, i_1) = \{s_1\} \cup \{s_2\}. \end{aligned}$$

Astfel că $f(s_0, i_1 i_2 i_1) \cap \Sigma_f = \{s_2\} \neq \emptyset$ și $p \in L(AF)$.

Pe diagrama de stări există traiectoriile:

$$\begin{array}{ccccccc} s_0 & \xrightarrow{i_1} & s_1 & \xrightarrow{i_2} & s_0 & \xrightarrow{i_1} & s_1; \\ s_0 & \xrightarrow{i_1} & s_1 & \xrightarrow{i_2} & s_1 & \xrightarrow{i_1} & s_2; \end{array}$$

A doua traiectorie ne duce într-o stare finală, deci $p \in L(AF)$.

Limbae de tipul trei și limbaje regulate. Vom arăta în cele ce urmează că familia limbajelor de tipul 3 coincide cu familia limbajelor regulate. În prealabil vom pune în evidență o formă specială a limbajelor de tipul 3, pe care convenim să o numim formă normală.

Definiție 2.2 *Vom spune că o gramatică de tipul 3 este în forma normală dacă are reguli de generare de forma*

$$\begin{cases} A \rightarrow iB, \\ C \rightarrow j, \end{cases} \quad \text{unde } A, B, C \in V_N, \quad i, j \in V_T$$

sau regula de completare $S \rightarrow \lambda$ și în acest caz S nu apare în dreapta vreunei reguli.

Lema 2.1 *Orice gramatică de tipul 3 admite o formă normală.*

Demonstrație. Dacă $G = (V_N, V_T, S, P)$ este gramatica dată, eliminăm în primul rând regulile de ștergere (teorema de la ierarhia Chomsky) apoi construim gramatica $G' = (V'_N, V_T, S, P')$, unde V'_N și P' se definesc astfel:

introducem în V'_N toate simbolurile din V_N iar în P' toate regulile din P care convin; fie acum în P o regulă de forma

$$A \rightarrow pB, \quad p = i_1 \dots i_n$$

Vom introduce în P' regulile:

$$\begin{aligned} A &\rightarrow i_1 Z_1, \\ Z_1 &\rightarrow i_2 Z_2, \\ &\dots, \\ Z_{n-1} &\rightarrow i_n B, \end{aligned}$$

iar simbolurile Z_1, \dots, Z_{n-1} le includem în V'_N .

În cazul unei reguli de forma $A \rightarrow p$ cu $|p| > 1$ procedăm analog, exceptând ultima regulă nou introdusă care va avea forma $Z_{n-1} \rightarrow i_n$. Să mai facem observația că simbolurile Z_1, \dots, Z_{n-1} le luăm distincte pentru fiecare caz.

Se poate arăta ușor că $L(G) = L(G')$. \square

Teorema 2.2 *Familia limbajelor de tipul 3 coincide cu familia limbajelor regulate.*

Demonstrație. Partea I: $E \in \mathcal{L}_3 \Rightarrow E \in \mathcal{R}$.

Fie E un limbaj de tipul 3 și $G = (V_N, V_T, S, P)$ gramatica care îl generează; putem presupune că G este în formă normală. Construim automatul finit $AF = (\Sigma, I, f, s_0, \Sigma_f)$ unde $\Sigma = V_N \cup \{X\}$ (X simbol nou), $I = V_T, s_0 = S$ și

$$\Sigma_f = \begin{cases} \{X, S\} & , \text{pentru } \Sigma \rightarrow \lambda \in P \\ \{X\} & , \text{pentru } \Sigma \rightarrow \lambda \notin P \end{cases}$$

Funcția de evoluție este definită de:

$$\begin{aligned} &\text{dacă } A \rightarrow iB \in P \text{ luăm } B \in f(A, i), \\ &\text{dacă } C \rightarrow j \in P \text{ luăm } X \in f(C, j), \\ &\text{în rest } \emptyset \end{aligned}$$

Observație. Automatul astfel definit este în general nedeterminist. De exemplu, dacă $A \rightarrow 0B|0$ atunci $f(A, 0) = \{B, X\}$.

Fie $p \in L(G)$, $p = i_1 \dots i_n$, deci $S \xrightarrow{*} p$. Detaliat, această derivare va avea forma

$$(1) \quad S \Rightarrow i_1 A_1 \Rightarrow i_1 i_2 A_2 \Rightarrow i_1 i_2 \dots i_{n-1} A_{n-1} \Rightarrow i_1 i_2 \dots i_n.$$

S-au aplicat regulile:

$$(2) \begin{array}{l} S \rightarrow i_1 A_1, \\ A_1 \rightarrow i_2 A_2, \\ \dots \\ A_{n-1} \rightarrow i_n. \end{array}$$

În automat avem corespunzător:

$$(3) : \begin{array}{l} A_1 \in f(S, i_1), \\ A_2 \in f(A_1, i_2), \\ \dots \\ X \in f(A_{n-1}, i_n) \end{array}$$

Putem scrie traiectoria

$$(4) S \xrightarrow{i_1} A_1 \xrightarrow{i_2} A_2 \xrightarrow{i_3} \dots \xrightarrow{i_n} X \in \Sigma_f$$

Deci $p \in L(AF)$.

Dacă $p = \lambda$, atunci $S \Rightarrow \lambda$ și $S \rightarrow \lambda \in P$. Dar atunci $\lambda \in L(AF)$, căci automatul este în starea S și rămîne în această stare după "citirea" lui λ ; cum însă în acest caz $S \in \Sigma_f$ rezultă că și în acest caz $p \in L(AF)$. În consecință $L(G) \subseteq L(AF)$.

Fie acum $p = i_1 \dots i_n \in L(AF)$; atunci avem traiectoria (4), relațiile (3), regulile de generare (2) și putem scrie derivarea (1), adică $p \in L(G)$ și $L(AF) \subseteq L(G)$. \square

Partea II $E \in \mathcal{R} \Rightarrow E \in \mathcal{L}_3$.

Vom indica numai modul de construcție a gramaticii. Fie $AF = (\Sigma, I, f, s_0, \Sigma_f)$ automatul finit care recunoaște limbajul E , pe care îl presupunem determinist. Construim gramatica $G = (\Sigma, I, s_0, P)$ unde mulțimea P este definită astfel

$$\begin{array}{l} f(A, i) = B \text{ generează regula } A \rightarrow iB \in P, \\ \text{în plus dacă } B \in \Sigma_f \text{ se generează și regula } A \rightarrow i \in P. \end{array}$$

Putem arăta că $L(G) = L(AF)$. \square

2.2 Proprietăți speciale ale limbajelor regulate

Caracterizarea limbajelor regulate. Limbajele regulate, fiind părți din \bar{I}^* , se pot caracteriza algebric, independent de mecanismele de generare (gramaticile de tipul 3) sau de cele de recunoaștere (automatele finite).

Teorema 2.3 Fie $E \subset I^*$ un limbaj. Următoarele afirmații sunt echivalente.

- (a) $E \in \mathcal{R}$;
- (b) E este o reuniune de clase de echivalențe a unei congruențe de rang finit;
- (c) Următoarea congruență

$$\mu = \{(p, q) \mid \chi_E(r_1 p r_2) = \chi_E(r_1 q r_2), \forall r_1, r_2 \in I^*\},$$

unde χ_E este funcția caracteristică a lui E , este de rang finit.

Demonstrație: Vom arăta următoarele implicații: (a) \Rightarrow (b), (b) \Rightarrow (c), (c) \Rightarrow (a).

(a) \Rightarrow (b).

Fie $AF = (\Sigma, I, f, s_0, \Sigma_f)$ automatul finit care recunoaște limbajul E . Definim pe I^* relația

$$\xi = \{(p, q) \mid f(s, p) = f(s, q), \forall s \in \Sigma\}.$$

Se poate vedea cu ușurință că ξ este o relație de echivalență (reflexivă, simetrică, tranzitivă). În plus, dacă $r \in I^*$ și $(p, q) \in \xi$, atunci $(pr, qr) \in \xi$ și $(rp, rq) \in \xi$. De exemplu, prima apartenență se deduce astfel

$$f(s, pr) = f(f(s, p), r) = f(f(s, q), r) = f(s, qr), \text{ etc.}$$

Prin urmare ξ este o relație de congruență.

Să arătăm că această congruență este de rang finit, adică mulțimea cât I^*/ξ este finită.

Fie $\alpha : \Sigma \longrightarrow \Sigma$ o aplicație oarecare și fie mulțimea

$$I^*(\alpha) = \{p \mid p \in I^*, f(s, p) = \alpha(s), \forall s \in \Sigma\}.$$

Să observăm că dacă α este funcția identică atunci $\lambda \in I^*(\alpha)$. Deci nu toate $I^*(\alpha)$ sunt vide; în acest caz $I^*(\alpha)$ este o clasă de echivalență. Într-adevăr, fie $p \in I^*(\alpha) \subseteq I^*$ fixat și fie C_p clasa de echivalență a lui p . Arătăm că $C_p = I^*(\alpha)$. Dacă $q \in I^*(\alpha)$, atunci $f(s, q) = \alpha(s), \forall s \in \Sigma$, ceea ce înseamnă că $f(s, q) = f(s, p), \forall s \in \Sigma$, și deci $(p, q) \in \xi$ adică $q \in C_p$ și $I^*(\alpha) \subseteq C_p$. Invers dacă $q \in C_p$ atunci $f(s, q) = f(s, p) = \alpha(s), \forall s \in \Sigma$ și $q \in I^*(\alpha)$, adică $C_p \subseteq I^*(\alpha)$. Aceasta înseamnă că $I^*(\alpha) = C_p$, adică $I^*(\alpha)$ este o clasă de echivalență.

Între mulțimea cât I^*/ξ și mulțimea funcțiilor definite pe Σ cu valori în Σ putem stabili următoarea corespondență biunivocă: unei funcții $\alpha : \Sigma \longrightarrow \Sigma$ îi corespunde clasa de echivalență $I^*(\alpha)$. Invers, fiind dată o clasă

de echivalență C , luăm $p \in C$ (oarecare) și atașăm lui C funcția $\alpha(s) = f(s, p) \forall s \in \Sigma$. Ținând cont că dacă $q \in C$ atunci $f(s, p) = f(s, q), \forall s \in \Sigma$, rezultă că funcția α nu depinde de elementul p ales.

Dar mulțimea funcțiilor definite pe Σ cu valori în Σ este finită, deci I^*/ξ este finită, adică congruența ξ este de rang finit.

Fie acum $p \in L(AF)$ și q astfel ca $(p, q) \in \xi$. Avem

$$f(s_0, q) = f(s_0, p) \in \Sigma_f;$$

adică $q \in L(AF)$. Aceasta înseamnă că odată cu elementul p , $L(AF)$ conține clasa de echivalență a lui p . De aici rezultă că $L(AF)$ este constituit dintr-un anumit număr de clase de echivalență a lui ξ . \square

(b) \Rightarrow (c)

Fie ξ o congruență de rang finit și E o reuniune de clase de echivalență. Fie apoi $(p, q) \in \xi$; aceasta înseamnă că $r_1 p r_2 \in E \Leftrightarrow r_1 q r_2 \in E$, deci

$$\chi_E(r_1 p r_2) = \chi_E(r_1 q r_2), \forall r_1, r_2 \in I^*.$$

Prin urmare, $(p, q) \in \mu$. Orice clasă de echivalență din I^*/ξ este inclusă într-o clasă de echivalență din I^*/μ , așa încât $\text{card}(I^*/\mu) < \text{card}(I^*/\xi)$, adică congruența μ este de rang finit. \square

(c) \Rightarrow (a)

Presupunem că μ este o congruență de rang finit; considerăm automatul finit $AF = (I^*/\mu, I, f, C_\lambda, \Sigma_f)$, unde funcția de evoluție f și mulțimea de stări finale sunt

$$f(C_p, i) = C_{pi}, \Sigma_f = \{C_p | p \in E\}.$$

Vom arăta că $E = L(AF)$. În primul rând să observăm că $f(C_p, q) = C_{pq}$ (se poate arăta prin inducție asupra lui $|q|$). Avem

$$p \in E \Leftrightarrow C_p \in \Sigma_f \Leftrightarrow f(C_\lambda, p) = C_{\lambda p} = C_p \in \Sigma_f \Leftrightarrow p \in L(AF)$$

adică $E = L(AF)$ și E este un limbaj regulat. \square

Corolar 2.4 Familia \mathcal{R} este închisă la operația de răsturnare.

Demonstrație. Fie $E \in \mathcal{R}$ și \tilde{E} răsturnatul lui E . Conform teoremei de caracterizare, $\text{card}(I^*/\mu^E) < \infty$. Avem

$$(p, q) \in \mu^E \Leftrightarrow \chi_E(r_1 p r_2) = \chi_E(r_1 q r_2) \Leftrightarrow$$

și

$$\chi_{\tilde{E}}(\tilde{r}_1 \tilde{p} \tilde{r}_2) = \chi_{\tilde{E}}(\tilde{r}_1 \tilde{q} \tilde{r}_2) \Leftrightarrow (\tilde{p}, \tilde{q}) \in \mu^{\tilde{E}}$$

Cu alte cuvinte dacă C este o clasă de echivalență a lui μ^E atunci \tilde{C} (răsturnatul lui C) este o clasă de echivalență a lui $\mu^{\tilde{E}}$. Aceasta înseamnă că $\text{card}(I^*/\mu^{\tilde{E}}) = \text{card}(I^*/\mu^E) < \infty$ și conform aceleiași teoreme de caracterizare $\tilde{E} \in \mathcal{R}$. \square

Observație. Am văzut că limbajele de tipul 3 pot fi definite de gramatici cu reguli de două categorii: drept liniare sau stâng liniare. Este evident că limbajele stâng liniare sunt răsturnatele limbajelor drept liniare. Cum familia limbajelor regulate (drept liniare) este închisă la operația de răsturnare, rezultă că cele două familii de limbaje coincid.

Închiderea familiei \mathcal{L}_3 la operațiile $Pref$ și complementariere.

Operațiile $Pref$ și complementariere se definesc în modul următor

$$Pref(E) = \{p \mid \exists r \in I^*, pr \in E\}, \quad C(E) = I^* \setminus E.$$

Teorema 2.5 *Familia \mathcal{L}_3 este închisă la operațiile $Pref$ și complementariere.*

Demonstrație. Fie $AF = (\Sigma, I, f, s_0, \Sigma_f)$ automatul finit care recunoaște limbajul E . Putem presupune că AF este determinist.

Limbaajul $Pref(E)$. Construim automatul finit $AF' = (\Sigma, I, f, s_0, \Sigma'_f)$ unde

$$\Sigma'_f = \{s \in \Sigma \mid s = f(s_0, q), q \in Pref(E)\}.$$

Este evident că $Pref(E) \subseteq L(AF')$, căci dacă $q \in Pref(E)$ atunci $s = f(s_0, q) \in \Sigma'_f$ conform definiției lui Σ'_f .

Să arătăm acum că $L(AF') \subseteq Pref(E)$. Fie $r \in L(AF')$, atunci $f(s_0, r) \in \Sigma'_f$, deci există $q \in Pref(E)$ astfel încât $f(s_0, r) = f(s_0, q)$. Cum q este prefixul unui cuvânt din E , există $w \in I^*$ astfel încât $qw \in E$, adică $f(s_0, q) \in \Sigma_f$. Dar

$$f(s_0, rw) = f(f(s_0, r), w) = f(f(s_0, q), w) = f(s_0, qw) \in \Sigma_f,$$

deci $rw \in E$ și $r \in Pref(E)$. Aceasta înseamnă că $L(AF') \subseteq Pref(E)$ și prin urmare $Pref(E) = L(AF')$, adică $Pref(E)$ este limbaj regulat. \square

Limbaajul $C(E)$. Avem

$$C(E) = \{p \in I^* \mid p \notin E\} = \{p \in I^* \mid f(s_0, p) \notin \Sigma_f\} = \{p \in I^* \mid f(s_0, p) \in C(\Sigma_f)\}.$$

Prin urmare $C(E) = L(AF_c)$ unde $AF_c = (\Sigma, I, f, s_0, C(\Sigma_f))$, adică $C(E)$ este un limbaj regulat. \square

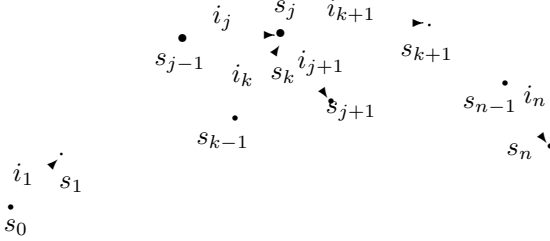


Figura 2.3: Traiectoria automatului finit

2.3 Lema de pompare pentru limbaje regulate

Sub această denumire (sau lema uvw) este cunoscută o proprietate a limbajelor regulate (ca și a altor familii de limbaje) care ne permit să descompunem cuvintele suficient de lungi ale limbajului în forma uvw și să multiplicăm subcuvântul v de un număr arbitrar de ori, obținând cuvinte care aparțin de asemenea limbajului. Cu alte cuvinte, putem să ”pompăm” în cuvântul dat o anumită parte a sa. Astfel de leme se utilizează deseori pentru a rezolva probleme de neapartenență, adică pentru a arăta că un anumit limbaj nu aparține unei familii date de limbaje.

Lema 2.2 Fie E un limbaj regulat și $AF = (\Sigma, I, f, s_0, \Sigma_f)$ automatul finit care îl recunoaște. Dacă $p \in E$ și $|p| \geq \text{card}(\Sigma)$ atunci p se descompune în forma $p = uvw$, $v \neq \lambda$ și $uv^m w \in E$, $\forall m \in \mathbb{N}$.

Demonstrație. Fie $p = i_1 \dots i_n$, $n \geq \text{card}(\Sigma)$; fie s_0, s_1, \dots, s_n stările parcurse de automat la citirea cuvântului p . Atunci, $s_j = f(s_{j-1}, i_j)$, $j = \overline{1, n}$, $s_n \in \Sigma_f$. Există în mod necesar două stări egale, $s_j = s_k$, $j < k$. Traiectoria va avea o buclă (vezi figura 2.3).

Descompunem cuvântul p în forma $p = uvw$ unde $u = i_1 \dots i_j$, $v = i_{j+1} \dots i_k$, $w = i_{k+1} \dots i_n$. Este clar că $v \neq \lambda$, căci $j < k$. Pe de altă parte, putem parcurge traiectoria făcând de mai multe ori bucla, adică

$$f(s_0, uv^m w) = s_n \in \Sigma_f.$$

Prin urmare $uv^m w \in E$. \square

Consecință 2.3 Incluziunea $\mathcal{L}_3 \subseteq \mathcal{L}_2$ este strictă.

În adevăr, fie limbajul $L_2 = \{0^n 1^n | n \geq 1\}$. Știm că acest limbaj este de tipul 2 și că poate fi generat de gramatica $G = (\{A\}, \{0, 1\}, A, \{A \rightarrow 0A1|01\})$. Să arătăm că L_2 nu este de tipul 3.

Să presupunem că L_2 este de tipul 3 și fie $AF = (\Sigma, I, f, s_0, \Sigma_f)$ automatul finit care îl recunoaște. Cum L_2 conține cuvinte oricât de lungi, fie $p \in L_2$ astfel încât $p \geq \text{card}(\Sigma)$. Conform lemei de pompare, p se descompune în forma $p = uvw$, $v \neq \lambda$ și $uv^m w \in E$. Putem avea una din situațiile:

$$\begin{aligned} (1) \quad p &= \underbrace{0\dots 00}_u \underbrace{0\dots 00}_v \underbrace{01\dots 1}_w, \\ (2) \quad p &= \underbrace{0\dots 01\dots 1}_u \underbrace{1\dots 1}_v \underbrace{1\dots 1}_w, \\ (3) \quad p &= \underbrace{0\dots 00}_u \underbrace{0\dots 1}_v \underbrace{1\dots 1}_w. \end{aligned}$$

Primele două cazuri nu pot avea loc deoarece multiplicându-l pe v , numărul de simboluri 0 și 1 nu s-ar păstra egal. În al treilea caz, luând de exemplu, $m = 2$ obținem

$$p_2 = 0\dots 00\dots 10\dots 11\dots 1 \in L_2$$

ceea ce din nou nu este posibil, întrucât se contrazice structura cuvintelor lui L_2 . Prin urmare L_2 nu este de tipul 3. \square

Observație. Este interesant de observat că limbajele simple de forma lui L_2 sunt semnificative pentru clasele din clasificarea Chomsky. Astfel

$$\begin{aligned} L_1 &= \{a^n | n \geq 1\}, L_1 \in \mathcal{L}_3; \\ L_2 &= \{a^n b^n | n \geq 1\}, L_2 \in \mathcal{L}_2, L_2 \notin \mathcal{L}_3; \\ L_3 &= \{a^n b^n c^n | n \geq 1\}, L_3 \in \mathcal{L}_1(?), L_3 \notin \mathcal{L}_2; \end{aligned}$$

Ne-am putea aștepta ca limbajul L_3 , un exemplu analog lui L_2 , să fie de tip 1, adică $L_3 \in \mathcal{L}_1$, $L_3 \notin \mathcal{L}_2$. În adevăr, se poate arăta că $L_3 \notin \mathcal{L}_2$, dar după cunoștința autorului, aparența $L_3 \in \mathcal{L}_1$ este o problemă deschisă.

Consecință 2.4 *Fie E un limbaj regulat și $AF = (\Sigma, I, f, s_0, \Sigma_f)$ automatul finit care îl recunoaște. Atunci E este infinit dacă și numai dacă există $p \in E$ astfel încât $|p| \geq \text{card}(\Sigma)$.*

Dacă limbajul este infinit, este clar că există $p \in E$ cu $|p| \geq \text{card}(\Sigma)$. Invers, dacă există $p \in E$ cu $|p| \geq \text{card}(\Sigma)$ atunci $p = uvw$, $v \neq \lambda$ și $uv^m w \in E, \forall m \in \mathbb{N}$, deci limbajul este infinit. \square

2.4 Expresii regulate

Expresii regulate și limbaje reprezentate Fie V un alfabet. Expresiile regulate sunt cuvinte peste alfabetul $V \cup \{\bullet, *, |\}$, unde simbolurile suplimentare introduse le vom considera operatori: "|" -sau, "•" -produs, "*" -închidere. Expresiile regulate se definesc astfel:

- (1) λ este o expresie regulată;
- (2) pentru orice $a \in V$, cuvântul a este o expresie regulată;
- (3) dacă R și S sunt expresii regulate, atunci $R|S$, $R \bullet S$ și R^* sunt expresii regulate.

Pentru a pune în evidență ordinea de aplicare a operatorilor vom utiliza paranteze; de exemplu $(R|S) \bullet P$. Vom considera că operatorul $*$ are ponderea cea mai mare, apoi operatorul \bullet și $|$ ponderea cea mai mică. Deci prin $R|S^*$ vom înțelege $R|(S^*)$.

Observație. Expresiile regulate se pot defini cu ajutorul gramaticii $G = (\{E, T, F\}, V \cup \{|\bullet, *, (\cdot)\}, E, P)$ unde

$$P = \left\{ \begin{array}{l} E \rightarrow E|T \mid E \bullet T \mid T, \\ T \rightarrow T^* \mid F \\ F \rightarrow (E) \mid a \mid \lambda. \end{array} \right.$$

Unei expresii regulate îi putem asocia un anumit limbaj peste V ; vom spune că expresia regulată reprezintă (desemnează, notează) acel limbaj. Modul în care asociem un limbaj unei expresii regulate este

- (1) λ reprezintă limbajul $\{\lambda\}$,
- (2) a reprezintă limbajul $\{a\}$,
- (3) dacă R și S sunt expresii regulate și reprezintă respectiv limbajele L_R și L_S atunci

(i) $R|S$ reprezintă limbajul $L_R \cup L_S$;

(ii) $R \bullet S$ reprezintă limbajul $L_R L_S$;

(iii) R^* reprezintă limbajul $(L_R)^*$.

