Prefaţă

Cartea conţine materia pentru studenţii informaticieni din anul III, învăţământ la distanţă. Conţinutul cărţii este împărţit în cinci teme. Fiecare temă este formată dintr-o prezentare teoretică, insistându-se pe clasificarea noţiunilor introduse şi mai puţin pe demonstrarea tuturor teoremelor, după care urmează câte un set de exerciţii rezolvate şi propuse spre rezolvare.

Cu toate că lipsesc demonstraţiile unor teoreme, cartea este utilă şi studenţilor din anul II de la învăţământ de la zi.

Sper ca munca depusă în redactarea acestui material să fie de folos studenţilor informaticieni.

Autorul

Cuprins

Tema 1

Limbaje şi gramatici generative

1 Noţiuni introductive

Prelucrarea informaţiilor cu ajutorul calculatoarelor electronice presupune lucrul cu şiruri de caractere, de aceea este important de studiat proprietăţile şi modurile de generare a şirurilor de caractere. În acest paragraf vom introduce noţiunile de bază pentru limbaje formale şi teoria automatelor.

Definiţia 1.1.1. O mulţime finită şi nevidă V se numeşte *alfabet*. Elementele lui V se numesc *simboluri* sau *simboli*.

Definiţia 1.1.2. Se numeşte cuvânt peste alfabetul V o aplicaţie $p : \{1, 2, \ldots, n\} \to V$. Numărul n se numeşte *lungimea cuvântului* p şi se notează cu $l(p)$ sau $|p|$.

Cuvântul p se notează cu $p(1)p(2)\ldots p(n)$. Mulţimea cuvintelor peste alfabetul V se notează cu V^+.

Exemplul 1.1.3. Fie $V = \{a, b, c, d\}$ şi cuvântul $p : \{1, 2, 3\} \to V$ definit prin $p(1) = a$, $p(2) = c$, $p(3) = d$. Acest cuvânt se scrie *acd*. Fie $u : \{1, 2, 3, 4, 5\} \to V$ definit prin $u(1) = b$, $u(2) = a$, $u(3) = c$, $u(4) = d$ şi $u(5) = a$. Acest cuvânt se scrie *bacda*.

Pe mulţimea cuvintelor peste un alfabet se poate defini operaţia de concatenare.

Definiţia 1.1.4. Fie $p : \{1, 2, \ldots, n\} \to V$ şi $q : \{1, 2, \ldots, m\} \to V$. Atunci *concatenarea* cuvintelor p şi q este cuvântul $p \cdot q : \{1, 2, \ldots, n + m\} \to V$ definit prin

$$p \cdot q(j) = \begin{cases} p(j) & \text{dacă } i \leq j \leq n \\ q(j - n) & \text{dacă } n + 1 \leq j \leq n + m \end{cases}$$

Concatenarea cuvintelor p şi q se notează cu $p \cdot q$ sau simplu pq. Din definiţie urmează că $pq = p(1) \ldots p(n)q(1) \ldots q(m)$.

Vom nota cu $V^* = V^+ \cup \{\lambda\}$, unde λ este un element nou, $\lambda \notin V^+$ pentru care extindem operaţia de concatenare $\lambda \cdot p = p \cdot \lambda = p$ pentru orice $p \in V^+$ si $\lambda \cdot \lambda = \lambda$. Vom defini $l(\lambda) = 0$. Cuvântul λ se numeşte *cuvânt vid* sau *cuvânt nul*.

Observaţia 1.1.5. Operaţia de concatenare este o operaţie **asociativă**, adică: $(pq)r = p(qr)\ \forall p, q, r \in V^*$.

Observaţia 1.1.6. $(V^*, \cdot)$ este monoid, se numeşte *monoidul liber generat de V*.

Observaţia 1.1.7. Aplicaţia $l : V^* \to N$ care asociază unui cuvânt $p \in V^*$ lungimea sa $l(p)$ este un **homomorfism**, adică $l(p \cdot q) = l(p) + l(q)$; lungimea unui cuvânt p se mai notează şi cu $|p|$.

Semigrupul $(V^*, \cdot)$ se numeşte *monoidul liber peste alfabetul V*. De asemenea vom identifica cuvintele de lungime unu cu simbolurile lui V. Vom nota cu $|V|$ numărul simbolurilor alfabetului V.

Lema 1.1.8. *Numărul cuvintelor de lungime k peste alfabetul V este egal cu $|V|^k$.*

Demonstraţie. Fie $n = |V|$. Numărul cuvintelor de lungime 1 este egal cu n; numărul cuvintelor de lungime 0 este 1. Presupunem proprietatea adevărată pentru cuvinte de lungime cel mult $k - 1$ şi o demonstrăm pentru cuvinte de lungime k. Deci numărul cuvintelor de lungime $k-1$ este n^{k-1}. Cuvintele de lungime k le putem obţine din cuvinte de lungime $k-1$ adăugând la sfârşit un simbol din V. Deci din fiecare cuvânt de lungime $k-1$ obţinem n cuvinte de lungime k. Deci numărul cuvintelor de lungime k este $n^{k-1} \cdot n = n^k$. ■

Observaţia 1.1.9. Mulţimea cuvintelor peste un alfabet V este o mulţime **numărabilă**, ca reuniune numărabilă de mulţimi finite.

$$V^* = \{\lambda\} \cup \bigcup_{k=1}^{\infty} \{p \mid p \in V^*, l(p) = k\}.$$

Observaţia 1.1.10. Monoidul $(V^*, \cdot)$ **nu este comutativ**.

De exemplu, dacă luăm $V = \{a, b\}$ şi cuvintele $p = ab$ şi $q = bb$, avem $p \cdot q \neq q \cdot p$.

Definiţia 1.1.11. Cuvântul q este *prefix* al cuvântului p dacă $\exists v \in V^*$, $p = qv$ şi este *prefix propriu* dacă $v \in V^+$. Cuvântul q este *sufix* al cuvântului p dacă $\exists u \in V^*$, $p = uq$ şi este *sufix propriu* dacă $u \in V^+$.

Lema 1.1.12. *Orice aplicaţie $\varphi : V \to S$, unde S este un semigrup cu operaţia $\oplus$, se extinde în mod unic la un homomorfism de semigrupuri $\varphi : V^+ \to S$.*

Demonstraţie. Fie $p = i_1 \dots i_n$ un cuvânt oarecare din V^+. Vom defini $\varphi(p)$ prin $\varphi(i_1 \dots i_n) = \varphi(i_1) \oplus \varphi(i_2) \oplus \dots \oplus \varphi(i_n)$. Se verifică imediat că aplicaţia φ astfel extinsă este un homomorfism de la V^+ la S.

Să demonstrăm unicitatea lui φ. Să presupunem că ar mai exista un homomorfism $\psi : V^+ \to S$ cu $\varphi(i) = \psi(i)$, oricare ar fi $i \in V$. Să demonstrăm că $\varphi(p) = \psi(p)$, oricare ar fi $p \in V^+$. Vom demonstra prin inducţie după lungimea cuvintelor. Pentru cuvinte de lungime 1, propoziţia este adevărată. Presupunem propoziţia adevărată pentru un cuvânt de lungime cel mult k şi o demonstrăm pentru cuvinte de lungime $k+1$. Fie p cu $l(p) = k+1$. Atunci $p = qi$, cu $l(q) = k$ şi $l(i) = 1$. Avem $\varphi(p) = \varphi(q \cdot i) = \varphi(q) \oplus \varphi(i) = \psi(q) \oplus \psi(i) = \psi(qi) = \psi(p)$. ■

Definiţia 1.1.13. Vom numi *limbaj peste* V orice submulţime a lui V^*.

Un limbaj fiind o mulţime de cuvinte, putem utiliza operaţiile cu mulţimi ca: reuniune, intersecţie şi complementariere.

Limbajele fiind mulţimi specifice ale căror elemente sunt cuvinte, putem defini operaţii proprii acestor mulţimi.

Definiţia 1.1.14. Operaţia de produs a limbajelor este definită pe $\mathcal{P}(V^*) \times \mathcal{P}(V^*)$ cu valori în $\mathcal{P}(V^*)$, care ataşează fiecărei perechi (L_1, L_2) un limbaj notat prin $L_1 \cdot L_2$ definit prin:

$$L_1 \cdot L_2 = \{p \mid p = uv,\ u \in L_1 \text{ şi } v \in L_2\}$$

Produsul $L_1 \cdot L_2$ se mai notează simplu prin $L_1 L_2$.

Prin definiţie avem $\phi \cdot L = L \cdot \phi = \phi$ pentru orice $L \in \mathcal{P}(V^*)$.

Observaţia 1.1.15. Limbajul $\{\lambda\}$ este **element neutru**, deoarece se poate verifica imediat că:

$$\{\lambda\} \cdot L = L \cdot \{\lambda\} = L,\ \forall L \in \mathcal{P}(V^*).$$

Observaţia 1.1.16. Se poate verifica uşor că operaţia de produs a două limbaje este **asociativă**.

Din definiţia produsului de limbaje, se poate verifica că pentru orice trei limbaje L_1, L_2 şi L_3 peste alfabetul V avem: $(L_1 \cdot L_2) \cdot L_3 = L_1 \cdot (L_2 \cdot L_3)$. Mulţimea $\mathcal{P}(V^*)$ înzestrată cu operaţia de produs este un semigrup cu element neutru $\{\lambda\}$.

Definiţia 1.1.17. *Puterea* unui limbaj L se defineşte inductiv prin:

$L^0 = \{\lambda\}$
$L^1 = L$
$L^{n+1} = L^n \cdot L,\ \forall n \in N.$

Vom nota cu $L^+ = \bigcup_{n=1}^{\infty} L^n$ şi cu $L^* = \bigcup_{n=0}^{\infty} L^n$. Limbajul L^* se numeşte *iteratul* limbajului L.

2 Gramatici generative şi limbaje generate de gramatici

Un program într-un limbaj de programare poate fi privit ca un şir de caractere generat după nişte reguli precizate, pornind de la un alfabet. Generarea şirurilor de caractere se poate formaliza prin introducerea noţiunii de *gramatică*.

Definiţia 1.2.1. O gramatică este un 4-uplu $G = (V_N, V_T, x_0, P)$ unde:
- V_N este o mulţime finită şi nevidă, elementele sale se numesc *neterminali* sau variabile;
- V_T este o mulţime finită şi nevidă, elementele sale se numesc *terminali* şi $V_N \cap V_T = \phi$;
- $x_0 \in V_N$ este un simbol special numit *simbol iniţial* sau simbol de start;
- fie $V = V_N \cup V_T$, atunci P este o mulţime finită şi $P \subseteq V^* V_N V^* \times V^*$.

Mulţimea P se numeşte *mulţimea regulilor de generare* sau mulţimea producţiilor gramaticii G. Vom nota perechea $(u, v) \in P$ prin $u \to v$. Se observă că în orice regulă $u \to v$ cuvântul u conţine cel puţin un simbol neterminal.

Vom defini o relaţie binară peste V^* numită *derivare directă*, notată cu $\underset{G}{\Rightarrow} \subseteq V^* \times V^*$.

Definiţia 1.2.2. Vom spune că perechea $(\alpha, \beta) \in \underset{G}{\Rightarrow}$ dacă există o regulă $u \to v \in P$ şi $\alpha = \alpha_1 u \alpha_2$, $\beta = \alpha_1 v \alpha_2$.

Deci două cuvinte α şi β sunt în relaţie de derivare directă dacă β se obţine din α prin înlocuirea membrului stâng al unei reguli prin membrul drept al regulii (adică u prin v). Dacă $(\alpha, \beta) \in \underset{G}{\Rightarrow}$, vom nota aceasta prin $\alpha \underset{G}{\Rightarrow} \beta$, care este mai sugestivă, β se obţine din α prin aplicarea unei reguli din G. Închiderea reflexivă şi tranzitivă a relaţiei

$\underset{G}{\Rightarrow}$ o vom nota cu $\underset{G}{\overset{*}{\Rightarrow}}$ şi o vom numi *relaţia de derivare.* Deci vom spune că $\alpha \underset{G}{\overset{*}{\Rightarrow}} \beta$ dacă are loc una din următoarele situaţii:

i) $\alpha = \beta$ (închiderea reflexivă);

ii) există cuvintele $u_1, \ldots, u_n \in V^*$ astfel încât $\alpha = u_1$, $\beta = u_n$ şi $u_i \underset{G}{\Rightarrow} u_{i+1}$, $n \geq 2$, $i \in \overline{1, n-1}$.

Definiţia 1.2.3. O derivare în gramatica G este un şir de cuvinte $u_1, \ldots, u_{n+1}$, $u_j \in V^*$, $1 \leq j \leq n+1$, astfel încât $u_i \underset{G}{\Rightarrow} u_{i+1}$, $1 \leq i \leq n$.

Deci o derivare este un şir de cuvinte care se obţine unul din altul prin aplicarea unei reguli din G, numărul n din definiţia de mai sus se numeşte lungimea derivării. Deci lungimea derivării este numărul regulilor care se aplică pentru a obţine u_{n+1} din u_1. Vom considera că $\alpha \overset{*}{\Rightarrow} \alpha$ printr-o derivare de lungime 0, adică nu se aplică nici o regulă din G.

Atunci când nu există posibilitatea de confuzie, vom renunţa la indicele G din notaţiile $\underset{G}{\Rightarrow}$ şi $\underset{G}{\overset{*}{\Rightarrow}}$ şi vom folosi $\Rightarrow$ şi $\overset{*}{\Rightarrow}$.

Definiţia 1.2.4. Limbajul generat de gramatica G, notat cu $L(G)$, este definit prin:

$$L(G) = \{p \mid p \in V_T^*,\ x_0 \underset{G}{\overset{*}{\Rightarrow}} p\}.$$

Remarcăm că limbajul generat de gramatica G este format din toate cuvintele peste V_T (alfabetul terminalilor) care se obţin din simbolul de start aplicând reguli din G. Aici se vede rolul special care îl are x_0; toate derivările care ne dau cuvinte din $L(G)$ încep cu simbolul x_0.

Definiţia 1.2.6. Două gramatici $G_1 = (V_{N_1}, V_{T_1}, x_{01}, P_1)$ şi $G_2 = (V_{N_2}, V_{T_2}, x_{02}, P_2)$ se numesc *echivalente* dacă $L(G_1) = L(G_2)$.

Deci gramaticile sunt echivalente dacă generează acelaşi limbaj. Vom considera o cale simplă de a obţine gramatici echivalente.

Vom da o metodă de construire a gramaticilor echivalente. Dată o gramatică $G = (V_N, V_T, x_0, P)$ construim gramatici $G' = (V_N', V_T, x_0', P')$ unde $V_N' = \{x' \mid x \in V_N\}$ iar P' se obţine înlocuind în fiecare regulă din P simbolii x cu x'.

Se poate arăta că cele două gramatici sunt echivalente.

Observaţia

1) Din cele de mai sus rezultă că nu contează cum notăm terminalii, contează numai forma regulilor.

2) Dacă lucrăm cu două sau mai multe gramatici şi ne interesează limbajele generate de gramatici putem considera că gramaticile au mulţimile simbolilor neterminale disjuncte.

Fie $G = (V_N, V_T, x_0, P)$ o gramatică. Vom construi o gramatică G' echivalentă cu G. Considerăm mulţimea $V'_N = \{x' \mid x \in V_N\}$. Considerăm bijecţia $\beta : V_N \to V'_N$ definită prin $\beta(x) = x'$. Vom extinde bijecţia β la V prin:

$$h(y) = \begin{cases} \beta(y) & \text{dacă } y \in V_N \\ y & \text{dacă } y \in V_T \end{cases}$$

Aplicaţia considerată $h : V \to V'$ se poate prelungi în mod unic, conform lemei 1.1.12., la un homomorfism de semigrupuri prin $h(pi) = h(p)h(i)$ de la V^* la V'^*, $V' = V_T \cup V'_N$. Cuvântul $h(p)$ se obţine din cuvântul p prin înlocuirea simbolurilor x din V_N prin x' din V'_N şi simbolii terminali rămân neafectaţi. Considerăm gramatica $G' = (V'_N, V_T, x'_0, P')$, unde $P' = \{h(u) \to h(v) \mid u \to v \in P\}$.

Vom arăta că gramaticile G şi G' sunt echivalente, adică $L(G) = L(G')$. Pentru aceasta vom demonstra că $u \underset{G}{\Rightarrow} v$ dacă şi numai dacă $h(u) \underset{G'}{\Rightarrow} h(v)$.

Într-adevăr, dacă $u \underset{G}{\Rightarrow} v$, avem că $u = u_1\alpha u_2$, $v = u_1\beta u_2$ şi $\alpha \to \beta \in P$. Din $\alpha \to \beta \in P$ avem că $h(\alpha) \to h(\beta) \in P'$ şi din $h(u) = h(u_1)h(\alpha)h(u_2)$ şi $h(v) = h(u_1)h(\beta)h(u_2)$ rezultă că $h(u) \underset{G'}{\Rightarrow} h(v)$.

Invers, din $h(u) \underset{G'}{\Rightarrow} h(v)$ avem că $h(u) = z_1h(\alpha)z_2$, $h(v) = z_1h(\beta)z_2$, $h(\alpha) \to h(\beta) \in P'$, $z_1, z_2 \in (V'_N \cup V_T)^*$. Există t_1 şi t_2 cu $z_1 = h(t_1)$, $z_2 = h(t_2)$; t_1 şi t_2 se obţin din z_1 si z_2 prin înlocuirea simbolilor x' din V'_N prin simboli x din V_N. Din $h(u) = h(t_1)h(\alpha)h(t_2) =$

$h(t_1\alpha t_2)$, $h(v) = h(t_1)h(\beta)h(t_2) = h(t_1\beta t_2)$ şi $h(\alpha) \to h(\beta) \in P'$, avem că $u = t_1\alpha t_2$, $v = t_1\beta t_2$ şi $\alpha \to \beta \in P$, deci $u \underset{G}{\Rightarrow} v$.

Aplicând la fiecare pas rezultatul de mai sus, avem derivarea

$$x_0 \underset{G}{\Rightarrow} u_1 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} u_n$$

dacă şi numai dacă

$$h(x_0) \underset{G'}{\Rightarrow} h(u_1) \underset{G'}{\Rightarrow} \ldots \underset{G'}{\Rightarrow} h(u_n)$$

Dacă $u_n \in V_T^*$ avem $h(u_n) = u_n$ şi $x_0 \overset{*}{\underset{G}{\Rightarrow}} u_n$ dacă şi numai dacă $x_0' \overset{*}{\underset{G'}{\Rightarrow}} u_n$, deci $L(G) = L(G')$.

Din cele de mai sus, rezultă că putem schimba simbolurile neterminale ale unei gramatici (prin punerea de accente ca mai sus sau în alt mod, schimbând corespunzător şi în regulile gramaticii) fără ca limbajul să se schimbe. Aceasta ne permite ca atunci când lucrăm cu două gramatici $G_1 = (V_{N_1}, V_{T_1}, x_{01}, P_1)$ şi $G_2 = (V_{N_2}, V_{T_2}, x_{02}, P_2)$ şi suntem interesaţi de limbajele generate, putem presupune că $V_{N_1} \cap V_{N_2} = \phi$.

După forma regulilor gramaticilor, gramaticile se pot împărţi în diverse clase. Vom da mai jos cea mai cunoscută ierarhie a gramaticilor, cunoscută sub numele de **ierarhia lui Chomsky**.

Definiţia 1.2.7. 1) Gramaticile de *tip 0* sunt acele gramatici care nu au restricţii asupra regulilor lor, altele decât cele prezentate în definiţia gramaticii.

2) Gramaticile de *tip 1* sunt acele gramatici care au regulile de forma $uxv \to urv$, unde $u, v \in V^*$, $x \in V_N$, $r \in V^+$ sau de forma $x_0 \to \lambda$, caz în care simbolul iniţial x_0 nu apare în partea dreaptă a vreunei reguli.

3) Gramaticile de *tip 2* sunt acele gramatici care au regulile de forma $x \to r$, unde $x \in V_N$ şi $r \in V^*$.

4) Gramaticile de *tip 3* sunt acele gramatici care au regulile de forma $x \to px'$ sau $x \to p$, unde $x, x' \in V_N$ şi $p \in V_T^*$.

Observaţia 1.2.8. În gramaticile de tip 1, regulile de forma $uxv \to urv$ descriu generarea cuvântului r din simbolul neterminal

x, dar numai în contextul $u - v$ (x si r sunt precedati de u şi urmaţi de v), de aceea acestea se mai numesc *gramatici dependente de context* (sensibile la context). Pentru gramaticile de tip 2, regulile de generare sunt de forma $x \to r$, deci generează cuvântul r din neterminalul x indiferent de context, de aceea aceste gramatici se numesc şi gramatici independente de context.

Definiţia 1.2.9. Un limbaj L este de tipul j dacă există o gramatică G de tipul j astfel încât $L(G) = L$, unde $j \in \{0, 1, 2, 3\}$.

Vom nota cu $\mathcal{L}_j$ clasa limbajelor de tipul j, unde $j \in \{0, 1, 2, 3\}$. Cu $\mathcal{L}_j(V)$ vom nota clasa limbajelor de tipul j peste alfabetul V, unde $j \in \{0, 1, 2, 3\}$.

Observaţia 1.2.10. Au loc incluziunile evidente $\mathcal{L}_1 \subseteq \mathcal{L}_0$ şi $\mathcal{L}_3 \subseteq \mathcal{L}_2$, deoarece orice gramatică de tipul 1 este şi de tip 0 şi orice gramatică de tip 3 este şi de tip 2.

Vom demonstra că are loc şi incluziunea $\mathcal{L}_2 \subseteq \mathcal{L}_1$, care nu este evidentă. De asemenea, vom arăta mai târziu că incluziunile $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$ sunt de fapt incluziuni stricte.

Teorema 1.2.11. (Teorema de localizare) *Fie* $G = (V_N, V_T, x_0, P)$, *o gramatică de tipul 2 şi derivarea* $y_1 \dots y_n \underset{G}{\overset{*}{\Rightarrow}} \alpha$. *Atunci* α *se poate scrie sub forma* $\alpha_1 \dots \alpha_n$ *astfel încât avem* $y_j \underset{G}{\overset{*}{\Rightarrow}} \alpha_j$, $1 \le j \le n$.

Demonstraţie. Vom proceda prin inducţie după lungimea derivării $y_1 \dots y_n \underset{G}{\overset{*}{\Rightarrow}} \alpha$. Fie această lungime k.

Pentru $\boldsymbol{k = 0}$ afirmaţia teoremei este evident adevărată luând $\alpha_j = y_j$, $1 \le j \le n$.

Să presupunem afirmaţia teoremei adevărată pentru derivări de lungime cel mult $k - 1$ şi fie $y_1 \dots y_n \underset{G}{\overset{*}{\Rightarrow}} \alpha$ o derivare de lungime k. Vom pune în evidenţă ultimul pas al derivării; avem

$$y_1 \dots y_n \underset{G}{\overset{*}{\Rightarrow}} u \underset{G}{\Rightarrow} \alpha.$$

Derivarea $y_1 \dots y_n \underset{G}{\overset{*}{\Rightarrow}} u$ are lungimea $k-1$ şi pe baza ipotezei inductive,

u are forma $u = u_1 \ldots u_n$ şi $y_j \overset{*}{\underset{G}{\Rightarrow}} u_j$, $1 \leq j \leq n$. În ultimul pas al derivării de lungime k, α se obţine din u prin aplicarea unei reguli de forma $x \rightarrow r$, simbolul x al regulii $x \rightarrow r$ aparţine lui u, deci există un l astfel încât $u_l = u_l' x u_l''$, $1 \leq l \leq n$. Atunci α se descompune în $\alpha_1 \ldots \alpha_n$ cu

$$\alpha_j = \begin{cases} u_j & \text{dacă } j \neq l \\ u_l' r u_l'' & \text{dacă } j = l \end{cases}$$

Astfel avem $y_j \overset{*}{\underset{G}{\Rightarrow}} u_j = \alpha_j$, $1 \leq j \leq n$, $j \neq l$ şi $y_l \overset{*}{\underset{G}{\Rightarrow}} u_l' x u_l'' \underset{G}{\Rightarrow} u_l' r u_l'' = \alpha_l$, deci $y_j \overset{*}{\underset{G}{\Rightarrow}} \alpha_j$, $1 \leq j \leq n$. ■

Observaţia 1.2.12. Din demonstraţia teoremei se observă că lungimea derivărilor $y_j \overset{*}{\underset{G}{\Rightarrow}} \alpha_j$, $1 \leq j \leq n$, este mai mică sau cel mult egală cu lungimea derivării $y_1 \ldots y_n \overset{*}{\underset{G}{\Rightarrow}} \alpha$.

Teorema Pentru orice gramatică $G = (V_N, V_T, x_0, P)$ de tipul 2 există o gramatică G, de tipul 2 care nu are reguli de forma $x \rightarrow \lambda$ şi pentru care avem $L(G_1) = L(G) \setminus \{\lambda\}$.

Definiţia Dacă gramatica G nu are reguli de forma $x \rightarrow \lambda$ atunci $\lambda \notin L(G)$ şi putem lua $G_1 = G$ şi $L(G_1) = L(G) = L(G) \setminus \{\lambda\}$.

Dacă gramatica G conţine reguli de forma $x \rightarrow \lambda$, vom considera următorul şir de mulţimi:

$U_1 = \{x \mid x \rightarrow \lambda \in P\}$

$U_{n+1} = U_n \cup \{x \mid x \rightarrow r \in P, r \in U_n^*\}, n \geq 1$.

Şirul de mulţimi are proprietăţile:

(1) $U_1 \subseteq U_2 \subseteq \ldots U_n \subseteq U_{n+1} \subseteq \ldots \subseteq V_N$.

(2) Deoarece V_N este finită rezultă că există un k astfel că $U_k = U_{k+1}$.

(3) Se poate arăta prin inducţie după j ca $\mathcal{U}_k = \mathcal{U}_{k+j}, j \geq 1$.

(4) Se poate demonstra că, $x \in \mathcal{U}_k$ dacă si numai dacă $x \overset{*}{\underset{G}{\Rightarrow}} \lambda$.

Construcţia gramaticii G_1.

$G_1 = (V_N, V_T, x_0, P_1)$, unde P_1 se obţine din P în modul următor: pentru fiecare regulă din P, $x \rightarrow r$, cu $r \neq \lambda$ se pun în P_1 toate regulile de forma $x \to r_1$ şi r_1 se obţine din r prin ştergerea a $0, 1, \ldots$ simboli din $\mathcal{U}_k$.

Se poate arăta că $L(G_1) = L(G) \setminus \{\lambda\}$.

Teorema 1.2.14. *Fie $G = (V_N, V_T, x_0, P)$ o gramatică de tip 2.* **Dacă $\boldsymbol{\lambda \in L(G)}$**, *atunci există o gramatică G' de tipul 2, echivalentă cu G care are ca unică regulă cu λ în partea dreapta,* **regula $\boldsymbol{x'_0 \to \lambda}$**, *unde x'_0 este simbolul de start al gramaticii G' şi x'_0 nu apare în partea dreaptă a vreunei reguli din G'.*

Demonstraţie. Considerăm gramatica $G' = (V_N \cup \{x'_0\}, V_T, x'_0, P_1 \cup \{x'_0 \to x_0, x'_0 \to \lambda\})$, unde $x'_0 \notin V_N$ şi P_1 se obţine din P ca în teorema precedentă. Să demonstrăm că $L(G) = L(G')$.

Incluziunea $\boldsymbol{L(G) \subseteq L(G')}$. Fie $p \in L(G)$; există atunci derivarea $x_0 \overset{*}{\underset{G}{\Rightarrow}} p$. Dacă $p = \lambda$, atunci în G' avem regula $x'_0 \to \lambda \in P'$ şi derivarea $x'_0 \underset{G'}{\Rightarrow} \lambda$, deci $p \in L(G')$. Dacă $p \neq \lambda$, atunci $p \in L(G) \setminus \{\lambda\} \subseteq L(G_1) \subseteq L(G')$, deoarece regulile din G_1 le avem şi în G' şi avem şi regula $x'_0 \to x_0$, care ne permite ca oricărei derivări de forma $x_0 \overset{*}{\underset{G_1}{\Rightarrow}} \alpha$ să-i corespundă $x'_0 \underset{G'}{\Rightarrow} x_0 \overset{*}{\underset{G'}{\Rightarrow}} \alpha$. Gramatica G_1 este gramatica din teorema precedentă. Deci şi în cazul $p \neq \lambda$ avem $p \in L(G')$.

Să demonstrăm acum incluziunea inversă, adică $\boldsymbol{L(G') \subseteq L(G)}$.

Fie $p \in L(G')$; deci există derivarea $x'_0 \overset{*}{\underset{G'}{\Rightarrow}} p$. Prima regulă care se aplică în această derivare poate fi $x'_0 \to \lambda$ sau $x'_0 \to x_0$. Dacă prima regulă care se aplică este $x'_0 \to \lambda$, atunci $p = \lambda \in L(G)$. Dacă prima regulă care se aplică este $x'_0 \to x_0$, atunci derivarea este de forma $x'_0 \underset{G'}{\Rightarrow} x_0 \overset{*}{\underset{G'}{\Rightarrow}} p$, dar în derivarea $x_0 \overset{*}{\underset{G'}{\Rightarrow}} p$ se aplică numai reguli din G_1 şi deci avem $x_0 \overset{*}{\underset{G_1}{\Rightarrow}} p$, $p \in L(G_1) \subseteq L(G)$. În ambele situaţii avem $p \in L(G)$ ceea ce demonstrează incluziunea $L(G') \subseteq L(G)$. ■

Dacă $\lambda \notin L(G)$, atunci considerăm $G' = (V_N \cup \{x'_0\}, V_T, x'_0, P_1 \cup \{x'_0 \to x_0\})$ şi avem $L(G') = L(G)$ şi G' este o gramatică fără reguli cu λ în partea dreaptă. Se demonstrează imediat că $L(G) = L(G')$.

Consecinţa 1.2.15. *Din teorema precedentă rezultă că $\mathcal{L}_2 \subseteq \mathcal{L}_1$.*

Într-adevăr, orice limbaj $L \in \mathcal{L}_2$ poate fi generat de o gramatică G' de tipul celei din teorema precedentă care este şi de tip 1, deci $L \in \mathcal{L}_1$.

Teorema 1.2.16. *Fiecare din familiile $\mathcal{L}_j$ cu $0 \leq j \leq 3$ conţine toate limbajele finite.*

Demonstraţie. Limbajul vid este generat de o gramatică de forma $G = (\{x, y\}, V_T, x, \{x \to y\})$ care este de tip j cu $j \in \{0, 1, 2, 3\}$. Fie $L = \{p_1, \ldots, p_n\}$ un limbaj finit şi fie $V_T = \{i_1, \ldots, i_m\}$. L este generat de gramatica

$$G = (\{x_0\}, \{i_1, \ldots, i_m\}, x_0, \{x_0 \to p_1, \ldots, x_0 \to p_n\}),$$

care este o gramatică de tip j cu $j \in \{0, 1, 2, 3\}$. ■

Teorema 1.2.17. *Fiecare din familiile $\mathcal{L}_j$cu $j \in \{0, 1, 2, 3\}$ este închisă la operaţia de reuniune.*

Demonstraţie. Va trebui să demonstrăm că dacă $L_1, L_2 \in \mathcal{L}_j$, $0 \leq j \leq 3$, atunci şi $L_1 \cup L_2 \in \mathcal{L}_j$. Fie j fixat cu $j \in \{0, 1, 2, 3\}$, atunci există gramaticile $G_k = (V_{N_k}, V_{T_k}, x_{0k}, P_k)$, $k \in \{1, 2\}$, de tipul j astfel încât
$L_1 = L(G_1)$ şi $L_2 = L(G_2)$. Putem presupune că $V_{N_1} \cap V_{N_2} = \phi$. Considerăm gramatica

$$G = (V_{N_1} \cup V_{N_2} \cup \{x_0\}, V_{T_1} \cup V_{T_2}, x_0, P_1 \cup P_2 \cup \{x_0 \to x_{01}, x_0 \to x_{02}\}),$$

unde $x_0 \notin V_{N_1} \cup V_{N_2}$. Gramatica G este de acelaşi tip cu gramaticile G_1 şi G_2 deoarece regulile $x_0 \to x_{01}$ şi $x_0 \to x_{02}$ nu schimbă tipul gramaticii.
Vom demonstra că $\boldsymbol{L(G) = L_1 \cup L_2}$.