Fie R, S, P trei expresii regulate și L_R, L_S, L_P limbajele reprezentate. Avem :

$$L_{(R|S)|P} = (L_R \cup L_S) \cup L_P = L_R \cup (L_S \cup L_P) = L_{R|(S|P)},$$

întrucât operația de reuniune este asociativă. Vom scrie $(R|S)|P = R|(S|P)$. În mod analog se pot obține și alte proprietăți. De exemplu:

$$\begin{aligned} S|R &= S|R, \\ R|R &= R, \end{aligned}$$

$$(R \bullet S) \bullet P = R \bullet (S \bullet P),$$

$$R \bullet (S|P) = (R \bullet S)|(R \bullet P), \text{ etc.}$$

În cazul în care nu există pericol de confuzie, vom nota cu L (fără indice) limbajul reprezentat de o anumită expresie regulată.

Exemple.

1. $R = a^*$; $L = \bigcup_{j=0}^{\infty} \{a^j\} = \{\lambda, a, a^2, \dots\}$.
2. $R = aa^*$; $L = a \bullet \{\lambda, a, a^2, \dots\} = \{a, a^2, a^3, \dots\}$.
3. $R = (a|b)^*$; $L = (L_a \cup L_b)^* = (\{a\} \cup \{b\})^* = \{a, b\}^*$;
 $\{a, b\}^* = \{\lambda\} \cup \{a, b\}^1 \cup \{a, b\}^2 \cup \dots = \{\lambda, a, b, aa, ab, ba, bb, \dots\}$,
 adică $(a|b)^*$ reprezintă mulțimea tuturor cuvintelor peste alfabetul $\{a, b\}$.
4. $R = a|ba^*$; $L = \{a\} \cup \{b\} \bullet \{\lambda, a, a^2, \dots\} = \{a, b, ba, ba^2, \dots\}$.

Limbajele reprezentate de expresii regulate constituie o anumită familie de limbaje; o vom nota cu \mathcal{L}_{lr} . Apare următoarea problemă: *care este poziția acestei familii în ierarhia Chomsky?* Vom arăta că \mathcal{L}_{lr} coincide cu familia limbajelor regulate.

2.5 Sisteme tranziționale

Definiție 2.3 *Un sistem tranzițional este un sistem de forma*

$$ST = (\Sigma, I, f, \Sigma_0, \Sigma_f, \delta)$$

unde:

- Σ este o mulțime (finită) de stări;
- I este alfabetul de intrare;
- $f : \Sigma \times I \rightarrow \mathcal{P}(\Sigma)$ este funcția de tranziție;
- $\Sigma_0 \subseteq \Sigma$ este mulțimea de stări inițiale;
- $\Sigma_f \subseteq \Sigma$ este mulțimea de stări finale;
- $\delta \subseteq \Sigma \times \Sigma$ este relația de tranziție.

Exemplu. $\Sigma = \{s_0, s_1, s_2\}$, $I = \{0, 1\}$, $\Sigma_0 = \{s_0, s_1\}$, $\Sigma_f = \{s_2\}$ iar funcția și relația de tranziție sunt date de:

f	s_0	s_1	s_2
0	$\{s_1\}$	$\{s_2\}$	$\{s_0\}$
1	λ	$\{s_0, s_1\}$	$\{s_0, s_2\}$

$\delta = \{(s_0, s_1), (s_2, s_1)\}$.

Ca și în cazul unui automat finit putem construi diagrama de stări completată cu relația δ (arcele punctate). În cazul exemplului nostru diagrama de stări este prezentă în figura ??

Observație: δ fiind o relație, are sens δ^* (închiderea tranzitivă și reflexivă).

Fie $i \in I \cup \{\lambda\}$; vom spune că sistemul tranzițional evoluează direct din starea s' în starea s'' dacă:

(1) $i = \lambda$ și $(s', s'') \in \delta^*$. Pe diagrama de stări vom avea o traiectorie punctată de la starea s' la starea s'' ;

$$s' \longrightarrow O \longrightarrow O \longrightarrow \dots \longrightarrow O \longrightarrow s''.$$

(2) $i \neq \lambda$ și există $s_1, s_2 \in \Sigma$ astfel încât $(s', s_1) \in \delta$, $s_2 \in f(s_1, i)$ și $(s_2, s'') \in \delta^*$. Pe diagrama de stări, putem ajunge din s' în s'' pe o traiectorie punctată, apoi un pas pe un arc plin și din nou pe o traiectorie punctată.

$$s' \longrightarrow O \longrightarrow \dots \longrightarrow s_1 \xrightarrow{i} s_2 \longrightarrow O \longrightarrow \dots \longrightarrow s''.$$

Vom scrie $s' \stackrel{i}{\vdash} s''$.

Fie acum $p = i_1 \dots i_n$. Vom spune că sistemul evoluează din starea s' în starea s'' dacă există s_0, s_1, \dots, s_n astfel încât

$$s' = s_0 \stackrel{i_1}{\vdash} s_1 \stackrel{i_2}{\vdash} \dots \stackrel{i_n}{\vdash} s_n = s''.$$

Vom scrie $s' \stackrel{p}{\vdash} s''$.

Definiție 2.4 *Limbaajul recunoscut de un sistem tranzițional ST este*

$$L(ST) = \{p \mid p \in I^*, \exists s_0 \in \Sigma_0, s_0 \stackrel{p}{\vdash} s, s \in \Sigma_f\}.$$

Vom nota cu \mathcal{L}_{ST} familia limbajelor recunoscute de sisteme tranziționale. Este evident că orice automat finit este un sistem tranzițional particular în care $\text{card}(\Sigma_0) = 1$ iar $\delta = \emptyset$ (nu există arce punctate). Prin urmare $\mathcal{R} \subseteq \mathcal{L}_{ST}$.

Teorema 2.6 $\mathcal{R} = \mathcal{L}_{ST}$.

Demonstrație. Evident, trebuie să arătăm incluziunea $\mathcal{L}_{ST} \subseteq \mathcal{R}$. Fie $ST = (\Sigma, I, f, \Sigma_0, \Sigma_f, \delta)$ un sistem tranzițional. Construim automatul finit $AF = (\mathcal{P}(\Sigma), I, f', \Sigma_0, \Sigma'_f)$ unde

$$\begin{aligned} f'(Z, i) &= \{s \mid \exists s' \in Z, s' \stackrel{i}{\vdash} s\}, \\ \Sigma'_f &= \{Z \mid Z \cap \Sigma_f \neq \emptyset\}. \end{aligned}$$

Fie $p = i_1 \dots i_n \in L(ST)$ și fie următoarea evoluție a sistemului tranzițional

$$s_0 \xrightarrow{i_1} s_1 \xrightarrow{i_2} \dots \xrightarrow{i_n} s_n \in \Sigma_f.$$

Putem construi o traiectorie a lui AF de forma

$$\Sigma_0 \xrightarrow{i_1} Z_1 \xrightarrow{i_2} \dots \xrightarrow{i_n} Z_n,$$

unde $Z_1 = f'(\Sigma_0, i_1)$, $Z_k = f'(Z_{k-1}, i_k)$, $k = 2, \dots, n$. Să observăm că $s_0 \in \Sigma_0$ și că dacă $s_{k-1} \in Z_{k-1}$, atunci conform definiției funcției f' , avem $s_k \in f'(Z_{k-1}, i_k) = Z_k$. Astfel, $s_k \in Z_k$, $k = 1, \dots, n$; pentru $k = n$ avem $s_n \in Z_n$ și cum $s_n \in \Sigma_f$ rezultă că $Z_n \cap \Sigma_f \neq \emptyset$, adică $Z_n \in \Sigma'_f$. Deci automatul ajunge într-o stare finală, $p \in L(AF)$ și $L(ST) \subseteq L(AF)$.

Incluziunea inversă se arată în mod analog. \square

Construcția sistemelor tranziționale pentru expresii regulate. Fiind dată o expresie regulată putem întotdeauna construi un sistem tranzițional care recunoaște limbajul reprezentat de expresia respectivă.

Construcția se face cu ajutorul diagramelor de stări.

Sistemele tranziționale (diagramele de stări) corespunzătoare expresiilor regulate λ , a și \emptyset sunt prezentate în figura ??.

Dacă R și S sunt expresii regulate și notăm cu ST_R și ST_S sistemele tranziționale corespunzătoare, atunci sistemele tranziționale pentru $R|S$, $R \bullet S$ și R^* sunt redată în figura ??.

În acest mod putem construi succesiv (recurent) un sistem tranzițional corespunzător unei expresii regulate.

Exemplu. Sistemul tranzițional corespunzător expresiei $R = a|b \bullet a^*$ este redat în figura ??.

Consecință Dându-se o expresie regulată, putem construi sistemul tranzițional care recunoaște limbajul reprezentat de expresia respectivă. Cum orice limbaj recunoscut de un sistem tranzițional este regulat, rezultă că limbajele reprezentate de expresii regulate sunt limbaje regulate.

1. Construiți automate finite pentru recunoașterea limbajelor:

- (a) $L = \{PSDR, PNL, PUNR\}$;
- (b) $L = \{w \mid \text{șiruri de } 0 \text{ și } 1 \text{ terminate cu } 1 \}$;
- (c) $L = \{w \mid w \text{ identificator PASCAL}\}$;
- (d) $L = \{w \mid w \text{ constantă întreagă cu semn în PASCAL}\}$;
- (e) $L = \{w \in \{0, 1\}^* \mid w \text{ multiplu de } 3\}$;
- (f) $L = \{a^i b^j \mid i, j > 0\}$;
- (g) $L = \emptyset$.

2. Construiți automate finite echivalente cu gramaticile de tipul trei de la problema 1 capitolul 1.
3. Construiți automate finite deterministe echivalente cu cele nedeterminate obținute la problema precedentă.
4. Găsiți gramatici regulate echivalente cu automatele de la problema 1.
5. Folosind lema de pompare pentru limbae regulate dovediți că următoarele limbae nu sunt regulate:
 - (a) $L = \{0^{i^2} | i \geq 1\}$;
 - (b) $L = \{0^{2^n} | n \geq 1\}$;
 - (c) $L = \{0^n | n \text{ este număr prim } \}$;
 - (d) $L = \{0^m 1^n 0^{m+n} | m \geq 1, n \geq 1\}$;
6. Specificați limbaele denotate de următoarele expresii regulate:
 - (a) $(11 + 0)^*(00 + 1)^*$;
 - (b) $(1 + 01 + 001)^*(\lambda + 0 + 00)$;
 - (c) $10 + (0 + 11)0^*1$;
 - (d) $((0 + 1)(0 + 1))^*$;
 - (e) $01^* + 1$;
 - (f) $((11)^* + 101)^*$.
7. Construiți sisteme tranzitionale ce recunosc limbaele specificate la problema precedentă. Pentru fiecare sistem tranzitional construiți un automat finit determinist echivalent.

Capitolul 3

Limbaje independente de context

3.1 Arbori de derivare

Caracterizarea limbajelor independente de context cu ajutorul arborilor de derivare.

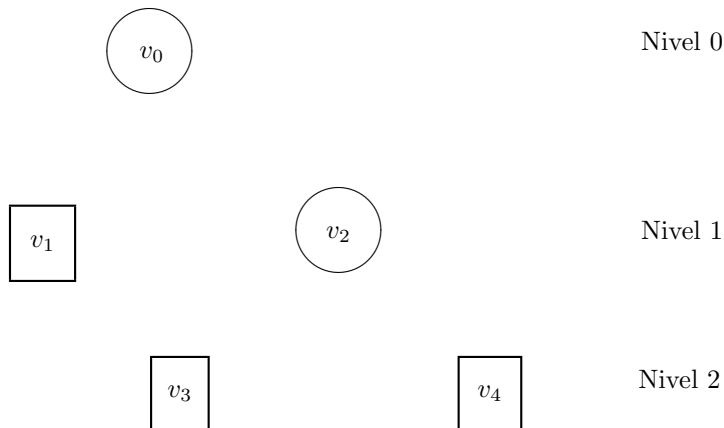
Una din caracteristicile de bază ale limbajelor independente de context este aceea că o derivare într-un astfel de limbaj poate fi reprezentată de un arbore, numit în acest context *arbore de derivare*. Această reprezentare este importantă în mod special pentru faptul că permite o imagine intuitivă simplă a unei derivări și deci posibilitatea de a lucra ușor cu limbaje de tipul 2.

Vom prezenta în primul rând câteva noțiuni elementare de teoria grafurilor, cu scopul de a preciza notațiile și terminologia.

Un *graf orientat* \mathcal{G} este o pereche $\mathcal{G} = (V, \Gamma)$ unde V este o mulțime finită iar Γ o aplicație $\Gamma : V \rightarrow \mathcal{P}(V)$. Mulțimea V se numește mulțimea vârfurilor (nodurilor) grafului iar dacă $v_2 \in \Gamma(v_1)$, perechea (v_1, v_2) este un arc în graf; v_1 este *originea* iar v_2 este *extremitatea* arcului. Un *drum* de la vârful v' la vârful v'' în graful \mathcal{G} este o mulțime de arce $(v_1, v_2)(v_2, v_3) \dots (v_{n-1}, v_n)$ cu $v' = v_1$ și $v'' = v_n$. Numărul $n - 1$ este *lungimea drumului*. Un drum pentru care $v_1 = v_n$ se numește *circuit*. Un circuit de lungime 1 poartă numele de *buclă*.

Definiție 3.1 *Un arbore este un graf fără circuite, cu $\text{card}(V) \geq 2$ și care satisface următoarele două condiții :*

1. $\exists v_0 \in V$ astfel încât $v_0 \notin \Gamma(v), \forall v \in V$; v_0 se numește **rădăcina arborelui**;

Figura 3.1: Reprezentarea grafică a arborelui $\mathcal{G} = (V, \Gamma)$

2. $\forall v \in V \setminus \{v_0\}, \exists! w$ cu $v \in \Gamma(w)$; altfel spus orice vârf diferit de v_0 este extremitatea unui singur arc.

Exemplu. $V = \{v_0, v_1, v_2, v_3, v_4\}$ iar funcția Γ este dată de:

$$\frac{x}{\Gamma(x)} \parallel \begin{array}{|c|c|c|c|c|c|} \hline v_0 & v_1 & v_2 & v_3 & v_4 \\ \hline \{v_1, v_2\} & \emptyset & \{v_3, v_2\} & \emptyset & \emptyset \\ \hline \end{array}$$

Reprezentarea în plan a acestui arbore este dată în figura 4.1

Nodurile v pentru care $\Gamma(v) = \emptyset$ se numesc noduri *terminale* (finale); celelalte se numesc *interne*. Mulțimea nodurilor terminale constituie *frontiera* arborelui. În general vom nota un arbore cu litere mari, specificînd ca indici rădăcina și frontiera; de exemplu $\mathcal{A}_{v_0, v_1 v_2 v_3}$. Un arbore comportă mai multe *ramuri*; în exemplu avem următoarele ramuri : $v_0 v_1, v_0 v_2 v_3, v_0 v_2 v_4$.

Fie $G = (V_N, V_T, S, P)$ o gramatică de tipul 2.

Definiție 3.2 Un arbore de derivare în gramatica G este un arbore cu următoarele două proprietăți .

- (1) Nodurile sunt etichetate cu elementele din V_G ;
- (2) Dacă un nod v are descendenți direcți v_1, \dots, v_n atunci $v \rightarrow v_1 v_2 \dots v_n \in P$.

Exemplu. $G = (\{A, B\}, \{a, b\}, A, P)$ unde

$$P = \{A \rightarrow aBA | Aa | a, B \rightarrow AbB | ba | abb\}.$$

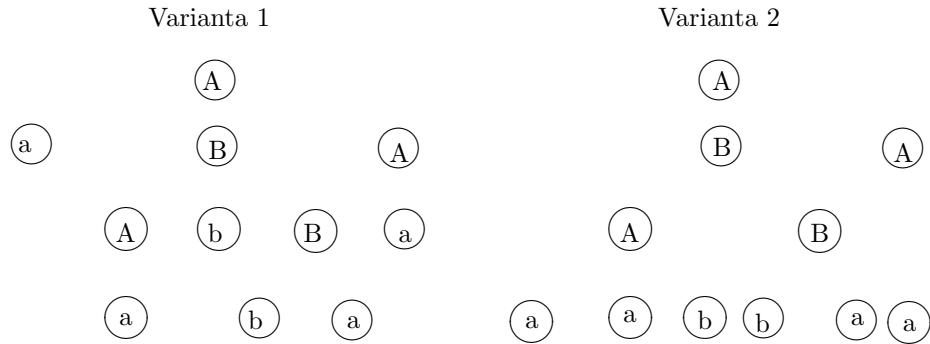


Figura 3.2: Variante de reprezentare a arborelui $\mathcal{A}_A, aabbbaa$

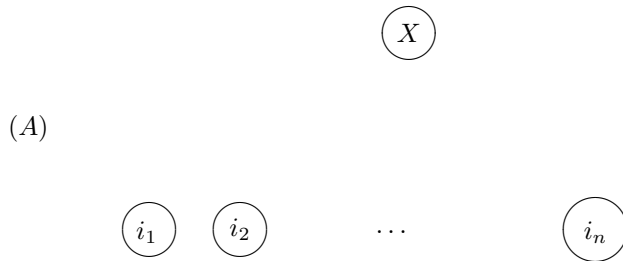


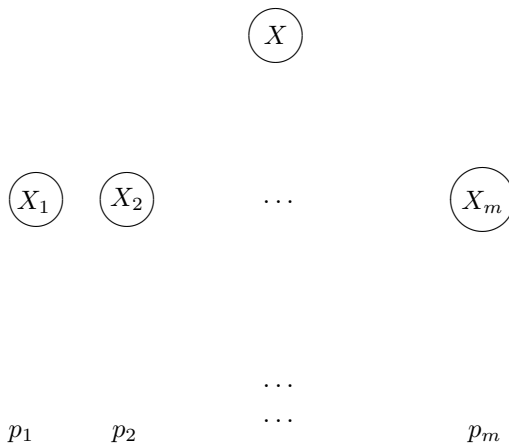
Figura 3.3: Arbore corespunzător unei derivări directe

Arborele $\mathcal{A}_A, aabbbaa$ reprezentat în figura 4.2 (Varianta 1) este un arbore de derivare (poate fi desenat coborând frontiera pe nivelul ultim , Varianta 2):

Teorema 3.1 Fie G o gramatică de tipul 2. Atunci $X \xrightarrow{*} p$ dacă și numai dacă există un arbore $\mathcal{A}_{X, p}$.

Demonstrație. $X \xrightarrow{*} p$ implică $\exists \mathcal{A}_{X, p}$.

Procedăm prin inducție asupra lungimii derivării l . Dacă $l = 1$, $X \Rightarrow p = i_1 \dots i_n$ și $X \rightarrow i_1 \dots i_n \in P$. Arborele din figura 4.3 corespunde cerințelor teoremei.

Figura 3.4: Construcția arborelui $\mathcal{A}_{X,p_1\dots p_m}$.

Presupunem că proprietatea este adevărată pentru derivări de lungime l și considerăm o derivare de lungime $l + 1$, $X \xRightarrow{*} p$. Punem în evidență prima derivare directă

$$X \Rightarrow X_1 \dots X_n \xRightarrow{*} p$$

Conform lemei de localizare, $p = p_1 \dots p_n$ și $X_j \xRightarrow{*} p_j$, $j = \overline{1, n}$. Putem face următoarea construcție: conform ipotezei inductive, fiecărei derivări $X_j \xRightarrow{*} p_j$ îi corespunde câte un arbore \mathcal{A}_{X_j, p_j} ; unim apoi toate nodurile X_j în nodul X plasat la nivelul zero. Obținem astfel un arbore $\mathcal{A}_{X, p_1 \dots p_m} = \mathcal{A}_{X, p}$ (vezi figura 4.4) care corespunde cerințelor teoremei.

Pentru implicația $\exists \mathcal{A}_{X, p} \Rightarrow X \xRightarrow{*} p$, se parcurge o cale inversă, făcând o inducție asupra numărului de nivele. De exemplu, dacă acest număr este 2, arborele de derivare trebuie să arate ca în 4.3 și deci avem $X \rightarrow i_1 i_2 \dots i_n = p \in P$ și $X \xRightarrow{*} p$, etc. \square

3.2 Decidabilitate și ambiguitate în familia \mathcal{L}_2 .

Decidabilitate. Problemele de decidabilitate sunt acele probleme în care se cere să decidem dacă un anumit fapt are sau nu loc. De obicei aceste probleme se rezolvă prin construirea unui algoritm de decizie.

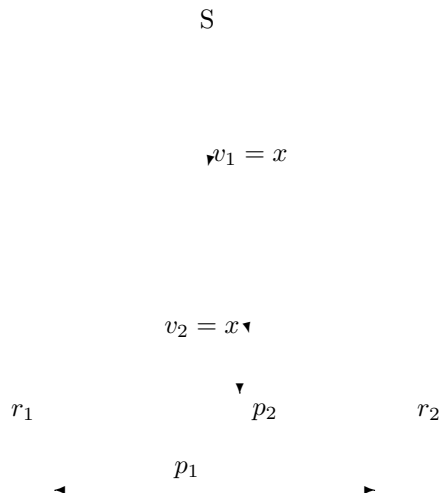


Figura 3.5:

Exemplu Fie gramatica

$$G = (\{A, B, C\}, \{a, b\}, A, \{A \rightarrow aA|bB|C, B \rightarrow abA|aC, C \rightarrow aabA\}).$$

Se poate ușor vedea că $L(G) = \emptyset$ (nu putem elimina neterminalele).

Problema: Putem decide în general dacă limbajul generat de o gramatică de tipul 2 este vid sau nu?

Teorema 3.2 Faptul că limbajul generat de o gramatică de tipul 2 este nevid este decidabil.

Demonstrație. Vom construi un algoritm cu ajutorul căruia se poate decide dacă limbajul generat de o gramatică de tipul 2 este vid sau nu.

Presupunem că limbajul nu este vid, $L(G) \neq \emptyset$ și fie $p \in L(G)$. Există arborele $\mathcal{A}_{S,p}$. Să presupunem că în acest arbore există un drum cu două noduri interne etichetate cu același simbol, $v_1 = v_2 = X$. Descompunem arborele ca în figură 4.5

Avem $p = r_1 p_1 r_2$; evident $p_2 \in \text{Sub}(p_1)$. Efectuăm următoarea modificare: scoatem subarborele $\mathcal{A}_{v_1, p_1} = \mathcal{A}_{X, p_1}$ și-l înlocuim cu subarborele $\mathcal{A}_{v_2, p_2} = \mathcal{A}_{X, p_2}$; obținem în acest fel un arbore $\mathcal{A}_{S, r_1 p_2 r_2}$ (care este într-adevăr un arbore de derivare). Conform cu teorema de caracterizare a

limbajelor de tipul 2, avem $S \xrightarrow{*} r_1 p_2 r_2 \in L(G)$. Dar arborele corespunzător nu mai conține perechea v_1, v_2 de noduri etichetate cu același simbol X .