Fie $p \in L(G)$; rezultă că există derivarea $x_0 \overset{*}{\underset{G}{\Rightarrow}} p$. Prima regulă aplicată în această derivare este $x_0 \to x_{01}$ sau $x_0 \to x_{02}$, deci derivarea $x_0 \overset{*}{\underset{G}{\Rightarrow}} p$ este de forma $x_0 \underset{G}{\Rightarrow} x_{01} \overset{*}{\underset{G}{\Rightarrow}} p$ sau de forma $x_0 \underset{G}{\Rightarrow} x_{02} \overset{*}{\underset{G}{\Rightarrow}} p$. În derivarea $x_{01} \overset{*}{\underset{G}{\Rightarrow}} p$ se aplică numai reguli din G_1 deoarece $x_{01} \in V_{N_1}$ şi $V_{N_1} \cap V_{N_2} = \phi$, deci putem scrie $x_{01} \overset{*}{\underset{G_1}{\Rightarrow}} p$ şi $p \in L(G_1) = L_1$. Analog, în cazul $x_{02} \underset{G}{\overset{*}{\Rightarrow}} p$ derivarea revine la $x_{02} \underset{G_2}{\overset{*}{\Rightarrow}} p$ şi $p \in L(G_2) = L_2$. Deci în ambele cazuri $p \in L_1 \cup L_2$.

Invers, fie $p \in L_1 \cup L_2$. Din $p \in L_1 \cup L_2$ rezultă că $p \in L_1$ sau $p \in L_2$. Dacă $p \in L_1 = L(G_1)$ rezultă că există derivarea $x_{01} \underset{G_1}{\overset{*}{\Rightarrow}} p$ şi atunci avem $x_0 \underset{G}{\Rightarrow} x_{01} \underset{G}{\overset{*}{\Rightarrow}} p$ şi $p \in L(G)$. În mod analog, pentru $p \in L_2 = L(G_2)$, avem că $p \in L(G)$. ■

Exemple rezolvate

1) Se dă gramatica $G = (\{x_0\}, \{a, b\}, x_0, \{x_0 \to ab, x_0 \to ax_0b\})$. Să se arate că $L(G) = \{a^nb^n \mid n \geq 1\}$.

Rezolvare: Conform definiţiei limbajului generat avem că $L(G) = \{p \mid pGV_T^* \exists x_0 \underset{G}{\overset{*}{\Rightarrow}} p\}$, deci sunt toate cuvintele din V_T^* care se obţin din x_0 prin derivări în G. Pentru a demonstra egalitatea din enunţ, trebuie să demonstrăm dubla incluziune.

$$\{a^nb^n \mid n \geq 1\} \subseteq L(G).$$

Pentru $n = 1$ avem: $x_0 \underset{G}{\overset{*}{\Rightarrow}} ab$, deci $ab \in L(G)$.

Pentru $n = 2$ avem: $x_0 \underset{G}{\overset{*}{\Rightarrow}} ax_0b \Rightarrow aabb$, deci $a^2b^2 \in L(G)$.

Pentru $n \geq 3$ avem: $x_0 \underset{G}{\overset{*}{\Rightarrow}} ax_0b \underset{G}{\overset{*}{\Rightarrow}} \dots \underset{G}{\overset{*}{\Rightarrow}} a^{n-1}x_0b^{n-1} \underset{G}{\overset{*}{\Rightarrow}} a^nb^n$, deci $a^nb^n \in L(G)$.

Să demonstrăm că $L(G) \subseteq \{a^nb^n \mid n \geq 1\}$.

Fie $p \in L(G)$. Din definiţia lui $L(G)$ rezultă că există derivarea $x_0 \underset{G}{\overset{*}{\Rightarrow}} p$ şi $p \in V_T^*$.

Dacă la primul pas al derivării se aplică regula $x_0 \to ab$, atunci avem $x_0 \underset{G}{\overset{*}{\Rightarrow}} ab$ şi $p = ab \in \{a^nb^n \mid n \geq 1\}$. Altă posibilitate este să aplicăm la început regula $x_0 \to ax_0b$ de ℓ ori şi apoi regula $x_0 \to ab$ ca să eliminăm neterminalul x_0, atunci avem:

$$x \underset{G}{\overset{*}{\Rightarrow}} ax_0b \underset{G}{\overset{*}{\Rightarrow}} \dots \underset{G}{\overset{*}{\Rightarrow}} a^\ell x_0 b^\ell \text{ şi } a^{\ell+1}b^{\ell+1}G\{a^nb^n \mid n \geq 1\}.$$

Acestea fiind regulile posibilităţilor avem $L(G) \in \{a^nb^n \mid n \geq 1\}$.

2. Se dă gramatica $G = (\{x_0, x\}), \{a, b\}, x_0, P)$ cu P dat prin

1) $x_0 \to ax_0$.

2) $x_0 \to ax$.

3) $x \to bx$.

4) $x \to b$.

Să se arate că $L(G) = \{a^n b^m / n, m \geq 1\}$.

Rezolvare: Incluziunea $\supseteq$.

Pentru $n = 1, m = 1,$, avem $x_0 \underset{G}{\Rightarrow} ax \underset{G}{\Rightarrow} ab$. Deci $ab \in L(G)$. Pentru n si m arbitrari se aplică regula 1 de $n-1$ ori, după care se aplică regula 2 şi apoi se aplică regula 3 de $m-1$ ori şi la sfârşit regula 4.

$$X_0 \underset{G}{\Rightarrow} ax_0 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} a^{n-1}x_0 \underset{G}{\Rightarrow} a^n x \underset{G}{\Rightarrow} a^n bx \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} a^n b^{m-1} x \underset{G}{\Rightarrow} a^n b^m.$$

Deci $a^n b^m \in L(G)$.

Să demonstrăm acum că $L(G) \subseteq \{a^n b^m / n, m \geq 1\}$ Fie $p \in L(G)$; atunci avem $x_0 \overset{*}{\underset{G}{\Rightarrow}} p$ şi $p \in \{a, b\}^*$.

Să analizăm cum poate fi derivarea. La început trebuie să se aplice regulile 1) sau 2) după care trebuie să se aplice 3) şi în final regula 4). Deci putem considera că se aplică regula 1) de l ori, cu $l \geq 0$ (dacă $l = 0$ regula 1) nu se aplică) după care obligatoriu se aplică regula 2) ca să eliminăm x_0. Aplicarea regulei 2) ne generează neterminalul x, deci mai departe se pot aplica regulile 3) de k ori cu $k \geq 0$ (dacă $k = 0$ regula 3) nu se aplică) şi apoi regula 4) pentru a elimina neterminalul x. Derivarea are forma:

$$x_0 \underset{G}{\Rightarrow} ax_0 \Rightarrow \ldots \underset{G}{\Rightarrow} a^l x_0 \underset{G}{\Rightarrow} a^{l+1} x \Rightarrow a^{l+1} bx \Rightarrow \ldots \underset{G}{\Rightarrow} a^{l+1} b^k x \Rightarrow a^{l+1} b^{k+1}.$$

şi avem $a^{l+1} b^{k+1} \in \{a^n b^m | n, m \geq 1\}$.

3. Fie gramatica $G = (\{x_0, x\}, \{a, b, c\}, x_0, P)$ cu P dat prin:

1. $x_0 \to ax_0xc$

2. $x_0 \to abc$

3. $cx \to xc$

4. $bx \to bb$

Să se arate că $L(G) = \{a^nb^nc^n | n \geq 1\}$.

Rezolvare. Să demonstrăm întâi că $\{a^nb^nc^n | n \geq 1\} \subseteq L(G)$.

Pentru $n = 1$. Din regula 2) avem $x_0 \Rightarrow abc$, deci $abc \in L(G)$.

Pentru $n = 2$. Aplicăm întâi regula 1) şi apoi regula 2) şi apoi regulile 3) şi 4). Avem:

$$x_0 \Rightarrow ax_0xc \Rightarrow a^2bcxc \Rightarrow a^2bxc^2 \Rightarrow a^2b^2c^2.$$

Pentru $n > 2$. Vom aplica regula 1) de $(n-1)$ ori şi apoi regula 2 şi obţinem:

$$x_0 \Rightarrow ax_0xc \Rightarrow^* a^{n-1}x_0(xc)^{n-1} \Rightarrow a^nbc(xc)^{n-1}.$$

Mai departe aplicăm regula 3) încât să comutăm toţi x cu c şi apoi aplicăm de $(n-1)$ în regula 4). Avem:

$$x_0 \overset{*}{\underset{G}{\Rightarrow}} a^nbc(xc)^{n-1} \overset{*}{\underset{G}{\Rightarrow}} a^nbx^{n-1}c^n \overset{*}{\underset{G}{\Rightarrow}} a^nb^nc^n.$$

Acum să demonstrăm că $L(G) \subseteq \{a^nb^nc^n | n \geq 1\}$.

Fie $p \in L(G)$. Atunci avem $x_0 \overset{*}{\underset{G}{\Rightarrow}} p$ şi $p \in \{a, b, c\}^*$. Derivarea $x_0 \overset{*}{\underset{G}{\Rightarrow}} p$ poate începe cu regula 2) sau cu regula 1). Dacă începe cu regula 2) avem $x_0 \underset{G}{\Rightarrow} abc$ şi $abc \in \{a^nb^nc^n | n \geq 1\}$. Dacă se aplică la început regula 1) de l ori după care trebuie să aplicăm regula 2) să eliminăm x_0.

Avem: $x_0 \overset{*}{\Rightarrow} a^lx_0(xc)^l \underset{G}{\Rightarrow} a^{l+1}bc(xc)^l$.

Mai departe trebuie să eliminăm toţi neterminalii x. Acest lucru se face prin comutarea neterminalilor x cu c şi apoi se aplică regula 4). Comutarea x cu c se poate face câte un x şi apoi aplicăm regula 4), efectul este acelaşi. Acestea sunt singurele posibilităţi de derivare. Deci obţinem $x_0 \overset{*}{\underset{G}{\Rightarrow}} a^{l+1}bc(xc)^l \overset{*}{\Rightarrow} a^{l+1}b^{l+1}c^{l+1}$, $a^{l+1}b^{l+1}c^{l+1} \in \{a^nb^nc^n | n \geq 1\}$ şi prin urmare.

4. Fie gramatica $G = (\{x_0, x_1, x_2\}, \{a, b, c\}, x_0P)$ cu P dat prin:

1. $x_0 \to abc$
2. $x_0 \to ax_1bc$
3. $x_1b \to bx_1$
4. $x_1c \to x_2bcc$
5. $bx_2 \to x_2b$
6. $ax_2 \to aax_1$
7. $ax_2 \to aa$

Să se arate că $L(G) = \{a^nb^nc^n | n \geq 1\}$.

Să demonstrăm că $\{a^nb^nc^n | n \geq 1\} \subseteq L(G)$.

Pentru **n=1**. Aplicând regula 1) avem $x_0 \Rightarrow abc$, deci $abc \in L(G)$.

Pentru **n=2**. Aplicăm întâi regula 2) şi apoi 3), 4), 5) şi 7) avem $x_0 \underset{G}{\Rightarrow} ax_1bc \underset{G}{\Rightarrow} abx_1c \underset{G}{\Rightarrow} abx_2bcc \underset{G}{\Rightarrow} ax_2bbcc \underset{G}{\Rightarrow} a^2b^2c^2$, deci $a^2b^2 \in L(G)$.

Vom demonstra că dacă avem un cuvânt

$$(*) \quad a^jx_1b^jc^j \begin{array}{l} \overset{I\ \ *}{\nearrow} a^{j+1}b^{j+1}c^{j+1} \\ \underset{II\ *}{\searrow} a^{j+1}x_1b^{j+1}c^{j+1} \end{array}$$

Să analizăm posibilităţile de derivare din $ajx_1b^jc^j$. Se poate analiza numai regula 3) de j ori.

$$a^jx_1b^jc^j \overset{*}{\Rightarrow} a^jb^jx_1c^j.$$

Mai departe se poate aplica regula 4) şi apoi regula 5) de j ori şi acestea sunt singurile reguli care se pot aplica. Avem:

$$a^jx_1b^jc^jCa^jb^jx_1c^j \Rightarrow a^jb^jx_2bc^{j+1} \overset{*}{\Rightarrow} a^jx_2b^{j+1}c^{j+1}.$$

Acum avem două posibilităţi. Putem aplica sau regula 6) sau regula 7).

$$a^jx_2b^{j+1}c^{j+1} \begin{array}{l} \overset{I\ \ *}{\nearrow} a^{j+1}b^{j+1}c^{j+1} \\ \underset{II\ *}{\searrow} a^{j+1}x_1b^{j+1}c^{j+1} \end{array}$$

Acum să arătăm că pentru $n \geq 3, a^n b^n c^n \in L(G)$. $x_0 \Rightarrow ax_1bc$, dar ax_1bc este de forma $a^j x_1 b^j c^j$ cu $j = 1$. Vom aplica relaţia (*) varianta II de $(n-2)$ ori, la fiecare aplicare creşte exponentul lui a, b şi c cu o unitate şi aplicăm I deci obţinem:

$$x_0 \underset{G}{\Rightarrow} ax_1bc \overset{II}{\Rightarrow}{}^* a^{n-1}x_1b^{n-1}c^{n-1} \overset{I}{\Rightarrow}{}^* a^n b^n c^n.$$

deci $a^n b^n c^n \in L(G)$.

Să demonstrăm că $L(G) \subseteq \{a^n b^n c^n / n \geq 1\}$.

Fie $p \in L(G)$. Din definiţia lui $L(G)$ rezultă că avem $x_0 \overset{*}{\underset{G}{\Rightarrow}} p$ şi $p \in \{a, b, c\}^*$.

In derivarea $x_0 \overset{*}{\Rightarrow} p$ se poate aplica la primul pas sau regula 1) sau regula 2). Dacă se aplică regula 1) avem $x_0 \Rightarrow abc$ şi $abc \in \{a^n b^n c^n | n \geq 1\}$.

Dacă se aplică regula 1), avem $x_0 \Rightarrow ax_1bc$ şi ax_1bc este de forma $a^j x_1 b^j c^j$ cu $j = 1$. Din relaţia (*) avem că singurile cuvinte care se obţin din x_0 şi conţin numai terminali sunt de forma $a^j b^j c^j$ şi $a^j b^j c^j \in \{a^n b^n c^n | n \geq 1\}$.

Tema de Control (1)

1) Să se arate că gramatica $G = (\{x_0, x, y\}, \{a, b, c\}, x_0, P)$.
Cu P :

1) $x_0 \rightarrow xy$

2) $x \rightarrow axb$

3) $x \rightarrow ab$

4) $y \rightarrow cy$

5) $y \rightarrow c$

generează limbajul $L(G) = \{a^n b^n c^m | n, m \geq 1\}$.

2) Să se construiască o gramatică care generează limbajele $\{a^n b^n c^m | n, m \geq 1\}$ şi $\{a^n b^m c^n | n, m \geq 1\}$.

3) Să se arate că gramatica $G = (\{x_0 x_3, x_2\}, \{a, b, c\}, x_0)P$, cu P :

1) $x_0 \to ax_0x_1x_2$

2) $x_0 \to ax_1x_2$

3) $x_1x_2 \to x_1x_2$

4) $ax_1 \to ab$

5) $bx_1 \to bb$

6) $bx_2 \to bc$

7) $cx_2 \to cc$

generează limbajul $\{a^nb^nc^n | n \geq 1\}$.

4) Să se arate că gramatica $G = (\{A, B, C, E, F\}, \{M, b, c\}, A, P)$ cu p :

1) $A \to aAB$

2) $A \to abC$

3) $CB \to EB$

4) $EB \to EF$

5) $EF \to BF$

6) $BF \to BC$

7) $C \to c$

8) $bB \to bbc$

generează limbajul $L(G) = \{a^nb^nc^n | n \geq 0\}$.

5) Să se arate că gramatica $G = (\{S, A, B, C\}, \{a\}, S, P)$ cu P :

1) $S \to BAB$

2) $BA \to BC$

3) $CA \to AAC$

4) $CB \to AAB$

5) $A \to a$

6) $B \to x$

generează limbajul $L(G) = \{a^{2^n} | n \geq 0\}$.

6) Să se arate că gramatica $G = (\{S, X, A, B\}, \{*, a\}, S, P)$ cu P :

1) $S \to **$

2) $S \to *B*$

3) $S \to *AABB*$

4) $*A \to *XA$

5) $XA \to AX$

6) $XB \to AAB$

7) $X* \to B*$

8) $A \to a$

9) $B \to a$

generează limbajul $L(G) = \{*a^{n^2} * | n \geq 0\}$.

Tema 2

Automate finite şi gramatici de tip trei

1 Automate finite

În acest capitol vom introduce noţiunea de automat finit, vom studia proprietăţile lui şi vom arăta că familia limbajelor acceptate de automatele finite este egală cu familia $\mathcal{L}_3$.

Definiţia 2.1.1. Un *automat determinist* este un 5-uplu $A = (S, \Sigma, \delta, s_0, F)$ unde:

- S este o mulţime **nevidă**, numită *mulţimea stărilor*;
- Σ este o mulţime **nevidă** şi **finită** numită *alfabet de intrare*;
- δ este o funcţie $\delta : S \times \Sigma \to S$ numită *funcţie de tranziţie directă*;
- $s_0 \in S$ se numeşte *stare iniţială*;
- $F \subseteq S$ este *mulţimea stărilor finale*.

Dacă S este finită, atunci automatul este *automat finit* şi în practică, acest tip de automate interesează.

Un model pentru un automat finit este format dintr-un dispozitiv de control care se poate afla la un moment dat într-o stare din S şi care este prevăzut cu un cap de citire cu acces la o bandă de intrare pe care este scris un cuvânt din Σ^*. Automatul aflându-se în starea s şi citind de pe bandă un simbol c va trece în starea $\delta(s, c)$ şi va deplasa capul de citire cu o locaţie la dreapta.

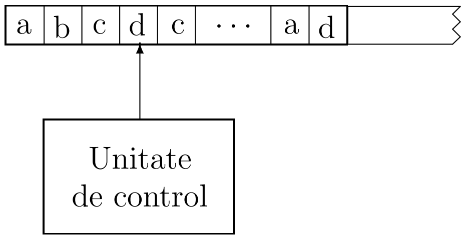


Fig. 2.1. Model pentru automat determinist

Specificarea unui automat finit revine la precizarea mulţimilor S, Σ, F, a stării iniţiale s_0 şi la definirea funcţiei de tranziţie directă δ. Definirea funcţiei de tranziţie se poate face fie tabelar, fie cu ajutorul unui graf orientat etichetat, care descrie funcţionarea automatului sau ca expresie analitică. Descrierea funcţiei de tranziţie tabelar se face utilizând o matrice de dimensiune $|S| \times |\Sigma|$, câte o linie pentru fiecare $s \in S$.

δ	i
	$\vdots$
s	$\cdots \delta(s,i) \cdots$
	$\vdots$

Fig. 2.2. Tabelul funcţiilor de tranziţie

Al doilea procedeu constă în ataşarea unui graf orientat etichetat numit *graful tranziţiilor* (sau uneori *diagrama de tranziţie a automatului*).

Definiţia 2.1.2. Graful tranziţiilor automatului finit $A = (S, \Sigma, \delta, s_0, F)$ este un cuplu (G, γ) unde $G = (S, \mathcal{U})$ este un graf orientat, având ca mulţime de noduri, mulţimea stărilor automatului S şi ca mulţime de arce mulţimea $\mathcal{U} = \{(s_j, s_k),\ \exists i \in \Sigma, \delta(s_j, i) = s_k\}$, iar $\gamma : \mathcal{U} \to \mathcal{P}(\Sigma)$ este o funcţie de etichetare a arcelor definită în modul următor: pentru fiecare arc $u \in \mathcal{U}$ $\gamma(u) = \{i \mid i \in \Sigma, u = (s_j, s_k), \delta(s_j, i) = s_k\}$.

Convenim ca pentru stările finale să desenăm două cercuri, iar starea iniţială să o marcăm printr-o săgeată.

Exemplul 2.1.3. Considerăm automatul

$$A = (\{s_0, s_1, s_2\}, \{0, 1\}, \delta, s_0, \{s_1, s_2\})$$

şi funcţia δ definită ca în Figura 2.3.

δ	0	1
s_0	s_1	s_2
s_1	s_1	s_0
s_2	s_2	s_1

Fig. 2.3. Funcţia de tranziţie definită tabelar

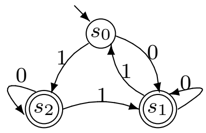


Fig. 2.4. Graful de tranziţii

Funcţia de tranziţie $\delta : S \times \Sigma \to S$ o putem extinde la $S \times \Sigma^*$ astfel încât pentru o stare s şi un cuvânt $p \in \Sigma^*$ starea $\delta(s, p)$ să fie starea în care ajunge automatul după parcurgerea celor $l(p)$ simboli ai cuvântului p.

Definiţia 2.1.4. Fie $A = (S, \Sigma, \delta, s_0, F)$ un automat. Definim extensia lui δ, $\widehat{\delta} : S \times \Sigma^* \to S$ prin:

1) $\widehat{\delta}(s, \lambda) = s, \ \forall s \in S$;

2) $\widehat{\delta}(s, ua) = \delta(\widehat{\delta}(s, u), a); \forall s \in S, \ \forall u \in \Sigma^*, \ \forall a \in \Sigma$.

Pentru simplitatea notaţiei convenim să notăm în continuare δ în loc de $\widehat{\delta}$. Se justifică acest lucru deoarece $\widehat{\delta}(s, u) = \delta(s, u) \ \forall s \in S, \forall a \in \Sigma$.

Teorema 2.1.5. Fie $A = (S, \Sigma, \delta, s_0, F)$ şi $p, q \in \Sigma^*$. Atunci are loc egalitatea:

(1) $$\delta(s, pq) = \delta(\delta(s, p), q), \ \forall s \in S.$$

Demonstraţie. Vom demonstra egalitatea (1) prin inducţie după lungimea cuvântului q.

Pentru $l(q) = 0$, deci $q = \lambda$, avem: $\delta(s, p\lambda) = \delta(\delta(s, p), \lambda)$, care revine la $\delta(s, p) = \delta(s, p)$ şi deci (1) este adevărată.

Presupunem egalitatea (1) adevărată pentru cuvinte de lungime cel mult n şi o demonstrăm pentru cuvinte de lungime $n + 1$. Fie $q' = qi$ cu $i \in \Sigma$ şi $q \in \Sigma^*$ cu $l(q) = n$, deci $l(q') = n + 1$.

Avem $\delta(s,pq') = \delta(s,pqi) = \delta(\delta(s,pq),i) = \delta(\delta(\delta(s,p),q),i) = \delta(\delta(s,p),qi) = \delta(\delta(s,p),q')$. S-a folosit ipoteza inductivă şi egalitatea 2) din Definiţia 2.1.4. ■

Definiţia 2.1.6. Limbajul acceptat de un automat $A = (S, \Sigma, \delta, s_0, F)$ este definit prin:

$$L(A) = \{w \mid w \in \Sigma^*, \delta(s_0, w) \in F\}.$$

Definiţia 2.1.7. Un limbaj L este *regulat* dacă există **un automat finit A** pentru care să avem $L = L(A)$.

Mulţimea limbajelor regulate o notăm cu $\mathcal{L}_R$.

Teorema 2.1.9. Orice limbaj $L \in \mathcal{P}(\Sigma^*)$ este acceptat de un automat convenabil ales.

Demonstraţie. Fie automatul $A = (\Sigma^*, \Sigma, \delta, \lambda, L)$, unde $\delta(p,i) = pi$, $\forall p \in \Sigma^*$ şi $i \in \Sigma$. Automatul A este infinit deoarece mulţimea stărilor Σ^* este infinită.
Funcţia δ o putem extinde $\delta : \Sigma^* \times \Sigma^* \to \Sigma^*$ ca în Definiţia 2.1.4. şi avem $\delta(p,q) = p \cdot q$.
Să arătăm că $L = L(A)$.

Fie $q \in L$ atunci avem $q = \delta(\lambda, q)$, de unde rezultă că $q \in L(A)$. Invers, fie $q \in L(A)$; rezultă că $\delta(\lambda, q) \in L$, dar $\delta(\lambda, q) = q \in L$. ■

2 Caracterizarea limbajelor regulate cu ajutorul relaţiilor de echivalenţă

În acest paragraf vom da o altă caracterizare a limbajelor regulate decât aceea că sunt acceptate de automate finite deterministe.

Fie L un limbaj, $L \subseteq \Sigma^*$. Vom considera funcţia caracteristică: $\chi_L : \Sigma^* \to \{0,1\}$ definită prin $\chi_L(p) = \begin{cases} 1 & \text{dacă } p \in L \\ 0 & \text{dacă } p \notin L. \end{cases}$

Definiţia 2.2.1. Fiecărui limbaj L îi ataşăm relaţia ν_L definită prin $\nu_L = \{(p, q) \mid p, q \in \Sigma^*, \chi_L(pr) = \chi_L(qr),\ \forall r \in \Sigma^*\}$.

Relaţia ν_L se poate defini şi în modul următor: cuvintele p şi q sunt în relaţia ν_L dacă pr şi qr aparţin sau nu limbajului L pentru orice $r \in \Sigma^*$.

Observaţia 2.2.2. *Relaţia ν_L este o relaţie de echivalenţă.*

Observaţia 2.2.3. *Echivalenţa ν_L este compatibilă cu concatenarea la dreapta, adică $(p, q) \in \nu_L$ implică $(pu, qu) \in \nu_L$ oricare ar fi $u \in \Sigma^*$.*

Într-adevăr, oricare ar fi $r \in \Sigma^*$ avem

$$\chi_L((pu)r) = \chi_L(p(ur)) = \chi_L(q(ur)) = \chi_L((qu)r).$$

Definiţia 2.2.4. O relaţie de echivalenţă α peste o mulţime M este de rang finit, dacă $|M/\alpha|$ este finit.

Definiţia 2.2.5. Fie $A = (S, \Sigma, \delta, s_0, F)$. Considerăm relaţia ν_{A,s_0} ataşată automatului A definită prin

$$\nu_{A,s_0} = \{(p, q) \mid p, q \in \Sigma^*, \delta(s_0, p) = \delta(s_0, q)\}.$$

Observaţia 2.2.6. *Relaţia ν_{A,s_0} este o relaţie de echivalenţă. Relaţia ν_{A,s_0} este compatibilă cu concatenarea la dreapta.*

Teorema 2.2.7. *Dacă limbajul L este acceptat de automatul $A = (S, \Sigma, \delta, s_0, F)$ atunci $\nu_{A,s_0} \subseteq \nu_L$.*

Demonstraţie. Trebuie să demonstrăm că $(p, q) \in \nu_{A,s_0}$ implică $(p, q) \in \nu_L$. Din $(p, q) \in \nu_{A,s_0}$ avem că $\delta(s_0, p) = \delta(s_0, q)$. Fie $r \in \Sigma^*$ un cuvânt arbitrar, atunci avem $pr \in L$ dacă şi numai dacă $\delta(s_0, pr) \in F$ dacă şi numai dacă $\delta(\delta(s_0, p), r) = \delta(\delta(s_0, q), r) = \delta(s_0, qr) \in F$ dacă şi numai dacă $qr \in L$. Deci $\chi_L(pr) = \chi_L(qr)$, ceea ce implică $(p, q) \in \nu_L$. ■

Teorema 2.2.8. *Fie $A = (S, \Sigma, \delta, s_0, F)$; există o injecţie*

$\varphi : \Sigma^*/\nu_{A,s_0} \to S$.

Demonstraţie. Definim funcţia $\varphi : \Sigma^*/\nu_{A,s_0} \to S$ prin $\varphi([p]) = \delta(s_0, p)$, oricare ar fi $[p] \in \Sigma^*/\nu_{A,s_0}$. Funcţia φ este bine definită; nu depinde de reprezentatul clasei. Fie $q \in [p]_{\nu_{A,s_0}}$, atunci $[q]_{\nu_{a,s_0}} = [p]_{\nu_{A,s_0}}$. Avem $\varphi([q]_{\nu_{A,s_0}}) = \delta(s_0, q) = \delta(s_0, p) = \varphi([p]_{\nu_{A,s_0}})$. Se verifică imediat că φ este injectivă. Într-adevăr, $\varphi([p]_{\nu_{A,s_0}}) = \varphi([q]_{\nu_{A,s_0}})$ implică $\delta(s_0, p) = \delta(s_0, q)$, care înseamnă $[p]_{\nu_{A,s0}} = [q]_{\nu_{A,s_0}}$. ■

Teorema 2.2.9. *Limbajul L este acceptat de un automat finit dacă şi numai dacă ν_L are rang finit.*

Demonstraţie. Să demonstrăm că, L acceptat de un automat A finit implică ν_L are rang finit. Din teorema 2.2.8. rezultă că există injecţia
$\varphi : \Sigma^*/\nu_{A,s_0} \to S$, de unde rezultă că $|\Sigma^*/\nu_{A,s_0}| \leq |S|$. Din teorema 2.2.7. rezultă că $\nu_{A,s_0} \subseteq \nu_L$ care implică că $|\Sigma^*/\nu_L| \leq |\Sigma^*/\nu_{A,s_0}| \leq |S|$. Cum $|S|$ este finit, rezultă că $|\Sigma^*/\nu_L|$ este finit, deci ν_L are rang finit.

Invers, să demonstrăm că, ν_L are rang finit implică L este limbaj regulat.
Considerăm automatul $A_L = (\Sigma^*/\nu_L, \Sigma, \delta, [\lambda]_{\nu_L}, \{[w]_{\nu_L} \mid w \in L\})$. Definim funcţia de tranziţie $\delta : \Sigma^*/\nu_L \times \Sigma \to \Sigma^*/\nu_L$ prin

$$\delta([p]_{\nu_L}, i) = [pi]_{\nu_L},\ \forall p \in \Sigma^* \text{ şi } \forall i \in \Sigma.$$

Să observăm că funcţia δ este bine definită, adică nu depinde de reprezentantul clasei. Fie $q \in [p]_{\nu_L}$. Din $q \in [p]_{\nu_L}$ rezultă $(p, q) \in \nu_L$ care implică $(pi, qi) \in \nu_L$, $\forall i \in \Sigma$ şi astfel $[pi]_{\nu_L} = [qi]_{\nu_L}$. Prelungim funcţia de tranziţie δ la Σ^* prin $\delta([p]_{\nu_L}, q) = [pq]_{\nu_L}$. Automatul A_L este automat finit, deoarece Σ este finit şi $|\Sigma^*/\nu_L|$ este finit.

Să demonstrăm că $L = L(A_L)$. Fie $p \in L$, rezultă că $[p]_{\nu_L} \in \{[w]_{\nu_L} \mid w \in L\}$. Dar $\delta([\lambda]_{\nu_L}, p) = [p]_{\nu_L} \in \{[w]_{\nu_L} \mid w \in L\}$, de unde $p \in L(A_L)$.