Repetând procedeul, eliminăm pe rând nodurile de pe aceleași ramuri etichetate identic. În final vom obține un arbore $\mathcal{A}_{S, q}$, $q \in L(G)$ care are proprietatea că pe orice ramură nodurile sunt etichetate cu simboluri distincte. Ținând cont că orice ramură are toate nodurile etichetate cu simboluri neterminale cu excepția ultimului (de pe frontieră) care este etichetat cu un simbol terminal, rezultă că în $\mathcal{A}_{S, q}$ orice ramură conține cel mult $\text{card}(V_N) + 1$ noduri. Dar mulțimea unor astfel de arbori este finită; obținem următorul algoritm de decizie:

Construim toți arborii cu rădăcina S și care au proprietatea de mai sus; dacă printre aceștia se găsește un arbore cu frontiera constituită numai din terminale, atunci $L(G) \neq \emptyset$ (evident) iar dacă nici unul din acești arbori nu au frontiera constituită numai din terminale, atunci $L(G) = \emptyset$. \square

Ambiguitate. Fie G o gramatică de tipul 2. O derivare $S = u_0 \Rightarrow u_1 \Rightarrow \dots \Rightarrow u_n$ în care la fiecare derivare directă se înlocuiește simbolul neterminal cel mai din stânga (dreapta) se numește *derivare extrem stângă (dreaptă)*. Să observăm că în particular într-o gramatică de tipul 3 orice derivare este extrem dreaptă (scrierea drept liniară).

Definiție 3.3 O gramatică G de tipul 2 în care există un cuvânt $p \in L(G)$ care se poate obține cu două derivări extrem stângi (drepte) distincte, se numește ambiguă. În caz contrar este neambiguă.

Exemplu. Gramatica $A \rightarrow aBA|Aa|a$, $B \rightarrow AbB|ba|abb$ este ambiguă. Într-adevăr, avem

$$A \Rightarrow aBA \Rightarrow aBa \Rightarrow aAbBa \Rightarrow aAbbaa \Rightarrow aabbaa;$$

$$A \Rightarrow Aa \Rightarrow aBAa \Rightarrow aBaa \Rightarrow aabbaa.$$

Definiție 3.4 Un limbaj este ambigu dacă toate gramaticile care îl generează sunt ambigue. În caz contrar (adică dacă există o gramatică neambiguă care să îl genereze) limbajul este neambigu.

Dacă G este ambigu și $p \in L(G)$ este un cuvânt care se poate obține cu două derivări extrem stângi distincte, atunci există arborii $\mathcal{A}_{S, p}$ și $\mathcal{A}'_{S, p}$, diferiți, dar care au aceeași rădăcină și frontieră.

Teorema 3.3 Dacă L_1 și L_2 sunt limbaie disjuncte neambigue, atunci $L_1 \cup L_2$ este neambigu.

Demonstrație. Fie $G_k = (V_{N_k}, V_{T_k}, S_k, P_k)$, $k = 1, 2$ două gramatici de tipul 2 neambigue și fie $G = (V_{N_1} \cup V_{N_2}, V_{T_1} \cup V_{T_2}, S, P_1 \cup P_2 \cup \{S \rightarrow S_1|S_2\})$ gramatica ce generează limbajul $L(G_1) \cup L(G_2)$.

Să presupunem că $L(G)$ este ambiguă. Atunci există $p \in L(G)$ care se poate obține cu două derivări extreme stângi diferite. Să presupunem că $p \in L(G_1)$, $p \notin L(G_2)$. Atunci obținem două derivări distincte în gramatica G

$$(1) S \xRightarrow{G} S_1 \xRightarrow{G}^* p, \text{ deci } S_1 \xRightarrow{G_1}^* p;$$

$$(2) S \xRightarrow{G} S_1 \xRightarrow{G}^* p, \text{ deci } S_1 \xRightarrow{G_2}^* p,$$

deci și două derivări extrem stângi în gramatica G_1 . Aceasta ar însemna că G_1 este ambiguă. Contradicție cu ipoteza! \square

Teorema 3.4 *Limbațele de tipul 3 sunt neambigue .*

Demonstrație. Fie L un limbaj de tipul 3 și G gramatica care îl generează; fie apoi AF automatul finit care recunoaște limbajul L și AFD automatul finit echivalent determinist. Construim gramatica G' astfel încât $L(G') = L(AF)$. Reamintim că regulile lui G' se construiesc astfel $f(A, a) = B \Rightarrow A \rightarrow aB$, $f(A, a) \in \Sigma_f \Rightarrow A \rightarrow a$.

Să presupunem acum că L este ambigu; atunci orice gramatică care îl generează, inclusiv G' , este ambiguă. Aceasta înseamnă că există un $p \in L(G')$ astfel încât

$$S \Rightarrow i_1 A_1 \Rightarrow i_1 i_2 A_2 \Rightarrow \dots \Rightarrow i_1 \dots i_{n-1} A_{n-1} \left\{ \begin{array}{l} \Rightarrow i_1 \dots i_n A'_n \Rightarrow \\ \Rightarrow i_1 \dots i_n A''_n \Rightarrow \end{array} \right\} \xRightarrow{*} p.$$

Deci există regulile $A_{n-1} \rightarrow i_n A'_n$ și $A_{n-1} \rightarrow i_n A''_n$, adică în automatul AFD avem

$$f(A_{n-1}, i_n) = A'_n, \quad f(A_{n-1}, i_n) = A''_n,$$

ceea ce contrazice faptul că AFD este determinist. \square

3.3 Forme normale pentru gramatici de tipul 2

Forma normală Chomsky.

Definiție 3.5 *O gramatică în forma normală Chomsky este o gramatică cu reguli de forma*

$$\begin{array}{l} A \rightarrow BC, \\ D \rightarrow i, \end{array}$$

unde $A, B, C, D \in V_N$ și $i \in V_T$. Se acceptă și regula de completare $S \rightarrow \lambda$ cu condiția ca S să nu apară în dreapta vreunei reguli.

Lema 3.1 (lema substituției). Fie G o gramatică de tipul 2 și $X \rightarrow uYv$ precum și $Y \rightarrow p_1 \dots p_n$ toate regulile din G care au Y în stânga. Atunci G este echivalentă cu o gramatică G' în care am făcut "substituțiile"; adică facem următoarea înlocuire

$X \rightarrow uYv$ se înlocuiește cu $X \rightarrow up_1v | \dots | up_nv$
(Regulile $Y \rightarrow p_1 | \dots | p_n$ le vom păstra neschimbate).

Demonstrație. Fie $p \in L(G)$ și $S \xrightarrow{*} p$. Punem în evidență doi pași consecutivi oarecare:

$$G : S \xrightarrow{*} r \Rightarrow s \Rightarrow t \xrightarrow{*} p.$$

Dacă în $r \Rightarrow s$ se utilizează regula $X \rightarrow uYv$ atunci în mod necesar în pasul următor se utilizează una din regulile $Y \rightarrow p_1 | \dots | p_n$, să presupunem $Y \rightarrow p_j$ (evident, este posibil ca această regulă să nu se aplice în pasul imediat următor, dar ea poate fi "adusă" în această poziție). Prin urmare

$$(A) \quad G : r = r'Xr'' \Rightarrow r'uYvr'' \Rightarrow r'up_jvr'' = t.$$

Acești doi pași se pot obține și în G' (într-un singur pas):

$$(B) \quad G' : r = r'Xr'' \Rightarrow r'up_jr'' = t.$$

Deci $S \xrightarrow[G']{*} p, p \in L(G')$ și $L(G) \subseteq L(G')$.

Invers, dacă $p \in L(G')$ și $S \xrightarrow[G']{*} p$, atunci dacă la un pas se utilizează o regulă nouă introdusă (pasul (B)), transformarea respectivă se poate obține și în G cu doi pași (pașii (A)); deci $p \in L(G)$ și $L(G') \subseteq L(G)$. \square

Corolar 3.5 Orice gramatică de tipul 2 este echivalentă cu o gramatică de același tip în care mulțimea de reguli nu conține redenumiri. (O redenumire este o regulă de forma $A \rightarrow B$, $A, B \in V_N$).

Intr-adevăr, dacă $A \rightarrow B \in P$ este o redenumire și $B \rightarrow p_1 | \dots | p_n$ sunt toate regulile care au B în stânga, efectuăm substituțiile, deci înlocuim regula $A \rightarrow B$ cu $A \rightarrow p_1 | \dots | p_n$. În cazul în care printre acestea apare o nouă redenumire, repetăm procedeuul. \square

Exemplu. Gramatica G_E care generează expresii aritmetice $E \rightarrow E + T | T, T \rightarrow T * F | F, F \rightarrow (E) | i$ se poate pune sub următoarea formă (fără redenumiri) :

$$\begin{aligned} E &\rightarrow E + T | T * F | (E) | i \\ T &\rightarrow T * F | (E) | i \\ F &\rightarrow (E) | i \end{aligned}$$

Teorema 3.6 (teorema lui Chomsky de existență a formei normale). *Orice gramatică independentă de context este echivalentă cu o gramatică în forma normală Chomsky. ema2*

Demonstrație. Putem porni de la o gramatică G care nu are redenumire și ale cărei reguli cu terminale au forma $A \rightarrow i$, $A \in V_N$, $i \in V_T$. De asemenea presupunem că G nu are reguli de ștergere.

Rezultă că regulile lui G au una din formele:

- (1) $A \rightarrow BC$,
- (2) $D \rightarrow i$,
- (3) $X \rightarrow X_1 \dots X_n$, $n > 2$.

Construim o gramatică $G' = (V'_N, V_T, S, P')$ unde $V_N \subseteq V'_N$ și P' conține toate regulile din P de forma (1) și (2). Fiecare regulă de forma (3) o înlocuim cu:

$$\begin{aligned} X &\rightarrow X_1 Z_1, \\ Z_1 &\rightarrow X_2 Z_2, \\ &\dots \\ Z_{n-2} &\rightarrow X_{n-1} X_n \end{aligned}$$

și includem neterminalele Z_1, \dots, Z_{n-2} (altele pentru fiecare regulă de forma (3) în V'_N).

Se poate relativ ușor arăta că $L(G) = L(G')$. De exemplu, dacă $u \Rightarrow v$ (direct) în G și de aplică o regulă de forma (1) sau (2), atunci evident derivarea respectivă se poate obține și în G' ; în cazul în care se aplică o regulă de forma (3), avem

$$G : u = u' X u'' \Rightarrow u' X_1 \dots X_n u'' = v.$$

Această derivare se poate obține și în G' în mai mulți pași și anume

$$G' : u = u' X u'' \Rightarrow u' X_1 Z_1 u'' \Rightarrow u' X_1 X_2 Z_2 u'' \Rightarrow \dots \Rightarrow u' X_1 \dots X_n u'' = v. \square$$

Observație. O gramatică ce are reguli de forma $A \rightarrow BC$, $A \rightarrow B$, $A \rightarrow a$ unde $A, B, C \in V_N$ și $a \in V_T$ spunem că este în *forma 2-canonice*. Este evident că orice gramatică de tip 2 este echivalentă cu o gramatică în formă 2-canonice.

Gramatici recursive

Definiție 3.6 *Un simbol neterminal X al unei gramatici de tipul 2 este recursiv dacă există o regulă de forma $X \rightarrow uXv$, $u, v \in V_G^*$.*

Dacă $u = \lambda$ ($v = \lambda$) simbolul X este stâng (drept) recursiv. O gramatică ce are cel puțin un simbol recursiv se numește recursivă. De exemplu,

gramatica G_E care generează expresiile aritmetice are două simboluri stâng recursive, E și T .

Existența simbolurilor stâng recursive poate provoca dificultăți în aplicarea algoritmilor de analiză top-down. Într-adevăr, într-o astfel de gramatică, încercarea de a construi arborele de derivare corespunzător unui cuvânt p prin aplicarea întotdeauna a primei reguli pentru simbolul cel mai din stânga, poate să conducă la un ciclu infinit (de exemplu în G_E s-ar obține $E \Rightarrow E + T \Rightarrow E + T + T \Rightarrow \dots$).

Teorema 3.7 *Orice limbaj de tipul 2 poate să fie generat de o gramatică fără recursie stângă.*

Demonstrație. Fie $G = (V_N, V_T, S, P)$ o gramatică de tipul 2; presupunem că G are un singur simbol recursiv X și fie

$$(A) \quad X \rightarrow u_1|u_2|\dots|u_n|Xv_1|\dots|Xv_m$$

toate regulile care au X în stânga. Construim gramatica $G' = (V'_N, V_T, S, P')$, unde $V_N \subset V'_N$, $P \subset P'$ cu excepția regulilor (A); acestea se înlocuiesc cu

$$\begin{aligned} X &\rightarrow u_1|u_2|\dots|u_n|u_1Y|u_2Y|\dots|u_nY, \\ Y &\rightarrow v_1|\dots|v_m|v_1Y|\dots|v_mY \end{aligned}$$

G' este de tipul 2 și nu are simboluri stâng recursive; se vede însă că Y este un simbol drept recursiv.

Fie $p \in L(G)$, $S \xrightarrow[G]{*} p$. Dacă în această derivare nu intervine simbolul recursiv, atunci evident că $S \xrightarrow[G']{*} p$. Să presupunem că X intervine la un anumit pas: $S \Rightarrow u \Rightarrow p$, unde $u = u'Xu''$. Putem aplica, începând de la u spre dreapta, în primul rând regulile pentru X și să urmărim numai subarboarele respectiv, deci

$$G : X \xrightarrow[G]{\Rightarrow} Xv_{j_1} \xrightarrow[G]{\Rightarrow} Xv_{j_2}v_{j_1} \xrightarrow[G]{\Rightarrow} \dots \xrightarrow[G]{\Rightarrow} Xv_{j_s}\dots v_{j_1} \xrightarrow[G]{\Rightarrow} u_jv_{j_s}\dots v_{j_1}.$$

Aceeași formă propozițională o putem obține și în gramatica G' astfel

$$G' : X \xrightarrow[G']{\Rightarrow} u_jY \xrightarrow[G']{\Rightarrow} u_jv_{j_s}Y \xrightarrow[G']{\Rightarrow} \dots \xrightarrow[G']{\Rightarrow} u_jv_{j_s}\dots v_{j_1}.$$

Prin urmare avem $S \xrightarrow[G']{\Rightarrow} u \xrightarrow[G']{\Rightarrow} p$, adică $p \in L(G')$ și $L(G) \subseteq L(G')$.

Analog, $L(G') \subseteq L(G)$. \square

Forma normală Greibach.

Definiție 3.7 *O gramatică în forma normală Greibach este o gramatică cu reguli de forma*

$$A \rightarrow ip, \text{ unde } A \in V_N, i \in V_T, p \in V_N^*.$$

Se acceptă și regula de completare $S \rightarrow \lambda$ cu condiția ca S să nu apară în dreapta vreunei reguli.

Teorema 3.8 *(Teorema de existență a formei normale Greibach). Orice gramatică de tipul 2 este echivalentă cu o gramatică în forma normală Greibach.*

Demonstrație. Fie G o gramatică de tipul 2 în forma normală Chomsky și fie $V_N = \{S = X_1, X_2, \dots, X_n\}$. Vom construi o gramatică echivalentă care să satisfacă cerințele din forma normală Greibach în mai multe etape.

Etapa I. Vom modifica regulile de generare astfel încât toate regulile care nu sunt de forma $X \rightarrow i$ să satisfacă condiția $X_j \rightarrow X_k p$, $j < k$, $p \in V_N^*$. Acest lucru îl facem cu un algoritm pe care îl prezentăm într-un limbaj nestandard de publicare (tip PASCAL):

```

      j := 1;
e1:  begin
      Se elimină recursiile stângi; neterminalele
      noi le notăm cu  $Y_1, Y_2, \dots$ 
      end
      if  $j = n$  then STOP;
      j := j + 1;
      l := 1;
e2:  begin
      Fie  $X_j \rightarrow X_l p$ ,  $p \in V_N^*$  și  $X_l \rightarrow p_1 \dots p_m$ 
      toate regulile care au  $X_l$  în stânga; se efectuează toate substituțiile.
      end
      l := l + 1;
      if  $l < j - 1$  then goto e2
      goto e1

```

Să observăm că pentru $j = 1$ și după eliminarea recursiilor stângi condiția cerută este evident îndeplinită; în plus, dacă au fost recursii, vom avea reguli cu partea stângă neterminale noi Y_1, Y_2, \dots . Mai departe, luăm toate regulile care au în stânga X_2 ($j := j + 1 = 2$) și efectuăm substituțiile; acestea se vor transforma în $X_2 \rightarrow X_k p$ cu $k \geq 2$ și după o nouă eliminare a recursiilor stângi vom avea $k > 2$ plus reguli cu partea stângă neterminale

noi. În felul acesta toate regulile care au în stânga X_1 și X_2 satisfac condiția cerută; în continuare $j := j + 1 = 3$, etc.

Etapa II. Avem acum trei categorii de reguli:

- (1) $X_j \rightarrow i$;
- (2) $X_j \rightarrow X_k p$, $j < k$, $p \in V_N^*$;
- (3) $Y \rightarrow iq$, $q \in V_N^*$, $i = 1, \dots, m$.

Aranjăm toate neterminalele într-un șir unic, la început Y_1, \dots, Y_m apoi X_1, \dots, X_n și le redenumim, de exemplu cu X_1, \dots, X_{m+n} :

$$\begin{array}{cccccccc} Y_1, & Y_2, & \dots, & Y_m, & X_1, & X_2, & \dots, & X_n \\ X_1, & X_2, & \dots, & X_m, & X_{m+1}, & X_{m+2}, & \dots, & X_{m+n} \end{array}$$

Vom nota $n + m = N$. În felul acesta regulile gramaticii vor avea numai formele (1) și (2).

Etapa III. Toate regulile care au X_N în stânga vor avea forma (1). Fie $X_{n-1} \rightarrow X_N p_1 | \dots | X_N p_n$ toate regulile care au X_{n-1} în stânga și care nu sunt de forma (1). Efecuăm substituțiile lui X_N ; în acest fel regulile care au X_N și X_{n-1} în stânga satisfac cerințele din forma normală Greibach. În continuare, considerăm toate regulile care au X_{n-2} în stânga și efectuăm substituțiile, etc. \square

Forma normală operator

Una din formele importante pentru gramatici independente de context, utilizată în analiza sintactică prin metoda precedenței, este forma operator a acestor gramatici.

Definiție 3.8 *O gramatică independentă de context $G = (V_N, V_T, S, P)$ se spune că este în forma normală operator dacă oricare ar fi producția $A \rightarrow \beta \in P$, în β nu apar două neterminale (variabile) consecutive, adică*

$$P \subseteq V_N \times [(V_N \cup V_T)^* \setminus (V_N \cup V_T)^* V^2 (V_N \cup V_T)^*].$$

Teorema 3.9 *Orice gramatică independentă de context este echivalentă cu o gramatică în forma normală operator.*

Demonstrație. Fie $G = (V_N, V_T, S, P)$ o gramatică de tipul 2 și $L(G)$ limbajul generat. Fără a restrânge generalitatea presupunem că $\lambda \notin L(G)$ și G este în forma 2-canonică (regulile sunt de forma $A \rightarrow BC$, $A \rightarrow B$, $A \rightarrow a$ vezi teorema ??). Definim o gramatică echivalentă $G' = (V'_N, V_T, S, P')$ astfel: $V'_N = \{S\} \cup (V_N \times V_T)$, iar $P' = P_1 \cup P_2 \cup P_3 \cup P_4$ unde

- i) $P_1 = \{S \rightarrow (S, a) a \mid a \in V_T\}$;
- ii) $P_2 = \{(A, a) \rightarrow \lambda \mid A \in V_N, a \in V_T, A \rightarrow a \in P\}$;
- iii) $P_3 = \{(A, a) \rightarrow (B, a) \mid A, B \in V_N, a \in V_T, A \rightarrow B \in P\}$;
- iv) $P_4 = \{(A, a) \rightarrow (B, b)b(C, a) \mid A, B, C \in V_N, a, b \in V_T, A \rightarrow BC \in P\}$.

Să observăm că G' este în forma normală operator. Pentru a demonstra că $L(G) = L(G')$ vom defini mai întâi o funcție $\phi : P_2 \cup P_3 \cup P_4 \rightarrow P$ astfel:

$$\begin{aligned}\phi((A, a) \rightarrow \lambda) &= A \rightarrow a \text{ pentru } (A, a) \rightarrow \lambda \in P_2; \\ \phi((A, a) \rightarrow (B, a)) &= A \rightarrow B \text{ pentru } (A, a) \rightarrow (B, a) \in P_3; \\ \phi((A, a) \rightarrow (B, b)b(C, a)) &= A \rightarrow BC \text{ pentru } (A, a) \rightarrow (B, b)b(C, a) \in P_4.\end{aligned}$$

Funcția ϕ se extinde în mod natural la $\phi' : (P_2 \cup P_3 \cup P_4)^* \rightarrow P^*$. Vom arăta că în gramatica G , $\forall w \in V_T^*$, $\forall a \in V_T$ are loc derivarea extrem dreaptă $A \xrightarrow[G]{*} wa$ folosind producțiile $\pi_1, \pi_2, \dots, \pi_n$ dacă și numai dacă există în P' producțiile $\pi'_1, \pi'_2, \dots, \pi'_n$ astfel ca $\phi(\pi'_i) = \pi_i$, $1 \leq i \leq n$ și în G' are loc derivarea extrem dreaptă $(A, a) \xrightarrow[G']{*} w$ folosind producțiile $\pi'_1, \pi'_2, \dots, \pi'_n$.

Să demonstrăm afirmația prin inducție după n , lungimea derivării.

Dacă $n = 1$ atunci $w = \lambda$, $A \rightarrow a \in P$ și în P' există producția $(A, a) \rightarrow \lambda$, deci $(A, a) \Rightarrow \lambda$ și $\phi((A, a) \rightarrow \lambda) = A \rightarrow a$.

Invers, dacă $(A, a) \Rightarrow w$ în G' atunci $w = \lambda$ (după forma producțiilor din G') și are loc proprietatea enunțată.

Să presupunem afirmația adevărată pentru derivări de lungime cel mult $n - 1$ și să o demonstrăm pentru derivări de lungime $n > 1$. Fie așadar $A \xrightarrow[G]{*} wa$ o derivare de lungime n în gramatica G și punem în evidență prima derivare directă. Distingem două cazuri:

I. Prima producție utilizată în derivare este $A \rightarrow B$. Atunci,

$$A \Rightarrow B \xrightarrow[G]{*} wa$$

și conform ipotezei inductive avem în G' o derivare $(B, a) \xrightarrow[G']{*} w$ (de lungime $n - 1$) cu producții satisfăcând condițiile arătate. Dar cum $A \rightarrow B \in P$, în P' avem producția $(A, a) \rightarrow (B, a)$ deci $(A, a) \xrightarrow[G']{*} w$ în gramatica G' .

II. Prima producție este de forma $A \rightarrow BC$. Atunci

$$A \Rightarrow BC \xrightarrow[G]{*} wa.$$

În acest caz $wa = ubva$ (conform lemei de localizare), astfel că $B \xrightarrow[G]{*} ub$ și

$C \xrightarrow[G]{*} va$. După ipoteza inductivă, vom avea în G' derivările:

$$(B, b) \xrightarrow[G']{*} u, \quad (C, a) \xrightarrow[G']{*} v.$$

Cum $A \rightarrow BC \in P$ vom avea în P' producția $(A, a) \rightarrow (B, b)b(C, a)$ și în G' putem scrie derivarea extrem dreaptă

$$(A, a) \xRightarrow[G']{*} (B, b)b(C, a) \xRightarrow[G']{*} (B, b)bv \xRightarrow[G']{*} ubv = w$$

și producțiile care s-au aplicat îndeplinesc condițiile din enunț.

În mod analog se demonstrează reciproca.

Din această afirmație, luând în particular $A = S$, obținem:

$$S \xRightarrow[G']{*} wa \Leftrightarrow (S, a) \xRightarrow[G']{*} w, \quad \forall w \in V_T^*, \forall a \in V_T.$$

Cum în G' există și producția $S \rightarrow (S, a)a$, am găsit: $wa \in L(G)$ dacă și numai $wa \in L(G')$, deci cele două gramatici sunt echivalente.

Pentru a încheia demonstrația trebuie să considerăm și cazul $\lambda \in L(G)$. Aplicăm construcția de mai sus unei gramatici ce generează $L(G) \setminus \{\lambda\}$ și obținem $G' = (V'_N, V_T, S, P')$ gramatica operator corespunzătoare. Considerăm acum gramatica $G_1 = (V_{N_1}, V_T, S_1, P_1)$ unde $V_1 = V' \cup \{S_1\}$, $P_1 = P' \cup \{S_1 \rightarrow \lambda, S_1 \rightarrow S\}$ care este în forma normală operator și $L(G_1) = L(G)$. \square

3.4 Automate push-down (APD)

Automatele push-down sunt mecanisme pentru recunoașterea limbajelor independente de context.

Un APD se compune din (vezi figura 4.6):

1. O bandă de intrare care conține simboluri ale unui *alfabet de intrare*; aceste simboluri constituie pe o bandă un anumit cuvânt peste alfabetul de intrare. Banda se mișcă numai spre stânga;
2. O *memorie push-down* (memorie inversă, stivă, pilă, etc) care conține simboluri ale unui alfabet propriu, numit *alfabetul memoriei push-down*. Această memorie funcționează ca o stivă - ultimul introdus, primul extras (Last In, First Out);
3. Un *dispozitiv de comandă* care se află permanent într-o anumită stare internă aparținând unei mulțimi finite de stări. Dispozitivul de comandă posedă un *dispozitiv de citire* de pe banda de intrare și un *dispozitiv de scriere-citire* în memoria push-down.

Ca și un automat finit, un automat push-down funcționează în pași discreți; un pas de funcționare comportă:

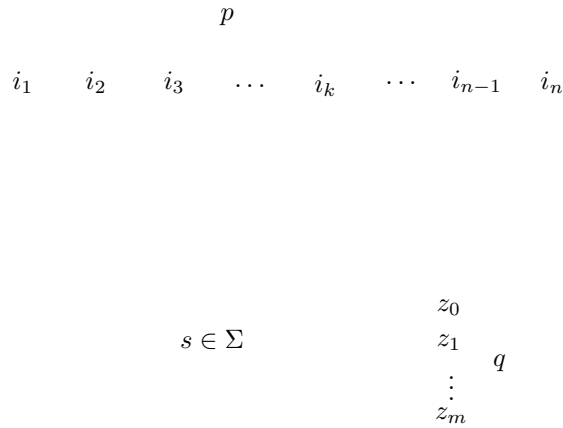


Figura 3.6: Reprezentare schematică a unui automat push-down

1. Dispozitivul de comandă citește simbolul de pe banda de intrare din dreptul dispozitivului de citire și mută banda cu o poziție spre stânga.
2. În funcție de starea internă, de simbolul citit și de simbolul din vârful memoriei push-down dispozitivul de comandă efectuează operațiile:
 - (a) Trece într-o nouă stare;
 - (b) Scrie în memoria push-down un anumit cuvânt peste alfabetul memoriei push-down; în particular, acesta poate să fie cuvântul vid, ceea ce are ca efect ștergerea simbolului din vârful memoriei push-down.

Funcționarea unui APD se termină în general după ce s-a citit ultimul simbol al cuvântului scris pe banda de intrare dar este posibil ca el să efectueze un anumit număr de pași, citind de fiecare dată de pe bandă cuvântul vid λ . De asemenea, este posibil ca în timpul funcționării, deci înainte de a ajunge la ultimul simbol, automatul să se blocheze. De exemplu, automatul ajunge într-o configurație (stare, simbol pe bandă, simbol în vârful memoriei push-down) inadmisibilă sau se golește memoria push-down dar nu s-a epuizat cuvântul de pe bandă, etc.

Matematic, un APD se definește astfel:

Definiție 3.9 *Un automat push-down este un sistem*

$$APD = (\Sigma, I, Z, f, s_0, z_0)$$

unde:

Σ este mulțimea de stări (finită și nevidă);
 I este alfabetul de intrare;
 Z este alfabetul memoriei push-down;
 $f : \Sigma \times (I \cup \{\lambda\}) \times Z \longrightarrow \mathcal{P}(\Sigma \times Z^*)$ este funcția de evoluție;
 $s_0 \in \Sigma$ este starea inițială;
 $z_0 \in Z$ este simbolul inițial al memoriei push-down.

Un APD are în general o funcțioanre nedeterministă, $\text{card } f(s, i, z) \geq 1$; mulțimea automatelor push-down deterministe formează o clasă specială.

În termenii funcției de evoluție, un pas de evoluție comportă citirea simbolului i de pe bandă, citirea simbolului z din vârful memoriei push-down, apoi, în funcție de starea internă s și de aceste două simboluri, automatul trece într-o nouă stare s' și scrie în memoria push-down un cuvânt $q \in Z^*$ astfel încât $(s', q) \in f(s, i, z)$. În cazul în care $f(s, i, z) = \emptyset$ evoluția este oprită; este situația în care automatul se blochează.

O stare a automatului (sau configurație) este un sistem $\delta = (s, p, q)$ unde $s \in S$ este starea internă, $p \in I^*$ este subcuvântul de pe banda de intrare rămas de citit (inclusiv simbolul din dreptul dispozitivului de citire), iar $q \in Z^*$ este subcuvântul din memoria push-down.

Vom spune că un APD trece direct din starea $\delta_1 = (s_1, p_1, q_1)$ în starea $\delta_2 = (s_2, p_2, q_2)$ și vom scrie $\delta_1 \mapsto \delta_2$ dacă se execută un pas de evoluție; dacă $p_1 = ip'_1, q_1 = zq'_1$ putem avea $(s_2, q) \in f(s_1, i, z)$ și atunci $p_2 = p'_1, q_2 = qq'_1$ sau $(s_2, q) \in f(s_1, \lambda, z)$ și atunci $p_2 = p_1, q_2 = qq'_1$.

Vom spune că automatul evoluează (fără specificația direct) din starea δ' în stare δ'' și vom scrie $\delta' \xrightarrow{*} \delta''$ dacă:

- (1) $\delta' = \delta''$;
- (2) $\exists \delta_1, \dots, \delta_n$ astfel încât $\delta' = \delta_1 \mapsto \delta_2 \mapsto \dots \mapsto \delta_{n-1} \mapsto \delta_n = \delta''$.

Limbaajul recunoscut de un APD se poate defini în două moduri:

- (1) Limbaajul recunoscut de un APD cu golirea memoriei push-down, este, prin definiție

$$L(APD) = \{p \mid p \in I^* (s_0, p, z_0) \xrightarrow{*} (s, \lambda, \lambda)\}.$$

Aceasta înseamnă că, pornind din starea internă s_0 și având în vârful memoriei push-down simbolul z_0 , cu ajutorul cuvântului p de pe banda de intrare, automatul poate să evolueze astfel încât să golească memoria push-down după citirea cuvântului. Menționăm că golirea memoriei push-down nu trebuie neapărat să coincidă cu citirea ultimului simbol al lui p ; este posibil ca automatul să mai efectueze câțiva pași citind de pe bandă simbolul λ .

- (2) Limbaajul recunoscut de un APD cu stări finale; în definiția automatului se adaugă o submulțime Σ_f a lui Σ numită mulțimea de stări finale. Prin definiție, limbaajul recunoscut de un APD cu stări finale este:

$$L(APD) = \{p \mid p \in I^* (s_0, p, z_0) \xrightarrow{*} (s, \lambda, q), s \in \Sigma_f, q \in Z^*\}.$$

Prin urmare, este necesar ca după citirea lui p , eventual după încă câțiva pași, APD să ajungă într-o stare finală. Vom vedea că cele două definiții sunt echivalente.

Limbaje recunoscute de automate push-down cu golirea memoriei.

Vom arăta că familia limbajelor recunoscute de APD cu stări finale coincide cu familia limbajelor independente de context. În felul acesta, APD constituie mecanisme analitice de definire a limbajelor de tipul 2.

Teorema 3.10 *Un limbaj este independent de context dacă și numai dacă este recunoscut de un automat push-down cu golirea memoriei push-down.*

Demonstrație. Partea I $E \in \mathcal{L}_2 \Rightarrow E = L(APD)$.

Fie $G = (V_N, V_T, S, P)$ o gramatică de tipul 2 în forma normală Greibach care generează limbajul E . Construim un automat pushdown astfel: $APD = (\{s\}, V_T, V_N, f, s, S)$, funcția de evoluție fiind definită de:

$$A \rightarrow ip \in P \Rightarrow (s, p) \in f(s, i, A),$$

altfel \emptyset .

Fie $p \in L(G)$, $p = i_1 \dots i_n$, $S \xrightarrow[G]{*} p$. Această derivare trebuie să aibă forma (extrem stângă):

$$(A) \quad S \Rightarrow i_1 X_1 u_1 \Rightarrow i_1 i_2 X_2 u_2 u_1 \Rightarrow i_1 i_2 i_3 X_3 u_3 u_2 u_1 \Rightarrow \dots \Rightarrow i_1 \dots i_n,$$

unde $u_1, u_2, u_3, \dots \in V_N^* = Z^*$.

Observație. Aparent, partea $u_s u_{s-1} \dots u_1$ se mărește cu fiecare derivare directă. În realitate, unele din cuvintele u_j sunt vide, și anume atunci când se aplică o regulă de forma $X \rightarrow i$; în particular, în ultimele derivări directe se aplică numai reguli de această formă.

Avem

$$\begin{aligned} S \rightarrow i_1 X_1 u_1 &\Rightarrow (s, X_1 u_1) \in f(s, i_1, S), \\ X_1 \rightarrow i_2 X_2 u_2 &\Rightarrow (s, X_2 u_2) \in f(s, i_2, X_1), \\ X_2 \rightarrow i_3 X_3 u_3 &\Rightarrow (s, X_3 u_3) \in f(s, i_3, X_2), \\ \dots & \end{aligned}$$

Prin urmare automatul poate să aibă următoarea evoluție:

$$(s, i_1 i_2 i_3 i_4 \dots i_n, S) \xrightarrow{*} (s, i_2 i_3 i_4 \dots i_n, X_1 u_1) \xrightarrow{*}$$

$$\vdash (s, i_3 i_4 \dots i_n, X_2 u_2 u_1) \vdash (s, i_4 \dots i_n, X_3 u_3 u_2 u_1) \vdash \dots$$

Dacă comparăm această evoluție cu derivarea (A) putem observa că pe banda de intrare avem la fiecare pas partea complementară a cuvântului (față de derivare) iar în memoria push-down se reproduce partea de neterminale din formele propoziționale ale derivării. Cum în derivare se ajunge la $i_1 \dots i_n$, în evoluție se va ajunge la (s, λ, λ) .

Deci $p \in L(APD)$ și $L(G) \subseteq L(APD)$.

Fie acum $p \in L(APD)$; va trebui să arătăm că $p \in L(G)$, deci că $S \xrightarrow[G]{*} p$. Vom arăta o implicație ceva mai generală, și anume, pentru orice $u \in V_N^*$, avem

$$(s, p, u) \vdash^* (s, \lambda, \lambda) \Rightarrow u \xrightarrow[G]{*} p.$$

În particular, dacă $u = S$ obținem implicația dorită.

Procedăm prin inducție asupra lungimii lui p .

Dacă $|p| = 1$, atunci $p = i, u = X$ iar evoluția va avea un singur pas $(s, i, X) \vdash (s, \lambda, \lambda)$, deci $(s, \lambda) \in f(s, i, X)$ și $X \rightarrow i \in P$. Putem scrie $u = X \xrightarrow[G]{*} i = p$.

Presupunem că implicația este adevărată pentru un cuvânt $|p| = l$ și considerăm un p astfel încât $|p| = l + 1$. Fie i și X primele simboluri din p și u , deci $p = ip'$ și $u = Xu'$. În evoluția $(s, p, u) \vdash^* (s, \lambda, \lambda)$ punem în evidență prima evoluție directă

$$(s, p, u) = (s, ip', Xu') \vdash (s, p', vu') \vdash^* (s, \lambda, \lambda).$$

Din definiția evoluției directe rezultă că $(s, v) \in f(s, i, X)$ deci $X \rightarrow iv \in P$. Pe de altă parte din ipoteza inductivă rezultă că $vu' \xrightarrow[G]{*} p'$. Avem

$$u = Xu' \xrightarrow[G]{*} ivu' \xrightarrow[G]{*} ip' = p,$$

ceea ce demonstrează implicația.

Prin urmare $p \in L(G)$ și $L(APD) \subseteq L(G)$, de unde $L(G) = L(APD)$. \square

Partea II. $E = L(APD)$ $E \in \mathcal{L}_2$

Fie $APD = (\Sigma, I, Z, f, s_0, z_0)$. Construim G de forma $G = (V_N, V_T, S, P)$ unde $V_N = \{s_0\} \cup \{(s, z, s') \mid s, s' \in \Sigma, z \in Z\}$, $V_T = I$, $S = s_0$, iar regulile de generare le definim astfel:

- (1) $s_0 \rightarrow (s_0, z_0, s)$, $\forall s \in \Sigma$;
- (2) Dacă $(s_1, z_1 \dots z_m) \in f(s, i, z)$ vom introduce în P reguli de forma

$$(s, z, s') \rightarrow i(s_1, z_1, s_2)(s_2, z_2, s_3) \dots (s_m, z_m, s'),$$

unde $s', s_2, \dots, s_m \in \Sigma$;

(3) Dacă $(s', \lambda) \in f(s, i, z)$ vom introduce în P reguli de forma

$$(s, z, s') \rightarrow i,$$

unde $s' \in \Sigma$.

Să observăm că gramatica astfel construită este independentă de context, și anume în forma normală Greibach.

Fie $p \in L(APD)$, deci $(s_0, p, z_0) \xrightarrow{*} (s', \lambda, \lambda)$; trebuie să arătăm că $s_0 \xrightarrow[G]{*} p$.

Vom arăta implicația ceva mai generală

$$(s, p, z) \xrightarrow{*} (s', \lambda, \lambda) \Rightarrow (s, z, s') \xrightarrow[G]{*} p.$$

În particular pentru $s = s_0, z = z_0$ rezultă $(s_0, z_0, s') \xrightarrow[G]{*} p$ și putem scrie

$$s_0 \Rightarrow (s_0, z_0, s') \xrightarrow[G]{*} p, \text{ adică } p \in L(G).$$

Procedăm prin inducție asupra lungimii evoluției l .

Dacă $l = 1$ atunci $(s, p, z) \xrightarrow{*} (s', \lambda, \lambda)$, deci $p = i$ și $(s', \lambda) \in f(s, i, z)$ și $(s, z, s') \rightarrow i$ este o regulă, adică putem scrie $(s, z, s') \Rightarrow i = p$.

Presupunem că implicația este adevărată pentru evoluții de lungime oarecare l și considerăm o evoluție de lungime $l + 1$; punem în evidență prima evoluție directă

$$(s, p, z) = (s, i_1 p', z) \xrightarrow{*} (s_1, p', z_1 \dots z_m) \xrightarrow{*} (s', \lambda, \lambda).$$

Descompunem cuvântul p' în forma $p' = p_1 \dots p_m$ astfel încât

$$\begin{aligned} (s_1, p_1, z_1) &\xrightarrow{*} (s_2, \lambda, \lambda), \\ (s_2, p_2, z_2) &\xrightarrow{*} (s_3, \lambda, \lambda), \\ &\dots \\ (s_m, p_m, z_m) &\xrightarrow{*} (s', \lambda, \lambda). \end{aligned}$$

Observație. Putem pune în evidență felul în care se definește cuvântul p_1 urmărind evoluția lui APD;

$$\begin{array}{ccc} (s_1, i_1 i_2 \dots i_n, z_1 z_2 \dots z_m) & \xrightarrow{*} & (s'_1, i_2 i_3 \dots i_n, q z_2 \dots z_m) & \xrightarrow{*} \\ (a) & & (b) & \\ \dots & \xrightarrow{*} & (s_2, i_{j_1} \dots i_n, z_2 \dots z_m) & \\ & & (c) & \end{array}$$

La primul pas (situația a) automatul este în starea s_1 , pe bandă este i_1 iar în memoria push-down este z_1 . După efectuarea unui pas, automatul trece în starea s'_1 , mută banda cu o poziție spre stânga, extrage pe z_1 și scrie în memoria push-down un cuvânt q (situația b). Se poate observa că z_2 a "coborât"; cum știm că memoria push-down se golește ($p \in L(APD)$), trebuie ca la un moment dat z_2 să ajungă în vârful stivei (situația c). În acest moment partea din p citită va fi p_1 iar starea în care a ajuns automatul o notăm cu s_2 . Este clar că dacă pe bandă am avea scris numai p_1 am avea evoluția $(s_1, p_1, z_1) \mapsto (s_2, \lambda, \lambda)$.

Analog p_2, \dots, p_m .

Din definiția derivării directe $(s, i_1 p', z) \mapsto (s_1, p', z_1 \dots z_m)$ avem $(s_1, z_1 \dots z_m) \in f(s, i_1, z)$ iar în P va exista regula

$$(s, z, s') \rightarrow i_1(s_1, z_1, s_2)(s_2, z_2, s_3) \dots (s_m, z_m, s')$$

unde luăm stările s_2, \dots, s_m cele rezultate la descompunerea lui p' . Pe de altă parte, din ipoteza inductivă, avem

$$\begin{aligned} (s_1, z_1, s_2) &\stackrel{*}{\Rightarrow} p_1, \\ (s_2, z_2, s_3) &\stackrel{*}{\Rightarrow} p_2, \\ \dots & \\ (s_m, z_m, s') &\stackrel{*}{\Rightarrow} p_m. \end{aligned}$$

Putem scrie derivarea

$$(s, z, s') \Rightarrow i_1(s_1, z_1, s_2)(s_2, z_2, s_3) \dots (s_m, z_m, s') \stackrel{*}{\Rightarrow} i_1 p_1 \dots p_m = i_1 p' = p.$$

După cum am văzut, rezultă mai departe $p \in L(G)$ și $L(APD) \subseteq L(G)$.

Pentru a demonstra incluziunea inversă, vom arăta mai întâi implicația

$$(s, z, s') \stackrel{*}{\Rightarrow} p(s_1, z_1, s_2) \dots (s_m, z_m, s') \text{ implică } (s, p, z) \mapsto (s_1, \lambda, z_1 \dots z_m).$$

Procedăm prin inducție asupra lungimii derivării l .

Dacă $l = 1$ atunci $p = i$ și se aplică regula

$$(s, z, s') \rightarrow i(s_1, z_1, s_2) \dots (s_m, z_m, s')$$

deci $(s_1, z_1 \dots z_m) \in f(s, i, z)$ și $(s, i, z) \mapsto (s_1, \lambda, z_1 \dots z_m)$.

Presupunem că implicația este adevărată pentru l oarecare și considerăm o derivare de lungime $l + 1$. Fie $p = p' i$ și punem în evidență ultimul pas.

$$\begin{aligned} (s, z, s') &\stackrel{*}{\Rightarrow} p'(s'_{j-1}, z'_{j-1}, s'_j)(s_j, z_j, s_{j+1}) \dots (s_m, z_m, s') \\ &\Rightarrow p' i(s_1, z_1, s_2) \dots (s_{j-1}, z_{j-1}, s_j)(s_j, z_j, s_{j+1}) \dots (s_m, z_m, s'), \end{aligned}$$

unde $s'_j = s_j$; la ultimul pas s-a aplicat regula

$$(s'_{j-1}, z'_{j-1}, s'_j) \rightarrow i(s_1, z_1, s_2) \dots (s_{j-1}, z_{j-1}, s_j).$$

Rezultă $(s_1, z_1 \dots z_{j-1}) \in f(s'_{j-1}, i, z'_{j-1})$ și putem scrie evoluția $(s'_{j-1}, i, z'_{j-1}) \mapsto (s_1, \lambda, z_1 \dots z_{j-1})$.

Pe de altă parte, conform ipotezei inductive, avem

$$(s, p', z) \xrightarrow{*} (s'_{j-1}, \lambda, z'_{j-1} z_j \dots z_m)$$

Prin urmare

$$(s, p, z) = (s, p' i, z) \xrightarrow{*} (s'_{j-1}, i, z'_{j-1} z_j \dots z_m) \mapsto (s_1, \lambda, z_1 \dots z_m)$$

și implicația este demonstrată.

Fie acum $p \in L(G)$, deci $s_0 \xrightarrow[G]{*} p$. Ținând seama de forma regulilor

din G , în această derivare se va aplica prima dată o regulă de forma (1), apoi regula de forma (2) iar la sfârșit reguli de forma (3). La aplicarea regulilor (2) putem rescrie la fiecare pas simbolul neterminal cel mai din stânga, deci să obținem o derivare extrem stângă. Să observăm că în acest caz structura formelor propoziționale intermediare este cea menționată, $p(s_1, z_1, s_2)(s_2, z_2, s_3) \dots (s_m, z_m, s')$.

Prin urmare, derivarea va avea forma

$$s_0 \Rightarrow (s_0, z_0, s') \xrightarrow{*} p(s_1, z_1, s_2) \dots (s_m, z_m, s') \xrightarrow{*} p.$$

Trebuie să avem regulile $(s_j, z_j, s_{j+1}) \rightarrow \lambda, j = 1, \dots, m, s_{m+1} = s'$ și putem scrie

$$(s_0, p, z_0) \xrightarrow{*} (s_1, \lambda, z_1 \dots z_m) \mapsto (s_2, \lambda, z_2 \dots z_m) \mapsto \dots \mapsto (s', \lambda, \lambda)$$

adică $p \in L(APD)$ și $L(G) \subseteq L(APD)$. \square

Automate push-down cu stări finale. Vom nota un automat push-down cu stări finale cu APD_f .

Teorema 3.11 *Un limbaj este recunoscut de un automat push-down dacă și numai dacă este recunoscut de un automat push-down cu stări finale.*

Demonstrație. Partea I $E \in L(APD) \Rightarrow E \in L(APD_f)$.

Dacă $APD = (\Sigma, I, Z, f, s_0, z_0)$ construim un automat push-down cu stări finale astfel

$$APD_f = (\Sigma \cup \{s'_0, s_f\}, I, Z \cup \{z'_0\}, f', s'_0, z'_0)$$

unde mulțimea de stări finale este $\{s_f\}$ iar funcția de evoluție este definită de:

$$\begin{aligned} f'(s, i, z) &= f(s, i, z), s \in \Sigma, i \in I \cup \{\lambda\}, z \in Z; \\ f'(s'_0, \lambda, z'_0) &= (s_0, z_0 z'_0); \end{aligned}$$

$f(s, \lambda, z'_0) = (s_f, \lambda), s \in \Sigma;$
 în rest \emptyset .

Fie $p \in L(APD)$; atunci $(s_0, p, z_0) \xrightarrow{*} (s, \lambda, \lambda)$. Evident că aceeași evoluție o poate avea și automatul push-down cu stări finale. Putem scrie în APD_f evoluția

$$(APD_f) : (s'_0, p, z'_0) \xrightarrow{*} (s_0, \lambda, z'_0) \xrightarrow{*} (s, \lambda, \lambda),$$

deci $p \in L(APD_f)$ și $L(APD) \subseteq L(APD_f)$.