Invers, fie $p \in L(A_L)$, rezultă că $\delta([\lambda]_{\nu_L}, p) \in \{[w]_{\nu_L} \mid w \in L\}$, de unde $[p]_{\nu_L} \in \{[w]_{\nu_L} \mid w \in L\}$. Deci există $w \in L$ cu $(p, w) \in \nu_L$, care implică $\chi_L(pr) = \chi_L(wr)$, oricare ar fi $r \in \Sigma^*$. Luăm $r = \lambda$, avem $\chi_L(p) = \chi_L(w) = 1$, de unde $p \in L$. ■

Consecinţa 2.2.10. *Dacă L este un limbaj regulat, atunci automatul A_L este, conform teoremei 2.2.9., automatul cu cele mai puţine stări care acceptă limbajul L.*

Demonstraţie. Într-adevăr, dacă $A = (S, \Sigma, \delta, s_0, F)$ un automat finit care acceptă limbajul L, atunci din demonstraţia teoremei precedente avem că $|\Sigma^*/\nu_L| \leq |S|$. Dar automatul A_L are ca mulţime de stări Σ^*/ν_L si am arătat că acceptă limbajul L.

Automatul A_L îl vom numi, din acest motiv, **automatul minimal** al limbajului L.

Teorema precedentă se poate utiliza pentru a arăta că unele limbaje nu sunt limbaje regulate. Pentru aceasta se arată că relaţia de echivalenţă ataşată limbajului nu are rang finit.

Fie limbajul $L = \{a^n b^n \mid n \geq 1\}$, peste alfabetul $\{a, b\}$. Acest limbaj este un limbaj independent de context, fiind generat de o gramatică care are regulile $x_0 \to ax_0b$ şi $x_0 \to ab$, care satisfac condiţiile pentru acest tip de gramatică. Să arătăm că acest limbaj nu este regulat. Presupunem că limbajul L este regulat. Considerăm echivalenţa ν_L ataşată limbajului L, care are rang finit deoarece L este regulat. Considerăm şirul de cuvinte $a, a^2, \ldots, a^n, \ldots$. Deoarece ν_L are rang finit, există $i < j$ astfel încât $(a^i, a^j) \in \nu_L$. Relaţia ν_L este compatibilă cu concatenarea la dreapta, deci $(a^i b^j, a^j b^j) \in \nu_L$. Aceasta este imposibil, deoarece $a^j b^j \in L$ şi $a^i b^j \notin L$, deci $\chi_L(a^i b^j) \neq \chi_L(a^j b^j)$.

Teorema precedentă ne permite să găsim pentru un limbaj regulat un automat finit A_L care să accepte acest limbaj.

Vom defini o altă relaţie de echivalenţă care să caracterizeze mulţimea limbajelor regulate.

3 Construcţia automatului minimal

Fie $A = (S, \Sigma, \delta, s_0, F)$ un automat finit determinist.

Definiţia 2.3.1. Definim funcţia caracteristică a mulţimii F,

$\Psi : S \to \{0,1\}$, prin $\Psi(s) = \begin{cases} 1 & \text{dacă } s \in F \\ 0 & \text{dacă } s \notin F. \end{cases}$

Definiţia 2.3.2. Stările s_1 şi s_2 se numesc k- *inseparabile în raport cu* F, notat $s_1 \overset{k}{\equiv} s_2$, dacă pentru orice $p \in \Sigma^*$ cu $l(p) \leq k$, avem $\Psi(\delta(s_1, p)) = \Psi(\delta(s_2, p))$. Se verifică imediat că relaţia $\overset{k}{\equiv} \subseteq S \times S$ este o relaţie de echivalenţă, adică este reflexivă, simetrică şi tranzitivă.

Definiţia 2.3.3. Stările s_1 şi s_2 sunt inseparabile în raport cu F dacă şi numai dacă ele sunt k-inseparabile în raport cu F pentru orice $k \in N$.
Vom nota această relaţie cu ρ. Se poate arăta că ρ este o relaţie de echivalenţă. Din definiţia lui ρ rezultă că $\rho = \bigcap_{k=0}^{\infty} \{\overset{k}{\equiv} | k \in N\}$.

Definiţia 2.3.4. Fie $A = (S, \Sigma, \delta, s_0, F)$ un automat finit determinist. O stare $s \in S$ este *accesibilă* dacă există $p \in \Sigma^*$ astfel încât $s = \delta(s_0, p)$ şi *inaccesibilă* în caz contrar.

Definiţia 2.3.5. Un automat $A = (S, \Sigma, \delta, s_0, F)$ se numeşte *redus* dacă nu are stări inaccesibile.

Fiind dat un automat finit determinist $A = (S, \Sigma, \delta, s_0, F)$, fie S' stările sale accesibile şi $F' = S' \cap F$. Considerăm automatul $A' = (S', \Sigma, \delta, s_0, F')$. Automatul A' este redus şi în plus $L(A) = L(A')$. Într-adevăr, $p \in L(A) \Leftrightarrow \delta(s_0, p) \in F \Leftrightarrow \delta(s_0, p) \in F' \Leftrightarrow p \in L(A')$.

Dăm fără demonstraţie următoarea lemă:

Lema 2.3.6 *Pentru orice* $k \geq 1$, $s_1, s_2 \in S$ *avem* $s_1 \overset{k}{\equiv} s_2$ *dacă şi numai dacă* $s_1 \overset{k-1}{\equiv} s_2$ *şi* $\delta(s_1, i) \overset{k-1}{\equiv} \delta(s_2, i)$, $\forall i \in \Sigma$.

Din lema precedentă rezultă că $\overset{k}{\equiv} \subseteq \overset{k-1}{\equiv}$, $\forall k \geq 1$.

Teorema 2.3.9. *Fie automatul* $A = (S, \Sigma, \delta, s_0, F)$ *cu* $|S| = n$. *Stările* s_1 *şi* s_2 *sunt inseparabile în raport cu* F *dacă şi numai dacă ele sunt* $(n-2)$*-inseparabile în raport cu* F.

Demonstraţie.

Dacă stările s_1 şi s_2 sunt inseparabile atunci ele sunt şi $(n-2)$-inseparabile. Acest fapt decurge din definiţiile date.

Invers, să demonstrăm că dacă stările s_1 şi s_2 sunt $(n-2)$-inseparabile atunci ele sunt şi inseparabile.

Din incluziunea $\overset{k}{\equiv} \subseteq \overset{k-1}{\equiv}$, oricare ar fi $k \geq 1$, rezultă următorul şir de incluziuni:

$$(1) \qquad \rho \subseteq \ldots \subseteq \overset{k}{\equiv} \subseteq \overset{k-1}{\equiv} \subseteq \overset{k-2}{\equiv} \subseteq \ldots \subseteq \overset{1}{\equiv} \subseteq \overset{0}{\equiv}$$

Clasele de echivalenţă pentru relaţia $\overset{0}{\equiv}$ sunt F şi $S - F$, deoarece singurul cuvânt de lungime 0 este λ şi $\delta(s, \lambda) = s$, oricare ar fi $s \in S$, aşadar avem că $s_1 \overset{0}{\equiv} s_2$ dacă şi numai dacă $s_1, s_2 \in F$ sau $s_1, s_2 \notin F$.

Din şirul de incluziuni (1) rezultă următorul şir de inegalităţi:

$$(2) \quad 2 = |S/\overset{0}{\equiv}| \leq |S/\overset{1}{\equiv}| \leq \ldots \leq |S/\overset{k}{\equiv}| \leq \ldots \leq |S/\rho| \leq |S| = n$$

Deoarece mulţimea S este finită, rezultă că există un k pentru care avem $|S/\overset{k}{\equiv}| = |S/\overset{k+1}{\equiv}|$. Fie k cel mai mic număr natural pentru care avem $|S/\overset{k}{\equiv}| = |S/\overset{k+1}{\equiv}|$. Deci în (2) avem

$$(3) \qquad 2 < |S/\overset{1}{\equiv}| < \ldots < |S/\overset{k}{\equiv}| \leq n$$

Deoarece primele k inegalităţi din (3) sunt stricte rezultă că

$$k + 2 \leq |S/\overset{k}{\equiv}| \leq n$$

de unde rezultă $k \leq n - 2$.

Se poate demonstra prin inducţie după j că $S/\overset{k}{\equiv} = S/\overset{k+j}{\equiv}$, oricare ar fi $j \geq 1$. Atunci relaţia (1) devine:

$$(1') \qquad \overset{0}{\equiv} \supseteq \overset{1}{\equiv} \supseteq \ldots \supseteq \overset{k}{\equiv} = \overset{k+1}{\equiv} = \ldots = \overset{n-2}{\equiv} \ldots = \rho$$

Deci $\rho = \overset{n-2}{\equiv}$. ■

Vom construi automatul minimal care să accepte un limbaj L, recunoscut de un automat $A = (S, \Sigma, \delta, s_0, F)$. Putem considera, după cum am văzut mai înainte, că automatul A este redus. Am arătat înainte că automatul A_L este minimal, adică are cele mai puţine stări şi este echivalent cu A (i.e. recunosc acelaşi limbaj).

Considerăm automatul $A' = (S/\rho, \Sigma, \delta_\rho, [s_0]_\rho, \{[s]_\rho\, , s \in F\})$ cu $\delta_\rho([s]_\rho, i) = [\delta(s,i)]_\rho$, $\forall s \in S$ şi $\forall i \in \Sigma$. Funcţia δ_ρ este bine definită, nu depinde de reprezentantul clasei $[s]_\rho$. Într-adevăr, din $(s_1, s_2) \in \rho$ avem că $(\delta(s_1, i), \delta(s_2, i)) \in \rho$ (Lema 2.3.6).

Definiţia 2.3.10. Două automate $A_1 = (S_1, \Sigma, \delta_1, s_{01}, F_1)$ şi $A_2 = (S_2,\ \Sigma,\ \delta_2,\ s_{02},\ F_2)$ sunt *izomorfe* dacă există o bijecţie $h : S_1 \to S_2$, compatibilă cu funcţiile de tranziţie, adică: $h(\delta_1(s,i)) = \delta_2(h(s), i))$, $\forall s \in S$ şi $i \in \Sigma$.

Se poate demonstra uşor următoarea lemă:

Lema 2.3.7 *Două automate izomorfe $A_1 = (S_1, \Sigma, \delta_1, s_{01}, F_1)$ şi $A_2 = (S_2,\ \Sigma,\ \delta_2,\ s_{02},\ F_2)$, cu izomorfismul $h : S_1 \to S_2$ care satisface condiţiile $h(s_{01}) = s_{02}$ şi $h(F_1) = F_2$, recunosc acelaşi limbaj.*

Teorema 2.3.11. *Fie $L \subseteq \Sigma^*$ un limbaj regulat acceptat de automatul redus $A = (S, \Sigma, \delta, s_0, F)$. Automatul A_L este izomorf cu automatul $A_\rho = (S/\rho, \Sigma, \delta_\rho, [s_0]_\rho,\{[s]_\rho\, , s \in F\})$.*

Demonstraţie. Ştim că $A_L = (\Sigma^*/\nu_L, \Sigma, \delta', [\lambda]_{\nu_L},\{[p]_{\nu_L}\, , p \in L\})$, cu $\delta'([p]_{\nu_L}, i) = [pi]_{\nu_L}$, $\forall p \in \Sigma^*$, $i \in \Sigma$.

Definim aplicaţia $h : \Sigma^*/\nu_L \to S/\rho$ prin $h([p]_{\nu_L}) = [\delta(s_0, p)]_\rho$. Se poate arăta că *funcţia h este bine definită*, adică nu depinde de reprezentantul clasei $[p]_{\nu_L}$.

Să arătăm că h este bijecţie. În primul rând h este surjectivă, deoarece oricare ar fi $[\delta(s_0, p)]_\rho \in S/\rho$ există $[p]_{\nu_L}$ astfel încât $h([p]_{\nu_L}) = [\delta(s_0, p)]_\rho$.

Funcţia h este injectivă. Într-adevăr, din $h([p]_{\nu_L}) = h([q]_{\nu_L})$ avem $[\delta(s_0, p)]_\rho = [\delta(s_0, q)]_\rho$, de unde rezultă $\Psi(\delta(\delta(s_0, p), r)) = \Psi(\delta(\delta(s_0, q), r))$, deci $\Psi(\delta(s_0, pr)) = \Psi(\delta(s_0, qr))$, $\forall r \in \Sigma^*$. Dar ultima egalitate implică faptul că $\delta(s_0, pr)$, $\delta(s_0, qr) \in F$ sau $\delta(s_0, pr), \delta(s_0, qr) \notin F$, de unde $pr, qr \in L$ sau $pr, qr \notin L$, $\forall r \in \Sigma^*$, deci $[p]_{\nu_L} = [q]_{\nu_L}$.

Se poate arăta că h este compatibilă cu funcţiile de tranziţie, deci A_ρ şi A_L sunt izomorfe, şi că $h([\lambda]_{\nu_L}) = [s_0]_\rho$ şi $h(\{[p]_{\nu_L} \mid p \in L\}) = \{[s]_\rho \mid s \in F\}$, deci $L(A_L) = L(A_\rho)$. Deoarece între stările celor două automate există o bijecţie, rezultă că cele două automate au acelaşi număr de stări, deci amîndouă sunt *minimale* pentru limbajul L. ■

În continuare vom prezenta algoritmul de construcţie a automatului minimal echivalent cu un automat A. Întâi vom construi un automat redus echivalent cu A, prin algoritmul ACC dat mai jos, ce determină stările accesibile, iar apoi vom construi automatul minimal prin algoritmul MIN.

Algoritmul ACC

Intrare: Un automat $A = (S, \Sigma, \delta, s_0, F)$

Ieşire: Stările accesibile S' , $F' = F \cap S'$ şi automatul redus $A' = (S', \Sigma, \delta', s_0, F')$.

Pas1. $G_0 = \{s_0\}$; $i = 0$;

Pas2. $i = i + 1$; $G_i = G_{i-1} \cup \delta(G_{i-1}, \Sigma)$;

Pas3. Dacă $G_i = G_{i-1}$, atunci $S' = G_i$; treci la Pas4, altfel treci la Pas2.

Pas4. $F' = F \cap S'$ şi $A' = (S', \Sigma, \delta', s_0, F')$, unde δ' este restricţia lui δ la S'.

Se poate demonstra uşor că o stare este accesibilă dacă şi numai dacă ea aparţine mulţimii S' calculate de algoritmul ACC.

Algoritmul de construcţie a automatului minimal este:

Algoritmul MIN

Intrare: Un automat $A = (S, \Sigma, \delta, s_0, F)$ care recunoaşte limbajul L.

Ieşire: Automatul minimal A_{min} care recunoaşte L.

Pas1 Se aplică algoritmul ACC şi se determină automatul redus $A' = (S', \Sigma, \delta', s_0, F')$.

Pas2 Se calculează S'/ρ.

Pas3 $A_{\min} = (S'/\rho, \Sigma, \delta_\rho, [s_0]_\rho, \{[s]_\rho, s \in F'\})$.

4 Automate nedeterministe

Definiţia 2.4.1. Un automat nedeterminist este un 5-uplu
$A = (S, \Sigma, \delta, s_0, F)$ unde:

- S este o mulţime nevidă, *mulţimea stărilor automatului nedeterminist*;

- Σ este o mulţime nevidă şi finită, *alfabetul de intrare a automatului nedeterminist*;

- $\delta : S \times \Sigma \to \mathcal{P}(S)$, *funcţia de tranziţie*;

- $s_0 \in S$ este *starea iniţială a automatului*;

- $F \subseteq S$ este *mulţimea stărilor finale*.

Dacă S este finită, atunci automatul nedeterminist este finit.

Vom considera mai departe numai automate nedeterministe finite. Faptul că pentru un $(s, i) \in S \times \Sigma$, $\delta(s, i)$ este o mulţime, se interpretează că automatul nedeterminist fiind în starea s şi primind ca intrare simbolul i, poate trece în oricare din stările mulţimii $\delta(s, i)$, de unde şi denumirea de *nedereminist* deoarece starea următoare nu este unic determinată, ea va fi "aleasă" dintr-o mulţime de stări.

Ca şi la automate deterministe finite, definiţia funcţiei de tranziţie se poate face fie tabelar fie cu ajutorul grafului de tranziţie ataşat automatului. Funcţia de tranziţie se poate da ca o matrice cu $|S|$ linii şi $|\Sigma|$ coloane, iar elementele matricei sunt $\delta(s, i)$.

Exemplul 2.4.2. Fie automatul finit nedeterminist
$A = (\{s_1, s_2, s_3\}, \{a, b\}, \delta, s_1, F)$ unde δ este definită prin:

$S \setminus \Sigma$	a	b
s_1	$\{s_1\}$	$\{s_1, s_2\}$
s_2	$\{s_1, s_3\}$	$\{s_3\}$
s_3	$\{s_1, s_2\}$	$\{s_2\}$

Fig. 2.4.2.

O altă posibilitate de a da funcţia de tranziţie este cu ajutorul grafului de tranziţie.

Definiţia 2.4.3. Graful tranziţiilor automatului finit nedeterminist $A = (S, \Sigma, \delta, s_0, F)$ este cuplul (G, γ) unde $G = (S, U)$ este un graf orientat având ca mulţime de vârfuri mulţimea stărilor S, iar ca mulţime de arce mulţimea U definită prin $U = \{(s_j, s_k) \mid s_j, s_k \in S,\ \exists i \in \Sigma,\ s_k \in \delta(s_j, i)\}$. Funcţia de etichetare $\gamma : U \to \mathcal{P}(\Sigma)$ este definită prin $\gamma(u) = \{i \mid i \in \Sigma, u = (s_j, s_k), s_k \in \delta(s_j, i)\}$, $\forall u \in U$.

Graful tranziţiilor automatului finit nedeterminist A din exemplul precedent este următorul:

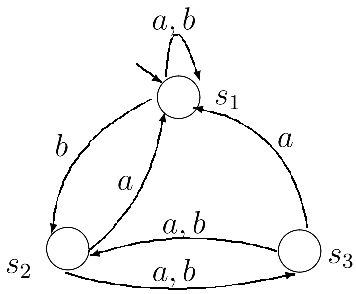


Fig. 2.4.3.

Un automat nedeterminist pentru care $|\delta(s, i)| = 1$, $\forall (s, i) \in S \times \Sigma$ poate fi privit ca un automat determinist. Deci automatele nedeterministe sunt “mai generale” (se identifică mulţimile cu un element, cu elementul însuşi).

Funcţia de tranziţie $\delta : S \times \Sigma \to \mathcal{P}(S)$ se poate extinde la $S \times \Sigma^*$ prin:

1. $\widehat{\delta}(s, \lambda) = \{s\}$, $\forall s \in S$;

2. $\widehat{\delta}(s, pi) = \bigcup\{\delta(s', i) \mid s' \in \widehat{\delta}(s, p)\}$ $\forall s \in S, p \in \Sigma^*, i \in \Sigma$.

Definiţia 2.4.4. Limbajul acceptat de automatul finit nedeterminist $A = (S, \Sigma, \delta, s_0, F)$ este $L(A) = \{p | p \in \Sigma^*, \delta(s_0, p) \cap F \neq \emptyset\}$.

Conform acestei definiţii, un cuvânt $p \in \Sigma^*$ este în limbajul $L(A)$ dacă automatul, primind la intrare simbolurile cuvântului p, are posibilitatea să ajungă într-o stare finală din starea iniţială, sau în graful tranziţiilor **există cel puţin un drum** cu arcele etichetate cu simbolurile lui p de la s_0 la o stare din F.

Observaţia 2.4.5. Se poate defini limbajul acceptat de un automat nedeterminist şi prin $L(A) = \{p \mid p \in \Sigma^* \text{ şi } \delta(s_0, p) \subseteq F\}$, dar această definiţie este mai restrictivă.

Fiecărui automat finit determinist $A = (S, \Sigma, \delta, s_0, F)$ îi corespunde un automat finit nedeterminist $A' = (S, \Sigma, \delta', s_o, F)$ astfel încât $L(A) = L(A')$.

Definim $\delta'(s, i) = \{\delta(s, i)\}$, $\forall (s, i) \in S \times \Sigma$. Se verifică imediat prin inducţie după lungimea cuvântului $p \in \Sigma^*$ că $\delta'(s, p) = \{\delta(s, p)\}$, $\forall (s, p) \in S \times \Sigma^*$.

Folosind definiţiile limbajelor acceptate de automate finite deterministe şi nedeterministe, avem:
$L(A') = \{p \mid p \in \Sigma^*, \delta'(s_0, p) \cap F \neq \emptyset\}$=$\{p \mid p \in \Sigma^*, \delta(s_0, p) \cap F \neq \emptyset\}$=$\{p \mid p \in \Sigma^*, \delta(s_0, p) \in F\} = L(A)$.

De aici rezultă că limbajele acceptate de automatele finite deterministe sunt acceptate şi de automatele finite nedeterministe.

Interesant este că mulţimea limbajelor acceptate de automatele finite nedeterministe coincide cu mulţimea limbajelor acceptate de automatele finite deterministe. Pentru aceasta, trebuie să mai arătăm că pentru orice limbaj care este acceptat de un automat finit nedeterminist există un automat finit determinist care să accepte acest limbaj.

Teorema 2.4.6. *Fie L un limbaj acceptat de un automat finit nedeterminist $A = (S, \Sigma, \delta, s_0, F)$. Există un automat finit determinist care acceptă limbajul L.*

Demonstraţie. Considerăm automatul finit determinist
$A' = (\mathcal{P}(S),\ \Sigma,\ \delta,\ \{s_0\}, F')$ unde $F' = \{S' \mid S' \subseteq S, S' \cap F \neq \emptyset\}$ şi $\delta' : \mathcal{P}(S) \times \Sigma \to \mathcal{P}(S)$ este definită prin:

$$\delta'(S',i) = \begin{cases} \bigcup\{\delta(s,i) | s \in S'\}, \forall S' \in \mathcal{P}(S), & \text{dacă } S' \neq \emptyset \\ \emptyset, & \text{dacă } S' = \emptyset. \end{cases}$$

Putem extinde $\delta' : \mathcal{P}(S) \times \Sigma^* \to \mathcal{P}(S)$ prin

1. $\delta'(S', \lambda) = S'$, $\forall S' \in \mathcal{P}(S)$;
2. $\delta'(S', pi) = \delta'(\delta'(S', p), i)$, $\forall S' \in \mathcal{P}(S), p \in \Sigma^*, i \in \Sigma$.

Să demonstrăm acum următoarea egalitate:

$$\delta(s,p) = \delta'(\{s\}, p), \ \forall s \in S, p \in \Sigma^*. \tag{1}$$

Egalitatea (1) o demonstrăm prin inducţie după lungimea cuvântului p.
Pentru $l(p) = 0$, adică $p = \lambda$, avem $\delta(s, \lambda) = \{s\} = \delta'(\{s\}, \lambda)$.
Presupunem proprietatea adevărată pentru cuvinte de lungime cel mult n şi o vom demonstra pentru cuvinte de lungime $n + 1$. Fie q cu $l(q) = n$ şi $i \in \Sigma$; atunci $p = qi$ are lungimea $n + 1$. Avem $\delta(s,p) = \delta(s,qi) = \bigcup\{\delta(s', i) | s' \in \delta(s,q)\}$. Pentru membrul drept al egalităţii (1) avem $\delta'(\{s\}, p) = \delta'(\{s\}, qi) = \delta'(\delta'(\{s\}, q), i) = \bigcup\{\delta(s', i) \mid s' \in \delta'(\{s\}, q)\} = \bigcup\{\delta(s', i) \mid s' \in \delta(s, q)\}$; din egalitatea (1). Aici am folosit ipoteza inductivă că $\delta'(\{s\}, q) = \delta(s, q) \neq \emptyset$. Dacă $\delta(s,q) = \delta'(\{s\}, q) = \emptyset$, atunci avem $\delta'(\{s\}, p) = \emptyset = \delta(s, p)$.
Luând $s = s_0$ avem:

$$\delta(s_0,p) = \delta'(\{s_0\}, p), \ \forall p \in \Sigma^*. \tag{1'}$$

Vom arăta că $L = L(A')$. Fie $p \in L$; rezultă că $p \in L(A) = L$, de unde $\delta(s_0, p) \cap F \neq \emptyset$, deci $\delta(s_0, p) \in F'$. Deoarece $\delta(s_0,p) = \delta'(\{s_0\}, p)$ avem că $\delta'(\{s_0\}, p) \in F'$, deci $p \in L(A')$.

Invers, fie $p \in L(A')$; atunci avem $\delta'(\{s_0\}, p) \in F'$, deci $\delta'(\{s_0\}, p) \cap F \neq \emptyset$. Din nou, din $\delta'(\{s_0\}, p) = \delta(s_0, p)$ avem că $\delta(s_0, p) \cap F \neq \emptyset$, deci $p \in L(A) = L$. ■

5 Limbaje regulate şi limbaje de tip 3

În acest paragraf vom arăta că mulţimea limbajelor regulate, $\mathcal{L}_R$, coincide cu mulţimea limbajelor de tip trei, $\mathcal{L}_3$.

Teorema 2.5.1. *Orice limbaj regulat este un limbaj de tip trei.*

Demonstraţie. Fie L un limbaj regulat. Există un automat finit determinist $A = (S, \Sigma, \delta, s_0, F)$ astfel încât $L = L(A)$.

Considerăm gramatica $G = (S, \Sigma, s_0, P)$, cu $P = \{s \to is' \mid s' = \delta(s, i)\} \cup \{s \to i \mid \delta(s, i) \in F\} \cup P_0$, unde

$$P_0 = \begin{cases} \{s_0 \to \lambda\}, & \text{dacă } s_0 \in F \\ \emptyset, & \text{dacă } s_0 \notin F. \end{cases}$$

Această gramatică G este de tipul trei. Să arătăm acum că $L = L(G)$. Să demonstrăm $L \subseteq L(G)$. Fie $p \in L$, $p \neq \lambda$; atunci $p = i_1 \dots i_n$, $n \geq 1$, $i_j \in \Sigma$, $1 \leq j \leq n$. Din $p \in L$ rezultă $p \in L(A)$, deci $\delta(s_0, p) \in F$. Considerăm următorul şir de stări: $s_0, s_1 = \delta(s_0, i_1), \dots, s_j = \delta(s_{j-1}, i_j), \dots, s_n = \delta(s_{n-1}, i_n)$. Deoarece $s_n = \delta(s_{n-1}, i_n) = \delta(\delta(s_{n-2}, i_{n-1}), i_n) = \delta(s_{n-2}, i_{n-1}i_n) = \dots = \delta(s_0, i_1 \dots i_n) = \delta(s_0, p) \in F$, avem că $s_n \in F$.

Corespunzător şirului de stări considerate mai sus, avem în gramatica G regulile: $s_0 \to i_1 s_1$, $s_1 \to i_2 s_2$, …, $s_{n-2} \to i_{n-1} s_{n-1}$, $s_{n-1} \to i_n$. Ultima regulă este $s_{n-1} \to i_n$ deoarece $s_n \in F$ şi $s_n = \delta(s_{n-1}, i_n)$. Utilizând aceste reguli obţinem în G următoarea derivare:
$s_0 \Rightarrow i_1 s_1 \Rightarrow i_1 i_2 s_2 \Rightarrow \dots \Rightarrow i_1 \dots i_{n-1} s_{n-1} \Rightarrow i_1 \dots i_n$ deci $p \in L(G)$. Dacă $\lambda \in L$, atunci $\delta(s_0, \lambda) = s_0 \in F$ şi prin urmare în G avem regula $s_0 \to \lambda$ şi derivarea $s_0 \Rightarrow \lambda$, deci $\lambda \in L(G)$.

Invers, să demonstrăm incluziunea $L(G) \subseteq L$.
Fie $p \in L(G)$ cu $p \neq \lambda$. Din $p \in L(G)$ rezultă că există în gramatica G derivarea $s_0 \overset{*}{\Rightarrow} p$. Dacă $p = i_1 \dots i_n$, $n \geq 1$, atunci în derivarea $s_0 \overset{*}{\Rightarrow} p$ se aplică succesiv regulile de forma $s_0 \to i_1 s_1$, $s_1 \to i_2 s_2$, …, $s_{n-2} \to i_{n-1} s_{n-1}$, $s_{n-1} \to i_n$, ceea ce antrenează $s_j = \delta(s_{j-1}, i_j)$, $1 \leq j \leq n$ şi $\delta(s_{n-1}, i_n) \in F$. Dar $\delta(s_{n-1}, i_n) = \delta(\delta(s_{n-2}, i_{n-1}), i_n) = \dots = \delta(s_0, i_1 \dots i_n) = \delta(s_0, p) \in F$, deci $p \in L$.

Dacă $p = \lambda \in L(G)$ atunci p se obţine din derivarea $s_0 \Rightarrow \lambda$. Deci în gramatica G există regula $s_0 \rightarrow \lambda$. Dar această regulă există numai dacă $s_0 \in F$ şi prin urmare, $\delta(s_0, \lambda) \in F$, deci $\lambda \in L$. ∎

Pentru a demonstra incluziunea $\mathcal{L}_3 \subseteq \mathcal{L}_R$ sunt necesare unele rezultate privind limbajele de tip trei. O parte din rezultate le vom demonstra în cadrul mai general al gramaticilor independente de context.

Definiţia 2.5.2. Fie $G = (V_N, V_T, x_0, P)$ o gramatica independentă de context. Vom spune că derivarea $w_1 \Rightarrow w_2 \ldots \Rightarrow w_n$, $n \geq 2$ este o *derivare stângă a lui* w_n dacă pentru fiecare i, $1 \leq i \leq n-1$, $w_i = \alpha_i A \beta_i$, $w_{i+1} = \alpha_i u_i \beta_i$, $A \rightarrow u_i \in P$ şi $\alpha_i \in V_T^*$.

O derivare stângă este acea derivare în care se înlocuieşte simbolul neterminal cel mai din stânga din cuvânt.

Teorema 2.5.3. *Fie* $G = (V_N, V_T, x_0, P)$ *o gramatică independentă de context şi* $w \in L(G)$. *Există o derivare stângă a lui* w *din* x_0 *în gramatica* G.

Demonstraţie. Vom demonstra mai general că fiecărei derivări $A \overset{*}{\Rightarrow} w$, cu $A \in V_N$ şi $w \in V^*$ îi corespunde o derivare stângă $A \overset{*}{\Rightarrow} w$. Demonstraţia acestei proprietăţi o facem prin inducţie după lungimea derivării.

Dacă $A \overset{*}{\Rightarrow} w$ este de lungime 1, atunci $A \rightarrow w \in P$ şi această derivare $A \overset{*}{\Rightarrow} w$ este o derivare stângă.

Presupunem proprietatea adevărată pentru toate derivările de lungime cel mult k şi fie $A \Rightarrow y \overset{*}{\Rightarrow} w$ o derivare de lungime $k+1$. Fie $y = y_1 \ldots y_n$ cu $y_i \in V_N \cup V_T$, $1 \leq i \leq n$. Aplicând teorema de localizare pentru derivarea $y_1 \ldots y_n \underset{G}{\overset{*}{\Rightarrow}} w$, avem că $w = w_1 \ldots w_n$ şi $y_i \underset{G}{\overset{*}{\Rightarrow}} w_i$, $1 \leq i \leq n$. Derivările $y_i \underset{G}{\overset{*}{\Rightarrow}} w_i$, $1 \leq i \leq n$ au lungimile cel mult k, deci putem aplica pasul inductiv şi există derivările stângi $y_i \overset{*}{\Rightarrow} w_i$ $1 \leq i \leq n$. Cum derivarea $A \Rightarrow y_1 \ldots y_n$ este o derivare stângă şi apoi aplicăm de la stânga spre dreapta derivările stângi $y_i \overset{*}{\Rightarrow} w_i$, $1 \leq i \leq n$, avem $A \Rightarrow y_1 \ldots y_n \Rightarrow w_1 y_2 \ldots y_n \Rightarrow \ldots \Rightarrow w_1 \ldots w_n$ care este o derivare stânga.