Invers, fie $p \in L(APD_f)$, atunci (în APD_f)

$$(s'_0, p, z'_0) \xrightarrow{*} (s, p, z_0 z'_0) \xrightarrow{*} (s_f, \lambda, q).$$

Ultimul pas trebuie să fie de forma $(s, \lambda, z'_0) \xrightarrow{*} (s_f, \lambda, \lambda)$ pentru că nu există altă valoare a lui f care să ne ducă într-o stare finală. Deci (în APD_f)

$$(s'_0, p, z_0 z'_0) \xrightarrow{*} (s, \lambda, z'_0) \xrightarrow{*} (s_f, \lambda, \lambda)$$

și putem scrie în APD evoluția $(s_0, p, z_0) \xrightarrow{*} (s, \lambda, \lambda)$, adică $p \in L(APD)$ și $L(APD_f) \subseteq L(APD)$. \square

Partea II $E = L(APD_f) \Rightarrow E \in L(APD)$.

Fie $APD_f = (\Sigma, I, Z, f, s_0, z_0, \Sigma_f)$ un automat push-down cu stări finale (mulțimea stărilor finale este Σ_f) și construim un APD astfel

$$APD = (\Sigma \cup \{s'_0, s'\}, I, Z \cup \{z'_0\}, f', s'_0, z'_0)$$

unde

$f'(s, i, z) = f(s, i, z), s \in \Sigma, i \in I \cup \{\lambda\}, z \in Z;$
 $f(s'_0, \lambda, z'_0) = (s, z_0 z'_0);$
 $f(s, \lambda, z) = (s', \lambda), s \in \Sigma_f \cup \{s'\}, z \in Z \cup \{z'_0\};$
 în rest \emptyset .

Fie $p \in L(APD_f)$, atunci $(s_0, p, z_0) \xrightarrow{*} (s, \lambda, q), s \in \Sigma_f$. Este evident că în APD avem evoluția $(s_0, p, z_0) \xrightarrow{*} (s, \lambda, q)$. Putem scrie

$$APD : (s'_0, p, z'_0) \xrightarrow{*} (s_0, p, z_0 z'_0) \xrightarrow{*} (s, \lambda, q z'_0) \xrightarrow{*} (s', \lambda, \lambda),$$

deci $p \in L(APD)$ și $L(APD_f) \subseteq L(APD)$.

Invers, fie $p \in L(APD)$. Avem

$$APD : (s'_0, p, z'_0) \xrightarrow{*} (s_0, p, z_0 z'_0) \xrightarrow{*} (s, \lambda, \lambda).$$

Simbolul z'_0 nu poate fi șters decât cu o regulă de forma $f(s, \lambda, z) = (s', \lambda), s \in \Sigma_f \cup \{s'\}$, deci APD trebuie să ajungă într-o stare $s \in \Sigma_f$, apoi să rămână în s' .

$$APD : (s_0, p, z_0 z'_0) \xrightarrow{*} (s, \lambda, q z'_0), s \in \Sigma_f.$$

Putem scrie

$$APD_f : (s_0, p, z_0) \xrightarrow{*} (s, \lambda, q), s \in \Sigma_f$$

și deci $p \in L(APD_f)$, adică $L(APD) \subseteq L(APD_f)$. \square

3.5 Automate push-down deterministe

Funcționarea unui APD este în general nedeterministă, $\text{card } f(s, i, z) \geq 1$. Pentru ca un APD să aibă o funcționare deterministă nu este suficient să impunem condiția $\text{card } f(s, i, z) = 1$, deoarece dacă pentru un anumit $s \in \Sigma$ și $z \in Z$ avem $f(s, \lambda, z) \neq \emptyset$ și $f(s, i, z) \neq \emptyset$, putem face un pas citind λ sau citind i .

Definiție 3.10 *Un automat push-down este determinist dacă*

- (1) $\text{card } f(s, i, z) \geq 1, s \in \Sigma, i \in I \cup \{\lambda\}, z \in Z$;
- (2) *dacă* $f(s, \lambda, z) \neq \emptyset$, *atunci* $f(s, i, z) = \emptyset, \forall i \in I$.

Un limbaj recunoscut de un APD determinist îl vom numi limbaj independent de context (sau de tipul doi) determinist. Familia limbajelor independente de context deterministe este inclusă (strict) în familia limbajelor de tipul 2 (după cum vom vedea).

Un APD se poate bloca în următoarele două situații

1. Automatul ajunge în starea s , în vârful memoriei push-down se află simbolul z , pe banda de intrare urmează i și $f(s, i, z) = f(s, \lambda, z) = \emptyset$;
2. Intră într-un ciclu infinit citind de pe bandă λ ; de exemplu $f(s, \lambda, z) = (s, z)$ și $f(s, i, z) = \emptyset$ pentru o anumită pereche (s, z) .

Definiție 3.11 *Un APD determinist este neblocabil dacă pentru orice cuvânt $p \in I^*$ există o evoluție de forma $(s_0, p, z_0) \xrightarrow{*} (s, \lambda, q)$.*

Într-un APD determinist neblocabil orice cuvânt peste I poate fi citit. Evident, de aici nu rezultă că orice cuvânt este recunoscut de APD.

Lema 3.2 *Un APD determinist cu stări finale este echivalent cu un APD determinist cu stări finale neblocabil (relativ la prima situație de blocare).*

Demonstrație. Fie $APD_f = (\Sigma, I, Z, f, s_0, z_0, \Sigma_f)$. Construim $APD'_f = (\Sigma \cup \{s'_0, s'\}, I, Z \cup \{z'_0\}, f', s'_0, z'_0, \Sigma_f)$ unde:

- (1) $f'(s, i, z) = f(s, i, z)$ dacă $f(s, i, z) \neq \emptyset, s \in \Sigma, i \in I \cup \{\lambda\}, z \in Z$;
- (2) $f'(s, i, z) = (s', z)$ dacă $f(s, i, z) = f(s, \lambda, z) = \emptyset, s \in \Sigma, i \in I, z \in Z$;
- (3) $f'(s', i, z) = (s', z), i \in I, z \in Z$;
- (4) $f'(s', \lambda, z'_0) = (s_0, z'_0)$.

Avem

$$p \in L(APD_f) \Leftrightarrow (s_0, p, z_0) \stackrel{APD_f}{\models} (s, \lambda, q), s \in \Sigma_f \Leftrightarrow$$

$$(s'_0, p, z'_0) \stackrel{APD'_f}{\models} (s_0, p, z_0 z'_0) \stackrel{APD'_f}{\models} (s, \lambda, qz'_0), s \in \Sigma_f \Leftrightarrow p \in L(APD'_f).$$

Deci $L(APD_f) = L(APD'_f)$. \square

Observație. Într-o situație de blocare (s, ip, zq) și $f(s, i, z) = f(s, \lambda, z) = \emptyset$ putem scrie

$$(s, ip, zq) \stackrel{APD'_f}{\models} (s', p, zq) \stackrel{APD'_f}{\models} \dots \stackrel{APD'_f}{\models} (s', \lambda, zq).$$

Teorema 3.12 *Orice limbaj independent de context determinist este recunoscut de un APD determinist neblocabil.*

Demonstrație. Fiind dat un APD determinist vom construi un APD determinist neblocabil echivalent. Conform lemei anterioare putem presupune că nu are loc prima situație de blocare.

Dacă are loc a doua situație de blocare, putem avea două cazuri:

1. conținutul memoriei push-down se mărește nelimitat;
2. lungimea cuvintelor scrise în memoria push-down nu depășește un anumit număr.

Fie $card(\Sigma) = n$, $card(Z) = k$, $l = \max\{|q|, q \in Z^* \mid (s', q) \in f(s, i, z)\}$.

Cazul 1. Există în total un număr nk de perechi de forma (s, z) . Dacă în evoluția lui APD s-ar succeda numai configurații cu perechi de forma (s, z) distincte atunci lungimea cuvântului din memoria push-down ar crește la maximum nk simboluri. Prin urmare, dacă $(s', \lambda, \alpha') \xrightarrow{*} (s'', \lambda, \alpha'')$ și $|\alpha''| - |\alpha'| > nk$, atunci în această evoluție trebuie să existe două configurații cu aceeași stare s și cu același simbol z în vârful memoriei push-down. Deci

$$(s', \lambda, \alpha') \xrightarrow{*} (s, \lambda, zr) \xrightarrow{*} (s, \lambda, zqr) \xrightarrow{*} (s'', \lambda, \alpha''),$$

de unde urmează că

$$(s', \lambda, \alpha') \xrightarrow{*} (s, \lambda, zr) \xrightarrow{*} (s, \lambda, q^m r), \forall m \in N.$$

Cazul II Lungimea cuvântului scris în memoria push-down nu depășește nk , căci dacă $|\alpha''| - |\alpha'| > nk$, ar rezulta că în memoria push-down am avea $zq^m, \forall m \in N$ și lungimea cuvântului nu ar fi finită. \square

Teorema 3.13 *Dacă E este un limbaj independent de context determinist atunci limbajul complementar $I^* \setminus E$ este de asemenea independent de context determinist.*

Demonstrație. Fie $APD_f = (\Sigma, I, Z, f, s_0, z_0, \Sigma_f)$ automatul push-down determinist care recunoaște limbajul E . Construim un automat push-down determinist care va recunoaște limbajul $I^* \setminus E$ în modul următor

$$APD'_f = (\Sigma \times \{\sigma_1, \sigma_2, \sigma_3\}, I, Z, f', s'_0, z_0, \Sigma'_f)$$

unde

$$s'_0 = \begin{cases} (s_0, \sigma_1) & \text{pentru } s_0 \in \Sigma_f, \\ (s_0, \sigma_2) & \text{pentru } s_0 \notin \Sigma_f, \end{cases}$$

mulțimea de stări finale este $\Sigma'_f = \{(s, \sigma_3) | s \in \Sigma\}$ iar funcția de evoluție este definită de

- (1) dacă $f(s, i, z) = (s', q)$ atunci
 $f'((s, \sigma_1), i, z) = ((s', k'), q),$
 $f'((s, \sigma_2), i, z) = ((s, \sigma_3), q),$
 $f'((s, \sigma_3), i, z) = ((s', k'), q);$
- (2) dacă $f(s, \lambda, z) = (s', q)$ atunci
 $f'((s, \sigma_1), \lambda, z) = ((s', k'), q),$
 $f'((s, \sigma_3), \lambda, z) = ((s', k'), q);$

unde $k' = 1$ dacă $s' \in \Sigma_f$ și $k' = 2$ dacă $s' \notin \Sigma_f$.

Fie $p \in L(APD'_f)$, atunci $((s_0, \sigma_k), p, z_0) \xrightarrow{*} ((s, \sigma_3), \lambda, q), k = 1, 2$. În mod necesar, ultima configurație trebuie să fie precedată de o configurație de forma $((s', \sigma_2), \lambda, q)$, deci

$$(A) ((s_0, \sigma_k), p, z_0) \xrightarrow{*} ((s', \sigma_2), \lambda, q) \longrightarrow ((s, \sigma_3), \lambda, q)$$

de unde rezultă că $(s_0, p, z_0) \xrightarrow{*} (s', \lambda, q)$ și $s' \notin \Sigma_f$. Deci $p \in I^* \setminus E$ și $L(APD'_f) \subseteq I^* \setminus E$.

Invers, fie $p \in I^* \setminus E$, atunci în APD'_f avem evoluția $(s_0, p, z_0) \xrightarrow{*} (s, \lambda, q), s \notin \Sigma_f$ și putem scrie evoluția (A). Prin urmare $p \in L(APD'_f)$, adică $I^* \setminus E \subseteq L(APD'_f)$. □

Consecință. Putem enunța proprietatea de mai sus astfel: Familia limbajelor independente de context deterministe este închisă la complementariere. Cum familia limbajelor independente de context nu este închisă la complementariere și cum ea coincide cu familia limbajelor recunoscute de automatele push-down nedeterministe putem mai departe obține următorul rezultat:

APD nedeterministe nu sunt echivalente cu APD deterministe.

3.6 Lema Bar–Hillel

Ca și în cazul limbajelor regulate, lema Bar–Hillel pune în evidență următoarea proprietate a unui limbaj independent de context: orice cuvânt al unui astfel de limbaj, suficient de lung, conține un subcuvânt (nevid) pe care multiplicându-l de un număr arbitrar de ori, obținem noi cuvinte care aparțin de asemenea limbajului. Mai poartă denumirea de ”lema de pompare” sau ”lema $uvwx^y$ ”.

Ne vom referi în cele ce urmează la gramatici de tipul 2 în formă normală Chomsky. Într-o gramatică de această formă arborii de derivare sunt întotdeauna arbori binari.

Lema 3.3 *Fie G o gramatică în forma normală Chomsky și $X \xRightarrow{*} p, p \in V_G^*$. Dacă cea mai lungă ramură din arborele $\mathcal{A}_{X,p}$ conține m noduri, atunci $|p| \leq 2^{m-1}$.*

Demonstrație. Procedăm prin inducție asupra lui m .

Pentru $m = 2$ arborele are două nivele și în mod evident $|p| \leq 2 = 2^{m-1}$.

Presupunem că proprietatea este adevărată pentru un m oarecare și considerăm un arbore care are pe cea mai lungă ramură $m + 1$ noduri. În derivarea $X \xRightarrow{*} p$ punem în evidență prima derivare directă

$$X \Rightarrow YZ \xRightarrow{*} p.$$

Conform lemei de localizare, $p = p_1p_2$ și $Y \xRightarrow{*} p_1, Z \xRightarrow{*} p_2$. Arborii de derivare \mathcal{A}_{Y,p_1} și \mathcal{A}_{Z,p_2} conțin, fiecare pe cea mai lungă ramură cel mult m noduri; conform ipotezei de inducție, avem $|p_1| \leq 2^{m-1}, |p_2| \leq 2^{m-1}$. Prin urmare

$$|p| = |p_1p_2| = |p_1| + |p_2| \leq 2^{m-1} + 2^{m-1} = 2^m. \square$$

Exemplu. Considerăm un arbore de formă dată în figura 4.7. Cea mai lungă ramură are $m = 5$ noduri (nivelul ultim plus 1). Se poate ușor vedea că pe ultimul nivel putem avea cel mult 2^{m-1} noduri (în acest caz toate nodurile sunt neterminale).

Observație. Dacă $X \xRightarrow{*} p$ și $|p| > 2^{m-1}$ atunci există în arborele $\mathcal{A}_{X,p}$ cel puțin o ramură cu $m + 1$ noduri.

Lema 3.4 *Fie G o gramatică de tipul 2 și fie $X \Rightarrow X_1 \dots X_m \xRightarrow{*} p$. Atunci, conform lemei de localizare, $p = p_1 \dots p_m$ și $X_j \xRightarrow{*} p_j, j = 1, \dots, m$. Fie $\mathcal{A}_{Y,q} \subseteq \mathcal{A}_{X,p}$. atunci q este subcuvânt într-unul din cuvintele p_j .*

Demonstrație. Neterminalul Y aparține unuia din subarborii \mathcal{A}_{X_j,p_j} , fie acesta \mathcal{A}_{X_k,p_k} ; atunci $\mathcal{A}_{Y,q} \subseteq \mathcal{A}_{X_k,p_k}$ și $q \in \text{Sub}(p_k)$. Imaginea grafică este dată de figura 4.8. \square

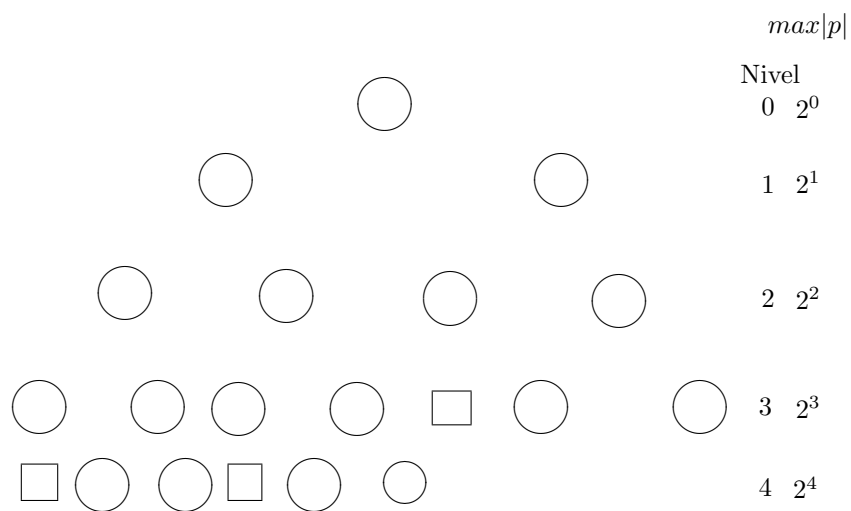


Figura 3.7: Exemplu de arbore

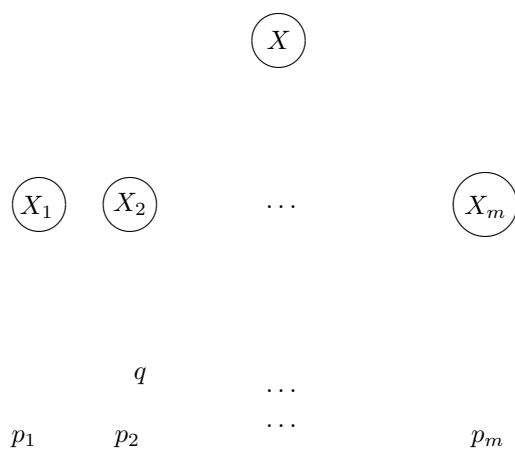
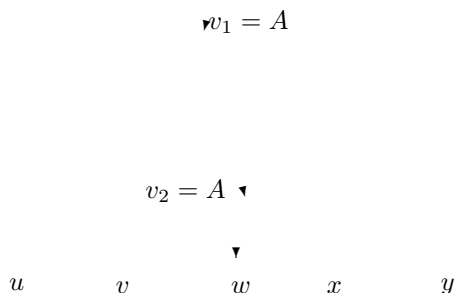


Figura 3.8: Reprezentarea grafică a incluziunii $q \in Sub(p_k)$.

Figura 3.9: Descompunerea cuvântului p .

Lema 3.5 (Lema Bar–Hillel) Fie E un limbaj independent de context. Atunci există un număr natural k astfel încât, dacă $p \in E$ și $|p| > k$ atunci p se poate descompune în forma $p = uvwx$ cu următoarele proprietăți:

1. $vx \neq \lambda$;
2. $|vwx| \geq k$;
3. $uv^jwx^jy \in E, \forall j \in \mathbb{N}$,

Demonstrație. Fie $n = \text{card}(V_N)$. Luăm $k = 2^n$. Cum $|p| > k = 2^n$, conform observației de la prima leamnă, există în arborele $\mathcal{A}_{S,p}$ cel puțin o ramură cu $n + 2$ noduri; pe această ramură ultimul nod este terminal, deci există $n + 1$ noduri etichetate cu același simbol A . Descompunem cuvântul p ca în figura 4.9, $p = uvwx$.

(1) Considerăm subarborele $\mathcal{A}_{A,vwx}$ căruia îi corespunde derivarea $A \xRightarrow{*} vwx$. Punem în evidență primul pas

$$A \Rightarrow BC \xRightarrow{*} vwx,$$

și vwx se descompune în $vwx = p_B p_C, B \xRightarrow{*} p_B, C \xRightarrow{*} p_C$ și $p_B, p_C \neq \lambda$. Cum $\mathcal{A}_{A,w} \subseteq \mathcal{A}_{A,vwx}$, rezultă că w este subcuvânt în p_B sau în p_C . Să presupunem că $w \in \text{Sub}(p_C)$; atunci $p_B \in \text{Sub}(v)$ și cum $p_B \neq \lambda$ rezultă $v \neq \lambda$.

(2) Putem alege nodurile v_1 și v_2 astfel încât pe ramura punctată începând de la v_1 în jos până la frontieră să avem exact $n + 1$ noduri. Rezultă conform lemei precedente, $|vwx| \geq 2^n = k$.

(3) putem scrie derivările $A \xrightarrow{*} w, A \xrightarrow{*} vAx$. Deci

$$S \xrightarrow{*} uAy \xrightarrow{*} uvAxy \xrightarrow{*} uv^2Ax^2y \xrightarrow{*} \dots \xrightarrow{*} uv^jwx^jy. \square$$

Problema închiderii familiei \mathcal{L}_2 la intersecție

Teorema 3.14 *Familia \mathcal{L}_2 nu este închisă la intersecție.*

Demonstrație. Considerăm limbajul $L_3 = \{a^n b^n c^n | n \geq 1\}$. Să rătăim că $L_3 \notin \mathcal{L}_2$. Într-adevăr, să presupunem că $L_3 \in \mathcal{L}_2$ și fie k constanta din lema Bar–Hillel. Luăm $n > k/3$ și fie $p = a^n b^n c^n$; avem $|p| > k$, deci conform acestei leme p se descompune în forma $p = uvwxy$ cu $vx \neq \lambda$ și $uv^jwx^jy \in L_3, j \in \mathbb{N}$.

Vom analiza posibilitățile de constituire a subcuvintelor v și x . Să presupunem că în v (sau x) intră două din simbolurile a, b, c ; de exemplu $v = aabbb$. atunci considerăm cuvântul $p_2 = uv^2wx^2y = uaabbaabbbwx^2y$ care nu are structura cuvintelor lui L_3 ("a" nu poate să urmeze după "b") dar conform lemei Bar–Hillel aparține lui L_3 . Deci, în v (sau x) nu pot intra două (sau trei) din simbolurile a, b, c . Să presupunem că intră un singur simbol; de exemplu $v \in \{a\}^*$ și $x \in \{b\}^*$. Atunci multiplicând în p subcuvintele v și x , puterile simbolurilor "a" și "b" se măresc iar "c" rămâne pe loc, contrazicându-se din nou structura cuvintelor din L_3 . În concluzie L_3 nu este independent de context.

Fie acum limbajele

$$L'_3 = \{a^m b^n c^n | m, n \geq 1\}$$

$$L''_3 = \{a^n b^n c^m | m, n \geq 1\}.$$

Se poate vedea ușor că aceste limbaje sunt de tipul 2; gramaticile care le generează sunt $S \rightarrow aS|aA, A \rightarrow bAc|bc$ și respectiv $S \rightarrow Sc|Ac, A \rightarrow aAb|ab$.

Avem $L'_3 \cap L''_3 = L_3$, deci intersecția a două limbaje de tipul 2 este un limbaj care nu aparține lui \mathcal{L}_2 . \square

Corolar 3.15 *Familia \mathcal{L}_2 nu este închisă la complementariere.*

Într-adevăr, să presupunem că \mathcal{L}_2 este închisă la complementariere și fie $E_1, E_2 \in \mathcal{L}_2$. Cum familia \mathcal{L}_2 este închisă la reuniune, ar rezulta $C(E_1) \cup C(E_2) \in \mathcal{L}_2$. Dar (de Morgan) $C(E_1) \cup C(E_2) = C(E_1 \cap E_2) \in \mathcal{L}_2$.

Complementul limbajului $C(E_1 \cap E_2)$ este $E_1 \cap E_2$ și conform presupunerii ar trebui ca $E_1 \cap E_2 \in \mathcal{L}_2$, oricare ar fi E_1, E_2 , ceea ce nu este adevărat. \square

Generalizări ale lemei Bar–Hillel Lema lui Bar–Hillel reprezintă o condiție necesară ca un limbaj să fie independent de context. cu ajutorul ei

se poate demonstra relativ ușor că anumite limbaje nu sunt independente de context. Ideea este aceea că prin multiplicarea subcuvintelor v și x , de un număr suficient de ori, se contrazice structura limbajului.