Luând $A = x_0$ avem afirmaţia teoremei. ■

Observaţia 2.5.4. Pentru gramatici de tipul 2 sau 3, atunci când ne va interesa limbajul generat de gramatică, vom putea considera că acest limbaj este generat numai prin derivări stângi. Derivările stângi le vom nota cu $\underset{\text{st}}{\Rightarrow}$.

Definiţia 2.5.5. O gramatică G este *fără redenumiri* dacă ea nu conţine reguli de forma $x \to y$ cu $x, y \in V_N$.

Evident, o regulă de forma $x \to y$ cu $x, y \in V_N$ nu generează nimic, schimbă numai numele simbolului neterminal din x în y, de aceea o astfel de regulă se numeşte *redenumire*.

Teorema 2.5.6. *Orice limbaj L de tipul 2 (sau 3) poate fi generat de o gramatică de tipul 2 (sau 3) fără redenumiri.*

Demonstraţie. Fie $G = (V_N, V_T, x_0, P)$ o gramatică de tipul 2 sau 3 cu $L = L(G)$. Putem considera că singura regulă cu λ în partea dreaptă, dacă există, este $x_0 \to \lambda$ şi în acest caz x_0 nu apare în partea dreaptă a vreunei reguli.
Dacă gramatica G nu are redenumiri, atunci ea satisface cerinţele teoremei. În caz contrar, dacă G are redenumiri, atunci considerăm gramatica $G_1 = (V_N, V_T, x_0, P_1)$, cu $P_1 = P' \cup P''$, unde:
$P' = \{x \to r \mid x \to r \in P$ şi $r \notin V_N\}$,
$P'' = \{x \to r \mid \exists\ x \overset{+}{\underset{G}{\Rightarrow}} y,\ y \to r \in P$ şi $r \notin V_N\}$.
Mulţimea P' conţine toate regulile din P care nu sunt redenumiri, iar P'' conţine reguli care le formăm cu ajutorul redenumirilor. Se poate arăta uşor că $L(G) = L(G_1)$, adică cele două gramatici sunt echivalente. ■

Observaţia 2.5.7. Conform acestei teoreme, limbajele de tipul 2 sau 3 le putem considera ca fiind generate de gramatici de tipul 2 sau 3 fără redenumiri.

Vom arăta mai departe că limbajele de tip 3 pot fi generate de gramatici de tip 3 care au o formă particulară numită *formă normală*.

Teorema 2.5.8. *Fie L un limbaj de tip trei. Există o gramatică de tip trei, $G = (V_N, V_T, x_0, P)$ care generează limbajul L şi G are regulile de forma $x_1 \to ix_2$, $x \to i$ sau $x_0 \to \lambda$, caz în care x_0 nu apare în partea dreaptă a vreunei reguli, unde $x_1, x_2, x \in V_N$ şi $i \in V_T$.*

Demonstraţie. Din $L \in \mathcal{L}_3$, rezultă că există o gramatică de tipul trei fără redenumiri $G_1 = (V_{N_1}, V_{T_1}, x_0, P_1)$ astfel încât $L = L(G_1)$. Dacă $\lambda \in L$, atunci avem regula $x_0 \to \lambda$ caz în care x_0 nu apare în partea dreaptă a vreunei reguli şi aceasta este singura regulă cu λ în partea dreaptă.

Regulile din P_1 de forma $x_1 \to px_2$ au $l(p) \geq 1$, deoarece nu avem redenumiri.
Dacă $l(p) = 1$, atunci $p \in V_T$ şi $x_1 \to px_2$ este de forma cerută în teoremă.
În cazul când $l(p) \geq 2$, fie $p = i_1 \dots i_n$, $n \geq 2$. Vom introduce $n-1$ simboli neterminali noi $z_1, \dots, z_{n-1}$. Regulii $x_1 \to px_2 \in P_1$ îi corespunde în G următorul şir de reguli $x_1 \to i_1 z_1$, $z_1 \to i_2 z_2, \dots, z_{n-2} \to i_{n-2} z_{n-1}$, $z_{n-1} \to i_n x_2$, care au forma cerută în teoremă. Acest lucru îl facem pentru fiecare regulă de acest tip.

O regulă de forma $x \to p \in P_1$, cu $l(p) = 1$, o punem şi în P. Unei reguli de forma $x \to p \in P_1$, cu $p = i_1 \dots i_n$, $n \geq 2$ şi $i_j \in V_T$, $1 \leq j \leq n$, îi va corespunde în P regulile $x \to i_1 v_1$, $v_1 \to i_2 v_2, \dots, v_{n-2} \to i_{n-1} v_{n-1}$, $v_{n-1} \to i_n$, unde $v_1, \dots, v_n$ sunt simboli neterminali noi. Procedăm analog pentru fiecare regulă $x \to p \in P_1$.

Fie gramatica $G = (V_N \cup Z \cup \mathcal{V}, V_T, x_0, F)$, unde Z este mulţimea simbolilor neterminali noi adăugaţi pentru toate regulile de forma $x_i \to px_2$ şi $\mathcal{V}$ este mulţimea simbolilor noi adăugaţi pentru reguli de forma $x \to p$. Se observă că aplicarea regulei $x_1 \to px_2$ şi aplicarea succesivă a regulilor $x_1 \to i_1 z_1$, $z_1 \to i_2 z_2, \dots, z_{n-1} \to i_n x_2$ are acelaşi efect, adică generarea cuvântului p şi în plus ultimul şir de reguli se pot aplica numai succesiv toate, deoarece $z_1, \dots, z_{n-1}$ sunt simboli noi. Acelaşi lucru se poate spune despre o regulă de forma $x \to p$ şi şirul $x \to i_1 v_1, \dots, v_{n-1} \to i_n$. Din cele de mai sus, avem că $x_0 \overset{*}{\underset{G_1}{\Rightarrow}} p \Leftrightarrow x_0 \overset{*}{\underset{G}{\Rightarrow}} p$, deci $L = L(G)$. ■

În continuare vom demonstra incluziunea $\mathcal{L}_3 \subseteq \mathcal{L}_R$. Acest rezultat este stabilit în teorema următoare.

Teorema 2.5.9. *Orice limbaj de tip trei este un limbaj regulat.*

Demonstraţie. Fie L un limbaj de tip 3. Atunci există o gramatică $G = (V_N, V_T, x_0, P)$ de tip 3 care generează L. Dacă $\lambda \in L(G)$ atunci există regula $x_0 \to \lambda \in P$ şi x_0 nu apare în partea dreaptă a vreunei reguli din P. Putem presupune că gramatica G este sub formă normală, adică regulile sale sunt de forma $x \to ix'$ sau $x \to i$, cu $x, x' \in V_N$ şi $i \in V_T$.

Pentru a arăta că L este regulat, vom arăta că există un automat finit nedeterminist care să accepte limbajul L. Pentru aceasta, construim automatul finit nedeterminist $A = (V_N \cup \{\bar{x}\}, V_T, \delta, x_0, F)$, unde δ este definită prin: $\delta(x, i) = \{x' \mid x \to ix' \in P\} \cup B$, unde

$$B = \begin{cases} \{\bar{x}\}, & \text{dacă } x \to i \in P \\ \emptyset, & \text{în caz contrar.} \end{cases}$$

Mulţimea F este dată prin:

$$F = \begin{cases} \{\bar{x}, x_0\}, & \text{dacă } x_0 \to \lambda \in P \\ \{\bar{x}\}, & \text{în caz contrar.} \end{cases}$$

Să arătăm că $\boldsymbol{L(G) = L(A)}$. Demonstrăm întâi incluziunea $L(G) \subseteq L(A)$. Fie $p \in L(G)$. Vom considera două cazuri:
1) dacă $p = \lambda$, atunci există în gramatica G regula $x_0 \to \lambda \in P$, şi prin urmare $\delta(x_0, \lambda) = \{x_0\} \subseteq F$, de unde rezultă că $\lambda \in L(A)$.
2) dacă $p \neq \lambda$, atunci $p = i_1 \dots i_k$, $k \geq 1$ şi $i_j \in V_T$ pentru orice $1 \leq j \leq k$. Deoarece $p \in L(G)$ rezultă că există derivarea $x_0 \underset{G}{\overset{*}{\Rightarrow}} p$. Din faptul că gramatica G este în formă normală rezultă că această derivare are forma:

$$x_0 \underset{G}{\Rightarrow} i_1x_1 \underset{G}{\Rightarrow} i_1i_2x_2 \underset{G}{\Rightarrow} \dots \underset{G}{\Rightarrow} i_1i_2 \dots i_{k-1}x_{k-1} \underset{G}{\Rightarrow} i_1 \dots i_k\,.$$

Deci în gramatica G există regulile

(1) $\qquad x_0 \to i_1x_i\,,\ x_1 \to i_2x_2\,, \dots,\ x_{k-2} \to i_{k-1}x_{k-1}$ şi $x_{k-1} \to i_k\,.$

Din definiţia funcţiei de tranziţie δ, corespunzător şirului (1) avem:
(2)
$x_1 \in \delta(x_0, i_1)\,,\ x_2 \in \delta(x_1, i_2)\,, \ldots,\ x_{k-1} \in \delta(x_{k-2}, i_{k-1})$ şi $\bar{x} \in \delta(x_{k-1}, i_k)\,.$

Ţinînd cont da faptul că $\delta(x,p) \subseteq \delta(x', i'p)$ dacă $x \in \delta(x', i')$, pentru orice $x, x' \in V_N$, $i \in V_T$ şi $p \in V_T^*$, proprietate ce se demonstrează prin inducţie după lungimea cuvântului p, avem:

$$\bar{x} \in \delta(x_{k-1}, i_k) \subseteq \delta(x_{k-2}, i_{k-1}i_k) \subseteq \ldots \subseteq \delta(x_0, i_1 \ldots i_k)\,,$$

deci $\bar{x} \in \delta(x_0, p) \cap F$, de unde rezultă că $\delta(x_0, p) \cap F \neq \emptyset$, deci $p \in L(A)$.

Să demonstrăm acum incluziunea inversă, $L(A) \subseteq L(G)$. Fie $p \in L(A)$, rezultă că $\delta(x_0, p) \cap F \neq \emptyset$. Vom considera două cazuri:
1) dacă $p = \lambda$, atunci avem $\delta(x_0, \lambda) \cap F \neq \emptyset$, de unde $x_0 \in F$ şi prin urmare există regula $x_0 \to \lambda \in P$. Din $x_0 \to \lambda \in P$ rezultă că există derivarea $x_0 \underset{G}{\Rightarrow} \lambda$, deci $\lambda \in L(G)$.
2) dacă $p \neq \lambda$, atunci $p = i_1 \ldots i_k$, $k \geq 1$ şi $i_j \in V_T$ pentru $1 \leq j \leq k$. Din $\delta(x_0, i_1 \ldots i_k) \cap F \neq \emptyset$ rezultă că există stările $x_1, \ldots, x_{k-1}, \bar{x}$ cu
(3)
$x_1 \in \delta(x_0, i_1)\,,\ x_2 \in \delta(x_1, i_2)\,, \ldots,\ x_{k-1} \in \delta(x_{k-2}, i_{k-1})$ şi $\bar{x} \in \delta(x_{k-1}, i_k)\,.$

Ultima apartenenţă este $\bar{x} \in \delta(x_{k-1}, i_k)$ şi nu $x_0 \in \delta(x_{k-1}, i_k)$ deoarece dacă am avea $x_0 \in \delta(x_{k-1}, i_k)$, atunci în gramatica G ar exista regula $x_{k-1} \to i_k x_0$, ceea ce contrazice presupunerea că în regulile lui G, x_0 nu apare în partea dreaptă. Din apartenenţele din şirul (3) şi din definiţia funcţiei δ rezultă că în gramatica G avem regulile

(4) $\quad x_0 \to i_1 x_1\,,\ x_1 \to i_2 x_2\,, \ldots,\ x_{k-2} \to i_{k-1} x_{k-1}$ şi $x_{k-1} \to i_k\,.$

Atunci în gramatica G putem scrie derivarea:

$$x_0 \underset{G}{\Rightarrow} i_1 x_1 \underset{G}{\Rightarrow} i_1 i_2 x_2 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} i_1 \ldots i_{k-1} x_{k-1} \underset{G}{\Rightarrow} i_1 i_2 \ldots i_k\,,$$

prin urmare $p \in L(G)$. ■

Teorema 2.5.10. *Familia $\mathcal{L}_3$ este închisă la produs.*

Demonstraţie. Fie $L_1, L_2 \in \mathcal{L}_3$; rezultă că există gramaticile de tip 3 $G_1 = (V_N^1, V_T^1, x_0^1, P_1)$ şi $G_2 = (V_N^2, V_T^2, x_0^2, P_2)$ cu $L_1 = L(G_1)$ şi

$L_2 = L(G_2)$ şi $V_N^1 \cap V_N^2 = \emptyset$. Considerăm gramatica $G = (V_N^1 \cup V_N^2, V_T^1 \cup V_T^2, x_0^1, \{x \to px' \mid x \to px' \in P_1\} \cup \{x \to px_0^2 \mid x \to p \in P_1\} \cup P_2)$. Gramatica G este de tip 3.

Vom demonstra prin dublă incluziune că $L_1 \cdot L_2 = L(G)$.

Să demonstrăm întâi că $\boldsymbol{L_1 \cdot L_2 \subseteq L(G)}$. Fie $p \in L_1 \cdot L_2$ atunci $p = p_1 \cdot p_2$ cu $p_1 \in L_1$ şi $p_2 \in L_2$. Din $p \in L_1$ rezultă că există derivarea $x_0^1 \overset{*}{\underset{G_1}{\Rightarrow}} p_1$ şi din $p \in L_2$ avem $x_0^2 \overset{*}{\underset{G_2}{\Rightarrow}} p_2$. În derivarea $x_0^1 \overset{*}{\underset{G_1}{\Rightarrow}} p_1$ se aplică reguli de forma $x \to qx'$ cu excepţia ultimei reguli care este de forma $x \to q$, cu $q \in V_T^*$; dar în G avem regula corespunzătoare $x \to qx_0^2$. Deci putem scrie în G următoarea derivare $x_0^1 \overset{*}{\underset{G}{\Rightarrow}} p_1 x_0^2 \overset{*}{\underset{G}{\Rightarrow}} p_1 p_2$, ceea ce înseamnă că $p \in L(G)$.

Invers, $\boldsymbol{L(G) \subseteq L_1 \cdot L_2}$. Fie $p \in L(G)$ rezultă că există derivarea $x_0^1 \overset{*}{\underset{G}{\Rightarrow}} p$. Din construcţia gramaticii G şi din $V_N^1 \cap V_N^2 = \emptyset$, deoarece numai regulile din P_2 pot elimina neterminali, avem că derivarea de sus este de forma $x_0^1 \overset{*}{\underset{G}{\Rightarrow}} p_1 x_0^2 \overset{*}{\underset{G}{\Rightarrow}} p_1 p_2$. În derivarea $x_0^1 \overset{*}{\underset{G}{\Rightarrow}} p_1 x_0^2$ regulile aplicate sunt cele din G_1 cu excepţia ultimei reguli care este de forma $x \to \alpha x_0^2$ cu $\alpha \in V_T^*$, dar atunci $x \to \alpha \in P_1$. Deci putem scrie în G_1, $x_0^1 \overset{*}{\underset{G_1}{\Rightarrow}} p_1$ şi prin urmare $p_1 \in L_1$. Din $p_1 x_0^2 \overset{*}{\underset{G}{\Rightarrow}} p_1 p_2$ avem că $x_0^2 \overset{*}{\underset{G}{\Rightarrow}} p_2$, dar aici se aplică numai reguli din P_2, deci $x_0^2 \overset{*}{\underset{G_2}{\Rightarrow}} p_2$ şi $p_2 \in L_2$. Dar $p_1 \in L_1$, $p_2 \in L$ şi $p = p_1 \cdot p_2$ implică $p \in L_1 \cdot L_2$. ■

Teorema 2.5.11. *Familia $\mathcal{L}_3$ este închisă la operaţia $*$.*

Demonstraţie. Fie $L \in \mathcal{L}_3$; trebuie să demonstrăm că $L^* \in \mathcal{L}_3$. Din $L \in \mathcal{L}_3$ rezultă că există o gramatică $G = (V_N, V_T, x_0, P)$ cu $L(G) = L$. Vom construi o gramatică $G_* = (V_N, V_T, x_0, P \cup \{x \to px_0 \mid x \to p \in P\} \cup \{x_0 \to \lambda\})$. Gramatica G_* este de tipul 3. Putem considera că în gramatica G, x_0 nu apare în membrul drept al vreunei reguli din P. Vom demonstra că $L(G_*) = L^*$.

Incluziunea $\boldsymbol{L(G_*) \subseteq L^*}$. Fie $p \in L(G_*)$; rezultă că $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p$. În această derivare punem în evidenţă cuvintele care conţin x_0. Simbolul x_0 poate să apară numai în urma aplicării unei reguli de forma $x \to \alpha x_0$. Deci avem $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_1 x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_1 p_2 x_0 \overset{*}{\underset{G_*}{\Rightarrow}} \ldots \overset{*}{\underset{G_*}{\Rightarrow}} p_1 \ldots p_{l-1} x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_1 \ldots p_{l-1} p_l$. În această derivare apar derivările $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_i x_0$, $1 \leq i \leq$

$l-1$ şi $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_l$. În derivările $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_i x_0$, $1 \le i \le l-1$ se aplică numai reguli din G cu excepţia ultimei reguli care este de forma $x \to \alpha x_0$, dar atunci în gramatica G avem regula $x \to \alpha$. Deci în gramatica G putem scrie derivările $x_0 \overset{*}{\underset{G}{\Rightarrow}} p_i$, $1 \le i \le l-1$, deci $p_i \in L$, $1 \le i \le l-1$. Iar derivarea $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_l$ este de fapt în G, deci $p_l \in L$. Ţinând cont că $p = p_1 \dots p_l$ si $p_i \in L$, $1 \le i \le l$, rezultă că $p \in L^*$.

Incluziunea $\boldsymbol{L^* \subseteq L(G_*)}$. Fie $p \in L^*$; rezultă că există $k \ge 0$ astfel încât $p \in L^k$. Vom considera trei cazuri:

1. $k = 0$. Atunci $p = \lambda$ şi în gramatica G_* avem $x_0 \underset{G_*}{\Rightarrow} \lambda$, deci $p \in L(G_*)$;

2. $k = 1$. Atunci $p \in L$ şi cum regulile lui G sunt şi în G_*, avem $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p$, deci $p \in L(G_*)$;

3. $k > 1$. În acest caz $p = p_1 \dots p_k$ şi $p_i \in L$, $1 \le i \le k$, deci avem $x_0 \overset{*}{\underset{G}{\Rightarrow}} p_i$, $1 \le i \le k$. Din construcţia gramaticii G_* avem derivările $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_i x_0$ $1 \le i < k$ şi $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_k$. Deci putem scrie $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_1 p_2 x_0 \overset{*}{\underset{G_*}{\Rightarrow}} \dots \overset{*}{\underset{G_*}{\Rightarrow}} p_1 \dots p_{k-1} x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_1 \dots p_k$, prin urmare $p \in L(G_*)$. ■

Teorema 2.5.12. *Familia $\mathcal{L}_3$ este închisă la operaţiile de complementariere şi intersecţie.*

Demonstraţie. Fie $L \in \mathcal{L}_3$; trebuie să demonstrăm că $\Sigma^* - L \in \mathcal{L}_3$. Din $L \in \mathcal{L}_3$ rezultă că există un automat finit determinist $A = (S, \Sigma, \delta, s_0, F)$ cu $L = L(A)$. Considerăm automatul finit determinist $A^* = (S, \Sigma, \delta, s_0, S - F)$. Să arătăm că $L(A^*) = \Sigma^* - L$, ceea ce este echivalent cu $p \in L(A^*)$ dacă şi numai dacă $p \notin L(A)$.
Din $p \in L(A^*)$ avem $\delta(s_0, p) \in S - F$, deci $p \notin L(A)$.
Din $p \notin L(A)$ avem $\delta(s_0, p) \notin F$, deci $\delta(s_0, p) \in S - F$ şi rezultă $p \in L(A^*)$. Vom nota $\overline{L} = \Sigma^* - L$.

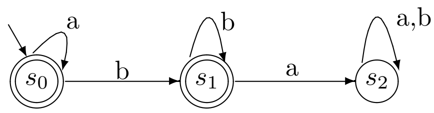
Pentru închiderea la intersecţie, folosim $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ şi cum $\mathcal{L}_3$ este închis la reuniune şi complementariere, rezultă că este închis la intersecţie. ■

O altă proprietate a limbajelor regulate este descrisă de următorul rezultat, pe care-l vom aminti fără demonstraţie:

Lema Bar–Hillel. *Fie L un limbaj de tip 3. Există un număr k astfel încât oricare ar fi $w \in L$ cu $|w| \geq k$, are o descompunere de forma $w = xyz$, unde $0 < |y| \leq k$ şi $xy^iz \in L$ oricare ar fi $i \geq 0$.*

Exerciţii rezolvate

1) Să se arate că automatul $A = (\{s_0, s_1, s_2\}, \{a, b\}, \delta, s_0, \{s_0, s_1\})$, cu δ definită prin graful de tranziţie din figura următoare, acceptă limbajul $L = \{a^ib^j \mid i, j \geq 0\}$.



Rezolvare: Trebuie să demonstrăm că $L(A) = L$. Arătăm mai întâi incluziunea $L \subseteq L(A)$. Să luăm cîteva cuvinte: pentru $i = j = 0$, $a^ib^j = \lambda$, $\delta(s_0, \lambda) = s_0 \in F$, deci $\lambda \in L(A)$. Pentru $i = j = 1$, $a^ib^j = ab$, $\delta(s_0, ab) = \delta(\delta(s_0, a), b) = \delta(s_0, b) = s_1 \in F$, deci $\lambda \in L(A)$. Pentru $i, j \geq 0$ arbitrari, avem:

$$\delta(s_0, a^ib^j) = \delta(\delta(s_0, a^i), b^j) = \delta(s_0, b^j) = \begin{cases} s_0 \in F, & \text{dacă } j = 0 \\ s_1 \in F, & \text{în caz contrar} \end{cases},$$

deci $a^ib^j \in L(A)$.

Invers, să arătăm incluziunea $L(A) \subseteq L$. Fie $p \in L(A)$. Atunci $p \in \{a, b\}^*$ şi $\delta(s_0, p) \in F$. Vom arăta că $p = a^ib^j$ cu $i, j \geq 0$. Pentru aceasta, considerăm mai multe cazuri posibile:

1. $p = \lambda$, atunci $\delta(s_0, \lambda) = s_0 \in F$ şi $p = \lambda = a^0b^0$.

2. $p = a^i$, cu $i \geq 1$, atunci $\delta(s_0, a^i) = s_0 \in F$ şi $p = a^ib^0$.

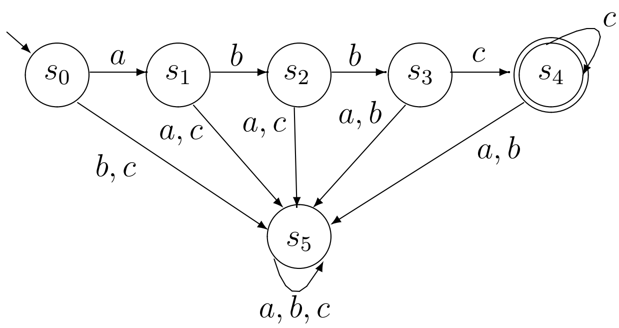
3. $p = b^j$, cu $j \geq 1$, atunci $\delta(s_0, b^j) = s_1 \in F$ şi $p = a^0 b^j$.

4. $p = a^i b^j$, cu $i, j \geq 1$, atunci $\delta(s_0, a^i b^j) = \delta(\delta(s_0, a^i), b^j) = \delta(s_0, b^j) = s_1 \in F$ şi $p = a^i b^j$.

5. $p = a^i b^j au$, cu $i \geq 0, j \geq 1$ şi $u \in \{a, b\}^*$. Atunci $\delta(s_0, a^i b^j au) = \delta(\delta(s_0, a^i), b^j au) = \delta(s_0, b^j au) = \delta(\delta(s_0, b^j), au) = \delta(s_1, au) = \delta(\delta(s_1, a), u) = \delta(s_2, u) = s_2 \notin F$ şi prin urmare acest caz este imposibil, deoarece contrazice faptul că $p \in L(A)$.

Deci cuvintele din $L(A)$ sunt de forma $a^i b^j$, cu $i, j \geq 0$.

2) Să se arate că automatul $A = (\{s_0, s_1, s_2, s_3, s_4, s_5\}, \{a, b, c\}, \delta, s_0, \{s_4\})$, cu δ definită prin graful de tranziţie din figura următoare, recunoaşte limbajul $L = \{ab^2c^n \mid n \geq 1\}$.



Rezolvare: Trebuie să demonstrăm că $L(A) = L$. Arătăm mai întâi incluziunea $L \subseteq L(A)$. Pentru $n \geq 1$ arbitrar, avem: $\delta(s_0, ab^2c^n) = \delta(\delta(s_0, a), b^2c^n) = \delta(s_1, b^2c^n) = \delta(\delta(s_1, b), bc^n) = \delta(s_2, bc^n) = \delta(\delta(s_2, b), c^n) = \delta(s_3, c^n) = \delta(\delta(s_3, c), c^{n-1}) = \delta(s_4, c^{n-1}) = s_4 \in F$, deci $ab^2c^n \in L(A)$.

Invers, să arătăm incluziunea $L(A) \subseteq L$. Fie $w \in L(A)$. Atunci $w \in \{a, b, c\}^*$ şi $\delta(s_0, w) \in F$. Vom arăta că $w = ab^2c^n$ cu $n \geq 1$. Pentru aceasta, convenim să notăm cu $first(w)$ primul simbol al cuvântului w, dacă $w \neq \lambda$. Considerăm mai multe cazuri:

1. $w = \lambda$, atunci $\delta(s_0, \lambda) = s_0 \notin F$, ceea ce contrazice ipoteza, deci acest caz nu este posibil.

2. dacă $first(w) = b$ sau $first(w) = c$, atunci $w = bw_1$ sau $w = cw_1$ şi $\delta(s_0, w) = \delta(s_5, w_1) = s_5 \notin F$, ceea ce contrazice ipoteza. Deci w nu poate începe cu b sau c, prin urmare avem $w = aw_1$.

3. $w = aw_1$, atunci $\delta(s_0, aw_1) = \delta(\delta(s_0, a), w_1) = \delta(s_1, w_1)$. Acum trebuie să ţinem cont de faptul că trebuie să ajungem în starea finală s_4.

   – Dacă $first(w_1) = a$ sau $first(w_1) = c$, atunci $\delta(s_1, w_1) = s_5 \notin F$, ceea ce contrazice ipoteza. Deci $first(w_1) = b$, prin urmare $w_1 = bw_2$ şi avem $\delta(s_1, bw_2) = \delta(\delta(s_1, b), w_2) = \delta(s_2, w_2)$.

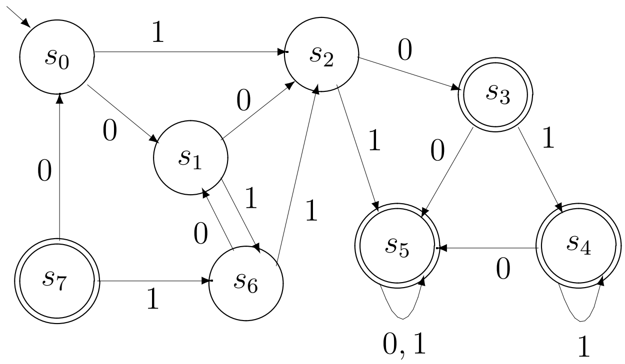
   – Dacă $first(w_2) = a$ sau $first(w_2) = c$, atunci $\delta(s_2, w_2) = s_5 \notin F$, ceea ce contrazice ipoteza. Deci $first(w_2) = b$, prin urmare $w_2 = bw_3$ şi avem $\delta(s_2, bw_3) = \delta(\delta(s_2, b), w_3) = \delta(s_3, w_3)$.

   – Dacă $first(w_3) = a$ sau $first(w_3) = b$, atunci $\delta(s_3, w_3) = s_5 \notin F$, ceea ce contrazice ipoteza. Deci $first(w_3) = c$, prin urmare $w_3 = cw_4$ şi avem $\delta(s_3, cw_4) = \delta(\delta(s_3, c), w_4) = \delta(s_4, w_4)$.

   – Cuvântul w_4 trebuie să conţină numai simboli c, deoarece în caz contrar, dacă ar conţine măcar un a sau b, atunci automatul ar trece în starea s_5 pe care nu ar mai părăsi-o, deci nu am putea ajunge în final la s_4. Prin urmare, $w_4 = c^k$, cu $k \geq 0$, şi avem că $w = aw_1 = abw_2 = abbw_3 = abbcw_4 = ab^2c^{k+1}$.

Deci cuvintele din $L(A)$ sunt de forma $w = ab^2c^n$ cu $n \geq 1$.

3) Să se construiască automatul minimal corespunzător automatului $A = (\{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}, \{0, 1\}, \delta, s_0, \{s_3, s_4, s_5, s_7\})$, cu δ definită prin graful de tranziţie din figura următoare.



Rezolvare: Pentru a construi automatul minimal parcurgem următorii paşi:
1) Determinăm stările accesibile utilizând algoritmul ACC şi găsim automatul redus echivalent cu A.
2) Calculăm clasele de echivalenţă ale relaţiei ρ în automatul redus.
3) Construim automatul minimal echivalent cu automatul A, $A_{\min} = (S'/\rho, \Sigma, \delta_\rho, [s_0]_\rho, \{[s]_\rho, s \in F'\})$.

Să parcurgem paşii de mai sus:
1) $G_0 = \{s_0\}$, $G_1 = G_0 \cup \delta(G_0, \Sigma) = \{s_0\} \cup \{s_1, s_2\} = \{s_0, s_1, s_2\}$; cum $G_0 \neq G_1$, continuăm calculul mulţimilor G_i. $G_2 = G_1 \cup \delta(G_1, \Sigma) = \{s_0, s_1, s_2\} \cup \{s_3, s_5, s_6\} = \{s_0, s_1, s_2, s_3, s_5, s_6\}$, $G_1 \neq G_2$. $G_3 = G_2 \cup \delta(G_2, \Sigma) = \{s_0, s_1, s_2, s_3, s_5, s_6\} \cup \{s_4, s_5\} = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$, $G_2 \neq G_3$. $G_4 = G_3 \cup \delta(G_3, \Sigma) = G_3$ şi deci algoritmul ACC se opreşte cu $S' = G_3 = G_4$.
Aşadar, stările accesibile sunt $S' = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$, iar automatul redus este $A' = (\{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}, \{0, 1\}, \delta', s_0, \{s_3, s_4, s_5\})$, unde δ' este restricţia lui δ la S'.
2) Să construim clasele de echivalenţă ale relaţiei ρ pentru A'.