Totuși, nu orice limbaj care nu este de tipul 2 poate fi respins de lema lui Bar–Hillel. De exemplu, această lemă nu poate respinge limbajul

$$L = \{a^n b^{2^m} \mid n, m \geq 1\} \cup \{b\}^*,$$

care nu este de tipul 2 (se demonstrează pe altă cale). Într-adevăr, pentru orice $p = a^n b^{2^m}$ luăm

$$u = \lambda, v = a, w = \lambda, x = \lambda, y = a^{n-1} b^{2^m}.$$

Putem itera subcuvintele v și x fără să contrazicem structura cuvintelor limbajului.

O generalizare a lemei Bar–Hillel este

Lema 3.6 (*Lema lui Ogden*) *pentru orice limbaj independent de context L există o constantă k astfel încât orice $p \in L$ pentru care cel puțin k simboluri sunt "marcate", se poate descompune în forma $p = uvwx$ astfel încât*

1. sau u, v, w sau w, x, y conțin fiecare câte un simbol marcat;
2. vwx conține cel mult k simboluri marcate;
3. $uv^jwx^jy \in L, \forall j \in \mathbb{N}$

Cu ajutorul acestei leme se poate arăta că limbajul de mai sus nu este de tipul 2, considerând marcate simbolurile b .

3.7 Închiderea familiei \mathcal{L}_2 la substituții

Definiție 3.12 *Vom spune că o familie de limbaje \mathcal{L} este închisă la substituții dacă oricare ar fi $L \in \mathcal{L}$ și oricare ar fi substituția $s : V^* \rightarrow \mathcal{P}(U^*)$ astfel încât $s(i) \in \mathcal{L}, \forall i \in V$, avem $s(L) \in \mathcal{L}$.*

Se cunosc o serie de proprietăți de închidere a familiilor de limbaje din clasificarea Chomsky față de substituții particulare; o parte din acestea sunt prezentate în tabelul următor.

	\mathcal{L}_0	\mathcal{L}_1	\mathcal{L}_2	\mathcal{L}_3
Substituție	DA	NU	DA	DA
Substituție λ -liberă	DA	DA	DA	DA
Homomorfisme	DA	NU	DA	DA
Homomorfisme λ -libere	DA	DA	DA	DA

În cele ce urmează vom considera noțiunea de substituție într-o accepțiune ceva mai generală (față de cea din Capitolul I)

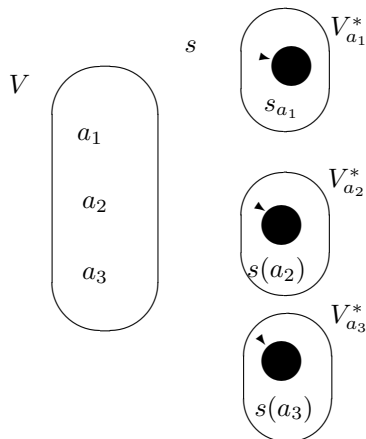


Figura 3.10: Reprezentare grafică a substituției

Definiție 3.13 Fie V un alfabet; pentru orice $a \in V$ considerăm un alfabet V_a și notăm $V_g = \bigcup_{a \in V} V_a$. Fie $s : V \rightarrow \mathcal{P}(V_g^*)$ o aplicație cu proprietatea $s(a) \in V_a^*, a \in V$. Prelungirea canonică a lui s la V^* o numim substituție.

Observație. Prelungirea canonică se definește recursiv astfel:

$$s(\lambda) = \{\lambda\}, \quad s(a_1 \dots a_n) = s(a_1) \dots s(a_n).$$

O reprezentare grafică este prezentată în figura 4.10. Observăm că $V_g^* \neq \bigcup_{a \in V} V_a^*$ și $V_a^* \subseteq V_g^*$.

Definiția substituției se poate da și direct:

Definiție 3.14 Vom spune că $s : V^* \rightarrow \mathcal{P}(V_g^*)$ este o substituție dacă

1. $s(\lambda) = \{\lambda\}$;
2. $s(a) \subseteq V_a^*, \forall a \in V$;
3. $s(a_1 \dots a_n) = s(a_1) \dots s(a_n), a_1, \dots, a_n \in V$.

Definiție 3.15 Imaginea unui limbaj $L \in V^*$ prin substituția s este

$$s(L) = \bigcup_{a \in L} s(a).$$

Definiție 3.16 Vom spune că o familie de limbaje \mathcal{L} este închisă la substituții dacă oricare ar fi $L \in \mathcal{L}$ și oricare ar fi substituția $s : V^* \rightarrow \mathcal{P}(V_g^*)$ astfel încât $s(a) \in \mathcal{L}, \forall a \in V$, avem $s(L) \in \mathcal{L}$.

Observație. În cazul particular în care $V_a = U, \forall a \in V$, avem $V_g = U$ și condiția $s(a) \subseteq V_a^*$ este în mod evident satisfăcută.

Teorema 3.16 *Familia limbajelor independente de context este închisă la substituții.*

Demonstrație. Fie $L \in \mathcal{L}_2$ și $s : V^* \rightarrow \mathcal{P}(V_g^*)$ o substituție astfel încât $s(a) \in \mathcal{L}_2, \forall a \in V$.

Fie

$$\begin{aligned} G &= (V_N, V, S, P) \text{ cu } L(G) = L, \\ G_a &= (V_N^a, V_a, S_a, P_a) \text{ cu } L(G_a) = s(a), \end{aligned}$$

G și G_a fiind gramatici de tipul 2. Vom presupune că aceste gramatici nu conțin reguli de ștergere (cu excepția regulilor de completare) și că alfabetele neterminale sunt disjuncte.

Definim $\mu : V_N \cup V \cup \{\lambda\} \rightarrow V_N \cup \{S_a | a \in V\} \cup \{\lambda\}$ prin

$$\begin{aligned} \mu(\lambda) &= \lambda, \\ \mu(X) &= X, X \in V_N, \\ \mu(a) &= S_a, a \in V \end{aligned}$$

și prelungim canonic aplicația la $(V_N \cup V)^*$.

Considerăm acum gramatica $G' = (V'_N, V'_T, S, P')$ unde

$$\begin{aligned} V'_N &= V_N \cup \left(\bigcup_{a \in V} V_N^a \right), \\ V'_T &= \bigcup_{a \in V} V_a, \\ P' &= \left(\bigcup_{a \in V} P_a \right) \cup \{X \rightarrow \mu(\alpha) | X \rightarrow \alpha \in P\}. \end{aligned}$$

Observație. P' conține toate regulile din toate gramaticile G_a nemodificate; regulile din G le modifică, și anume terminalele din părțile drepte le înlocuim cu simbolurile de start din gramaticile G_a .

Să mai observăm că G' este de tipul 2.

Fie $u \in s(L)$. Deoarece $s(L) = \bigcup_{v \in L} s(v)$ rezultă că există $v \in L$ astfel încât $u \in s(v)$.

Dacă $v = \lambda$ înseamnă că $s \rightarrow \lambda \in P$ și $S \rightarrow \mu(\lambda) = \lambda \in P'$. Ținând cont că $s(\lambda) = \{\lambda\}$ rezultă $u = \lambda$ și $\lambda \in L(G')$.

Să presupunem că $v \neq \lambda$ și fie $v = v_1 \dots v_n$; avem $s(v) = s(v_1) \dots s(v_n)$. Prin urmare $u = u_1 \dots u_n$ și $u_i \in s(v_i), \dots, u_n \in s(v_n)$.

Avem $S \xrightarrow[G]{*} v$ și $S_{a_i} \xrightarrow{*} u_i$ în $G_{a_i}, i = 1, \dots, n$.

Atunci

$$G' : S \Rightarrow \mu(v_1) \Rightarrow \mu(v_2) \Rightarrow \dots \Rightarrow \mu(v_n) = \mu(v).$$

Cum v conține numai terminale, rezultă că $\mu(v) = S_{a_1} \dots S_{a_n}$. Deci

$$G' : S \xrightarrow{*} X_{a_1} \dots X_{a_n} \xrightarrow{*} u_1 \dots u_n = u,$$

3.8. CARACTERIZAREA LIMBAJELOR INDEPENDENTE DE CONTEXT 73

și $u \in L(G')$, adică $s(L) \subseteq L(G')$.

Invers, fie $u \in L(G')$. deoarece $(\cup V_N^a) \cap V_N = \emptyset$, derivările din G' care utilizează reguli din P_a nu introduc simboluri din V_N . În derivarea $S \xrightarrow[G']{*} u$ efectuăm mai întâi derivările directe care utilizează reguli de forma $X \rightarrow \mu(\alpha)$. Vom avea

$$G' : S \xrightarrow{*} S_{a_1} \dots S_{a_n} \xrightarrow{*} u.$$

Cuvântul u se descompune în $u = v_1 \dots v_n$ și $S_{a_j} \xrightarrow[G']{*} v_j$. Deoarece alfabetele gramaticilor G_a sunt disjuncte, putem scrie $S_{a_j} \xrightarrow[G']{*} v_j$ (în gramatica G_{a_j}) și deci $v_j \in L(G_{a_j}) = s(a_j)$. Așadar

$$u = v_1 \dots v_n \in s(a_1 \dots s(a_n)) = s(a_1 \dots a_n) \subseteq s(L).$$

Rezultă $L(G') \subseteq s(L)$ și în final $s(L) = L(G')$. \square

Proprietatea de închidere a familiei \mathcal{L}_2 la substituții se poate utiliza pentru a reobține proprietăți cunoscute. De exemplu:

Corolar 3.17 *Familia \mathcal{L}_2 este închisă la operațiile regulate (reuniune, produs, închidere Kleene).*

Demonstrație. Limbajele $\{a, b\}$, $\{ab\}$, $\{a\}^*$ sunt regulate, deci și independente de context.

Definim o substituție s astfel: $V = \{a, b\}, V_a, V_b$ oarecare și fie $L_1 \subseteq V_a^*, L_2 \subseteq V_b^*$ două limbaje independente de context. Considerăm $s : \{a, b\} \rightarrow \mathcal{P}((V_a \cup V_b)^*)$ definită de

$$s(a) = L_1, s(b) = L_2$$

Conform teoremei anterioare rezultă că limbajele $s(\{a, b\}), s(\{ab\}), s(\{a\}^*)$ sunt independente de context; dar

$$s(\{a, b\})L_1 \cup L_2, s(\{ab\}) = L_1L_2, s(\{a\}^*) = L_1^*.$$

De exemplu

$$s(\{a, b\}) = s(a) \cup s(b) = L_1 \cup L_2, \text{ etc.}$$

3.8 Caracterizarea limbajelor independente de context

Limbajul lui Dyck Cuvintele limbajului Dyck, numit și limbaj parametric, constituie șiruri de paranteze corect scrise într-o expresie aritmetică. De exemplu, dacă în expresia

$$\{(a + b) * [(c + d) - (e + f)]\}$$

se elimină toți operanzii se obține șirul de paranteze $\{()[(())]\}$, care după cum se poate observa sunt corect scrise. Definiția generală a unui șir de paranteze corect scrise nu este foarte simplă; de exemplu, nu este suficient să spunem că fiecărei paranteze deschise trebuie să-i corespundă o paranteză închisă.

Șirurile de paranteze corect scrise se pot defini elegant cu ajutorul gramaticilor de tipul 2. Fie $G_n = (\{X\}, V_T, X, P)$ o gramatică de tipul 2, unde

$$\begin{aligned} V_T &= \{i_1, \dots, i_n, i'_1, \dots, i'_n\}, \\ P &= \{X \rightarrow X i_k X i'_k X, k = 1, \dots, n \ X \rightarrow \lambda\}. \end{aligned}$$

Prin definiție, *limbajul lui Dyck de ordinul n* este $D_n = L(G_n)$.

Exemplu. Presupunem $n = 3$; putem scrie derivarea

$$\begin{aligned} X &\Rightarrow X i_3 X i'_3 X \Rightarrow X i_3 X i_1 X i'_1 X i'_3 X \Rightarrow X i_3 X i_1 X i'_1 X i_2 X i'_2 X i'_3 X \\ &X i_3 X i_1 X i'_1 X i_2 X i_1 X i'_1 X i'_2 X i'_3 X \Rightarrow X i_3 X i_1 X i'_1 X i_2 X i_1 X i_1 X i'_1 X i'_2 X i'_3 X. \end{aligned}$$

În final, aplicând regula de ștergere, obținem

$$X \xRightarrow{*} i_3 i_1 i'_1 i_2 i_1 i'_1 i'_2 i'_3.$$

Dacă asociem simbolurilor i_k, i'_k perechi de paranteze, de exemplu: $i_1, i'_1 \leftrightarrow (,)$; $i_2, i'_2 \leftrightarrow [,]$; $i_3, i'_3 \leftrightarrow \{, \}$ obținem $\{()[(())]\}$.

Proprietăți ale limbajului lui Dyck

1. $i_k D_n i'_k \subseteq D_n, k = 1, \dots, n$;
2. $D_n D_n \subseteq D_n$;
3. $D_n \subseteq (\bigcup_{k=1}^n i_k D_n i'_k D_n) \cup \{\lambda\}$;
4. dacă $p, pq \in D_n$ atunci $q \in D_n$.

Demonstrație. (1) Un cuvânt din $i_k D_n i'_k$ trebuie să aibă forma $i_k p i'_k, p \in D_n$ (deci $X \xRightarrow{*} p$). Putem scrie

$$X \Rightarrow X i_k X i'_k X \xRightarrow{*} i_k X i'_k \xRightarrow{*} i_k p i'_k \in D_n. \square$$

(2) Fie $p \in D_n D_n$, deci $p = p_1 p_2, p_1 \in D_n, p_2 \in D_n$ sau $X \xRightarrow{*} p_1, X \xRightarrow{*} p_2$; fie $p = i_1 \dots i_n$. Derivarea $X \xRightarrow{*} p_1$ o putem detalia lăsând la sfârșit ștergerile:

$$X \xRightarrow{*} X i_1 X_2 X \dots X i_n X \xRightarrow{*} i_1 \dots i_n.$$

Putem scrie

$$X \xRightarrow{*} X i_1 X_2 X \dots X i_n X \xRightarrow{*} i_1 \dots i_n X = p_1 X \xRightarrow{*} p_1 p_2 = p \in D_n. \square$$

3.8. CARACTERIZAREA LIMBAJELOR INDEPENDENTE DE CONTEXT 75

(3) fie $p \in D_n$; trebuie să arătăm că $p \in (\bigcup_{k=1}^n i_k D_n i'_k D_n) \cup \{\lambda\}$.
Procedăm prin inducție asupra lungimii lui p . Dacă $|p| = 0$, atunci $p = \lambda$ și apartenența este evidentă.

Observație. Putem lua $|p| = 2$, deci $p = i_k i'_k$; avem

$$i_k D_n i'_k D_n = i_k \{\lambda, \dots\} i'_k \{\lambda, \dots\} = \{i_k i'_k, \dots\}$$

și deci $i_k i'_k \in i_k D_n i'_k D_n$.

Presupunem acum că proprietatea este adevărată pentru $|p|$ oarecare și considerăm cazul $|p| + 1$. punem în evidență prima derivare directă $X \Rightarrow X i_k X i'_k X \xrightarrow{*} p$. Conform lemei de localizare, $p = p_1 i_k p_2 i'_k p_3$ și $X \xrightarrow{*} p_j$, $j = 1, 2, 3$, adică $p_j \in D_n$. Dacă $p_1 = \lambda$ proprietatea este demonstrată. Presupunem că $p_1 \neq \lambda$. Cum $p_1 \in D_n$, avem $p_1 = i_j p'_1 i'_j p''_1, p'_1, p''_1 \in D_n$, în conformitate cu ipoteza inductivă. Prin urmare

$$p = i_j p'_1 i'_j p''_1 \underbrace{i_k p_2 i'_k p_3}_{\substack{\in D_n \\ = q \in D_n}} = i_j p'_1 i'_j q. \square$$

(4) Fie $p, pq \in D_n$. Procedăm prin inducție asupra lui $|pq|$.

Dacă $|pq| = 0$, rezultă $q = \lambda \in D_n$.

Presupunem că proprietatea este adevărată pentru $|pq|$ oarecare și considerăm cazul $|pq| + 1$.

Din $p, q \in D_n$ rezultă

$$pq = i_k s i'_k t, \quad s, t \in D_n.$$

Avem două cazuri

$$I. pq = \underbrace{i_k s_1 s_2 \dots s_\alpha i'_k}_{p} \overbrace{t_1 t_2 \dots t_p}^{t'}$$

Deci

$$\underbrace{p}_{\in D_n} = \underbrace{i_k s i'_k}_{\in D_n} \Rightarrow (ip.ind.) t' \in D_n,$$

$$\underbrace{t}_{\in D_n} = \underbrace{t' q}_{\in D_n} \Rightarrow (ip.ind.) q \in D_n.$$

$$II. pq = \underbrace{i_k s_1 s_2 \dots s_\alpha}_{p} \underbrace{i'_k}_{s'} \underbrace{t_1 t_2 \dots t_p}_{q}.$$

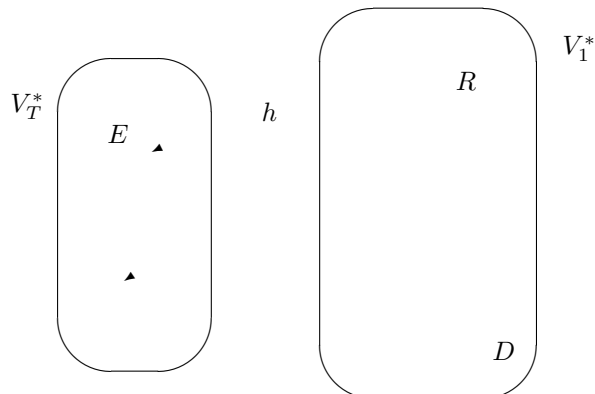


Figura 3.11: Imagine grafică a teoremei de caracterizare

Avem

$$i_k s i'_k = p s' i'_k = i_k p' i'_k p'' s' i'_k, p', p'' \in D_n,$$

$$\underbrace{s}_{\in D_n} = \underbrace{p'}_{\in D_n} i'_k p'' s' \Rightarrow (ip.ind.) i'_k p'' s' \in D_n$$

ceea ce nu este posibil. \square

Teorema de caracterizare

Teorema 3.18 *Un limbaj este independent de context dacă și numai dacă este imaginea printr-un homomorfism a intersecției dintre un limbaj regulat și un limbaj Dyck.*

Demonstrație. Vom arăta numai implicația directă, adică dacă E este un limbaj independent de context atunci există un limbaj regulat R , un limbaj Dyck și un homomorfism h astfel încât $E = h(R \cap D)$.

Imaginea geometrică este prezentată în figura 4.11. Fie $G = (V_N, V_T, S, P)$ în forma normală Chomsky astfel încât $E = L(G)$. Presupunem că $V_T = \{i_1, \dots, i_m\}$.

(a) Construcția lui R .

Luăm gramatica $G' = (V_N, V_1, S, P')$ unde $V_1 = \{i_1, i_2, i_m, i_{m+1}, \dots, i_{m+n}, i'_1, \dots, i'_{m+n}\}$ iar P' se construiește astfel:

$$X \rightarrow i_k \in P \Rightarrow \begin{cases} X \rightarrow i'_k i_k \in P', \\ X \rightarrow i_k i'_k i_{n+l} Z \in P', \quad l = \overline{1, n}, Z \in V_N; \\ X \rightarrow YZ \in P \text{ atunci } X \rightarrow i_{m+l} Y \in P', l = \overline{1, n}. \end{cases}$$

3.8. CARACTERIZAREA LIMBAJELOR INDEPENDENTE DE CONTEXT 77

De fiecare dată alegem un i_{m+l} diferit (de aici rezultă n).

(b) Construcția lui D .

Luăm $D = D_{m+n}$ cu alfabetul terminal V_1 și cu regulile cunoscute.

(c) Definiția lui h .

Luăm $h : V_1 \rightarrow V_T \cup \{\lambda\}$ astfel:

$$h(i_k) = \begin{cases} i_k, k \geq m \\ \lambda, k < m, \end{cases} \quad h(i'_k) = \lambda, k = 1, \dots, m+n.$$

Prelungim h în mod canonic la V_1^* . Se poate arăta ușor că h este un homomorfism.

Să arătăm că $E \subseteq h(R \cap D)$.

Fie $p \in E = L(G)$; va trebui să arătăm că există $q \in R, D$ astfel încât $p \in h(q)$. Vom arăta implicația ceva mai generală:

Dacă $X \xrightarrow[G]{*} p$ atunci $\exists q, X \xrightarrow[G']{*} q, q \in D, p = h(q)$.

Procedăm prin inducție asupra lungimii l a derivării.

Dacă $l = 1$, atunci $X \xrightarrow[G]{*} p = i_k, X \rightarrow i_k i'_k \in P'$. Luăm $q = i_k i'_k$ și

avem în G' derivarea $X \rightarrow i_k i'_k$, iar $h(q) = h(i_k)h(i'_k) = i_k = p$.

Presupunem că implicația este adevărată pentru derivări de lungime oarecare l și considerăm cazul $l + 1$; punem în evidență prima derivare directă

$$X \xrightarrow[G]{*} YZ \xrightarrow[G']{*} p.$$

Conform lemei de localizare, $p = p_1 p_2, Y \xrightarrow[G]{*} p_1, Z \xrightarrow[G']{*} p_2$. Ținând cont de ipoteza inductivă avem în gramatica G'

$$G' : \begin{cases} \exists q_1, Y \xrightarrow[G']{*} q_1, q_1 \in D, h(q_1) = p_1; \\ \exists q_2, Z \xrightarrow[G']{*} q_2, q_2 \in D, h(q_2) = p_2. \end{cases}$$

Ne fixăm atenția asupra derivării $Y \xrightarrow[G']{*} q_1$; fiind o derivare într-o gramatică de tipul 3, în toate derivările directe se aplică reguli de categoria I ($A \rightarrow pB$) cu excepția ultimei unde se aplică o regulă de categoria II ($c \rightarrow q$). În cazul nostru, această ultimă regulă trebuie să fie de forma $X \rightarrow i_k i'_k$. Odată cu această regulă avem și $X \rightarrow i_k i'_k k i'_{m+l} Z$; aplicând-o pe aceasta, avem

$$Y \xrightarrow[G']{*} q_1 i'_{m+l} Z \xrightarrow[G']{*} q_1 i'_{m+l} q_2.$$

Pe de altă parte, deoarece $X \rightarrow YZ \in P$ avem $X \rightarrow i_{m+l} Y \in P'$, așa încât

$$X \xrightarrow[G']{*} i_{m+l}Y \xrightarrow[G']{*} i_{m+l}q_1 i'_{m+l}q_2 \stackrel{def}{=} q.$$

Avem $X \xrightarrow[G']{*} q, q \in D$ și $h(q) = h(q_1)h(q_2) = p_1p_2 = p$.

Să arătăm acum că $h(R \cap D) \subseteq E$.

Fie $q \in R, D$ și $p \in h(q)$, atunci $p \in L(G)$.

Mai general

$$X \xrightarrow[G']{*} q, q \in D \text{ atunci } X \xrightarrow[G]{*} h(q).$$

Procedăm prin inducție asupra lungimii derivării l .

Dacă $l = 1$ atunci $X \Rightarrow q = i_k i'_k; q \in D_n, h(q) = i_k$. S-a aplicat regula $X \rightarrow i_k i'_k \in P'$; rezultă $X \rightarrow i_k \in P$ deci $X \xrightarrow[G]{*} i_k = h(q)$.