Clasele de echivalenţă pentru relaţia $\overset{0}{\equiv}$ sunt F' şi $S' - F'$, deoarece singurul cuvânt de lungime 0 este λ şi $\delta(s, \lambda) = s$, oricare ar fi $s \in S$, aşadar avem că $s_1 \overset{0}{\equiv} s_2$ dacă şi numai dacă $\Psi(\delta(s_1, \lambda)) = \Psi(\delta(s_2, \lambda))$, dacă şi numai dacă $\Psi(s_1) = \Psi(s_2)$, dacă şi numai dacă $s_1, s_2 \in F'$ sau $s_1, s_2 \notin F'$. Deci $S'/\overset{0}{\equiv} = \{F', S' - F'\} = \{\{s_3, s_4, s_5\}, \{s_0, s_1, s_2, s_6\}\}$.

Pentru a determina clasele de echivalenţă pentru relaţia $\overset{1}{\equiv}$ aplicăm lema: “ $s_1 \overset{k}{\equiv} s_2$ *dacă şi numai dacă* $s_1 \overset{k-1}{\equiv} s_2$ *şi* $\delta(s_1, i) \overset{k-1}{\equiv} \delta(s_2, i)$, $\forall i \in \Sigma$ ”. Deci pentru ca două stări să fie în aceeaşi clasă faţă de relaţia $\overset{k}{\equiv}$, ele trebuie să fi fost în aceeaşi clasă faţă de relaţia $\overset{k-1}{\equiv}$ şi, în plus, $\delta(s_1, i)$ şi $\delta(s_2, i)$ să fi fost şi ele în aceeaşi clasă faţă de relaţia $\overset{k-1}{\equiv}$, şi aceasta pentru toţi simbolii i ai alfabetului de intrare.

Pentru $\overset{1}{\equiv}$, luăm deci fiecare clasă pentru $\overset{0}{\equiv}$ şi verificăm dacă este îndeplinită condiţia a doua.

I. Pentru clasa $\{s_3, s_4, s_5\}$ avem:

1) Pentru perechea $\{s_3, s_4\}$ avem $\delta(s_3, 0) = s_5$, $\delta(s_4, 0) = s_5$, şi $\delta(s_3, 1) = s_4$, $\delta(s_4, 1) = s_4$. Deoarece $s_5 \overset{0}{\equiv} s_5$ şi $s_4 \overset{0}{\equiv} s_4$, rezultă că $s_3 \overset{1}{\equiv} s_4$.

2) Pentru perechea $\{s_3, s_5\}$ avem $\delta(s_3, 0) = s_5$, $\delta(s_5, 0) = s_5$, şi $\delta(s_3, 1) = s_4$, $\delta(s_5, 1) = s_5$. Deoarece $s_5 \overset{0}{\equiv} s_5$ şi $s_4 \overset{0}{\equiv} s_5$, rezultă că $s_3 \overset{1}{\equiv} s_5$.

Cum $\overset{1}{\equiv}$ este o relaţie de echivalenţă, rezultă că $\{s_3, s_4, s_5\}$ formează o clasă de echivalenţă faţă de relaţia $\overset{1}{\equiv}$.

II. Pentru clasa $\{s_0, s_1, s_2, s_6\}$ avem:

1) Pentru perechea $\{s_0, s_1\}$ avem $\delta(s_0, 0) = s_1$, $\delta(s_1, 0) = s_2$, şi $\delta(s_0, 1) = s_2$, $\delta(s_1, 1) = s_6$. Deoarece $s_1 \overset{0}{\equiv} s_2$ şi $s_2 \overset{0}{\equiv} s_6$, rezultă că $s_0 \overset{1}{\equiv} s_1$.

2) Pentru perechea $\{s_0, s_2\}$ avem $\delta(s_0, 0) = s_1$, $\delta(s_2, 0) = s_3$, şi $\delta(s_0, 1) = s_2$, $\delta(s_2, 1) = s_5$. Deoarece $s_1 \overset{0}{\not\equiv} s_3$, rezultă că $s_0 \overset{1}{\not\equiv} s_2$.

3) Pentru perechea $\{s_0, s_6\}$ avem $\delta(s_0, 0) = s_1$, $\delta(s_6, 0) = s_1$, şi $\delta(s_0, 1) = s_2$, $\delta(s_6, 1) = s_2$. Deoarece $s_1 \overset{0}{\equiv} s_1$ şi $s_2 \overset{0}{\equiv} s_2$, rezultă că $s_0 \overset{1}{\equiv} s_6$.

Cum $\overset{1}{\equiv}$ este o relaţie de echivalenţă, rezultă că $\{s_0, s_1, s_6\}$ formează o clasă de echivalenţă faţă de relaţia $\overset{1}{\equiv}$, iar $\{s_2\}$ formează separat o altă clasă faţă de $\overset{1}{\equiv}$.

Deci pentru $\stackrel{1}{\equiv}$ clasele de echivalenţă sunt:
$S'/\stackrel{1}{\equiv} = \{\{s_3, s_4, s_5\}, \{s_0, s_1, s_6\}, \{s_2\}\}$.

Similar, pentru $\stackrel{2}{\equiv}$, luăm fiecare clasă pentru $\stackrel{1}{\equiv}$ şi verificăm dacă este îndeplinită condiţia a doua din lema amintită mai sus.

III. Pentru clasa $\{s_3, s_4, s_5\}$ se constată, făcând calculele ca la etapa precedentă, că $\{s_3, s_4, s_5\}$ formează o clasă de echivalenţă faţă de relaţia $\stackrel{2}{\equiv}$.

IV. Pentru clasa $\{s_0, s_1, s_6\}$ avem:

1) Pentru perechea $\{s_0, s_1\}$ avem $\delta(s_0, 0) = s_1$, $\delta(s_1, 0) = s_2$, şi $\delta(s_0, 1) = s_2$, $\delta(s_1, 1) = s_6$. Deoarece $s_1 \stackrel{1}{\not\equiv} s_2$, rezultă că $s_0 \stackrel{2}{\not\equiv} s_1$.

2) Pentru perechea $\{s_0, s_6\}$ avem $\delta(s_0, 0) = s_1$, $\delta(s_6, 0) = s_1$, şi $\delta(s_0, 1) = s_2$, $\delta(s_6, 1) = s_2$. Deoarece $s_1 \stackrel{1}{\equiv} s_1$ şi $s_2 \stackrel{1}{\equiv} s_2$, rezultă că $s_0 \stackrel{2}{\equiv} s_6$.

Deci pentru $\stackrel{2}{\equiv}$ clasele de echivalenţă sunt:
$S'/\stackrel{2}{\equiv} = \{\{s_3, s_4, s_5\}, \{s_0, s_6\}, \{s_1\}, \{s_2\}\}$.

Apoi, pentru $\stackrel{3}{\equiv}$, luăm fiecare clasă pentru $\stackrel{2}{\equiv}$ şi verificăm dacă este îndeplinită condiţia a doua din lema amintită mai sus.

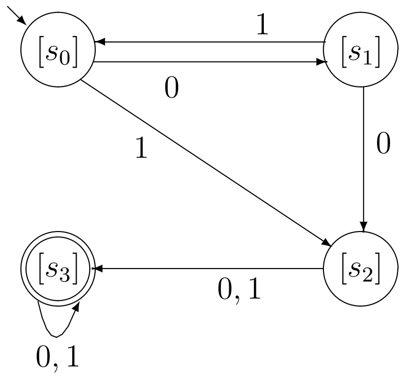
Făcând calculele ca la etapa precedentă, se constată că clasele $\{s_3, s_4, s_5\}$, $\{s_1\}$, şi $\{s_2\}$ rămîn neschimbate. Pentru perechea $\{s_0, s_6\}$ din calculele de mai sus se observă că $s_0 \stackrel{3}{\equiv} s_6$, şi prin urmare $\stackrel{3}{\equiv} = \stackrel{2}{\equiv}$.

Deci clasele de echivalenţă ale relaţiei ρ pentru A' sunt:

$$S'/\rho = \{\{s_0, s_6\}, \{s_1\}, \{s_2\}, \{s_3, s_4, s_5\}\} = \{[s_0], [s_1], [s_2], [s_3]\},$$

unde prin $[s_0] = \{s_0, s_6\}$ am notat clasa lui s_0, prin $[s_1] = \{s_1\}$ clasa lui s_1, ş.a.m.d.

3) În final obţinem că automatul minimal echivalent cu automatul A este următorul: $A_{\min} = (\{[s_0], [s_1], [s_2], [s_3]\}, \{0, 1\}, \delta_\rho, [s_0], \{[s_3]\})$, cu δ_ρ definită prin graful de tranziţie din figura următoare.



4) Să se arate că limbajul $L = \{a^n b^n \mid n \geq 1\}$ nu este limbaj regulat (de tip 3).

Rezolvare: Presupunem prin reducere la absurd că L este limbaj regulat. Atunci, aplicând lema Bar-Hillel (lema de pompare), rezultă că există un număr $k \in N$ astfel încât cuvintele $w \in L$ cu $|w| \geq k$ au proprietăţile din lemă.

Alegem un cuvânt $w = a^n b^n$ cu $|w| = 2n \geq k$. Considerăm toate cazurile posibile de descompunere a lui w şi vom arăta că în nici un caz nu sunt satisfăcute toate proprietăţile din lemă. Vom face raţtionamentul după cuvântul y (cel care se pompează).

Cazul I: $w = xyz$ cu $y = a^l b^t$, $l, t \geq 1$, şi $x = a^{n-l}$, $z = b^{n-t}$.
Atunci, ştim din lemă că $xy^i z \in L$, $\forall\, i \geq 0$. Dar, luând $i = 2$, avem că $xy^2z = a^{n-l}a^l b^t a^l b^t b^{n-t} = a^n b^t a^l b^n \notin L$, din cauză că apare a după b în acest cuvânt (întrucît $l, t \geq 1$).

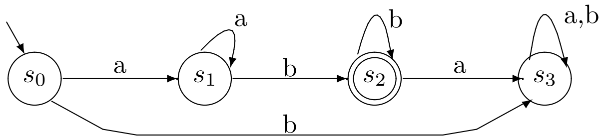
Cazul II: $y = a^l$, $l \geq 1$, şi $x = a^{n-l-t}$, $z = a^t b^n$, cu $t \geq 0$.
Luăm $i = 0$ şi avem că $xy^0z = xz = a^{n-l-t}a^t b^n = a^{n-l}b^n \notin L$, din cauză că $n - l < n$.

Cazul III: $y = b^l$, $l \geq 1$, şi $x = a^n b^t$, $z = b^{n-l-t}$, cu $t \geq 0$.
Luăm $i = 0$ şi avem că $xy^0z = xz = a^n b^t b^{n-l-t} = a^n b^{n-l} \notin L$, din cauză că $n > n - l$.

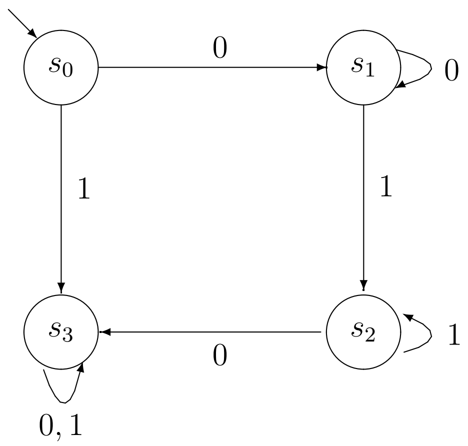
Acestea fiind singurele posibilităţi de descompunere $w = xyz$, rezultă că presupunerea pe care am făcut-o, aceea că limbajul L ar fi regulat, este falsă, deci L nu este limbaj regulat.

Tema de control (2)

1) Să se arate că automatul $A = (\{s_0, s_1, s_2, s_3\}, \{a, b\}, \delta, s_0, \{s_2\})$, cu δ definită prin graful de tranziţie din figura următoare, recunoaşte limbajul $L = \{a^n b^m \mid n, m \geq 1\}$.

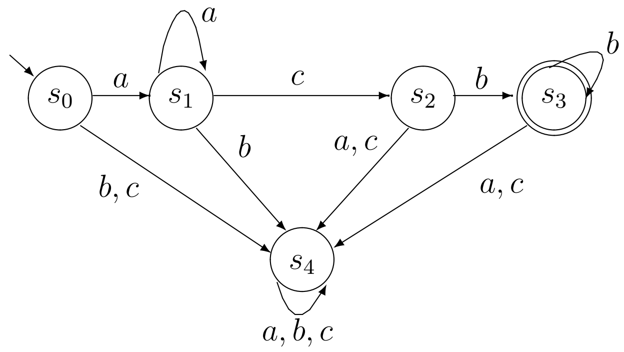


2) Se consideră automatul $A = (\{s_0, s_1, s_2, s_3\}, \{0, 1\}, \delta, s_0, F)$, cu δ definită prin graful de tranziţie din figura următoare.



a) Să se găsească $L(A)$ pentru $F = \{s_0\}$;
b) Să se găsească $L(A)$ pentru $F = \{s_1\}$;
c) Să se găsească $L(A)$ pentru $F = \{s_2\}$;
d) Să se găsească $L(A)$ pentru $F = \{s_3\}$.

3) Să se arate că automatul $A = (\{s_0, s_1, s_2, s_3, s_4\}, \{a, b, c\}, \delta, s_0, \{s_3\})$, cu δ definită prin graful de tranziţie din figura următoare, recunoaşte limbajul $L = \{a^n c b^m \mid n, m \geq 1\}$.



4) Să se arate că limbajul $L_m = \{a^n b^{n+m} \mid n \geq 1\}$, cu $m \geq 1$ fixat, nu este limbaj regulat (de tip 3).

Tema 3

Limbaje independente de context şi automate pushdown

1 Automate pushdown nedeterministe

În capitolul precedent am văzut că familia $\mathcal{L}_3$ coincide cu familia limbajelor acceptate de automate deterministe sau nedeterministe. Pentru limbajele din clasa $\mathcal{L}_2$ există un alt tip de automat care să accepte limbajele din $\mathcal{L}_2$ şi anume automate pushdown nedeterministe.

Un model pentru un automat pushdown nedeterminist este dat în Figura 3.1.1.

Modelul prezentat mai sus constă dintr-o *bandă de intrare* pe care sunt plasate simboluri dintr-un alfabet de intrare, Σ, o memorie numită *memorie pushdown* în care sunt plasate simboluri dintr-un alfabet Γ şi o *unitate de comandă*. Unitatea de comandă este prevăzută cu două capete de citire, unul pentru banda de intrare, care se poate deplasa spre dreapta, şi unul pentru memoria pushdown care vizează numai primul simbol din memorie. Unitatea de comandă care se află într-o stare s şi vizează pe banda de intrare simbolul a_i şi în memoria pushdown simbolul z_1 va trece într-o nouă stare s', capul de citire de pe banda de intrare se va deplasa cu o locaţie spre dreapta sau rămâne pe loc, iar simbolul z_1 din memoria pushdown se înlocuieşte

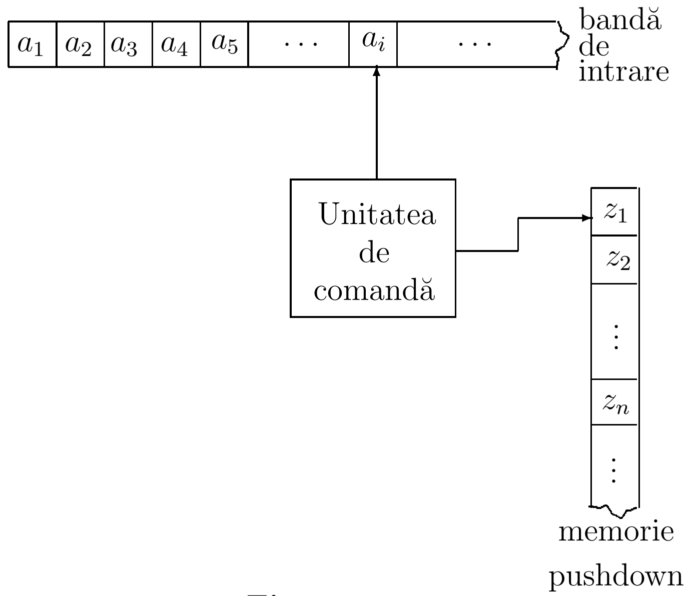


Fig. 3.1.1.

cu un cuvânt α peste alfabetul Γ. Înlocuirea lui z_1 cu α se face cu împingerea în jos a simbolurilor $z_2, \ldots, z_n$, dacă $|\alpha| > 1$, astfel că α se plasează în capul stivei, de aici denumirea de pushdown (cu toate că dacă $|\alpha| = 0$ simbolurile din memoria pushdown urcă, iar când $|\alpha| = 1$ aceste simboluri rămân pe loc).

Definiţia 3.1.1. Un automat pushdown nedeterminist este un 7-uplu $P = (S, \Sigma, \Gamma, \delta, s_0, z_0, F)$, unde:

1. S este o mulţime finită şi nevidă, numită *mulţimea stărilor automatului pushdown*;

2. Σ este o mulţime finită şi nevidă, numită *alfabetul de intrare*;

3. Γ este o mulţime finită şi nevidă numită *alfabetul pushdown*;

4. δ este o funcţie $\delta : S \times (\Sigma \cup \{\lambda\}) \times \Gamma \to \mathcal{P}(S \times \Gamma^*)$, numită *funcţie de tranziţie*;

5. $s_0 \in S$ se numeşte *starea iniţială a automatului pushdown*;

6. $z_0 \in \Gamma$ se numeşte *simbolul pushdown iniţial*;

7. $F \subseteq S$ se numeşte *mulţimea stărilor finale.*

Definiţia 3.1.2. Se numeşte *configuraţie* a unui automat pushdown o tripletă $(s, w, \alpha) \in S \times \Sigma^* \times \Gamma^*$.

Pe modelul dat, cele trei elemente ale unei configuraţii le putem interpreta în felul următor:

1. $s \in S$ reprezintă starea în care se află unitatea de comandă în momentul considerat;

2. $w \in \Sigma^*$ reprezintă partea necitită de pe banda de intrare, adică partea delimitată la stânga de capul de citire şi la dreapta de ultimul simbol diferit de blanc, inclusiv;

3. α reprezintă conţinutul memoriei pushdown, primul caracter al lui α este cel vizat. În cazul $\alpha = \lambda$ spunem că memoria pushdown este vidă.

O configuraţie reprezintă starea automatului la un moment dat. Pe mulţimea configuraţiilor $\mathcal{C}$, definim o relaţie binară $\underset{P}{\vdash} \subseteq \mathcal{C} \times \mathcal{C}$ care defineşte mişcarea automatului pushdown.

Definiţia 3.1.3. Configuraţia $(s, aw, z\alpha)$ este în relaţia $\underset{P}{\vdash}$ cu configuraţia $(s', w, \gamma\alpha)$, scris $(s, aw, z\alpha)\underset{P}{\vdash}(s', w, \gamma\alpha)$, dacă $(s', \gamma) \in \delta(s, a, z)$, unde $s, s' \in S$, $a \in \Sigma \cup \{\lambda\}$, $z \in \Gamma$, $w \in \Sigma^*$ şi $\alpha, \gamma \in \Gamma^*$.

Observăm că pentru a fi posibilă mişcarea automatului **dintr-o configuraţie în altă configuraţie, memoria pushdown trebuie să fie nevidă**. Dacă memoria pushdown este vidă, mişcarea automatului pushdown nu mai este posibilă.

Considerăm închiderea reflexivă şi tranzitivă a relaţiei $\underset{P}{\vdash}$ pe care o notăm cu $\underset{P}{\overset{*}{\vdash}}$. Atunci când automatul pushdown este subînţeles, noi vom renunţa la indicele P şi vom scrie $\overset{*}{\vdash}$ în loc de $\underset{P}{\overset{*}{\vdash}}$.

Definiţia 3.1.4. Limbajul acceptat de un automat pushdown cu stări finale $P = (S, \Sigma, \Gamma, \delta, s_0, z_0, F)$ este

$$L(P) = \{w \mid w \in \Sigma^*, \exists (s_0, w, z_0) \overset{*}{\underset{P}{\vdash}} (s, \lambda, \alpha), s \in F, \alpha \in \Gamma^*\}$$

Exemplu. Fie $P = (\{s_0, s_1, s_2, s_3\}, \{0, 1\}, \{0, z\}, \delta, s_0, z, \{s_3\})$ unde:

1. $\delta(s_0, 0, z) = \{(s_1, 0z)\}$
2. $\delta(s_1, 0, 0) = \{(s_1, 00)\}$
3. $\delta(s_1, 1, 0) = \{(s_2, \lambda)\}$
4. $\delta(s_2, 1, 0) = \{(s_2, \lambda)\}$
5. $\delta(s_2, \lambda, z) = \{(s_3, \lambda)\}$
6. $\delta(s, a, \gamma) = \emptyset$ în celelalte cazuri.

Să vedem dacă cuvântul 0011 aparţine limbajului $L(P)$. Considerăm secvenţa de tranziţii:
$(s_0, 0011, z) \vdash (s_1, 011, 0z) \vdash (s_1, 11, 00z) \vdash (s_2, 1, 0z) \vdash (s_2, \lambda, z) \vdash (s_3, \lambda, \lambda)$; cum s_3 este starea finală, rezultă că $0011 \in L(P)$.

În Definiţia 3.1.4 observăm că pentru ca un cuvânt w să aparţină limbajului trebuie ca să existe un şir de tranziţii din configuraţia iniţială (s_0, w, z) într-o configuraţie (s, λ, α) cu $s \in F$ şi $\alpha \in \Gamma^*$. Putem defini un nou limbaj acceptat de un automat pushdown în care să slăbim condiţia ca starea s să aparţină mulţimii F şi vom lăsa această stare arbitrară din S, dar vom întări condiţia $\alpha \in \Gamma^*$, vom cere ca $\alpha = \lambda$.

Definiţia 3.1.5. Fie $P = (S, \Sigma, \Gamma, \delta, s_0, z_0, \emptyset)$ un automat pushdown. Vom numi *limbaj acceptat de automatul pushdown P cu memorie pushdown vidă*, următorul limbaj:

$$L_\lambda(P) = \{w \mid w \in \Sigma^*, \exists (s_0, w, z_0) \overset{*}{\vdash} (s, \lambda, \lambda), s \in S\}$$

În exemplul dat înainte se observă că 0011 este un cuvânt acceptat de automatul pushdown cu memorie pushdown vidă.

Problema care se pune este ce legătură este între cele două familii de limbaje? În următoarele două teoreme vom arăta că cele două familii de limbaje sunt egale.

Teorema 3.1.6. *Fie* $P = (S, \Sigma, \Gamma, \delta, s_0, z_0, \emptyset)$ *un automat pushdown cu memorie pushdown vidă. Există un automat pushdown* P' *astfel încât* $L(P') = L_\lambda(P)$.

Demonstraţie. Fie $P' = (S \cup \{s'_0, s_f\}, \Sigma, \Gamma \cup \{x\}, \delta', s'_0, x, \{s_f\})$, unde $s'_0, s_f \notin S$, $x \notin \Gamma$ şi funcţia δ' este definită prin:

1. $\delta'(s'_0, \lambda, x) = \{(s_0, z_0x)\}$;
2. $\delta'(s, a, z) = \delta(s, a, z)$ pentru $s \in S$, $a \in \Sigma \cup \{\lambda\}$ şi $z \in \Gamma$;
3. $\delta'(s, \lambda, x) = \{(s_f, \lambda)\}$ pentru $s \in S$;
4. $\delta'(s, a, z) = \emptyset$ în celelalte cazuri.

Egalitatea $L(P') = L_\lambda(P)$ o vom demonstra prin dublă incluziune.

a) Să demonstrăm $\boldsymbol{L_\lambda(P) \subseteq L(P')}$. Fie $w \in L_\lambda(P)$; atunci există $s \in S$ astfel încât avem

(1) $$(s_0, w, z_0) \overset{*}{\underset{P}{\vdash}} (s, \lambda, \lambda)$$

Folosind definiţia lui δ' şi anume punctul 1), avem:

(2) $$(s'_0, w, x) \underset{P'}{\vdash} (s_0, w, z_0x).$$

Folosind punctul 2) din definiţia lui δ', avem:

(1′) $$(s_0, w, z_0) \overset{*}{\underset{P'}{\vdash}} (s, \lambda, \lambda)$$

Folosind relaţiile (2), (1') şi punctul 3) din definiţia lui δ', putem scrie următorul şir de tranziţii în P':
$(s'_0, w, x) \underset{P'}{\vdash} (s_0, w, z_0x) \overset{*}{\underset{P'}{\vdash}} (s, \lambda, x) \underset{P'}{\vdash} (s_f, \lambda, \lambda)$,
deci $(s'_0, w, x) \overset{*}{\underset{P'}{\vdash}} (s_f, \lambda, \lambda)$, ceea ce înseamnă $w \in L(P')$.

b) Să demonstrăm incluziunea inversă, $\boldsymbol{L(P') \subseteq L_\lambda(P)}$. Fie $w \in L(P')$; atunci există un $\alpha \in (\Gamma \cup \{x\})^*$ astfel încât

$$(3) \qquad (s'_0, w, x) \overset{*}{\underset{P'}{\vdash}} (s_f, \lambda, \alpha)$$

Din definiţia lui δ' rezultă că primul pas al tranziţiilor din (3) se face după punctul 1) şi anume avem:

$$(3') \qquad (s'_0, w, x) \underset{P'}{\vdash} (s_0, w, z_0x) \overset{*}{\underset{P'}{\vdash}} (s_f, \lambda, \alpha)$$

Dar modul de definire al funcţiei δ' ne arată că automatul P' poate intra în starea s_f numai conform punctului 3) din definiţia lui δ', deci ultimul pas al tranziţiilor din (3') trebuie să fie după cum urmează:

$$(3'') \qquad (s'_0, w, x) \underset{P'}{\vdash} (s_0, w, z_0x) \overset{*}{\underset{P'}{\vdash}} (s, \lambda, x) \underset{P'}{\vdash} (s_f, \lambda, \lambda)$$

şi în plus, pe porţiunea $(s_0, w, z_0x) \vdash^*_{P'} (s, \lambda, x)$ se aplică numai punctul 2) din definiţia lui δ', deci aceste tranziţii le avem şi în automatul pushdown P:

$$(4) \qquad (s_0, w, z_0) \overset{*}{\underset{P}{\vdash}} (s, \lambda, \lambda),$$

ceea ce arată că $w \in L_\lambda(P)$. ■

Cu aceasta am arătat că orice limbaj acceptat de un automat pushdown cu memorie pushdown vidă este acceptat de un automat pushdown cu stări finale. Următoarea teoremă stabileşte incluziunea inversă.

Acum să demonstrăm incluziunea inversă şi anume că orice limbaj acceptat de un APD cu stări finale poate fi acceptat de un APD cu memoria vidă.

Teorema 3.1.7. Un limbaj L acceptat de un automat pushdown cu stări finale este acceptat şi de un automat pushdown cu memorie pushdown vidă.

Demonstraţie. Fie $P = (S, \sum, \Gamma, \delta, s_0, z_0, F)$ cu $L = l(P)$. Fie automatul pushdown cu memorie vidă: $P' = (S \cup \{s'_0, s'\}, \sum, \Gamma \cup \{x\}$, $\delta', s'_0, x, \phi)$, unde $s'_0, s' \notin S, x \notin \Gamma$ şi δ' este definit prin:

1. $\delta'(s'_0, \lambda, x) = \{(s_0, z_0x)\}$;

2. (a) $\delta'(s, a, z) = \delta(s, a, z)$ pentru $s \in S, a \in \sum$ şi $z \in \Gamma$;

   (b) $\delta'(s, \lambda, z) = \delta(s, \lambda, z)$ pentru $s \in S \setminus F, z \in \Gamma$;

   (c) $\delta'(s, \lambda, z) = \delta(s, \lambda, z) \cup \{(s', \lambda)$ pentru $s \in F$ şi $z \in \Gamma$;

3. $\delta'(s, \lambda, x) = \{(s', \lambda)\}$ pentru $s \in F$;

4. $\delta'(s', \lambda, z) = \{(s', \lambda)\}$ pentru $z \in \Gamma \cup \{x\}$;

5. $\delta'(s, a, z) = \emptyset$ în celelalte cazuri.

Se poate arăta prin dublă incluziune că $L(P) = L_\lambda(P')$.

Cele două teoreme demonstrează că $\mathcal{L}^f_{APD} = \mathcal{L}^\lambda_{APD}$, adică familia limbajelor acceptate de automatele pushdown ca stări finale este egală cu familia limbajelor acceptate de automatele pushdown cu memorie vidă, de aceea vom nota această familie cu $\mathcal{L}_{APD}$, adică familia limbajelor acceptate de automatele pushdown. Vom arăta mai departe că $\mathcal{L}_{APD} = \mathcal{L}_2$. Această egalitate o demonstrăm în următoarele două teoreme.

Teorema 3.1.8. Fie L un limbaj independent de contact, adică $L \in \mathcal{L}_2$. Există un automat pushdown P' astfel încât $L = L_\lambda(P')$.

Teorema 3.1.9.*Fie L un limbaj independent de context. Există un automat pushdown P' astfel încât $L = L_\lambda(P')$.*

Demonstraţie. Din $L \in \mathcal{L}_2$, rezultă că există o gramatică de tipul doi $G = (V_N, V_T, x_0, P)$ astfel încât $L = L(G)$. Considerăm următorul automat pushdown $P' = (\{s\}, V_T, V_N \cup V_T, \delta, s, x_0, \emptyset)$. Funcţia de tranziţie δ este:

1. $\delta(s, \lambda, X) = \{(s, \alpha) \mid X \to \alpha \in P\}$;

2. $\delta(s, a, a) = \{(s, \lambda)\},\ \forall a \in \Sigma$;

3. $\delta(s, a, z) = \emptyset$ în celelalte cazuri.

Pentru a demonstra că $L(G) = L_\lambda(P')$ vom demonstra în prealabil următoarea proprietate:

(1) $X \overset{m*}{\underset{G}{\Rightarrow}} w$, $w \in V_T^*$, $X \in V_N$ dacă şi numai dacă $(s, w, X) \overset{n}{\underset{P'}{\vdash}} (s, \lambda, \lambda)$

pentru $m \geq 1$ şi $n \geq 1$ convenabil ales; m şi n reprezintă numărul de paşi în derivare sau în tranziţie, respectiv.

Implicaţia în sens direct o demonstrăm prin inducţie după m.

Pentru $m = 1$, fie $w = i_1 \ldots i_k$, $k \geq 0$. Atunci avem $X \underset{G}{\Rightarrow} i_1 \ldots i_k$, deci există regula $X \to i_1 \ldots i_k \in P$ şi după 1) şi 2) putem scrie următorul şir de tranziţii:

$$(s, i_1 \ldots i_k, X) \underset{P'}{\vdash} (s, i_1 \ldots i_k, i_1 \ldots i_k) \overset{*}{\underset{P'}{\vdash}} (s, \lambda, \lambda).$$

Presupunem implicaţia directă (1) adevărată pentru toate derivările de lungime cel mult $l - 1$ şi fie $X \overset{l*}{\underset{G}{\Rightarrow}} w$, cu $l > 1$. Din această derivare vom pune în evidenţă primul pas al derivării:

$$A \underset{G}{\Rightarrow} x_1 \ldots x_t \text{ cu } x_i \in V_N \cup V_T,\ 1 \leq i \leq t.$$

Pentru derivarea $x_1 \ldots x_t \overset{l-1*}{\underset{G}{\Longrightarrow}} w$ se aplică teorema de localizare şi obţinem $w = w_1 \ldots w_t$ şi există derivările

(2) $$x_i \overset{l-1*}{\underset{G}{\Rightarrow}} w_i,\ 1 \leq i \leq t.$$

Derivările (2) au lungimea cel mult $l - 1$. Dacă $x_i \in V_N$, atunci, conform pasului inductiv, avem:

(3) $$(s, w_i, x_i) \overset{n_i*}{\underset{P'}{\vdash}} (s, \lambda, \lambda).$$

Dacă $x_i \in V_T$, atunci $x_i = w_i$ şi utilizând punctul 2) din definiţia lui δ, avem:

(3′) $$(s, x_i, x_i) \underset{P'}{\vdash} (s, \lambda, \lambda)$$

Deci din (3) şi (3') avem:

(3″) $$(s, w_i, x_i) \overset{*}{\underset{P'}{\vdash}} (s, \lambda, \lambda),\ 1 \leq i \leq t.$$

Din $X \to x_1 \dots x_k \in P$, avem, conform punctului 1) din definiţia lui δ, următoarea tranziţie:

$$(s, w, X) \underset{P'}{\vdash} (s, w, x_1 \dots x_k) \tag{4}$$

Din (4) şi (3") obţinem:

$$(s, w, X) \underset{P'}{\vdash} (s, w_1 \dots w_k, x_1 \dots x_k) \overset{*}{\underset{P'}{\vdash}} (s, \lambda, \lambda).$$

Cu aceasta, implicaţia directă este demonstrată.

Să demonstrăm implicaţia inversă. Această implicaţie va fi demonstrată prin inducţie după n.

Pentru $n = 1$, avem $(s, w, X) \vdash (s, \lambda, \lambda)$. Deoarece $X \in V_N$ şi $w \in V_T^*$ rezultă că în această tranziţie nu se poate aplica direct punctul 1) din definiţia lui δ, deci există $X \to \lambda \in P$ şi în plus, $w = \lambda$. Din $X \to \lambda \in P$, rezultă că avem $X \underset{G}{\Rightarrow} \lambda$.

Presupunem implicaţia adevărată pentru $n \leq l$ şi să o demonstrăm pentru $n = l + 1$. Fie deci tranziţia

$$(s, w, X) \overset{l+1*}{\underset{P'}{\vdash}} (s, \lambda, \lambda), \text{ cu } l \geq 1. \tag{5}$$

Prima mişcare a lui P' trebuie să fie de forma

$$(s, w, X) \underset{P'}{\vdash} (s, w, x_1 \dots x_k),$$

unde $X \to x_1 \dots x_k \in P$ şi $x_i \in V_N \cup V_T$, $1 \leq i \leq k$.

Dar din configuraţia $(s, w, x_1 \dots x_k)$ trebuie să se ajungă în l paşi în (s, λ, λ) conform relaţiei (5), deci avem:

$$(s, w, x_1 \dots x_k) \overset{l*}{\underset{P'}{\vdash}} (s, \lambda, \lambda) \tag{6}$$

Din (6) rezultă că $w = \alpha_1 \dots \alpha_k$, unde α_1 este format din simbolii lui w care sunt parcurşi până ce x_2 ajunge primul simbol din memoria pushdown, α_2 este format din simbolii lui w de la prima vizare a lui x_2 şi până ce x_3 ajunge primul simbol din memoria pushdown, s.a.m.d. Deci putem scrie:

$$(s, \alpha_i, x_i) \overset{l_i*}{\underset{P'}{\vdash}} (s, \lambda, \lambda), \; 1 \leq i \leq k \tag{7}$$

şi numărul de paşi l_i ai acestor tranziţii este mai mic sau cel mult egal cu l.

Dacă $x_i \in V_T$, se poate aplica numai punctul 2) din definiţia lui δ, ceea ce implică $\alpha_i = x_i$ şi tranziţia este de forma:

$$(7') \qquad (s, x_i, x_i) \vdash (s, \lambda, \lambda)$$

într-un pas.
Dacă $x_i \in V_N$, atunci, aplicând ipoteza inductivă, avem $x_i \overset{*}{\underset{G}{\Rightarrow}} \alpha_i$. Deci în total avem

$$x_1 \dots x_k \overset{*}{\underset{G}{\Rightarrow}} \alpha_1 \dots \alpha_k.$$

În plus, ţinând cont că în P există regula $X \to x_1 \dots x_k$, avem

$$X \underset{G}{\Rightarrow} x_1 \dots x_k \overset{*}{\underset{G}{\Rightarrow}} \alpha_1 \dots \alpha_k = w, \text{ deci } X \overset{*}{\underset{G}{\Rightarrow}} w.$$

Luând $X = x_0$, proprietatea (1) devine: $x_0 \overset{*}{\underset{G}{\Rightarrow}} w$ dacă şi numai dacă $(s, w, x_0) \overset{*}{\underset{P'}{\vdash}} (s, \lambda, \lambda)$ ceea ce înseamnă că $w \in L(G)$ dacă şi numai dacă $w \in L_\lambda(P')$. ■

Să demonstrăm acum că limbajul acceptat de un automat push-down este independent de context.

Teorema 3.1.10. *Fie $R = (S, \Sigma, \Gamma, \delta, s_0, z_0, \emptyset)$ un automat push-down. Limbajul $L_\lambda(R)$ este un limbaj independent de context.*

Demonstraţie. Vom construi o gramatică independentă de context care să genereze limbajul $L_\lambda(R)$. Gramatica G o considerăm de forma $G = (V_N, \Sigma, x_0, P)$ unde:

- $V_N = (S \times \Gamma \times S) \cup \{x_0\}$, unde $x_0 \notin S \times \Gamma \times S$;
- regulile din P ale gramaticii G le definim în felul următor:

1. Pentru $a \in \Sigma \cup \{\lambda\}$ şi $(r, \alpha_1 \dots \alpha_k) \in \delta(s, a, z)$, $k \geq 1$ şi $\alpha_i \in \Gamma$, $1 \leq i \leq k$. Vom considera în G toate regulile de forma:

   $[sz\sigma_k] \to a[r\alpha_1\sigma_1] \dots [\sigma_{k-1}\alpha_k\sigma_k]$

   pentru fiecare secvenţă de stări $\sigma_1 \dots, \sigma_k \in S$.

2. Pentru $a \in \Sigma \cup \{\lambda\}$ şi $(r, \lambda) \in \delta(s, a, z)$ considerăm regula $[szr] \rightarrow a$.

3. Pentru fiecare $s \in S$ considerăm regula $x_0 \rightarrow [s_0 z_0 s]$.

În definirea regulilor de mai sus, am notat un element $(s, z, s') \in S \times \Gamma \times S$ cu $[szs']$. Se observă că gramatica G definită mai sus este independentă de context. Se poate demonstra prin inducţie că:

$$(1) \qquad \text{"}[szr] \overset{*}{\underset{G}{\Rightarrow}} w \text{ dacă şi numai dacă } (s, w, z) \vdash_R^* (r, \lambda, \lambda)\text{"} \quad \text{pentru orice } w \in \Sigma^*.$$

Această dublă implicaţie ne permite să demonstrăm că $L(G) = L_\lambda(R)$.

2 Proprietăţi de închidere pentru familia $\mathcal{L}_2$

În acest paragraf vom studia închiderea familiei $\mathcal{L}_2$ la unele operaţii ca: produs, iteraţie şi substituţie. Reamintim că am demonstrat că $\mathcal{L}_2$ este închisă la reuniune.

Teorema 3.2.1. *Familia $\mathcal{L}_2$ este închisă la operaţia de produs.*

Demonstraţie. Vom demonstra că dacă $L_1, L_2 \in \mathcal{L}_2$, atunci $L_1 \cdot L_2 \in \mathcal{L}_2$. Din $L_1, L_2 \in \mathcal{L}_2$ rezultă că există gramaticile de tipul doi $G_1 = (V_{N_1}, V_{T_1}, x_{01}, P_1)$ şi $G_2 = (V_{N_2}, V_{T_2}, x_{02}, P_2)$ cu $L_1 = L(G_1)$ şi $L_2 = L(G_2)$ şi în plus $V_{N_1} \cap V_{N_2} = \emptyset$.

Considerăm gramatica $G = (V_{N_1} \cup V_{N_2} \cup \{x_0\}, V_{T_1} \cup V_{T_2}, x_0, P_1 \cup P_2 \cup \{x_0 \rightarrow x_{01} x_{02}\})$, unde $x_0 \notin V_{N_1} \cup V_{N_2}$. Gramatica G este de tipul 2 deoarece regulile din P_1 şi P_2 sunt reguli ale gramaticilor de tip doi şi la fel, regula $x_0 \rightarrow x_{01} x_{02}$.

Trebuie să demonstrăm că $L(G) = L_1 \cdot L_2$. Această egalitate o vom demonstra prin dublă incluziune.

Incluziunea $\boldsymbol{L(G) \subseteq L_1 \cdot L_2}$. Fie $p \in L(G)$; rezultă că există derivarea $x_0 \overset{*}{\underset{G}{\Rightarrow}} p$, dar primul pas al derivării este $x_0 \underset{G}{\Rightarrow} x_{01}x_{02} \overset{*}{\underset{G}{\Rightarrow}} p$. Pentru derivarea $x_{01}x_{02} \overset{*}{\underset{G}{\Rightarrow}} p$ aplicăm teorema de localizare şi obţinem $p = p_1p_2$ şi $x_{01} \overset{*}{\underset{G}{\Rightarrow}} p_1$ şi $x_{02} \overset{*}{\underset{G}{\Rightarrow}} p_2$. Din faptul că $V_{N_1} \cap V_{N_2} = \emptyset$, rezultă că derivările precedente din G sunt în G_1 şi G_2, respectiv. Deci avem $x_{01} \overset{*}{\underset{G_1}{\Rightarrow}} p_1$ şi $x_{02} \overset{*}{\underset{G_2}{\Rightarrow}} p_2$, de unde $p_1 \in L_1$ şi $p_2 \in L_2$, ceea ce ne dă că $p = p_1p_2 \in L_1 \cdot L_2$.

Incluziunea inversă $\boldsymbol{L_1 \cdot L_2 \subseteq L(G)}$. Fie $p \in L_1 \cdot L_2 = L(G_1) \cdot L(G_2)$; rezultă că $p = p_1p_2$ cu $p_1 \in L(G_1)$ şi $p_2 \in L(G_2)$, de unde avem $x_{01} \overset{*}{\underset{G_1}{\Rightarrow}} p_1$ şi $x_{02} \overset{*}{\underset{G_2}{\Rightarrow}} p_2$. Dar cum regulile din G_1 şi G_2 sunt şi în G, iar în G avem în plus regula $x_0 \to x_{01}x_{02}$, rezultă că avem derivarea $x_0 \underset{G}{\Rightarrow} x_{01}x_{02} \overset{*}{\underset{G}{\Rightarrow}} p_1p_2$, deci $p \in L(G)$. ■

Lema 3.2.2. *Fie G o gramatică de tipul 2; atunci dacă într-un cuvânt u se pot aplica două sau mai multe reguli, nu are importanţă ordinea de aplicare a acestor reguli; aplicând regulile posibile în orice ordine, se obţine acelaşi cuvânt.*

Demonstraţie. Vom considera cazul când în u se pot aplica două reguli; cazul general rezultă imediat, din cel considerat. Fie $u = u_1xu_2yu_3$ cu $x, y \in V_N$ şi în gramatica G există regulile:

1. $x \to r_1$ şi

2. $y \to r_2$.

Atunci dacă aplicăm cele două reguli în ordinea 1) şi apoi 2), sau 2) şi apoi 1) obţinem:

a) $u \underset{G}{\Rightarrow} u_1r_1u_2yu_3 \underset{G}{\Rightarrow} u_1r_1u_2r_2u_3$;

b) $u \underset{G}{\Rightarrow} u_1xu_2r_2u_3 \underset{G}{\Rightarrow} u_1r_1u_2r_2u_3$;

de unde se vede că în cele două derivări, cuvântul obţinut din u este acelaşi, adică $u_1r_1u_2r_2u_3$. ■

Teorema 3.2.3. *Familia limbajelor libere de context, $\mathcal{L}_2$, este închisă la operaţia de iteraţie.*

Demonstraţie. Fie L un limbaj de tipul doi; trebuie să demonstrăm că $L^* \in \mathcal{L}_2$. Din $L \in \mathcal{L}_2$ rezultă că există o gramatică G de tipul doi, $G = (V_N, V_T, x_0, P)$ care să genereze L, adică $L = L(G)$.

Pentru a arăta că L^* este de tipul doi, vom construi o gramatică G_* de tipul doi care să genereze limbajul L^*. Să considerăm gramatica:

(1) $G_* = (V_N \cup \{y_0\}, V_T, y_0, P \cup \{y_0 \to \lambda, y_0 \to y_0 x_0\})$, unde $y_0 \notin V_N$.

Se observă imediat, din forma regulilor de generare, că G_* este o gramatică de tipul doi. Să demonstrăm egalitatea $L^* = L(G_*)$.

a) Incluziunea $\boldsymbol{L^* \subseteq L(G_*)}$. Fie $p \in L^* = \bigcup\limits_{n=0}^{\infty} L^n$; rezultă că există $k \geq 0$ astfel încât $p \in L^k$.

Dacă $k = 0$, rezultă $p = \lambda$ şi în gramatica G_* avem regula $y_0 \to \lambda$, deci şi derivarea $y_0 \underset{G_*}{\Rightarrow} \lambda$, de unde $p = G_*$.

Dacă $k \geq 1$, atunci din $p \in L^k$, rezultă că $p = p_1 \ldots p_k$ şi $p_j \in L(G)$, $1 \leq j \leq k$. Din $p_j \in L(G)$, $1 \leq j \leq k$, rezultă că există derivările:

$$x_0 \overset{*}{\underset{G}{\Rightarrow}} p_j,\ 1 \leq j \leq n. \tag{2}$$

Folosind derivările (2) putem scrie în G_*:

$$\begin{aligned} &y_0 \underset{G_*}{\Rightarrow} y_0 x_0 \underset{G_*}{\Rightarrow} y_0 x_0 x_0 \underset{G_*}{\Rightarrow} \ldots \underset{G_*}{\Rightarrow} y_0 \underbrace{x_0 \ldots x_0}_{k-\text{ori}} \underset{G_*}{\Rightarrow} \underbrace{x_0 \ldots x_0}_{k-\text{ori}} \underset{G_*}{\Rightarrow} \\ &\overset{*}{\underset{G_*}{\Rightarrow}} p_1 x_0 \ldots x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_1 p_2 x_0 \ldots x_0 \overset{*}{\underset{G_*}{\Rightarrow}} \ldots \overset{*}{\underset{G_*}{\Rightarrow}} p_1 p_2 \ldots p_k. \end{aligned} \tag{3}$$

Deci avem $y_0 \overset{*}{\underset{G_*}{\Rightarrow}} p$, de unde $p \in L(G_*)$.

b) Incluziunea inversă, $\boldsymbol{L(G_*) \subseteq L^*}$. Fie $p \in L(G_*)$, deci există derivarea $y_0 \overset{*}{\underset{G_*}{\Rightarrow}} p$. Singurele reguli cu y_0 sunt $y_0 \to \lambda$ şi $y_0 \to y_0 x_0$; deci în derivarea $y_0 \overset{*}{\underset{G_*}{\Rightarrow}} p$ la primul pas se aplică una din aceste reguli. Dacă se aplică prima regulă, atunci derivarea devine $y_0 \overset{*}{\underset{G_*}{\Rightarrow}} \lambda$ şi avem $\lambda = p \in L^*$. Dacă la primul pas se aplică cea de a doua regulă cu y_0 în partea stângă, atunci derivarea devine $y_0 \underset{G_*}{\Rightarrow} y_0 x_0$. Mai departe se pot aplica reguli din P sau din nou o regulă cu y_0 în partea stângă. Conform lemei precedente, putem considera că toate regulile care se aplică în derivarea $y_0 \underset{G_*}{\overset{*}{\Rightarrow}} p$ şi au y_0 în partea stângă se aplică la început, ultima regulă cu y_0 este $y_0 \to \lambda$, deoarece y_0 trebuie eliminat şi aceasta este singura regulă care şterge y_0. Deci putem considera că în derivarea $y_0 \underset{G_*}{\overset{*}{\Rightarrow}} p$ se aplică de m ori regula $y_0 \to y_0x_0$, după care se aplică $y_0 \to \lambda$. Deci putem scrie:

$$y_0 \underset{G_*}{\overset{m*}{\Rightarrow}} y_0 \underbrace{x_0 \ldots x_0}_{\text{m ori}} \underset{G_*}{\Rightarrow} \underbrace{x_0 \ldots x_0}_{\text{m ori}} \underset{G_*}{\overset{*}{\Rightarrow}} p$$

Acum considerăm derivarea $x_0 \ldots x_0 \underset{G_*}{\overset{*}{\Rightarrow}} p$ şi aplicăm teorema de localizare, rezultă $p = p_1 \ldots p_m$ şi

$$x_0 \underset{G_*}{\overset{*}{\Rightarrow}} p_j,\ 1 \leq j \leq m. \tag{4}$$

Dar în derivările (4) se aplică numai reguli din P, celelalte le-am aplicat înainte. Deci avem:

$$x_0 \underset{G}{\overset{*}{\Rightarrow}} p_j,\ 1 \leq j \leq m, \tag{4'}$$

de unde $p_j \in L$, $1 \leq j \leq m$, deci $p \in L^m \subseteq L^*$. ■

3 Exemple rezolvate

I. Să se arate că automatul pushdown nedeterminist $P = (\{s_0, s_1, s_2, s_3\}, \{0, 1\}, \{z, 0\}, \delta, s_0, z, \{s_3\})$ cu δ definit prin:

1) $\delta(s_0, 0, z) = \{(s_1, 0z)\}$

2) $\delta(s_1, 0, 0) = \{(s_1, 00)\}$

3) $\delta(s_1, 1, 0) = \{(s_2, \lambda)\}$

4) $\delta(s_2, 1, 0) = \{(s_2, \lambda)\}$

5) $\delta(s_2, \lambda, z) = \{(s_3, \lambda)\}$

6) $\delta(s, 0, z) = \emptyset$ în celelalte cazuri.

recunoaşte limbajul $\{0^n1^n|n \geq 1\}$.

Rezolvare:

Considerăm câteva cazuriparticulare. Să arătăm că $\{0^n1^n|n \geq 1\} \subseteq L(P)$.

Pentru $n = 1$ avem:

$(s_0, 01, z) \vdash (s_1, 1, 0z) \vdash (s_2, \lambda, z) \vdash (s_3, \lambda, \lambda)$

Pentru $n = 2$ avem:

$(s_0, 0011, z) \vdash (s_1, 011, 0z) \vdash (s_1, 11, 00z) \vdash (s_2, 1, 0z) \vdash (s_2, \lambda, z) \vdash (s_3, \lambda, \lambda)$

Pentru $n > 2$ avem:

$(s_0, 0^n1^n, z) \vdash (s_1, 0^{n-1}1^n, 0z) \vdash^* (s_1, 1^n, 0^nz) \vdash (s_2, 1^{n-1}, 0^{n-1}z) \vdash^* (s_2, \lambda, z) \vdash (s_3, x, \lambda)$.

Invers, să arătăm acum că $L(P) \subseteq \{0^n1^n|n \geq 1\}$.

Fie $w \in L(P)$. Să arătăm că $w = 0^n1^n$. Vom arăta că w nu poate avea altă formă. Din $w \in L(P)$ avem conform definiţiei că $(s_0, w, z) \vdash^* (s_3, \lambda, \alpha)$.

Considerăm $w = \lambda$, avem (s_0, λ, z) se blochează, deci $w = \lambda$ nu aparţine lui $L(P)$. Deci $w \in L(P)$ trebuie să fie diferit de λ.

2) $w = 1w'$, adică w începe cu 1, avem: $(s_0, 1w', z)$ se blochează, deci $w \notin L(P)$. De aici rezultă că w nu începe cu 1, deci începe cu 0.

3) $w = 0^n, n \geq 1$, avem $(s_0, 0^n, z) \vdash^* (s_1, \lambda, 0^nz)$ şi se blochează. Nu putem ajunge în s_3 deci $0^n \notin L(P)$.

4) $w = 0^n1^kw'$ cu $w' = \lambda$ sau w' începe cu 0.

Vom arăta mai întâi că nu putem avea $n < k$ şi nici $n > k$, deci n trebuie să fie egal cu k. Apoi arătăm că w trebuie să fie egal cu λ.

a) Cazul $n < k$. Atunci avem:

$(s_0, 0^n1^kw', z) \vdash^* (s_1, 0^{n-1}1^k, 0z) \vdash^* (s_1, 1^k, 0^nz) \vdash (s_2, 1^{k-1}, 0^{n-1}z) \vdash^* (s_2, 1^{k-n}, z) \vdash^* (s_3, 1^{k-n}, \lambda)$. Am ajuns în starea finală s_3 dar $k - n > 0$, cuvântul $0^n1^kw'$ nu este acceptat.

b) Cazul $n > k$.

$(s_0, 0^n1^kw', z)\vdash^*(s_1, 0^{n-1}1^k, 0z)\vdash^*(s_1, 1^k, 0^nz) \vdash (s_2, 1^{k-1}, 0^{n-1}z) \vdash$
$(s_2, x, 0^{n-k}z)$ şi se blochează deoarece $n-k>0$. Deci trebuie să avem $n=k$.
Să arătăm că $w'=\lambda$. Avem $(s_0, 0^n1^nw', z) \vdash (s_1, 0^{n-1}1^n, w', 0z)\vdash^*$ $(s_1, 1^nw', 0^n, z) \vdash (s_2, 1^{n-1}w', 0^{n-1}z)\vdash^*(s_2, w', z)\vdash^*(s_3, w', \lambda)$ şi că să fie acceptat trebuie să ajungem la (s_3, λ, λ), deci $w'=\lambda$. Prin urmare $w=0^n1^n$ cu $n\geq 1$.

II. Fie automatul pushdown nedeterminist $P=(\{s_0, s_1, s_2\}, \{a, b\}, \{z, a, b\}, s_0, z, \{s_2\})$ cu δ definit prin:

1) $\delta(s_0, a, z)=\{(s_0, az)\}$;

2) $\delta(s_0, b, z)=\{(s_0, bz)\}$;

3) $\delta(s_0, a, a)=\{(s_0, aa), (s_1, \lambda)\}$;

4) $\delta(s_0, a, b)=\{(s_0, ab)\}$;

5) $\delta(s_0, b, a)=\{(s_0, ba)\}$;

6) $\delta(s_0, b, b)=\{(s_0, bb), (s_1, \lambda)\}$;

7) $\delta(s_1, a, a)=\{(s_1, \lambda)\}$;

8) $\delta(s_1, b, b)=\{(s_1, \lambda)\}$;

9) $\delta(s_1, \lambda, z)=\{(s_2, \lambda)\}$;

10) $\delta=\emptyset$ în alte cazuri.

Să se arate că limbajul $L(P)\{w\tilde{w}|w\in\{a, b\}^+\}$, unde $w\tilde{w}$ este inversatul lui w.

Rezolvare. Să arătăm întâi că $\{w\tilde{w}|w\in\{a, b\}^+\}\subseteq L(P)$.
Câteva ocazii particulare:

1) $w=ab, w\tilde{w}=ba$.

$s_0, abba, z) \vdash (s_0, bba, az) \vdash (s_0, ba, baz) \vdash (s_1, a, az) \vdash (s_1, \lambda, z) \vdash (s_2, \lambda, \lambda)$

2) $w = abb$ şi $w\tilde{w} = bba$.

$(s_0, abbbba, z) \vdash (s_0, bbbba, az) \vdash (s_0, bbba, baz) \vdash (s_0, bba, bbaz) \vdash (s_1, ba, baz) \vdash (s_1, a, az) \vdash (s_1, \lambda, z) \vdash (s_2, \lambda, \lambda)$

Cazul general. Fie $w = i_1 \dots i_n, n \geq 1$ şi $i_j \in \{a, b\}$ pentru $1 \leq j \leq n$. Atunci $\tilde{w} = i_n \dots i_1$. Avem
$(s_0, i_1 \dots i_n i_n \dots i_1, z) \vdash (s_0, i_2 \dots i_n i_n \dots i_1, i_1, z) \vdash^* (s_0, i_n \dots i_1, i_n \dots i_1 z)$
$\vdash^* (s_1, i_{n-1} \dots i_1, i_{n-1} \dots i_1 z) \vdash^* (s_1, i_1, i_1 z) \vdash (s_1, i_1, i_z) \vdash (s_1, \lambda, z) \vdash (s_2, \lambda, \lambda)$.

Deci $w\tilde{w} \in L(P)$.

Să demonstrăm incluziunea inversă $L(P) \subseteq \{w\tilde{w} | w \in \{0, b\}^+\}$. Fie $p \in L(P)$. Din definiţia limbajului avem că $(s_0, p, z) \vdash^* (s_2, \lambda, \alpha)$. Vom demonstra că $p = w\tilde{w}$. Vom considera mai multe cazuri:

1) $p = \lambda$. Avem (s_0, λ, z) se blochează de ci $\lambda \notin L(P)$.

2) $p \neq \lambda$. Fie $p = i_1 \dots i_k, k \geq 1, i_j \in \{a, b\}, 1 \leq j \leq k$. Din definiţie avem că $(s_0, i_1 \dots i_k, z) \vdash^* (s_2, \lambda, \alpha)$. Să vedem cum poate evolua această tranziţie.

$$(s_0, i_1 \dots i_k, z) \vdash (s_0, i_2 \dots i_k, i_1, z) \vdash^* (s_0, i_{l+1} \dots i_k, i_l \dots i_1 z).$$

Ca să nu se blocheze şi să poată evolua pentru a ajunge în starea s_2, trebuie ca, $l = k - l$ deci $k = 2l$ şi în plus $i_{l+1} = i_l, i_{l+2} = i_{l-1}, \dots, i_{2l} = i_1$. In aceste condiţii avem:

$$(s_0, i_{l+1} \dots i_{2l}, i_l \dots i, z) \vdash^* (s_1, i_{l+2} \dots I_{2l}, i_{l-1} \dots i_1 z) \vdash^* (s_1, \lambda, z) \vdash (s_2, \lambda, \lambda).$$

Dar atunci $p = i_1 \dots i_l i_l \dots i_1 = w\tilde{w}$.

Temă de control (3).

1. Fie automatul pushdown nedetrminist $P = (\{s_0, s_1, s_2, s_3\}, \{a, b, c\}, \{z, a\}, \delta, s_0, z, \{s_3\})$ cu δ definit prin:

1) $\delta(s_0, a, z) = \{(s_1, aaz)\}$;

2) $\delta(s_1, a, a) = \{(s_1, aaa)\}$;

3) $\delta(s_1, b, a) = \{(s_2, \lambda)\}$;

4) $\delta(s_1, c, a) = \{(s_2, \lambda)\}$;

5) $\delta(s_2, b, a) = \{(s_2, \lambda)\}$;

6) $\delta(s_2, c, a) = \{(s_2, \lambda)\}$;

7) $\delta(s_2, \lambda, z) = \{(s_3, \lambda)\}$;

8) $\delta = \emptyset$ în celelalte cazuri.

Să se arate că $L(P) = \{a^n w | w \in \{b, c\}^+, |w| = 2n, n \geq 1\}$.

2. Fie automatul pushdown nedeterminist $P = (\{s_0, s_1, s_2, s_2\}, \{a, b, c\}, \{a, z\}, \delta,\ s_0, z, \{s_3\})$ cu δ definit prin:

1) $\delta(s_0, a, z_0) = \{(s_1 a z_0)\}$;

2) $\delta(s_1, a, a) = \{(s_1, aa\}$;

3) $\delta(s_2, b, a) = \{(s_1 \lambda)\}$;

4) $\delta(s_2, b, a) = \{(s_2, \lambda)\}$;

5) $\delta(s_2, c, z) = \{(s_3, z)\}$;

6) $\delta(s_3, c, z) = \{(s_3, z)\}$;

7) $\delta = \emptyset$ în celelalte cazuri.

Să se arate că $L(P) = \{a^n b^n c^m | n, m \geq 1\}$.

3. Fie automatul pushdown nedeterminist $P = (\{s_0, s_1, s_2, s_3, s_4\}, \{a, b, c\}, \{b, z\}, \delta, s_0, z, \{s_4\})$ cu δ definit prin:

1) $\delta(s_0, a, z) = \{(s_1, z)\}$;

2) $\delta(s_1, a, z) = \{(s_1, z)\}$;

3) $\delta(s_1, b, z) = \{(s_2, bz)\}$;

4) $\delta(s_2, b, z) = \{(s_2, bz)\}$;

5) $\delta(s_2, c, b) = \{(s_3, \lambda)\}$;

6) $\delta(s_3, c, b) = \{(s_3, \lambda)\}$;

7) $\delta(s_3, \lambda, z) = \{(s_4, \lambda)\}$;

8) $\delta = \emptyset$ în celelalte cazuri.

Să se arate că $L(P) = \{a^n b^m c^m | n, m \geq 1\}$.

Tema 4

Forme normale. Arbori de derivare. Teorema lui Ogden şi aplicaţii. Automate pushdown deterministe

1 Forme normale

În unele demonstraţii cât şi în unele aplicaţii este util să considerăm gramatici de tipul doi de anumite forme particulare dar care să nu micşoreze puterea de generare, adică aceste gramatici de forme particulare să genereze tot $\mathcal{L}_2$.

Vom considera trei clase de gramatici, numite *forme normale* şi anume: *forma normală Chomsky, formă normală Greibach* şi *formă normală operator*.

Definiţia 4.1.1. O gramatică independentă de context este sub *formă normală Chomsky* dacă regulile sale sunt de forma $x \to yz$ sau $x \to a$ unde $x, y, z \in V_N$ şi $a \in V_T$.

Teorema 4.1.2. *Orice limbaj independent de context, L, care nu conţine cuvântul vid λ, poate fi generat de o gramatică independentă de context sub formă normală Chomsky.*

Demonstraţie. Deoarece L este un limbaj independent de context şi nu conţine λ, rezultă că L poate fi generat de o gramatică $G = (V_N, V_T, x_0, P)$, care nu conţine reguli de forma $x \to y$ sau $x \to \lambda$, cu $x, y \in V_N$. Deci regulile în G sunt de forma:

$$(1) \qquad x \to y_1 \dots y_n, \text{ cu } n \geq 1 \text{ şi } x \in V_N.$$

În (1) pentru $n = 1$, obţinem : $x \to y_1$ şi din faptul că gramatica G nu conţine redenumiri, rezultă că $y_1 \in V_T$, deci regula $x \to y_1$ este de forma cerută pentru gramatici sub formă normală Chomsky. Fiecărui $i \in V_T$ îi punem în corespondenţă un nou simbol neterminal x_i. Pentru fiecare regulă de forma (1) cu $n \geq 2$ procedăm în felul următor: *fiecare simbol terminal y_i cu $1 \leq i \leq n$, îl înlocuim în regula $x \to y_1 \dots y_n$ cu un simbol nou x_{y_i} şi adăugăm regula $x_{y_i} \to y_i$, astfel că regula care s-a obţinut din $x \to y_1 \dots y_n$ prin înlocuirea lui y_i cu x_{y_i} şi regula $x_{y_i} \to y_i$ împreună au acelaşi efect ca regula $x \to y_1 \dots y_n$.* Mulţimea regulilor obţinute din regulile de forma $x \to y_1 \dots y_n$ în care am înlocuit simbolurile terminale y_i cu simbolurile noi x_{y_i} şi împreună cu noile reguli de forma $x_{y_i} \to y_i$ o vom nota cu P_1. De asemenea notăm cu $Z = \{x_i \mid i \in V_T\}$ mulţimea tuturor simbolurilor x_{y_i} adăugaţi. Considerăm gramatica $G_1 = (V_N \cup Z, V_T, x_0, P_1)$ care este o gramatică de tip doi şi este echivalentă cu G.