Presupunem că implicația este adevărată pentru un l oarecare și considerăm cazul $l + 1$. Punem în evidență prima derivare directă (evident, aceasta nu poate să fie de forma $X \rightarrow i_k i'_k$). Distingem două cazuri

I. Se aplică o regulă de forma $X \rightarrow i_k i'_k i'_{m+l}Z$. Avem

$$X \Rightarrow i_k i'_k i'_{m+l}Z \xrightarrow[G']{*} q,$$

$$\underbrace{q}_{\in D} = \underbrace{i_k i'_k}_{\in D} i'_{m+l}q_1 \Rightarrow i'_k i'_{m+l}q_1 \in D, \text{ nu este posibil.}$$

II. Se aplică o regulă de forma $X \rightarrow i_{m+l}Y$. Avem

$$G' : X \Rightarrow i_{m+l}Y \xrightarrow{*} q, q = i_{m+l}q' i'_{m+l}q'', q', q'' \in D.$$

Detaliat, derivarea de mai sus trebuie să aibă forma

$$G' : X \Rightarrow i_{m+l}Y \xrightarrow{*} i_{m+l}q' Z \xrightarrow{*} i_{m+l}q' i'_{m+l}q''.$$

De aici $Y \xrightarrow[G']{*} q', Z \xrightarrow[G']{*} q''$ și conform ipotezei inductive avem $Y \xrightarrow[G]{*} h(q')$,

$Z \xrightarrow[G]{*} h(q'')$. Putem scrie

$$G : X \Rightarrow YZ \xrightarrow{*} h(q')h(q'') = h(i_{m+l})h(q')h(i_{m+l})h(q'') = h(i_{m+l}q' i'_{m+l}q'') = h(q). \square$$

1. Să se arate că $L(G) = \{(ab)^n a | n \geq 0\}$ unde G are producțiile $S \rightarrow SbS, S \rightarrow a$. Să se construiască o gramatică echivalentă cu G care să fie neambiguă.

3.8. CARACTERIZAREA LIMBAJELOR INDEPENDENTE DE CONTEXT 79

2. Eliminați redenumirile din gramatica G_E ce generează expresiile aritmetice simple.
3. Aduceți la forma normală Chomsky gramaticile ce au regulile:
 - (a) $S \rightarrow T b T, T \rightarrow T a T | c a;$
 - (b) $S \rightarrow a A C, A \rightarrow a B | b A B, B \rightarrow b, C \rightarrow c;$
 - (c) $S \rightarrow A . A, A \rightarrow 0 A | 1 A | \dots 9 A | \lambda.$
4. Găsiți forma normală Greibach pentru gramaticile:
 - (a) $A_1 \rightarrow A_2 A_3, A_2 \rightarrow A_1 A_2 | 1, A_3 \rightarrow A_1 A_3 | 0.$
 - (b) G_E care generează expresiile aritmetice simple.
5. Construiți un automat push-down pentru recunoașterea limbajului:
 - (a) $L = \{w | w \in \{a, b\}^*, \text{ numărul literelor } a \text{ în } w \text{ este egal cu numărul literelor } b \text{ în } w \};$
 - (b) $L = \{w | w \in \{a, b\}^*, w = \tilde{w}\};$
 - (c) $L = \{w | w \in \{(,)\}^*, w \text{ este un cuvânt în care fiecare paranteză deschisă are o pereche, paranteză închisă } \}.$
6. Folosind lema de pompare să se arate că următoarele limbaje nu sunt independente de context:
 - (a) $L = \{a^i b^j c^k | i < j < k\};$
 - (b) $L = \{a^i b^j | j = i^2\};$
 - (c) $L = \{a^n b^n c^n | n \geq 1\};$
 - (d) $L = \{a^i | i \text{ prim } \};$
 - (e) $L = \{a^i b^i c^j | j \geq i\};$

Capitolul 4

Limbaje dependente de context

4.1 Gramatici monotone

Reamintim că o gramatică este monotonă dacă orice regulă $u \rightarrow v$ satisface condiția $|u| \leq |v|$. Este permisă și λ -regula, $S \rightarrow \lambda$ (care evident nu satisface condiția de monotonie) cu condiția ca S să nu apară în dreapta vreunei reguli.

Lema 4.1 *Fie $G_1 = (V_N, V_T, S, P \cup \{u \rightarrow v\})$ unde $u \rightarrow v$ este o regulă care satisface condiția de monotonie iar P sunt reguli de tipul 1. Atunci G_1 este echivalentă cu o gramatică dependentă de context.*

Demonstrație. Presupunem că P satisface condiția din lema 1.2, deci $u = X_1 \dots X_m$ și $v = Y_1 \dots Y_n$ cu $X_i, Y_j \in V_N$ și $m \leq n$. Putem presupune că $m \geq 2$. Fie Z_1, \dots, Z_m neterminale noi și considerăm regulile R :

$$\begin{aligned} X_1 X_2 \dots X_m &\rightarrow Z_1 X_2 \dots X_m, \\ Z_1 X_2 \dots X_m &\rightarrow Z_1 Z_2 \dots X_m, \\ &\dots \\ Z_1 Z_2 \dots Z_{m-1} X_m &\rightarrow Z_1 Z_2 \dots Z_m Y_{m+1} \dots Y_n, \\ Z_1 Z_2 \dots Z_m Y_{m+1} \dots Y_n &\rightarrow Y_1 Z_2 \dots Z_m Y_{m+1} \dots Y_n, \\ &\dots \\ Y_1 Y_2 \dots Y_{m-1} Z_m Y_{m+1} \dots Y_n &\rightarrow Y_1 Y_2 \dots Y_n. \end{aligned}$$

Luăm $G'_1 = (V_N \cup \{Z_1, \dots, Z_m\}, V_T, S, P \cup R)$. Evident G'_1 este de tipul 1. Să arătăm că $L(G'_1) = L(G)$.

Fie $p \in L(G)$. Dacă în derivarea $S \xrightarrow[G_1]{*} p$ nu se utilizează regula $u \rightarrow v$, este clar că putem scrie în G'_1 derivarea $S \xrightarrow{*} p$ și $p \in L(G'_1)$. Să presupunem că la un anumit pas se utilizează regula $u \rightarrow v$; detaliem

$$(G_1) : S \xrightarrow{*} \alpha = \alpha' u \alpha'' \Rightarrow \alpha' v \alpha'' = \beta \xrightarrow{*} p.$$

Pasul $\alpha \Rightarrow \beta$ se poate obține și în gramatica G'_1 utilizând regulile R :

$$(G'_1) : \alpha = \alpha' u \alpha'' = \alpha' X_1 \dots X_m \alpha'' \Rightarrow \dots \Rightarrow \alpha' v \alpha'' = \beta$$

Rezultă că $S \xrightarrow{*} p$ și deci $p \in L(G'_1)$, adică $L(G_1) \subseteq L(G'_1)$.

Invers, fie $p \in L(G'_1)$. Dacă în derivarea $S \xrightarrow{*} p$ s-au folosit numai reguli din P atunci evident $S \xrightarrow[G_1]{*} p$. Să presupunem că la un anumit pas s-a utilizat pentru prima dată o regulă din R ; aceasta nu poate să fie decât prima deoarece până în acest moment nu pot apărea simboluri Z . În continuare trebuie aplicate succesiv toate regulile din R , altfel nu putem elimina neterminalele Z . Pașii respectivi (de la α la β) vor avea forma (G'_1) de unde rezultă forma (G_1) . Prin urmare, $p \in L(G_1)$, mai departe $L(G'_1) \subseteq L(G_1)$ și $L(G'_1) = L(G_1)$. \square

Să observăm că orice gramatică dependentă de context este în mod evident monotonă. Vom demonstra în cele ce urmează afirmația reciprocă.

Teorema 4.1 *Orice gramatică monotonă este echivalentă cu o gramatică dependentă de context.*

Demonstrație. Orice gramatică monotonă se poate pune sub forma $G_n = (V_N, V_T, S, P \cup \{u_i \rightarrow v_i, i = 1, \dots, n\})$ unde regulile $u_i \rightarrow v_i$ satisfac condiția de monotonie iar P sunt reguli de tipul 1 (P conține cel puțin o regulă, anume cea care are S în stânga). Este clar că G_n este o generalizare a lui G_1 ; procedând ca în lema, putem construi o gramatică dependentă de context G'_n echivalentă cu G_n . \square

Corolar 4.2 *Familia limbajelor generate de gramatici monotone coincide cu familia \mathcal{L}_2 .*

Gramatici liniar mărginite.

Definiție 4.1 *O gramatică monotonă este de ordinul n dacă orice regulă $u \rightarrow v$ satisface condiția $|v| \leq n$.*

Lema 4.2 *O gramatică monotonă de ordinul $n \geq 3$ este echivalentă cu o gramatică monotonă de ordinul $n - 1$.*

Demonstrație. Fie $G = (V_N, V_T, S, P)$ o gramatică monotonă de ordinul $n \geq 3$. Putem presupune din nou că G are forma din lema 1.2. Construim $G' = (V'_N, V'_T, S, P')$ cu $V_N \subset V'_N$ și în plus

1. Dacă $u \rightarrow v \in P, |u|, |v| \leq 2$ atunci $u \rightarrow v \in P'$;
2. Fie $u \rightarrow v \in P, |v| > 2$; atunci $v = Y_1 Y_2 Y_3 v'$.

(a) dacă $u = X_1$ introducem în P' regulile

$$\begin{aligned} X_1 &\rightarrow Z_1 Z_2, \\ Z_1 &\rightarrow Y_1, \\ Z_2 &\rightarrow Y_2 Y_3 v'. \end{aligned}$$

(b) dacă $u = X_1 X_2 u'$ introducem în P' regulile

$$\begin{aligned} X_1 X_2 &\rightarrow Z_1 Z_2, \\ Z_1 &\rightarrow Y_1, \\ Z_2 u' &\rightarrow Y_2 Y_3 v'. \end{aligned}$$

În ambele cazuri $Z_1, Z_2 \in V'_N$.

Este clar că G' este monotonă, are ordinul $n - 1$ și $L(G') = L(G)$. \square

Observație. Procedând în mod analog putem reduce ordinul unei gramatici monotone până la valoarea 2. Prin urmare, orice gramatică monotonă este echivalentă cu o gramatică monotonă de ordinul 2.

Forma normală Kuroda.

Definiție 4.2 O gramatică se numește liniar mărginită (sau în forma normală Kuroda) dacă are reguli de generare de forma

1. $S \rightarrow SA,$
2. $B \rightarrow C|i,$
3. $DE \rightarrow FH,$

unde, ca de obicei, S este simbolul de start, $A, B, C, D, E, F, H \in V_N$ și $i \in V_T$ cu condiția $C, F, H \neq S$.

Teorema 4.3 Orice gramatică monotonă este echivalentă cu o gramatică liniar mărginită.

Demonstrație. Fie G o gramatică monotonă pe care o presupunem de ordinul 2. Prin urmare regulile din G au formele:

1. $B \rightarrow C|i,$

2. $DE \rightarrow FH$,
3. $X \rightarrow YZ$.

Regulile (1) și (2) satisfac condițiile de la gramaticile liniar mărginite, iar cele de forma (3) presupunem că nu satisfac aceste condiții, deci că $X \neq S$ sau $Y \neq S$. În general, putem avea mai multe astfel de reguli; pentru simplificarea presupunem că există două reguli de forma (3):

$$(3'); \quad X_1 \rightarrow Y_1Z_1, \quad X_2 \rightarrow Y_2Z_2.$$

Construim o gramatică liniar mărginită $G' = (V_N \cup \{S', \bar{X}_1, \bar{X}_2\}, V_T, S', P')$, unde P' conține toate regulile care convin (de formele (1) și (2)) precum și

$$\begin{aligned} S' &\rightarrow S, \\ S' &\rightarrow S'\bar{X}_1 | S'\bar{X}_2, \\ X_1\bar{X}_1 &\rightarrow Y_1Z_1, X_2\bar{X}_2 \rightarrow Y_2Z_2. \end{aligned}$$

De asemenea vom include în P' următoarele reguli de comutare (relativ la simbolurile nou introduse \bar{X}_1, \bar{X}_2):

$$\begin{aligned} A\bar{X}_1 &\rightarrow \bar{X}_1A, \\ A\bar{X}_2 &\rightarrow \bar{X}_2A, \end{aligned} \quad , \forall A \in V_N.$$

Se vede că G' este o gramatică liniar mărginită. Să arătăm că $L(G) = L(G')$.

Fie $p \in L(G)$, deci $S \xrightarrow[G]{*} p$. Presupunem că în această derivare există o singură secvență $u \xrightarrow{*} v$ în care se aplică reguli de forma (3'), adică

$$G : \quad S \xrightarrow{*} u \xrightarrow{*} v \xrightarrow{*} p.$$

Presupunem că secvența $u \xrightarrow{*} v$ are forma

$$u = u_1X_1u_2X_2u_3X_2u_4 \xrightarrow{*} u_1Y_1Z_1u_2Y_2Z_2u_3Y_2Z_3u_4 = v,$$

unde $u_1, u_2, u_3, u_4 \in V_N^*$. În G' putem scrie

$$\begin{aligned} S' &\Rightarrow S'\bar{X}_2 \Rightarrow S'\bar{X}_2\bar{X}_2 \Rightarrow S'\bar{X}_1\bar{X}_2\bar{X}_2 \Rightarrow S\bar{X}_1\bar{X}_2\bar{X}_2 \\ &\xrightarrow{*} u\bar{X}_1\bar{X}_2\bar{X}_2 = u_1X_1u_2X_2u_3X_2u_4\bar{X}_1\bar{X}_2\bar{X}_2 \xrightarrow{*} u_1X_1\bar{X}_1u_2X_2\bar{X}_2u_3X_2\bar{X}_2u_4 \\ &\xrightarrow{*} u_1Y_1Z_1u_2Y_2Z_2u_3Y_2Z_2u_4 = v \xrightarrow{*} p \end{aligned}$$

Prin urmare, $S' \xrightarrow[G']{*} p$ și $p \in L(G')$.

Invers, fie $p \in L(G')$, deci $S' \xrightarrow[G']{*} p$. Dacă în derivarea $S' \xrightarrow[G']{*} p$ apar simbolurile \bar{X}_1, \bar{X}_2 ele nu pot apărea decât în urma aplicării regulilor $S' \rightarrow$

$S' \bar{X}_1$ și $S' \rightarrow S' \bar{X}_2$, deci la începutul derivării, apoi singura posibilitate de continuare este $S' \rightarrow S$; de asemenea, aceste simboluri nu pot fi eliminate decât cu regulile $X_1 \bar{X}_1 \rightarrow Y_1 Z_1$ și $X_2 \bar{X}_2 \rightarrow Y_2 Z_2$, etc. Astfel derivarea noastră trebuie să aibă forma descrisă mai sus, de unde rezultă $S \xrightarrow{*}_G p, p \in$

$L(G)$. □

Observație. Ținând cont că gramaticile dependente de context sunt echivalente cu gramaticile monotone, rezultă că orice gramatică dependentă de context este echivalentă cu o gramatică liniar mărginită, cu alte cuvinte admite forma normală.

Structura derivărilor într-o gramatică liniar mărginită. Deoarece

$C, F, H \neq S$ rezultă că în orice derivare dintr-o gramatică liniar mărginită se aplică mai întâi un număr oarecare (să zicem m) de reguli de forma (2),(3) și la sfârșit reguli de forma (4). Este clar că lungimea cuvântului este $m + 1$.

Exemplu. Considerăm gramatica

$$\begin{aligned} S &\rightarrow SA|SB, \\ A &\rightarrow B, AB \rightarrow BA, \\ S &\rightarrow 0, A \rightarrow 0, B \rightarrow 1. \end{aligned}$$

Putem scrie derivarea

$$S \Rightarrow SA \Rightarrow SBA \Rightarrow SABA \Rightarrow SBAA \Rightarrow SBBA \xrightarrow{*} 0110.$$

Corespunzător acestei derivări putem figura hiperarboarele din figura 5.1. Să observăm că până la linia punctată s-au aplicat trei reguli de forma $S \rightarrow SA$, obținându-se cuvântul $SABA$ apoi două reguli de forma (2),(3) și apoi reguli de forma $X \rightarrow i$.

4.2 Automate liniar mărginite

Un automat liniar mărginit (*ALM*) se compune dintr-o bandă de intrare și un dispozitiv de comandă care posedă un dispozitiv de scriere-citire pe banda de intrare. Banda de intrare conține simboluri ale unui alfabet de intrare, constituind un cuvânt de intrare. Alfabetul de intrare conține un simbol special numit "marcaj" și notat "#" (diez) utilizat pentru delimitarea cuvintelor. Banda se poate mișca în ambele sensuri, sau să rămână pe loc; prin convenție vom nota, deplasarea spre stânga cu +1, deplasarea spre dreapta cu -1 și rămânerea pe loc cu 0. Dispozitivul de comandă se află întotdeauna într-o anumită stare internă, element al unei mulțimi finite de stări.

Schematic un *ALM* are forma din figura 5.2. Un *ALM* funcționează în

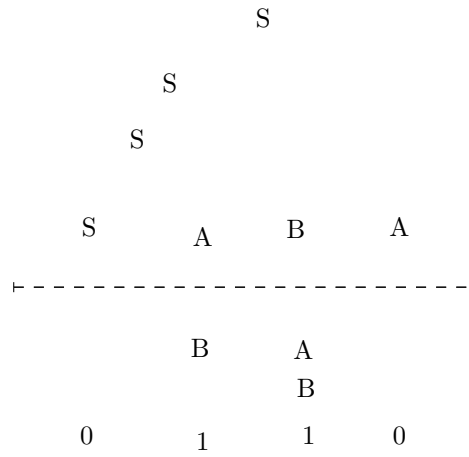
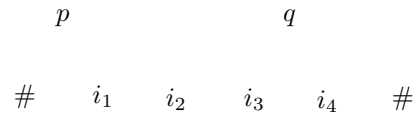


Figura 4.1: Hiperarbori asociați unei derivări



$s \in \Sigma$

Figura 4.2: Reprezentarea schematică a unui automat liniar mărginit

pași discreți. Un pas de funcționare comportă:

(a) Dispozitivul de comandă citește simbolul din dreptul dispozitivului de scriere–citire;

(b) În funcție de starea internă și de simbolul citit, automatul trece într-o nouă stare, scrie pe bandă un nou simbol (în locul simbolului citit) și mută banda cu o poziție (stânga, dreapta) sau o lasă pe loc.

Observație. Oricare ar fi starea internă a dispozitivului de comandă, citirea marcajului provoacă scrierea din nou a marcajului; de asemenea scrierea marcajului are loc numai în această situație (adică dacă s-a citit marcaj). Prin urmare, cele două marcaje care delimitează un cuvânt nu se modifică în urma funcționării.

Funcționarea unui *ALM* încetează în momentul în care s-au citit toate simbolurile cuvântului scris pe banda de intrare, mai exact în momentul în care s-a citit al doilea marcaj. Starea dispozitivului de comandă în acest moment este starea finală a automatului.

Matematic, un *ALM* este un sistem de forma $ALM = (\Sigma, I, f, s_0, \Sigma_f)$ unde:

- Σ este mulțimea de stări (finită și nevidă);
- I este alfabetul de intrare;
- $f : \Sigma \times I \longrightarrow \mathcal{P}(\Sigma \times I \times \{+1, 0, -1\})$ este funcția de evoluție;
- $s_0 \in \Sigma$ și constituie starea inițială;
- $\Sigma_f \subseteq \Sigma$ este mulțimea de stări finale.

Observație. Avem

$$f(s, i) = (s', \#, k) \Leftrightarrow i = \#.$$

O stare sau o configurație instantanee a unui *ALM* este un triplet de forma (s, p, q) unde p este partea din cuvântul de pe banda de intrare din stânga dispozitivului de scriere–citire, q este partea din dreapta a acestui cuvânt, inclusiv simbolul din dreptul dispozitivului de scriere–citire, iar s starea internă. Zicem că o stare $\sigma_1 = (p_1, s_1, q_1)$ se transformă (sau evoluează) direct în starea $\sigma_2 = (p_2, s_2, q_2)$ dacă

- (1) $q_1 = i_1 q'_1$, $f(s_1, i_1) = (s_2, i_2, +1)$, $p_2 = p_1 i_2$, $q_2 = q'_1$;
- (2) $q_1 = i_1 q'_1$, $f(s_1, i_1) = (s_2, i_2, 0)$, $p_2 = p_1$, $q_2 = i_2 q'_1$;
- (3) $q_1 = i_1 q'_1$, $p_1 = p'_1 j$, $f(s_1, i_1) = (s_2, i_2, -1)$, $p_2 = p'_1$, $q_2 = j i_2 q'_1$.

Vom spune că starea σ' evoluează (fără specificația direct) în starea σ'' și vom scrie $\sigma' \vdash^* \sigma''$ dacă $\sigma' = \sigma''$ sau dacă $\exists \sigma_1, \dots, \sigma_n$ astfel încât

$$\sigma' = \sigma_1 \vdash \sigma_2 \vdash \dots \vdash \sigma_n = \sigma''.$$

Definiție 4.3 *Limbaajul recunoscut de un ALM este*

$$L(ALM) = \{p \mid p \in I^*, (\lambda, s_0, p) \xrightarrow{*} (q, s, \lambda), s \in \Sigma_f\}.$$

Limbajele recunoscute de ALM. Automatele liniar mărginite sunt mecanisme care recunosc limbajele dependente de context, deci reprezintă o modalitate analitică de definire a acestor limbaje.

Teorema 4.4 *Un limbaj este dependent de context dacă și numai dacă este recunoscut de un automat liniar mărginit.*

Demonstrație. Partea I: $E \in \mathcal{L}_1 \Rightarrow E = L(ALM)$.

Fie $G = (V_N, V_T, S, P)$ o gramatică de tipul 1 presupusă în forma normală, astfel încât $E = L(G)$. Construim automatul liniar mărginit $ALM = (\Sigma, I, f, s_0, \Sigma_f)$ unde

$$\begin{aligned} \Sigma &= \{s_0, s_1, s_2, s_3, s_4\} \cup \{s_D \mid D \in V_N, DE \rightarrow FH \in P\}, \\ I &= V_N \cup V_T \cup \{\#, \flat\}, \\ \Sigma_f &= \{s_3\}, \end{aligned}$$

iar funcția de evoluție o definim punând în evidență patru categorii de relații:

$$Cat.I; \quad \begin{aligned} (1) f(s_0, \#) &= (s_0, \#, 1), \\ (2) f(s_3, \#) &= (s_3, \#, 1), \end{aligned}$$

$$Cat.II; \quad \left. \begin{aligned} (3) f(s_0, \#) &= (s_4, \#, 1), \\ (4) f(s_4, \#) &= (s_3, \#, 1), \end{aligned} \right\} \text{pentru } \lambda \in E;$$

$$Cat.III; \quad \begin{aligned} (5) f(s_0, x) &= \{(s_0, x, 1), (s_0, x, -1)\}, \forall x \in V_N \cup V_T; \\ (6) f(s_0, y) &= (s_0, B, 0), B \rightarrow y \in P, y \in V_N \cup V_T, \\ (7) f(s_0, F) &= (s_D, D, 1), DE \rightarrow FH \in P \\ (8) f(s_0, H) &= (s_0, E, 0), DE \rightarrow FH \in P \end{aligned}$$

$$Cat.IV; \quad \begin{aligned} (9) f(s_3, A) &= (s_0, S, 0), S \rightarrow SA \in P \\ (10) f(s_0, S) &= (s_1, S, -1), \\ (11) f(s_1, \#) &= (s_2, \#, 1), \\ (12) f(s_2, S) &= (s_3, \flat, 1), \\ (13) f(s_1, \flat) &= (s_2, \flat, 1), \\ (14) f(s_0, x_0) &= (s_1, x_0, -1), \\ (15) f(s_2, x_0) &= (s_3, \flat, 1), \\ (16) f(s_3, A) &= (s_0, x_0, 0) \end{aligned}$$

Relațiile din categoria I servesc la "pornirea" și respectiv "oprirea" automatului. Categoria II apar numai dacă gramatica conține regula $S \rightarrow \lambda$

B	D	E
y	F	H

Figura 4.3:

...	F	H	...
	s_0		
...	D	H	...
	s_D		
...	D	E	...
	s_0		

Figura 4.4:

și servesc la recunoașterea cuvântului vid λ . Regulile din categoria III se utilizează pentru reducerea succesivă a hiperarborelui de derivare și anume partea construită cu reguli de forma (2),(3), (4) (partea situată sub linia punctată de la exemplul din figura 5.1). În adevăr, cu ajutorul relației (5) putem plasa dispozitivul de comandă sub oricare din simbolurile cuvântului curent. Apoi, cu ajutorul relațiilor (7) și respectiv (8),(9) putem reduce subhiperarbori de forma prezentată în figura 5.3. corespunzător regulilor de forma $B \rightarrow y$ și $DE \rightarrow FH$. De exemplu, avem următoarea funcționare (vezi figura 5.4).