Vom construi o gramatică G_2 independentă de context sub formă normală Chomsky şi care să fie echivalentă cu G_1 deci şi cu G. Regulile în G_1 sunt de forma $x \to a$ sau de forma $x \to A_1 \dots A_n$, $n \geq 2$, unde $x, A_i \in V_N \cup Z$, $1 \leq i \leq n$ şi $a \in V_T$. Regulile de forma $x \to a$ şi $x \to A_1 A_2$ le lăsăm neschimbate, adică le punem şi în G_2, pentru că ele sunt corespunzătoare gramaticilor sub forma normală Chomsky.

Unei reguli din G_1 de forma:

$$(2) \qquad x \to A_1 \dots A_n, \; n > 2, \; x, a_i \in V_N \cup Z, \; 1 \leq i \leq n,$$

îi punem în corespondenţă în G_2, următoarele reguli:

$$(3)\qquad \begin{cases} x \to A_1B_1 \\ B_1 \to A_2B_2 \\ \dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots \\ B_{n-3} \to A_{n-2}B_{n-2} \\ B_{n-2} \to A_{n-1}A_n \end{cases}$$

unde $B_1, B_2, \dots B_{n-2}$ sunt simboluri noi. Aplicarea regulilor de forma (2) sau a şirului (3) au acelaşi efect, adică generarea din x a şirului $A_1 \dots A_n$. Deoarece $B_1, \dots, B_{n-2}$ sunt simboluri noi, rezultă că fiecare din ei nu sunt afectaţi de alte reguli. Fie $\mathcal{B}$ mulţimea tuturor simbolurilor B_i adăugaţi în regulile de forma (3). Considerăm gramatica independentă de context: $G_2 = (V_N \cup Z \cup \mathcal{P}, V_T, x_0, P_2)$ – unde P_2 este formată din regulile de forma $x \to a$ sau $x \to A_1A_2$, cu $x, A_1, A_2 \in V_N \cup Z$ şi $a \in V_T$ din G_1 - şi mulţimea tuturor regulilor de forma (3), corespunzătoare regulilor de forma (2) din G_1. Din observaţiile făcute mai sus, asupra legăturii dintre regulile de forma (2) şi cele de forma (3), rezultă că G_2 este echivalentă cu G_1. ■

Observaţia 4.1.3. Faptul că $\lambda \notin L$ nu este esenţial. Deoarece dacă $\lambda \in L$, atunci considerăm $L' = L - \{\lambda\}$, care este tot un limbaj de tip doi şi el poate fi generat de o gramatică sub formă normală Chomsky G', deci $L' = L(G')$. Atunci dacă $G' = (V'_N, V'_T, x'_0, P')$, considerăm gramatica $G = (V'_N \cup \{x_0\}, V'_T, x_0, P' \cup \{x_0 \to \lambda, x_0 \to x'_0\})$ care generează L.

Definiţia 4.1.4. O gramatică $G = (V_N, V_T, x_0, P)$ de tipul doi este sub *formă normală Greibach* dacă regulile sale sunt de forma: $X \to a\alpha$, unde $x \in V_N$, $a \in V_T$ şi $\alpha \in V^*$.

Gramatica G este sub *formă normală m-standard* dacă este sub formă normală Greibach şi $|\alpha| \leq m$.

Definiţia 4.1.5. Fie $G = (V_N, V_T, x_0, P)$ o gramatică independentă de context. Un neterminal $x \in V_N$ se numeşte *stâng-recursiv* (*drept-recursiv*) dacă în gramatica G există derivarea $x \overset{+}{\Rightarrow} x\beta$, $\beta \in V^*$ (respectiv $x \overset{+}{\Rightarrow} \beta x$). O gramatică G se numeşte *stâng-recursivă* (*drept-recursivă*) dacă ea are cel puţin un simbol neterminal stâng-recursiv (drept-recursiv).

Se poate demonstra următoarea teoremă:
Teorema Orice limbaj $L \in \mathcal{L}_{\in}$ cu $X \notin L$ poate fi generat de o gramatică sub formă normală Greibach.

A treia formă normală pentru gramaticile de tip doi este *forma normală operator*.

Definiţia 4.1.6. O gramatică de tip doi $G = (V_N, V_T, x_0, P)$ este sub *formă normală operator*, dacă oricare ar fi o regulă $x \to \alpha \in P$, în α nu apar doi simboli din V_N consecutivi.
Se poate demonstra că oricare ar fi un limbaj $L \in \mathcal{L}_2$ cu $\lambda \notin L$ există o gramatică G sub formă normală operator cu $L = L(G)$.

2 Arbori de derivare

Reprezentarea derivărilor în gramatici independente de context prin arbori de derivare, este o modalitate utilă pentru simplificarea anumitor demonstraţii.
Fie $\mathcal{G} = (X, E)$ un graf orientat. Dacă $(x, y) \in E$, vom spune că x este precedent direct al lui y şi y este succesor direct al lui x.

Definiţia 4.2.1. Un graf orientat $\mathcal{G}$ este un arbore orientat şi ordonat dacă sunt îndeplinite următoarele proprietăţi:

1. există în $\mathcal{G}$ un nod, numit *rădăcină*, care nu are precedenţi direcţi şi de la care există un drum la fiecare nod al grafului;

2. orice nod diferit de rădăcină are exact un precedent;

3. mulţimea succesorilor direcţi ai unui nod este ordonată.

Vom reprezenta arborii pe nivele, rădăcina va fi reprezentată pe primul nivel, iar succesorii direcţi ai unui nod vor fi reprezentaţi pe nivelul următor nivelului nodului, ordinea succesorilor fiind dată de reprezentarea lor de la stânga spre dreapta. Nodurile care au descendenţi direcţi se numesc *noduri interioare* şi cele care nu au descendenţi în $\mathcal{G}$ se numesc *noduri frunze* sau *noduri terminale.*

Definiţia 4.2.2. Un nod x este precedent al lui y dacă drumul de la rădăcină la y trece prin x (y este succesor al lui x sau y este descendent al lui x).

Convenţia de reprezentare pe nivele permite renunţarea la săgeţi, orientarea fiind de sus în jos ca în figura ce urmează.

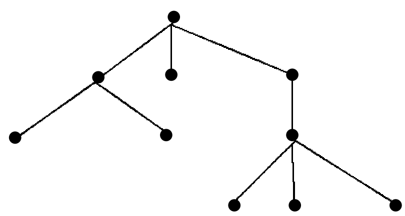


Fig. 4.2.1.

Definiţia 4.2.3. Fie $G = (V_N, V_T, x_0, P)$ o gramatică independentă de context. Un **arbore de derivare** pentru gramatica G este un arbore $A = (X, E)$ împreună cu funcţia de etichetare a nodurilor $f : X \to V_N \cup V_T \cup \{\lambda\}$ cu proprietăţile:

1. $f(r) = x_0$, dacă r este rădăcina arborelui;

2. dacă $x \in X$ are descendenţi, atunci $f(x) \in V_N$;

3. dacă x are descendenţi direcţi $x_1, x_2, \ldots, x_n$ în această ordine, atunci în G avem $f(x) \to f(x_1)f(x_2)\ldots f(x_n)$;

4. dacă $f(x) = \lambda$, atunci x este singurul descendent al precedentului său.

Exemplul 4.2.4. Fie $G = (V_N, V_T, x_0, P)$ cu $V_N = \{x_0, x_1, x_2\}$, $V_T = \{a, b\}$, $P = \{x_0 \to ax_1x_2|bx_2a,\ x_1 \to x_0x_1,\ x_2 \to \lambda\}$. Un exemplu de arbore de derivare este:

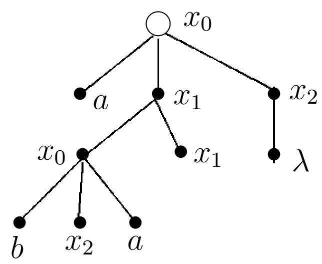


Fig. 4.2.2.

Definiţia 4.2.5. Un subarbore al unui arbore de derivare este un nod din arbore care devine rădăcină pentru subarbore împreună cu toţi descendenţii săi. Dacă rădăcina subarborelui are eticheta x, îl vom numi x-*arbore*. Astfel un arbore de derivare este un x_0-*arbore*.

În figura de mai jos avem un x_1-arbore

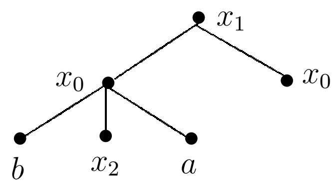


Fig. 4.2.3.

Ordinea de la stânga la dreapta a descendenţilor direcţi ai unui nod induce o ordine de la stânga la dreapta în mulţimea frunzelor unui arbore de derivare.

Definiţia 4.2.6. Fie x, y descendenţii direcţi ai aceluiaşi nod şi x la stânga lui y; atunci

i) x este la stânga oricărui descendent al lui y;

ii) orice descendent al lui x este la stânga lui y;

ii) orice descendent al lui x este la stânga oricărui descendent al lui y.

Considerăm atunci cuvântul obţinut din etichetele frunzelor unui arbore de derivare, în ordinea în care apar ele în arbore. Numim acest cuvânt **frontiera arborelui de derivare**.

Teorema 4.2.7. *Fie $G = (V_N, V_T, x_0, P)$ o gramatică independentă de context. Oricare ar fi $x \in V_N$ şi $\gamma \in (V_N \cup V_T)^*$, avem $x \overset{*}{\Rightarrow} \gamma$ dacă şi numai dacă există un x-arbore de derivare cu frontiera γ.*

Fie $w \in L(G)$. Considerăm că anumite poziţii din w le marcăm.

Teorema 4.2.8. (Teorema iteraţiei. Teorema lui Ogden)
Pentru orice limbaj L independent de context, există o constantă $n \in N$ (care depinde numai de limbajul L) astfel că orice $w \in L$ cu m poziţii marcate, $m \geq n$, admite o factorizare $w = xyzuv$ cu proprietăţile:

1. *yu are cel puţin o poziţie marcată;*

2. *yzu are cel mult n poziţii marcate;*

3. *$\forall i \geq 0$, $xy^izu^iv \in L$.*

Demonstraţie. Fie $L \in \mathcal{L}_2$ şi fie $G = (V_N, V_T, x_0, P)$ sub formă normală Chomsky, astfel că $L(G) = L - \{\lambda\}$. Dacă $|V_N| = k$ atunci considerăm $n = 2^k + 1$. Fie $w \in L(G)$ cu $|w| \geq n$ şi în w avem $m \geq n$ poziţii marcate. În arborele de derivare corespunzător derivării $x_0 \underset{G}{\overset{*}{\Rightarrow}} w$ construim inductiv un drum D astfel:

i) $D = \{r\}$, r este rădăcina arborelui.

ii) Fie t ultimul nod plasat în D. Dacă t este un nod final (frunză), drumul este construit. Dacă t nu este nod terminal, el are doi descendenţi (pentru că gramatica G este sub formă normală Chomsky). Avem 2 cazuri:

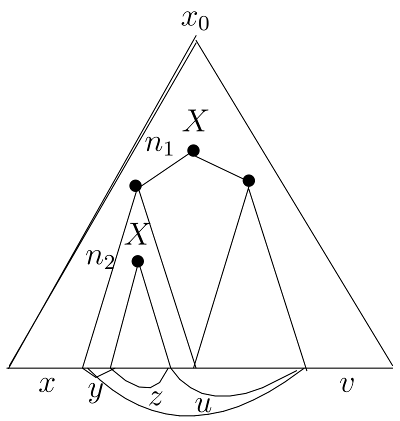
a) numai unul din descendenţii lui t are descendenţi finali marcaţi. atunci îl adăugăm pe acesta la D şi repetăm ii).

b) ambii descendenţi ai lui t au descendenţi finali marcaţi. Atunci t se numeşte **punct de ramificare**. Se adaugă la D nodul cu cel mai mare număr de descendenţi marcaţi şi apoi se repetă ii).

Din construcţia drumului D se constată că fiecare punct de ramificare are cel puţin jumătate din descendenţii finali marcaţi ai precedentului punct de ramificare din D. Cum în arborele de derivare avem cel puţin $n = 2^k + 1$ poziţii marcate în w şi toate aceste poziţii marcate sunt descendenţi ai rădăcinii, înseamnă că avem cel puţin $k+1$ puncte de ramificare în drumul D.

Cum $|V_N| = k$, există cel puţin 2 puncte de ramificare în D, n_1, n_2 etichetate la fel, cu un neterminal $\boldsymbol{x}$. Alegem aceste puncte aşa fel că n_1 este mai aproape de rădăcină decât n_2 şi **pe porţiunea de drum D de la n_1 la frontiera arborelui nu mai avem altă pereche de puncte etichetate cu aceeaşi etichetă**. Deci porţiunea de drum **D de la n_1 la frontieră conţine cel mult $k + 1$ puncte de ramificare**. Descompunem $w = xyzuv$ ca în figura 3.4.7. Considerăm subarborele cu rădăcina în n_1; frontiera sa yzu conţine **cel mult n poziţii marcate** (altfel ar exista de la el la frontieră, pe drumul D, mai mult de $k + 1$ puncte de ramificare). Deci am demonstrat 2).

Fie z frontiera subarborelui cu rădăcina în n_2. cum n_1 şi n_2 sunt pe drumul D, z este subcuvânt din w_1, unde $w_1 = yzu$ şi yu are cel puţin o poziţie marcată, deoarece n_1 şi n_2 sunt puncte de ramificare. Deci am demonstrat 1).



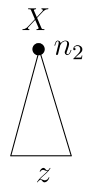


Fig. 4.2.7 **Fig. 4.2.8**

Considerăm subarborele cu rădăcina în n_2 ca în Figura 3.4.8. Dacă X este eticheta celor două noduri, avem:

(1) $$X \overset{*}{\underset{G}{\Rightarrow}} z$$

deoarece z este frontiera subarborelui cu rădăcina n_2 care are eticheta X.

Considerăm subarborele cu rădăcina în n_1 din care scoatem subarborele cu rădăcina în n_2, dar lăsăm pe n_2.

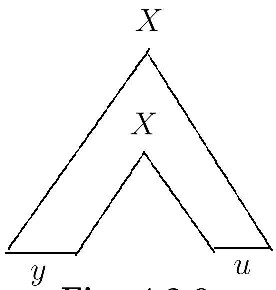


Fig. 4.2.9.

atunci avem

(2) $$X \overset{*}{\underset{G}{\Rightarrow}} yXu$$

Considerăm arborele de derivare din care am scos subarborele cu rădăcina în n_1 dar lăsăm n_1. Avem

(3) $$x_0 \overset{*}{\underset{G}{\Rightarrow}} xXv.$$

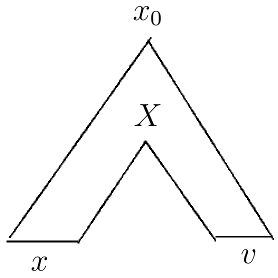


Fig. 4.2.10.

Să demonstrăm acum proprietatea 3) din teoremă utilizând (1), (2), (3). Pentru $i = 0$ avem:

$x_0 \overset{*}{\underset{G}{\Rightarrow}} xXv \overset{*}{\underset{G}{\Rightarrow}} xzv.$

Pentru $i = k > 0$ avem:

$x_0 \overset{*}{\Rightarrow} xXv \overset{*}{\underset{G}{\Rightarrow}} xy^kXu^kv \overset{*}{\underset{G}{\Rightarrow}} xy^kzu^kv.$ ■

Corolar 4.2.9.(Lema Bar-Hillel pentru limbaje de tip 2).

Pentru orice limbaj de tip 2 există o constantă n astfel că dacă $w \in L$, $|w| \geq n$, atunci există o descompunere $w = xyzuv$ cu proprietăţile:

1. $|yu| \geq 1$;
2. $|yzu| \leq n$;
3. $\forall i \geq 0,\ xy^izu^iv \in L$.

Demonstraţie. Dacă în lema lui Ogden considerăm toate poziţiile cuvântului w marcate, se obţine lema Bar-Hillel. ■

Folosind lema lui Bar-Hillel, putem demonstra că familia $\mathcal{L}_2$ nu este închisă la intersecţie şi nici la complementară. Acest rezultat este dat de următoarea teoremă:

Teorema 4.2.10. *Familia $\mathcal{L}_2$ nu este închisă la intersecţie şi nici la complementară.*

Demonstraţie. Limbajele $L_1 = \{a^i b^i c^j | i, j \geq 1\}$ şi $L_2 = \{a^i b^j c^j | i, j \geq 1\}$ sunt limbaje independente de context; ele sunt generate de gramaticile $G_1 = (\{S, A, B\}, \{a, b, c\}, S, \{S \to AB, A \to aAb, A \to ab, B \to Bc, B \to c\})$ şi $G_2 = (\{S, A, B\}, \{a, b, c\}, S, \{S \to AB, A \to aA, A \to a, B \to bBc, B \to bc\})$, respectiv.
Se poate verifica, aplicând lema Bar-Hillel, că limbajul $L_1 \cap L_2 = \{a^i b^i c^i | i \geq 1\}$ nu este limbaj independent de context, deci $\mathcal{L}_2$ nu este închisă la intersecţie. Cum $\mathcal{L}_2$ este închisă la reuniune şi nu este închisă la intersecţie, rezultă că $\mathcal{L}_2$ nu este închisă la complementară. ■

Interesant este că dacă intersectăm un limbaj independent de context cu un limbaj regulat, se obţine tot un limbaj independent de context. Acest rezultat este dat în teorema ce urmează.

Teorema 4.2.11. *Dacă $L \in \mathcal{L}_2$ şi $R \in \mathcal{L}_3$ atunci $L \cap R \in \mathcal{L}_2$.*

Demonstraţie. Din $L \in \mathcal{L}_2$ rezultă că există un automat pushdown $P = (S_1, \Sigma, \Gamma, \delta_1, s_0^1, z_0, F_1)$ cu $L = L(P)$. Din $R \in \mathcal{L}_3$ rezultă că există un automat finit determinist $A = (S_2, \Sigma, \delta_2, s_0^2, F_2)$ cu $R = L(A)$. Vom construi un automat pushdown P' care să recunoască limbajul $L \cap R$. Automatul pushdown P' este: $P' = (S_1 \times S_2, \Sigma, \Gamma, \delta, (s_0^1, s_0^2), z_0, F_1 \times F_2)$ unde δ este definit prin: $((s_1', s_2'), \gamma) \in \delta((s_1, s_2), a, z)$ dacă şi numai dacă $(s_1', \gamma) \in \delta_1(s_1, a, z)$ şi $s_2' = \delta_2(s_2, a)$.

Se poate arăta că $L(P') = L \cap R$. Construcţia lui P' este ilustrată în figura de mai jos.

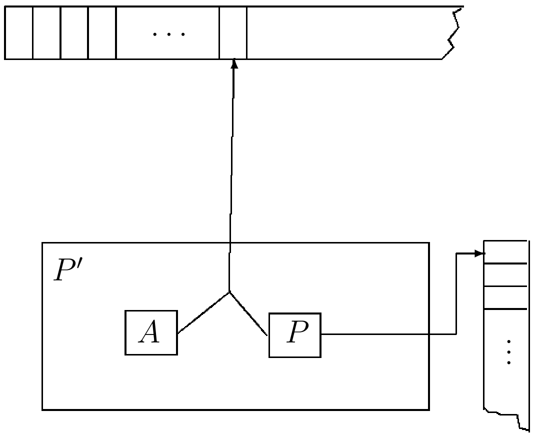


Fig. 4.2.11.

3 Automate pushdown deterministe

Pentru automate finite am văzut că puterea de recunoaştere este aceeaşi pentru automate finite deterministe sau automate finite nedeterministe. În cazul automatelor pushdown acest lucru nu mai este adevărat.

Definiţia 4.3.1. Un automat pushdown $M = (S, \Sigma, \Gamma, \delta, s_0, F)$ este *determinist* dacă sunt îndeplinite condiţiile:

1. $|\delta(s, a, z)| \leq 1$, $\forall a \in \Sigma \cup \{\lambda\}$, $\forall s \in S$, $\forall z \in \Gamma$;

2. dacă $\delta(s, \lambda, z) \neq \emptyset$, atunci $\delta(s, a, z) = \emptyset$, $\forall a \in \Sigma$.

Vom nota cu $\mathcal{L}_{2DET}$ clasa limbajelor acceptate de automate pushdown deterministe prin stări finale, adică $\mathcal{L}_{2DET} = \{L \mid \exists M$ automat pushdown determinist astfel că $L = L(M)\}$.

O subclasă de automate pushdown deterministe sunt cele sub formă normală.

Definiţia 4.3.2. Un automat pushdown determinist $M = (S, \Sigma, \Gamma, \delta, \ s_0, z_0, F)$ este sub *formă normală* dacă $\forall s \in S$, $\forall a \in \Sigma \cup \{\lambda\}$, $\forall z \in \Gamma$ şi $(s', \gamma) = \delta(s, a, z)$ are loc una din situaţiile:

1. $\gamma = \lambda$ (automatul şterge un simbol din memoria pushdown);
2. $\gamma = z$ (nu modifică memoria pushdown);
3. $\gamma = yz$ (se adaugă simbolul y la memoria pushdown).

Se poate demonstra că pentru orice limbaj $L \in \mathcal{L}_{2DET}$ există un automat pushdown determinist sub formă normală care să accepte L.

Deoarece automatele pushdown deterministe sunt o subclasă a automatelor pushdown nedeterministe rezultă că $\mathcal{L}_{2DET} \subseteq \mathcal{L}_2$. De fapt această incluziune este strictă deoarece $\mathcal{L}_{2DET}$ este închisă la complementariere şi $\mathcal{L}_2$ nu este închisă la complementariere, deci cele două familii de limbaje nu pot fi egale.

Exerciţii rezolvate

I) Să se arate că limbajul $L = \{a^n b^n c^n \,|\, n \geq 1\}$ nu este limbaj independent de context.

Rezolvare: Vom aplica teorema lui Ogden. Presupunem că limbajul L este independent de context. Alegem un cuvânt $w = a^n b^n c^n$, marcăm a^n şi alegem n mai mare decât numărul k din teorema lui Ogden. Conform teoremei lui Ogden $w = xyzuv$ cu proprietăţile din teoremă.

Vom arăta întâi că y nu poate conţine decât o literă. Să presupunem că $y = a^l b^n c^j$, $n, l, j \geq 1$. Atunci $u = c^m$, $m \geq 0$ şi luând xy^2zu^2v avem $a^{n-l}a^l\underline{b^n c^j a^l}\,b^n c^j c^{n-j+m}$ care ar trebui să aparţină limbajului L conform teoremei lui Ogden, dar acest cuvânt nu aparţine lui L deoarece are b-uri şi c-uri înainte de simbolul a. Analog se arată că y nu poate conţine doi simboli diferiţi (adică a şi b, sau b şi c). În aceeaşi manieră se arată că nici u nu poate conţine trei sau doi simboli diferiţi.

Prin urmare atât y cât şi u trebuie să conţină un singur simbol. Cuvântul y trebuie să conţină numai simboluri a, deoarece yu trebuie să aibă cel puţin un simbol marcat şi numai simbolurile a sunt marcate. Deci descompunerea lui $w = xyzuv$ se poate face într-unul din următoarele moduri: 1) y şi u conţin a-uri; 2) y conţine a-uri şi u conţine b-uri; 3) y conţine a-uri şi u conţine c-uri. Să considerăm cele trei cazuri posibile:

1) $x = a^l$, $y = a^t$, $z = a^s$, $u = a^r$, $v = a^{n-(l+t+s+r)}b^nc^n$ cu $t + r \geq 1$. Conform teoremei lui Ogden trebuie ca $xy^izu^iv \in L, \forall\, i \geq 0$.
Luăm $i = 0$ şi atunci $xy^0zu^0v = a^la^sa^{n-(l+t+s+r)}b^nc^n = a^{n-(t+r)}b^nc^n \notin L$ deoarece $n - (t + r) < n$. Contradicţie.
2) $x = a^l$, $y = a^t$, $z = a^{n-(l+t)}b^r$, $u = b^s$, $v = b^{n-(r+s)}c^n$ cu $t \geq 1$. Considerăm din nou $i = 0$ şi avem $xy^0zu^0v = a^la^{n-(l+t)}b^rb^{n-(r+s)}c^n = a^{n-t}b^{n-s}c^n \notin L$ deoarece $n - t < n$. Deci din nou contradicţie.
3) $x = a^l$, $y = a^t$, $z = a^{n-(l+t)}b^nc^s$, $u = c^r$, $v = c^{n-(r+s)}$ cu $t \geq 1$. Considerăm $i = 0$ şi avem $xy^0zu^0v = a^la^{n-(l+t)}b^nc^sc^{n-(r+s)} = a^{n-t}b^nc^{n-r} \notin L$ deoarece $n - t < n$. Contradicţie.

Deoarece în toate cele trei cazuri posibile am obţinut contradicţie, rezultă că limbajul $L = \{a^nb^nc^n \,|\, n \geq 1\}$ nu este limbaj independent de context.

II) Să se arate că limbajul $L = \{a^ib^jc^k \,|\, i, j, k \geq 1, i \neq j, i \neq k\}$ nu este limbaj independent de context.

Rezolvare: Considerăm cuvântul de forma $w = a^nb^{n+n!}c^{n+2n!}$ cu poziţiile lui a marcate şi $n \geq k$, k fiind cel din teorema lui Ogden. Conform teoremei lui Ogden w se descompune în $xyzuv$ cu proprietăţile din teoremă. Analog ca la exerciţiul precedent se arată că y şi u nu pot conţine mai mult de un simbol. Deci cazurile de descompunere posibile sunt: 1) y şi u conţin simboluri a; 2) y conţine simboluri a şi u conţine simboluri b; 3) y conţine a-uri şi u conţine c-uri. Să considerăm cele trei cazuri posibile:
1) $x = a^l$, $y = a^t$, $z = a^s$, $u = a^r$, $v = a^{n-(l+t+s+r)}b^{n+n!}c^{n+2n!}$ cu

$t + r \geq 1$. Notez cu $q = t + r$. Conform teoremei lui Ogden avem: $1 \leq q \leq n$. Deci q divide $n!$. Avem $n! = q \cdot j$ şi atunci $xy^izu^iv = a^l a^{ti} a^s a^{ri} a^{n-(l+t+s+r)} b^{n+n!} c^{n+2n!} = a^{n+(i-1)(t+r)} b^{n+n!} c^{n+2n!}$. Alegem i astfel încât $(i-1)q = n!$, deci $i = \frac{n!}{q} + 1 = j + 1$. În acest caz xy^izu^iv aparţine lui L, dar pe de altă parte $xy^izu^iv = a^{n+n!} b^{n+n!} c^{n+2n!} \notin L$ deoarece exponentul lui a este egal cu exponentul lui b. Contradicţie.
2) $x = a^l$, $y = a^t$, $z = a^{n-(l+t)} b^s$, $u = b^r$, $v = b^{n+n!-(r+s)} c^{n+2n!}$ cu $1 \leq t \leq n$. Calculăm
$xy^izu^iv = a^l a^{ti} a^{n-(l+t)} b^s b^{ri} b^{n+n!-(r+s)} c^{n+2n!} = a^{n+t(i-1)} b^{n+n!+r(i-1)} c^{n+2n!}$.

Deoarece $1 \leq t \leq n$, avem că t divide $n!$. Alegem i astfel încât $t(i-1) = 2n!$, deci $i = \frac{2n!}{t} + 1$.

Pe de o parte $xy^izu^iv \in L$ pentru orice $i \geq 0$, iar pe de altă parte pentru $i = \frac{2n!}{t} + 1$ avem $xy^izu^iv = a^{n+2n!} b^{n+n!+\frac{r}{t}2n!} c^{n+2n!} \notin L$, deoarece are acelaşi exponent pentru a şi c. Deci din nou contradicţie.
3) $x = a^l$, $y = a^t$, $z = a^{n-(l+t)} b^{n+n!} c^s$, $u = c^r$, $v = c^{n+2n!-(r+s)}$ cu $1 \leq t \leq n$. Calculăm
$xy^izu^iv = a^l a^{ti} a^{n-(l+t)} b^{n+n!} c^s c^{ri} c^{n+2n!-(r+s)} = a^{n+t(i-1)} b^{n+n!} c^{n+2n!+r(i-1)}$.

Deoarece $1 \leq t \leq n$, avem că t divide $n!$. Alegem i astfel încât $t(i-1) = n!$, deci $i = \frac{n!}{t} + 1$.

Din teorema lui Ogden $xy^izu^iv \in L$ pentru orice $i \geq 0$. Pe de altă parte pentru $i = \frac{n!}{t} + 1$ avem $xy^izu^iv = a^{n+n!} b^{n+n!} c^{n+2n!+\frac{r}{t}n!} \notin L$, deoarece are acelaşi exponent pentru a şi b. Contradicţie.

Deoarece în toate cele trei cazuri posibile am obţinut contradicţie, rezultă că limbajul $L = \{a^ib^jc^k \mid i, j, k \geq 1, i \neq j, i \neq k\}$ nu este de tip 2.

Tema de control (4)

1) Să se arate că limbajul $L = \{a^nb^{n+2}c^{n+3} \mid n \geq 1\}$ nu este limbaj de tip 2.

2) Să se arate că limbajul $L = \{a^ib^jc^k \mid i, j, k \geq 1, i \neq j, i \neq k, j \neq k\}$ nu este de tip 2.

Tema 5

Familiile de limbaje L_0 şi L_1

Dupã ce am studiat familiile de limbaje L_3 şi L_2, acum ne vom preocupa de proprietãţile familiilor L_0 şi L_1.

1. Proprietãţi de închidere pentru familiile L_0 şi L_1

Înainte de a prezenta proprietãţile de închidere vom da un rezultat care ne aratã cã o subclasã de gramatici de tip 0 sau de tip 1 au aceeaşi putere de generare ca şi întreaga clasã de gramatici de tip 0 sau 1.

Teorema 5.1.1. Pentru orice gramaticã $G=(V_N,V_T,x_0,P)$ de tip 0 sau 1, existã o gramaticã $G=(V_N,V_T,x_0,P')$ de acelaşi tip cu G şi echivalentã cu G astfel încît dacã o regulã din P, $u \to v$, conţine un simbol terminal ea are forma $x \to i$, cu $x \in V_N'$ şi $i \in V_T$.

Observaţie 5.1.2. O gramaticã de tip 0 sau 1 este sub formã standard dacã regulile sale sunt de forma $\alpha \to$ cu $\alpha, \in V_N^*$ sau de forma $x \to i$ cu $x \in V_N$ şi $i \in V_T$.

Observaţie 5.1.3. Conform teoremei 5.1 pentru orice gramaticã G de tip 0 sau 1, existã o gramaticã G sub formã standard echivalentã cu G.

Observaţie 5.1.4. Orice limbaj L$\in L_0$ (L$\in L_1$) poate fi generat de o gramaticã sub formã standard de tip 0 (sau 1).

Teorema 5.1.5. Familiile limbajelor L_0 şi L_1 sunt închise la produs.

Demonstraţie. Vom face demonstraţia pentru L_1; analog se face şi pentru $\mathcal{L}_0$. Fie $L_1, L_2 \in \mathcal{L}_1$. Există gramaticile $G_1 = (V_N^1, V_T^1, x_0^1, P_1)$ și $G_2 = (V_N^2, V_T^2, x_0^2, P_2)$ de tip 1 care satisfac condițiile din teorema precedentă și în plus $L_1 = L(G_1)$ și $L_2 = L(G_2)$. Putem considera $V_N^1 \cap V_N^2 = \emptyset$. Considerăm gramatica $G = (V_N^1 \cup V_N^2 \cup \{x_0\}, V_T^1 \cup V_T^2, x_0, P_1 \cup P_2 \cup \{x_0 \to x_0^1 x_0^2\})$, cu $x_0 \notin V_N^1 \cup V_N^2$. Această gramatică este de același tip cu G_1 și G_2. Vom demonstra că $L(G) = L_1 \cdot L_2$.