În acest fel ajungem la partea triunghiulară a hiperarborelui (situată în exemplu deasupra liniei punctate) care va avea pe frontieră un cuvânt de forma $\#SXY \dots Z\#$ cu $X, Y, \dots, Z \in V_N$. Această parte va fi redusă cu relațiile din categoria IV. Astfel, după plasarea dispozitivului de citire-scriere pe S , cu ajutorul relației (10), dispozitivul se plasează pe marcajul de început și trece în starea s_1 , după care revine pe S în starea s_2 (relația 11). În continuare au loc secvențe de forma din figura 5.5.

În urma acestor secvențe pe bandă se va scrie cuvântul $\#bb \dots b\#$ iar dispozitivul se va plasa pe marcajul de sfârșit în starea s_3 . În sfârșit, cu ajutorul relației (2) funcționarea automatului este oprită.

\cdots	S	X	Y	\cdots		#	X_0	A	B	\cdots
	s_2						s_0			
\cdots	b	X	Y	\cdots		#	X_0	A	B	\cdots
		s_3					s_1			
\cdots	b	S	Y	\cdots		#	X_0	A	B	\cdots
		s_0					s_2			
\cdots	b	S	Y	\cdots		#	b	A	B	\cdots
		s_1					s_0			
\cdots	b	S	Y	\cdots		\cdots	b	X_0	B	\cdots
		s_2					s_0			

Figura 4.5:

Este clar că dacă $p \in E$ atunci automatul definit mai sus va avea o evoluție de acest tip și reciproc, deci $E = L(ALM)$. \square

Exemplu: Evoluția ALM corespunzător gramaticii de la exemplul din paragraful precedent pentru cuvântul $\#0110\#$ este:

$$\begin{aligned}
&(\lambda, s_0, \#0110\#) \mapsto (\#, s_0, 0110\#) \mapsto (\#0, s_0, 110\#) \mapsto (\#01, s_0, 10\#) \mapsto \\
&(\#01, s_0, B0\#) \mapsto \dots \mapsto (\#S, s_0, BAA\#) \mapsto (\#SA, s_D, AA\#) \mapsto \\
&(\#SA, s_0, BA\#) \mapsto \dots \mapsto (\#, s_0, SABAA\#) \mapsto (\lambda, s_1, SBAA\#) \mapsto \\
&(\#, s_2, ABA\#) \mapsto (\#b, s_3, ABA\#) \mapsto (\#b, s_0, SBA\#) \mapsto (\#, s_1, SBA\#) \mapsto \\
&(\#b, s_2, SBA\#) \mapsto (\#bb, s_3, BA\#) \mapsto \dots \mapsto (\#bbb, s_3, \#)
\end{aligned}$$

Capitolul 5

Limbaje de tipul zero

5.1 Forma normală

Limbajele de tipul 0 sunt generate de gramatici Chomsky fără restricții asupra regulilor de derivare. Se poate arăta că orice gramatică de tipul zero acceptă următoarea formă normală

$$\begin{aligned} S &\rightarrow SA, \\ B &\rightarrow C|i|\lambda, \\ DE &\rightarrow FH, \end{aligned}$$

unde $A, C, F, H \neq S$. Se observă că singura deosebire față de gramaticile dependente de context este aceea că putem avea și λ -reguli.

Demonstrația existenței formei normale urmează în linii mari aceeași cale cu cea de la limbaje dependente de context. În prealabil gramatica se completează cu neterminale noi astfel încât orice regulă care nu este de ștergere să respecte condiția de monotonie. Acest lucru se poate realiza după cum urmează. Fie

$$X_1 \dots X_n \rightarrow Y_1 \dots Y_m, n > m,$$

o regulă care nu respectă condiția de monotonie. Vom pune

$$X_1 \dots X_n \rightarrow Y_1 \dots Y_m Z_{m+1} \dots Z_n, Z_{m+1} \rightarrow \lambda, \dots, Z_n \rightarrow \lambda.$$

Este clar că orice derivare directă $u \Rightarrow v$ în gramatica inițială se poate obține și în gramatica modificată și reciproc (neterminalele Z nu pot apărea în stânga altor relații). În acest fel, gramatica se va transforma astfel

încât să conțină două categorii de reguli: reguli care respectă condiția de monotonie și λ -reguli.

În continuare reducem ordinul regulilor monotone, urmând aceiași procedură ca la limbaje dependente de context, până la ordinul 2 inclusiv (λ -regulile nu intervin cu nimic în acest proces). În final obținem reguli de următoarele categorii:

- (1) $B \rightarrow C|i|\lambda$,
- (2) $DE \rightarrow FH$,
- (3) $X \rightarrow YZ$.

Regulile de forma (3) cu $X \neq S$ sau $Y \neq S$ le eliminăm cu procedura cunoscută plus un simbol de start nou, cu reguli stâng recursive cunoscute pentru neterminalele nou introduse.

Relativ la puterea generativă a gramaticilor de tipul zero se poate arăta că $\mathcal{L}_1 \subset \mathcal{L}_0$ (strict). Demonstrația se face pe o cale indirectă (nu s-a găsit un exemplu de limbaj de tipul zero care să nu poată fi generat de o gramatică dependentă de context).

După cum am văzut în primul capitol limbajele de tipul zero sunt închise față de operațiile regulate. Se poate arăta că familia \mathcal{L}_0 este închisă la intersecție.

Menționăm și următoarea "teoremă a spațiului de lucru". Fie G o gramatică de tipul zero. Pentru o derivare

$$D : S = u_0 \Rightarrow u_1 \Rightarrow \dots \Rightarrow u_n = p \in V_T^*$$

definim

$$WS_D(p) = \max_{i=1, \dots, n} |u_i|$$

și

$$WS(p) = \min_D \{WS_D(p)\}.$$

Observație. WS este prescurtare de la *working space*.

Spunem că o gramatică de tipul zero G are spațiul de lucru mărginit dacă există $k \in \mathbb{N}$ astfel încât pentru $\forall p \in L(G)$ să avem $WS(p) \leq k|p|$. Să observăm că orice gramatică care nu are reguli de ștergere (excepție $S \rightarrow \lambda$ și atunci S nu apare în dreapta) satisface această condiție cu $k = 1$.

Teorema 5.1 (teorema spațiului de lucru) *dacă o gramatică G are spațiul de lucru mărginit, atunci $L(G) \in \mathcal{L}_1$.*

$$p = i_1 i_2 i_3 i_4 i_5, n = 4$$

$$i_1 \quad i_2 \quad i_3 \quad i_4 \quad i_5$$

$$s \in \Sigma$$

Figura 5.1: Reprezentarea schematică a mașinii Turing

5.2 Mașina Turing

Conceptul de mașină Turing. Mașina Turing este un mecanism de recunoaștere a limbajelor de tipul zero. În felul acesta, familiilor de limbaje din clasificarea Chomsky le corespund automate specifice de recunoaștere.

- Familia \mathcal{L}_3 - automate finite,
- Familia \mathcal{L}_2 - automate push-down,
- Familia \mathcal{L}_1 - automate liniar mărginite,
- Familia \mathcal{L}_0 - mașina Turing.

O mașină Turing (prescurtat MT) se compune dintr-un dispozitiv de comandă care posedă un dispozitiv de scriere-citire și o bandă de intrare. Banda de intrare se consideră mărginită la stânga și nemărginită la dreapta; ea conține simboluri ale unui alfabet de intrare, începând din prima poziție. Alfabetul are un simbol special numit blanc și notat b (sau spațiu), care se consideră înregistrat în toată partea neocupată a benzii. Indicele simbolului din dreptul dispozitivului de scriere-citire îl notăm cu n și el va constitui un element al stării mașinii Turing. Dispozitivul de comandă se află într-o anumită stare internă, element al unei mulțimi finite de stări.

Schema unei mașini Turing este dată în figura 6.1. O mașină Turing funcționează în pași discreți. Un pas de funcționare constă din:

Dispozitivul de comandă citește simbolul aflat în dreptul dispozitivului de scriere-citire; în funcție de simbolul citit, mașina Turing trece într-o nouă stare internă, scrie pe bandă un nou simbol (întotdeauna diferit de blanc) în locul simbolului citit și mută banda cu o poziție (spre stânga sau dreapta) sau o lasă pe loc. Convențional, vom nota cu $+1$ o mișcare spre stânga, cu

–1 spre dreapta și cu 0 pe loc.

Din punct de vedere matematic, o mașină Turing este un sistem

$$MT = (\Sigma, I, f, s_0, \Sigma_f)$$

unde:

Σ este mulțimea de stări;

I este alfabetul de intrare;

$f : (\Sigma \times I)' \rightarrow \Sigma \times (I - \{b\}) \times \{1, 0, -1\}$, $(\Sigma \times I)' \subseteq \Sigma \times I$,
este funcția de evoluție;

$s_0 \in \Sigma$ este simbolul de start;

$\Sigma_f \subseteq \Sigma$ este mulțimea de stări finale.

Observație. Funcția f nu este definită pe întregul produs $\Sigma \times I$; dacă mașina ajunge într-o stare s iar în dreptul dispozitivului de scriere–citire se află simbolul i și $(s, i) \notin (\Sigma \times I)'$ spunem că mașina se blochează. Există și alte cazuri de blocare, de exemplu situația în care dispozitivul de scriere–citire se află în dreptul primului simbol, iar pasul de funcționare prevede o mișcare a benzii spre dreapta.

O stare (sau configurație) a unei mașini Turing este un triplet de forma $\sigma = (s, p, n)$ unde s este starea internă, p este cuvântul scris pe bandă iar n este indicele simbolului din dreptul dispozitivului de scriere–citire.

Vom spune că starea $\sigma_1 = (s_1, p_1, n_1)$ *evoluează direct* în starea $\sigma_2 = (s_2, p_2, n_2)$ și vom scrie $\sigma_1 \mapsto \sigma_2$ dacă se efectuează un pas de evoluție. În termenii funcției de evoluție, avem

$$(1) f(s_1, i_{n_1}) = (s_2, i, +1), p_2 = i_1 \dots i_{n_1-1} i i_{n_1+1} \dots i_m, n_2 = n_1 + 1;$$

$$(2) f(s_1, i_{n_1}) = (s_2, i, -1), p_2 = i_1 \dots i_{n_1-1} i i_{n_1+1} \dots i_m, n_2 = n_1 - 1;$$

$$(3) f(s_1, i_{n_1}) = (s_2, i, 0), p_2 = i_1 \dots i_{n_1-1} i i_{n_1+1} \dots i_m, n_2 = n_1;$$

$$(4) f(s_1, b) = (s_2, i, +1), p_2 = p_1 i, n_2 = n_1 + 1;$$

$$(5) f(s_1, b) = (s_2, i, -1), p_2 = p_1 i, n_2 = n_1 - 1;$$

$$(6) f(s_1, b) = (s_2, i, 0), p_2 = p_1 i, n_2 = n_1;$$

Vom spune că σ' *evoluează* (fără specificația direct) în σ'' și vom nota $\sigma' \mapsto^* \sigma''$ dacă $\sigma' = \sigma''$ sau dacă există $\sigma_1, \dots, \sigma_n$ astfel încât

$$\sigma' = \sigma_1 \mapsto \sigma_2 \mapsto \dots \mapsto \sigma_n = \sigma''.$$

Limbaajul recunoscut de o mașină Turing este prin definiție

$$L(MT) = \{p \mid p \in I^*, (s_0, p, 1) \mapsto^* (s, q, \epsilon), s \in \Sigma_f\}.$$

Observație. Este posibil ca mașina să ajungă într-o stare finală înainte de citirea integrală a lui p ; analiza stării finale trebuie făcută numai după parcurgerea cuvântului.

Exemplu. Considerăm mașina turing $MT = (\Sigma, I, f, s_0, \Sigma_f)$ unde $\Sigma = \{s_0, s_1, s_2\}$, $I = \{0, 1\}$, $\Sigma_f = \{s_1\}$ iar funcția de evoluție este dată de

f	s_0	s_1	s_2
b	$(s_2, 1, 1)$		$(s_0, 0, -1)$
0	$(s_1, 0, 1)$		
1	$(s_0, 1, 1)$	$(s_0, 0, 1)$	

Evoluția mașinii pentru $p = 001$ este

$$(s_0, 011, 1) \mapsto (s_1, 011, 2) \mapsto (s_0, 001, 3) \mapsto (s_0, 001, 4) \mapsto \\ (s_2, 0011, 5) \mapsto (s_0, 0011, 4) \mapsto (s_0, 00110, 5) \mapsto (s_1, 00110, 6)$$

Se poate observa că după citirea întregului cuvânt mașina poate să efectueze un număr de pași suplimentari până la ajungerea într-o stare finală, deci este posibil ca $|p| < |q|$.

Situații în care o mașină Turing se blochează (în aceste situații cuvântul scris pe bandă nu este recunoscut):

1. MT ajunge într-o stare s , în dreptul dispozitivului de citire-scriere se află simbolul i și $(s, i) \notin (\Sigma \times I)'$;
2. MT este în starea s , a citit simbolul din prima poziție i și $f(s, i) = (s', i', -1)$;
3. MT efectuează un ciclu infinit în interiorul cuvântului.

Definiție 5.1 Vom spune că o mașină Turing este *nestaționară* dacă

$$f : (\Sigma \times I)' \rightarrow \Sigma \times (I \setminus \{b\}) \times \{-1, +1\}.$$

Prin urmare, o mașină Turing *nestaționară* nu lasă în nici o situație banda pe loc.

Lema 5.1 Orice mașină Turing este echivalentă cu o mașină Turing *nestaționară*.

Demonstrație. Pornind de la o MT dată construim o MT' *nestaționară* astfel:

Dacă $f(s, i) = (s', i', \pm 1)$ vom pune $f'(s, i) = f(s, i)$;

Dacă $f(s, i) = (s', i', 0)$ vom pune $f'(s, i) = (s'', i', -1)$ și $f'(s'', j) = (s', j, +1)$, $\forall j \in I$, unde s'' este o stare nouă introdusă.

Astfel, în cazul unei rămăneri pe loc a lui MT, noua mașină va face un pas spre stânga și unul spre dreapta, fără să modifice nici starea și nici conținutul benzii. Evident, cele două mașini sunt echivalente.

Limbațele recunoscute de mașini Turing.

Teorema 5.2 *Un limbaj este recunoscut de o mașină Turing dacă și numai dacă este de tipul zero.*

Demonstrație. Partea I. $E = L(MT) \Rightarrow E \in \mathcal{L}_0$.

Fie $MT = (\Sigma, I, f, s_0, \Sigma_f)$ o mașină Turing astfel încât $E = L(MT)$. Putem presupune că MT este nestaționară. Fie $I_\lambda = (I \setminus \{b\}) \cup \{\lambda\}$.

Construim o gramatică de tipul zero astfel: $G = (V_N, V_T, S, P)$ unde

$$V_N = \Sigma \cup \{(i, j) \mid i \in I_\lambda, j \in I\} \cup \{S, X_1, X_2\}; \quad V_T = I \setminus \{b\}.$$

Definiția lui P :

$$S \rightarrow s_0 X_1, \quad X_1 \rightarrow (i, i) X_1, \quad i \in I \setminus \{b\},$$

$$X_1 \rightarrow X_2, \quad X_2 \rightarrow (\lambda, b) X_2, \quad X_2 \rightarrow \lambda,$$

$$\text{dacă } f(s, i) = (s', i', 1) \text{ atunci } s(i_1, i) \rightarrow (i_1, i') s', \quad \forall i_1 \in I_\lambda,$$

$$\text{dacă } f(s, i) = (s', i', -1) \text{ atunci } (i_1, i_2) s(i_3, i) \rightarrow s'(i_1, i_2) (i_3, i'), \quad i_1, i_3 \in I_\lambda, \quad i_2 \in I,$$

$$\text{dacă } s \in \Sigma_f, \text{ atunci } s(i_1, i_2) \rightarrow s i_1 s \text{ și } (i_1, i_2) s \rightarrow s i_1 s.$$

Fie $p \in L(MT), p = i_1 \dots i_n$. Presupunem că MT utilizează în recunoaștere m poziții de pe bandă situate la dreapta cuvântului p . În G avem

$$S \Rightarrow s_0 X_1 \xrightarrow{*} s_0 (i_1, i_1) \dots (i_n, i_n) X_2 \xrightarrow{*} s_0 (i_1, i_1) \dots (i_n, i_n) (\lambda, b)^m.$$

Cuvântul p fiind recunoscut de MT avem

$$(1) \quad (s_0, i_1 \dots i_n, 1) \xrightarrow{*} (s_f, i'_1 \dots i'_h, k), \quad h \geq n.$$

Vom arăta că (1) implică existența unei derivări de forma

$$2) s_0 (i_1, i_1) \dots (i_n, i_n) (\lambda, b)^m \xrightarrow{*} (i_1, i'_1) \dots (i_{k-1}, i'_{k-1}) s_f (i_k, i'_k) \dots (i_{n+m}, i'_{n+m}),$$

unde

$$\begin{aligned} i_1, \dots, i_n &\in I \setminus \{b\}, \quad i_{n+1}, \dots, i_{n+m} = \lambda, \\ i'_1, \dots, i'_h &\in I \setminus \{b\}, \quad i'_{h+1}, \dots, i'_{n+m} = \lambda. \end{aligned}$$

Demonstrație prin inducție asupra lungimii l a evoluției.

Pentru $l = 0$ avem

$$(s_0, i_1 \dots i_n, 1) \xrightarrow{*} (s_0, i_1 \dots i_n, 1), \quad k = 1, \quad h = n, \quad i'_j = i_j.$$

Partea dreaptă a lui 2) va avea forma $s_0 (i_1, i_1) \dots (i_n, i_n) (\lambda, b)^m$ și deci 2) este adevărată.

$$\begin{array}{ccc}
i_1'' & i_2'' & i_3'' & \dots & i_g'' & & i_1'' & i_2'' & i_3'' & \dots & i_g'' \\
s & & & & & & & & & & s \\
h = g & & & & & & h = g + 1 & & & &
\end{array}$$

Figura 5.2: Configurații posibile ale mașinii Turing

Presupunem că implicația este adevărată pentru l oarecare și considerăm o evoluție de lungime $l + 1$. Punem în evidență ultima evoluție directă

$$(s_0, i_1 \dots i_n, 1) \xrightarrow{*} (s, i_1'' \dots i_g'', j) \xrightarrow{} (s_f, i_1' \dots i_h', k).$$

În general $h = g$ sau $h = g + 1$ în conformitate cu următoarele două cazuri (figura 6.2).

Întotdeauna $k = j \pm 1$. apoi $i_t'' = i_t'$, $t = 1, \dots, g, t \neq j$, adică

$$\begin{array}{c}
i_1'', i_2'', \dots, i_{j-1}'', i_j'', i_{j+1}'', \dots, i_g''; \\
i_1', i_2', \dots, i_{j-1}', i_j', i_{j+1}', \dots, i_g'.
\end{array}$$

În urma ultimei evoluții directe vor diferi numai simbolurile i_j'', i_j' .

Din ipoteza inductivă rezultă

$$s_0(i_1, i_1) \dots (i_n, i_n)(\lambda, b)^m \xrightarrow{*} (i_1, i_1') \dots (i_{j-1}, i_{j-1}') s(i_j, i_j'') \dots (i_{n+m}, i_{n+m}'').$$

În conformitate cu definiția evoluției directe avem

$$\begin{array}{l}
f(s, i_j'') = (s_f, i_j', 1), \quad k = j + 1, \quad s(i_j, i_j'') \rightarrow (i_j, i_j'') s_f; \\
f(s, i_j') = (s_f, i_j', -1), \quad k = j - 1, \quad (i_{j-1}, i_{j-1}') s(i_j, i_j') \rightarrow s_f(i_j, i_{j-1}') (i_j, i_j').
\end{array}$$

În ambele cazuri

$$s_0(i_1, i_1) \dots (i_n, i_n)(\lambda, b)^m \xrightarrow{*} (i_1, i_1') \dots (i_{k-1}, i_{k-1}') s_f(i_k, i_k') \dots (i_{n+m}, i_{n+m}').$$

Acum, deoarece $s_f \in \Sigma_f$, putem scrie

$$\begin{array}{l}
S \xrightarrow{*} s_0(i_1, i_1) \dots (i_n, i_n)(\lambda, b)^m \\
\xrightarrow{*} (i_1, i_1') \dots (i_{k-1}, i_{k-1}') s_f(i_k, i_k') \dots (i_{m+n}, i_{m+n}') \\
\Rightarrow (i_1, i_1') \dots (i_{k-1}, i_{k-1}') s_f i_k s_f (i_{k+1}, i_{k+1}') \dots (i_{m+n}, i_{m+n}') \\
\Rightarrow (i_1, i_1') \dots (i_{k-2}, i_{k-2}') s_f i_{k-1} s_f i_k s_f i_{k+1} s_f (i_{k+2}, i_{k+2}') \dots (i_{m+n}, i_{m+n}') \\
\xrightarrow{*} s_f i_1 s_f i_2 s_f \dots s_f i_n s_f \xrightarrow{*} i_1 \dots i_n = p.
\end{array}$$

Prin urmare $p \in L(G)$ și $L(MT) \subseteq L(G)$. Analog se arată și incluziunea inversă și deci $L(MT) = L(G)$. \square

Cuprins

1	Limbaje formale	3
1.1	Limbaje, gramatici, proprietăți generale	3
1.2	Ierarhia Chomsky	11
1.3	Proprietăți de închidere a familiilor Chomsky	15
2	Limbaje regulate	23
2.1	Automate finite și limbaje regulate	23
2.2	Proprietăți speciale ale limbajelor regulate	29
2.3	Lema de pompare pentru limbaje regulate	33
2.4	Expresii regulate	35
2.5	Sisteme tranzitionale	36
3	Limbaje independente de context	41
4	Limbaje independente de context	43
4.1	Arbori de derivare	43
4.2	Decidabilitate și ambiguitate în familia \mathcal{L}_2	46
4.3	Forme normale pentru gramatici de tipul 2	49
4.4	Automate push-down (APD)	56
4.5	Automate push-down deterministe	65
4.6	Lema Bar-Hillel	68
4.7	Închiderea familiei \mathcal{L}_2 la substituții	72
4.8	Caracterizarea limbajelor independente de context	75
5	Limbaje dependente de context	83
5.1	Gramatici monotone	83
5.2	Automate liniar mărginite	87

6	Limbaje de tipul zero	93
6.1	Forma normală	93
6.2	Maşina Turing	95