Să demonstrăm incluziunea $L(G) \subseteq L_1 \cdot L_2$. Fie $p \in L(G)$; există derivarea $x_0 \underset{G}{\Rightarrow} x_0^1 x_0^2 \overset{*}{\underset{G}{\Rightarrow}} p$. Deoarece $V_N^1 \cap V_N^2 = \emptyset$, avem că lui x_0^1 și simbolilor care se obțin din el se aplică numai reguli din P_1 și analog pentru x_0^2. Deci cuvântul $p = p_1 p_2$ cu $x_0^1 \overset{*}{\underset{G_1}{\Rightarrow}} p_1$ și $x_0^2 \overset{*}{\underset{G2}{\Rightarrow}} p_2$, prin urmare $p_1 \in L_1$, $p_2 \in L_2$ și $p \in L_1 \cdot L_2$.

Incluziunea inversă $L_1 \cdot L_2 \subseteq L(G)$. Din $p \in L_1 \cdot L_2$ avem $p = p_1 \cdot p_2$ și $p_1 \in L_1$, $p_2 \in L_2$. Deci avem $x_0^1 \overset{*}{\underset{G_1}{\Rightarrow}} p_1$, $x_0^2 \overset{*}{\underset{G2}{\Rightarrow}} p_2$ și $x_0 \underset{G}{\Rightarrow} x_0^1 x_0^2 \overset{*}{\underset{G}{\Rightarrow}} p_1 p_2$, $p \in L(G)$.

Dăm fără demonstrație următoarea teoremă.

Teorema 5.1.6. Pentru orice gramatică de tip 1 sau 0 există o gramatică G\` de același tip cu G, astfel încât $L(G') = L(G) - \{\lambda\}$

Teorema 5.1.7. Familiile de limbaje $\mathcal{L}_0$ și $\mathcal{L}_1$ sunt închise la iterație.

Demonstrație. Deoarece $(L - \{\lambda\})^* = L^*$, rezultă că nu restrângem generalitatea dacă presupunem că $\lambda \notin L$. Conform Teoremei 5.1., putem presupune că $L = L(G)$, unde $G = (V_N, V_T, x_0, P)$ și regulile care conțin terminali sunt de forma $x \to i, x \in V_N$ și $i \in V_T$.

Considerãm gramatica

$$G_* = (V_N \cup \{y_0, y_1\}, V_T, y_0, P \cup \{y_0 \to x_0, y_0 \to y_1 x_0\} \cup \{y_1 i \to y_1 x_0 i, y_1 i \to x_0 i \mid i \in V_T\})$$

Gramatica G_* este de acelaşi tip cu G, pentru cã regulile adaugate nu schimbã tipul (0 sau 1) gramaticii G_*. Sã demonstrãm cã $L^* = L(G_*)$.

Sã demonstrãm întâi incluziunea $\boldsymbol{L^* \subseteq L(G_*)}$. Fie $p \in L^*$; atunci $p \in L^k, k \geq 0$.

1) $k = 0$, atunci $p = \lambda$ şi avem $y_0 \underset{G_*}{\Rightarrow} \lambda$.

2) $k \geq 1$, atunci $p = p_1 ... p_k$ şi $p_j \in L, p_j \neq \lambda, 1 \leq j \leq k$. Din $p_j \in L$ rezultă $x_0 \overset{*}{\underset{G}{\Rightarrow}} p_j, 1 \leq j \leq k$. În acest caz considerăm derivarea

$$y_0 \underset{G^*}{\Rightarrow} y_1 x_0 \overset{*}{\underset{G^*}{\Rightarrow}} y_1 p_k \overset{*}{\underset{G^*}{\Rightarrow}} y_1 x_0 p_k \overset{*}{\underset{G^*}{\Rightarrow}} ... \overset{*}{\underset{G^*}{\Rightarrow}} y_1 p_2 ... p_k \overset{*}{\underset{G^*}{\Rightarrow}} x_0 p_2 ... p_k \underset{G^*}{\Rightarrow} p_{1...} p_k.$$

Deci avem $y_0 \overset{*}{\underset{G_*}{\Rightarrow}} p, p \in L(G_*)$ si $L^* \subseteq L(G_*)$.

Incluziunea $\boldsymbol{L(G_*) \subseteq L^*}$. Fie $p \in L(G_*)$; reazultă că avem derivarea $y_0 \underset{G_*}{\Rightarrow} q \underset{G_*}{\Rightarrow} ... \underset{G_*}{\Rightarrow} p$.

Dacă $q = \lambda$, avem $y_0 \Rightarrow \lambda$, deci $p = \lambda$.

Dacă $q = x_0$, atunci avem $x_0 \overset{*}{\underset{G}{\Rightarrow}} p$, deci $p \in L$.

Daca $q = y_1 x_0$, atunci avem derivarea

$y_0 \underset{G_*}{\Rightarrow} y_1 x_0 \overset{*}{\underset{G_*}{\Rightarrow}} y_1 p_k \underset{G_*}{\Rightarrow} y_1 x_0 p_k \underset{G_*}{\Rightarrow} y_1 p_{k-1} p_k \underset{G_*}{\Rightarrow} ... \overset{*}{\underset{G_*}{\Rightarrow}} p_1 ... p_k$ şi în plus $x_0 \overset{*}{\underset{G_*}{\Rightarrow}} p_j, 1 \leq j \leq k$, deci

$p_j \in L$. Din $p_j \in L, 1 \leq j \leq k$, rezulta $p \in L^k \subseteq L^*$, deci $L(G_*) \subseteq L^*$.

Din cele două incluziuni, rezulta egalitatea dorită.

Dăm în continuare o teoremă de închidere pentru toate familiile de limbaje Lj, $0 \leq j \leq 3$.

Teorema 5.1.8. Familiile Lj, $1 \le j \le 1 \le j \le 3$, sunt închise la operația de oglindire.

Demonstratie. Fie L un limbaj de tip $j, 0 \le j \le 3$. Există o gramatica $G = (V_N, V_T, x_0, P)$ de tip $j, 0 \le j \le 3$, cu $L(G) = L$. Consideram gramatica $G^{'} = (V_N, V_T, x_0, P^{'})$, unde $P^{'} = \{\tilde{u} \to \tilde{v} \mid u \to v \in P\}$. Gramatica $G^{'}$ este de același tip cu G. Să aratam ca $\mathring{L} = L(G^{'})$, unde $\mathring{L} = \{\mathring{p} \mid p \in L\}$. Deci trebuie sa aratam ca $p \in L$ dacă și numai daca $\mathring{p} \in L(G^{'})$, adica $x_0 \overset{*}{\Rightarrow} P$ daca și numai daca $x_0 \overset{*}{\underset{G}{\Rightarrow}} \mathring{p}$.

Pentru aceasta este suficient sa aratam ca $w_1 \underset{G}{\Rightarrow} w_2$ daca și numai daca $\tilde{w}_1 \underset{G^{'}}{\Rightarrow} \tilde{w}_2$.

Daca $w_1 \underset{G}{\Rightarrow} w_2$, atunci $w_1 = w_1^{'} u w_1^{''}$, $w_2 = w_1^{'} v w_1^{''}$ și $u \to v \in P$.

Dar atunci avem $\tilde{u} \to \tilde{v} \in P^{'}$, $\mathring{w}_1 = \tilde{w}_1^{''} \tilde{u} \mathring{w}_1^{'}$ și $\mathring{w}_1 = \tilde{w}_1^{''} \tilde{v} \mathring{w}_1^{'}$, deci $\mathring{w}_1 \underset{G'}{\Rightarrow} \tilde{w}_2$.

Invers, daca $\mathring{w}_1 \underset{G'}{\Rightarrow} \tilde{w}_2$, atunci avem $\mathring{w}_1 = \tilde{w}_1^{''} \tilde{u} \mathring{w}_1^{'}$, $\tilde{w}_2 = \tilde{w}_1^{''} \tilde{v} \mathring{w}_1^{'}$ și $\tilde{u} \to \tilde{v} \in P'$. Dar atunci avem $w_1 = w_1^{'} u w_1^{''}$, $w_2 = w_1^{'} v w_1^{''}$ și $u \to v \in P$, deci $w_1 \underset{G}{\Rightarrow} w_2$.

2. Gramatici monotone și limbaje independente de context

Definitie 5.2.1. O gramtica $G = (V_N, V_T, x_0, P)$ se numeste monotona daca pentru orice regula $u \to v \in P$, avem $|u| \le |v|$.

ObservatiŠe 5.2.2. O gramaticaŠ de tip 1 care nu contiŠne regula $x_0 \to \lambda$ este o gramatica monotonã.

Definitie 5.2.3. Se numeste *ponderea* gramaticii G, $pond(G) = \max\{|v| \,/\, u \to v \in P\}$.

TeoremaŸ 5.2.4. Pentru orice gramaticaŸ monotonaŸ $G = (V_N, V_T, x_0, P)$, exista o gramatica monotona $G' = (V_N', V_T', x_0, P')$, de pondere cel mult doi, echivalentaz cu G.

Teorema– 5.2.5. Orice limbaj L generat de o gramatica– monotona– poate fi generat de o gramatica) senzitiva) de context.

Demonstratie™: Fara™™a restrânge generalitatea putem considera, conform teoremei precedente, ca¼ gramatica monotona¼ $G = (V_N, V_T, x_0, P)$ care genereaza limbajul L are ponderea cel mult doi. Deci regulile gramaticii G sunt de forma:

1) $A \to a$, $A \in V_N$ si $a \in V_T$, contextul (λ, λ);
2) $A \to B$, $A, B \in V_N$ contextul (λ, λ);
3) $A \to BC$, $A, B, C \in V_N$ contextul (λ, λ);
4) $AB \to AD$, $A, B, D \in V_N$ contextul (A, λ);
5) $AB \to CB$, $A, B, C \in V_N$ contextul (λ, B);
6) $AB \to CD$, $A \neq C$ si $B \neq D$, nu mai este senzitiva` de context.

O regula, de forma 6) o vom înlocui cu reguli de forma 4) si 5), si anume:

6.1) $AB \to EB$, contextul (λ, B)
6.2) $EB \to EF$, contextul (E, λ)
6.3) $EF \to CF$, contextul (λ, F)
6.4) $CF \to CD$, contextul (C, λ)

Consideram $G' = (V_N', V_T, x_0, P')$ unde

$V_N' = V_N \cup \{E, F \mid AB \to CD \in P, A \neq C, B \neq D\}$

$P' = P - \{AB \to CD \in P\} \cup \{AB \to EB, EB \to EF, EF \to CF, CF \to CD \mid AB \to CD \in P\}$

Se poate arata ca cele doua gramatici sunt echivalente.

Teorema 5.2.6. Orice limbaj L$\in \mathcal{L}_1$ care nu contine λ poate fi generat de o gramatica monotona.

Demonstratie. Din L$\in \mathcal{L}_1$ si $\lambda \notin L$ rezulta ca exista o gramatica $G = (V_N, V_T, x_0, P)$ senzitiva de context care nu contine $x_0 \to \lambda$ si care genereaza L. Dar regulile gramaticii G sunt de forma $uxv \to urv$ cu $x \in V_N$ si $r \in V^+$, deci avem $|uxv| \le |urv|$ si prin urmare gramatica G este monotona.

3. Masini Turing si limbaje de tip 0

Vom prezenta masinile Turing ca acceptori de limbaje.

3.1. Masini Turing cu o banda de intrare cu o cale infinita

Un model pentru o masina Turing cu o banda de intrare cu o cale infinita este format dintr-o banda marginita la stânga, împartita în locatii – în fiecare locatie se plaseaza câte un simbol dintr-un alfabet – si o unitate de control prevazuta cu un cap de citire/scriere. În urma operatiei de citire/scriere capul se poate deplasa la stânga sau dreapta cu o locatie. Prezentam mai jos în Figura 5.1 modelul considerat.

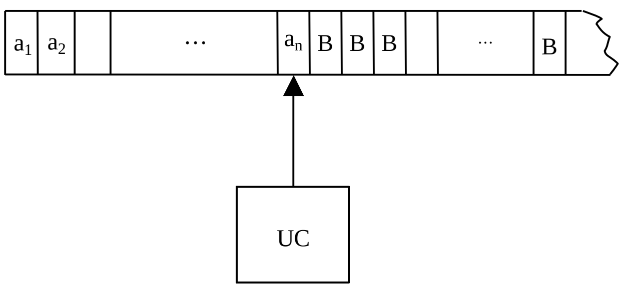


Figura 5.1. Model masina Turing

O mașină Turing face următoarele operații:

- cerceteaza o locatie pe banda de intrare si înlocuieste simbolul scris în ea;
- schimbă starea;
- mișcă capul de citire la dreapta sau la stânga cu o locație;

Formal, o mașină Turing (TM) o definim prin:

$$M = (S, \Sigma, \Gamma, \delta, s_0, B, F)$$

unde:

- S este o *mulțime finită de stări*;
- Σ este *alfabetul de intrare*, $\Sigma \subseteq \Gamma$ și $B \notin \Sigma$;
- Γ este o mulțime finită de simboli, *alfabetul de lucru al mașinii Turing*;
- B un simbol din Γ, numit *blanc*;
- $s_0 \in S$ este *starea initiala*;
- $F \subseteq S$ este *mulțimea stărilor finale*;
- $\delta : S \times \Gamma \to S \times \Gamma \times \{L, R\}$ este *funcția de tranziție* și este o funcție parțială, deci pot exista perechi $(s, z) \in S \times \Gamma$ pentru care δ nu este definită.

Vom numi *configurație* (descriere instantanee) a lui M, o tripletă $\alpha_1 s \alpha_2$ cu $\alpha_1, \alpha_2 \in \Gamma^*$ și $s \in S$. Fie C mulțimea configurațiilor mașinii M.

Pe modelul considerat mai sus, o configurație $\alpha_1 s \alpha_2$ are următoarea semnificație: s semnifică starea curentă a unității centrale; α_1 reprezintă cuvântul format din simbolii cuprinși între marginea din stânga a benzii și capul de citire iar α_2 este cuvântul format din simbolii cuprinși între capul de citire (inclusiv simbolul vizat) și ultimul simbol neblank de pe bandă.

Pe mulțimea configurațiilor C definim o relație binară $\vdash_M \subseteq C \times C$, numită "*relație de tranziție*".

1) Configurația $x_1 x_2 \ldots x_{i-1} s x_i \ldots x_n \vdash_M x_1 \ldots x_{i-2} s' x_{i-1} y x_{i+1} \ldots x_n$ dacă $i > 1$ și $\delta(s, x_i) = (s', y, L)$

În cazul $i = 1$, pentru că nu există poziția $i - 1$, ceea ce înseamnă - pe model - capul de citire/scriere nu se poate misca la stânga marginii din stânga.

Dacă există sufix în $x_{i-1} y x_{i+1} \ldots x_n$ format din blanc, acest sufix se sterge din acest cuvânt.

2) Configuratia $x_1x_2\ldots x_{i-1}sx_i\ldots x_n \vdash_M x_1\ldots x_{i-1}ys'x_{i+1}\ldots x_n$ daca $\delta(s,x_i)=(s',y,R)$.

Vom nota cu $\vdash_M^*$ închiderea tranzitiva si reflexiva a relatiei $\vdash_M$. Putem renunta la indicele M atunci când nu este posibilitatea de confuzie si scriem $\vdash^*$ în loc de $\vdash_M^*$.

Definim limbajul acceptat de o masina Turing M prin $L(M)=\{w \mid w\in\Sigma^*, s_0w \vdash_M^* \alpha_1 s\alpha_2, s\in F \text{ si } \alpha_1,\alpha_2\in\Gamma^*\}$.

Limbajul acceptat de masina Turing M este multimea cuvintelor $w\in\Sigma^*$, care, plasate pe banda de intrare în partea stânga a benzii, unitatea centrala fiind în starea s_0 si capul de citire/scriere pozitionat pe prima locatie, în urma tranzitiilor definite ca mai înainte, se ajunge într-o stare finala.

Vom presupune ca o masina Turing se opreste ori de câte ori intrarea este acceptata, iar pentru cuvintele neacceptate este posibil sa nu se opreasca.

Definitia 5.3.1.1. Un limbaj acceptat de o masina Turing se numeste *limbaj recursiv-enumerabil*.

Enumerabilitatea deriva din faptul ca aceste limbaje sunt formate din cuvinte care pot fi listate.

3.2. Masîni Turing cu banda de intrare cu doua cai infinite

O masina Turing cu banda de intrare infinita în ambele directii, este definita prin $M=(S,\Sigma,\Gamma,\delta,s_0,B,F)$, unde cele sapte elemente $S,\Sigma,\Gamma,,\delta,s_0,B$ si F au aceleasi semnificatii ca în modelul precedent.

Functia de tranzitie $\vdash_M$ este definita ca în modelul original cu, exceptia cazului când $sx\alpha \vdash_M s'By\alpha$, daca. $\delta(s,x)=(s',y,L)$.

Pentru masini Turing cu banda de intrare cu o cale nu se face tranzitia.

În plus, $sx\alpha \vdash_M s'\alpha$ daca $\delta(s,x)=(s',B,R)$ (în cazul precedent B apare în stânga lui s').

Limbajul acceptat de o masina Turing cu banda de intrare cu doua cai este:

$$L(M)=\{w \mid w\in\Sigma^*, s_0w \vdash_M^* \alpha_1 s\alpha_2, s\in F, \alpha_1,\alpha_2\in\Gamma^*\}.$$

Teorema 5.3.2.1. Un limbaj L este acceptat de o maşină Turing cu bandă de intrare cu două căi infinite dacă si numai dacă el este acceptat de o maşină Turing cu o cale infinită.

Demonstraţie. Demonstraţia că o maşină Turing cu banda de intrare cu două căi infinite poate simula o maşină Turing cu o bandă de intrare cu o cale infinită este uşoară. În primul rând marcăm locaţia din stânga poziţiei iniţiale cu un simbol special "$*$" şi apoi simulăm maşina Turing cu o cale infinita. Daca în timpul simularii se atinge locatia marcata cu "$*$", se trece într-o stare nouă care nu o mai părăseşte şi care nu este starea finală. Fie $M_1 = (S_1, \Sigma_1, \Gamma_1, \delta_1, s_0^1, B, F_1)$ o maşină Turing cu banda de intrare cu o cale infinită şi fie $M_2 = (S_1 \cup \{s', s_0^2, \bar{s}\}, \Sigma_1, \Gamma_1 \cup \{*\}, \delta_2, s_0^2, B, F_1)$ cu δ_2 definita prin:

1. $\delta_2(s_0^2, a) = (\bar{s}, a, L)$;
2. $\delta_2(\bar{s}, B) = (s_0, *, R)$;
3. $\delta_2(s, a) = \delta_1(s, a), \forall s \in S_1, \forall a \in \Gamma_1, \forall (s, a) \in dom(\delta_1)$;
4. $\delta_2(s, *) = (s', *, L)$.

Este evident că $L(M_1) = L(M_2)$.

Invers, să arătăm că o maşină Turing M_2 cu o bandă de intrare cu două căi infinite poate fi simulata de o masina Turing M_1 cu o banda de intrare cu o cale infinită. Vom construi M_1 cu banda de intrare cu două piste - o pistă reprezintă locaţiile din dreapta poziţiei iniţiale (inclusiv poziţia iniţială), cealaltă pistă reprezintă locaţiile din stânga locaţiei iniţiale ca în Figura 5.2.

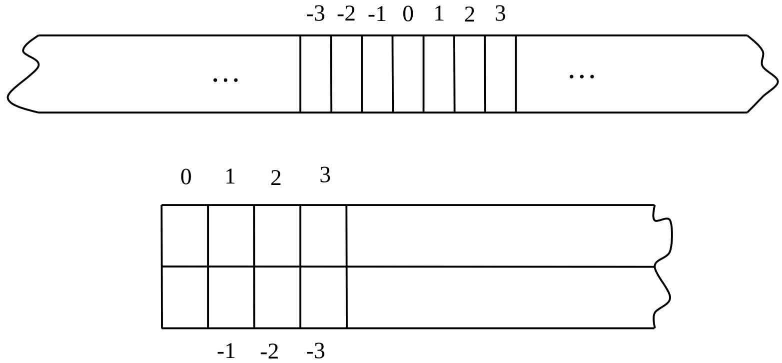


Figura 5.2. Construcţia benzii lui M_1 din cea a lui M_2

Prima locatie a lui M_1 contine pe pista de jos, un simbol special "μ" care semnifica ca este cea mai din stânga celula si ca M_1 nu se poate misca la stânga acestei locatii. Masina Turing M_1 va fi construita sa simuleze masina Turing M_2 în modul urmator:

când M_2 lucreaza la dreapta pozitiei initiale, masina M_1 lucreaza pe pista de sus, facând aceleasi operatii ca si M_2; când M_2 lucreaza la stânga pozitiei initiale, M_1 lucreaza pe pista de jos, facând aceleasi înlocuiri ca si M_2 si miscându-se în directia opusa lui M_2.

Fie masina Turing $M_2 = (S_2, \Sigma_2, \Gamma_2, \delta_2, s_2, B, F_2)$; atunci $M_1 = (S_1, \Sigma_1, \delta_1, s_1, B, F_1)$ unde:

- $S_1 = \{s_1\} \cup \{[s,U],[s,D] \mid s \in S_2\}$, U semnifica ca M_1 lucreaza pe pista de sus si D semnifica faptul ca M_1 lucreaza pe pista de jos.
- $\Sigma_1 = \{[a,B], a \in \Sigma\}$;
- $\Gamma_1 = \{[x,y] \mid x, y \in \Gamma_2, y$ poate fi si $\mu\}$
- Blancul în M_1 este identificat cu $[B,B]$;
- $F_1 = \{[s,U],[s,D] \mid s \in F_2\}$.

Functia δ_1 o definim dupa cum urmeaza:

1. $\delta_1(s_1,[a,B]) = ([s,U],[x,\mu],R)$ daca $\delta_2(s_2,a) = (s,x,R)$ pentru $a \in \Sigma_2 \cup \{B\}$;
2. $\delta_1(s_1,[a,B]) = ([s,D],[x,\mu],R)$ daca $\delta_2(s_2,a) = (s,x,L)$ pentru $a \in \Sigma_2 \cup \{B\}$;
3. $\delta_1([s,U],[x,y]) = ([s',U],[z,y],A)$ daca $\delta_2(s,x) = (s',z,A)$ pentru $[x,y] \in \Gamma_1$, $y \neq \mu$ si $A = L$ sau $A = R$;
4. $\delta_1([s,D],[x,y]) = ([s',D],[x,z],\bar{A})$ daca $\delta_2(s,y) = (s',z,A)$ pentru $[x,y] \in \Gamma_1$, $y \neq \mu$, $A = L$ daca $\bar{A} = R$ si invers;
5. $\delta_1([s,U],[x,\mu]) = \delta_1([s,D],[x,\mu]) = ([s',C],[y,\mu],R)$ daca; $\delta_2(s,x) = (s',y,A)$, unde $C = U$, daca $A = R$ si $C = D$, daca $A = L$.

Din definirea lui δ_1, se vede ca M_1 simuleaza M_2, pastrând starea lui M_2 în prima componenta a starii, facând aceleasi înlocuiri ca si pe pista

de sus - dacă M_2 lucrează la dreapta poziției inițiale - și pe pista de jos - dacă M_2 lucrează la stânga poziției inițiale. Când M_2 trece prin poziția inițială, de la dreapta la stânga, sau invers, mașina M_1 trece de pe pista de sus pe pista de jos, sau invers.
Este evident că $L(M_1) = L(M_2)$.

3.3. Mașini Turing cu mai multe benzi

Un model pentru mașini Turing, este cel din Figura 5.3.

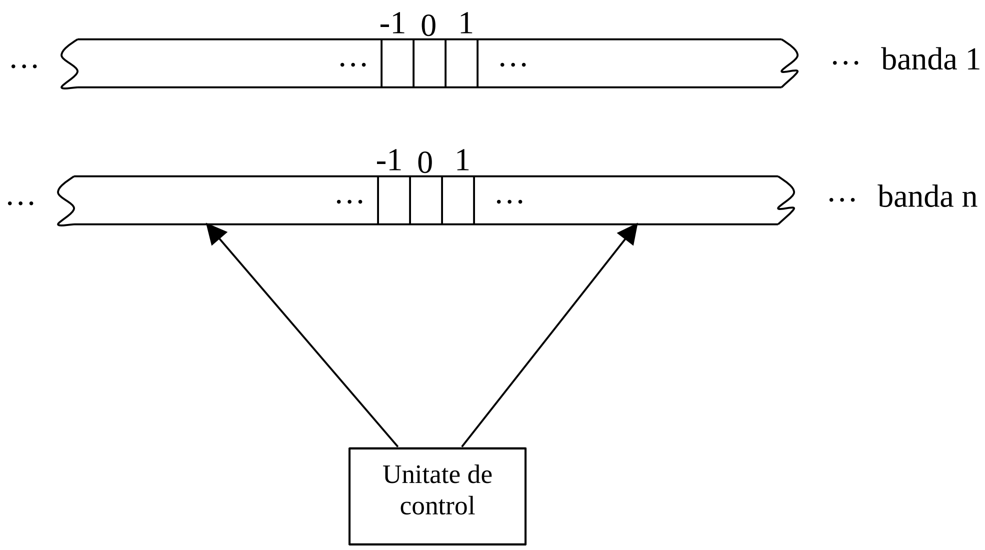


Figura 5.3. Mașina Turing cu mai multe benzi

Modelul pentru mașina Turing cu mai multe benzi constă din:

- n benzi infinite în ambele direcții;
- o unitate de control prevăzută cu n capete de citire/scriere care vizează câte o locație de pe fiecare bandă și care se mișcă independent. Mișcarea depinde de starea unității de control și de simbolul cercetat pe fiecare bandă de capul de citire/scriere.

În urma cercetării simbolurilor de pe cele n benzi şi de starea în care se afla unitatea de control, masina Turing face urmatoarele operatii:

- schimbă starea unităţii de control;
- scrie un nou simbol în fiecare locaţie cercetată de capetele de citire/scriere;
- mişcă capetele sale de citire/scriere, independent unul de altul, cu o locaţie la stânga sau la dreapta.

Formal, o maşină Turing cu n benzi se poate defini în felul următor: $M=(S,\Sigma,\Gamma,\delta,s_0,B,F)$, unde

- S,Σ,Γ,s_0,B şi F sunt ca la maşina Turing cu o bandă;
- $\delta : S\times\Gamma^n \to S\times\Gamma^n\times\{L,R\}^n$.

O configuraţie este $(\alpha_1 s_1, \alpha_2 s_2, \ldots a_k s_k)$. O tranziţie de la o configuraţie la alta, se face ca la maşinile Turing cu o bandă, lucrând pe fiecare componentă, ţinând cont de modificarea şi de mişcarea definită de funcţia de tranziţie δ pentru componenta respectivă.

$$L(M)=\{w \mid w\in\Sigma^*, (s_0 w, s_0, \ldots, s_0) \overset{*}{\underset{M}{\vdash}} (\alpha_1 s_1, \ldots, \alpha_k s_k), s\in F\}$$

Pentru a vedea dacă un cuvânt w este acceptat, se pune cuvântul pe banda 1, pe celelalte benzi se pune blanc, se poziţionează capetele de citire/scriere pe poziţiile originale, unitatea de control se pune în starea s_0 şi se dă drumul maşinii să funcţioneze, după funcţia δ. Dacă maşina intră într-o stare finală, se acceptă w; în caz contrar, cuvântul w nu se acceptă.

Dăm, fără demonstraţie, următoarea teoremă:

Teorema 5.3.3.1. *Un limbaj, acceptat de o maşină Turing cu mai multe benzi, este acceptat de o maşină Turing cu o bandă.*

3.4. Maşini Turing nedeterministe

Un model pentru o maşină Turing nedeterministă constă dintr-o unitate de control şi o bandă de intrare cu o cale infinită. Pentru fiecare stare şi simbol cercetat pe banda de intrare, maşina are un număr finit de posibilităţi de alegere, fiecare alegere constând din o nouă stare a unităţii de control, un simbol scris pe banda si o deplasare stânga sau dreapta a capului de citire/scriere.

Formal, o masina Turing nedeterminista este:
$M = (S, \Sigma, \Gamma, \delta, s_0, B, F)$, unde
$\delta : S \times \Gamma \to P_f(S \times \Gamma \times \{L, R\})$.
Celelalte elemente ale masinii Turing M sunt definite ca la masina Turing determinista, iar P_f înseamna multimea partilor finite.
Configuratiile, tranzitiile $\vdash_M$ si $\vdash_M^*$ se definesc în mod analog ca la masini Turing deterministe. Limbajul acceptat de o masina Turing nedeterminista este $L(M) = \{w \mid w \in \Sigma^*, \exists s_0 w \vdash_M^* \alpha_1 s \alpha_2, s \in F\}$.
Ca si la automate finite, nedeterminismul nu mareste puterea de generare a masinilor Turing. De fapt, combinarea nedeterminismului cu orice extensie privind benzile de intrare (benzi de intrare cu doua capete infinite, benzi de intrare multiple) nu mareste puterea de generare.

Teorema 5.3.4.1. *Daca L este un limbaj acceptat de o masina Turing nedeterminista M_1, atunci L este acceptat de o masina Turing determinista M_2.*

Se poate arãta cã familia limbajelor recursiv enumerabile (familia limbajelor acceptate de masini Turing) este egalã cu familia limbajelor de tio 0. Au loc urmãtoarele teoreme:

Teorema 5.3.4.2. Dacã L$\in$ L_0 atunci el este un limbaj acceptat de o masinã Turing.

Are loc si teorema reciprocã.

Teorema 5.3.4.3. Dacã L este un limbaj recursiv enumerabil (adicã acceptat de o masinã Turing) atunci el este un limbaj de tip 0.

4. Automate liniar mãrginite si limbaje de tip 1

Vom introduce un nou dispozitiv de acceptare de limbaje si vom arata ca acest dispozitiv va accepta tocmai clasa limbajelor senzitive de context L_1.
Definitia 5.4.1. Un *automat liniar marginit* (LBA) este o masina Turing

nedeterminista care satisface urmatoarele doua conditii:
1) Alfabetul de intrare Σ include doi simboli speciali μ si $ª$, care se numesc *marcatori la stânga si la dreapta*, respectiv;
2) $\delta(s,\mu) = (s,\mu,R)$ si $\delta(s,ª) = (s,ª,L)$.
Un automat liniar marginit este de forma $M = (S,\Sigma,\Gamma,\delta,s_0,\mu,ª,F)$, unde $S,\Sigma,\Gamma,\delta,s_0$ si F sunt, ca la masina Turing, nedeterministe; μ si $ª$ sunt doi simboli speciali din Σ si blancul nu face parte din Γ .
Limbajul acceptat de un automat liniar marginit, M , este $L(M) = \{w \mid w \in (\Sigma - \{\mu, ª\})^*,\ s_0\mu wª \overset{*}{\underset{M}{\vdash}} \alpha s\ ,\ s \in F\}$.
Din definitia lui $L(M)$, se vede ca marcatorii μ si $ª$ nu fac parte din cuvântul acceptat de automatul liniar marginit; acesti simboli sunt pusi pe banda numai pentru a delimita cuvântul de intrare.
Se poate arata ca familia limbajelor acceptate de automatele liniar marginite este familia L_1 .

Teorema 5.4.2. *Daca un limbaj* $L \in L_1$ *si* $\lambda \notin L$ *, atunci exista un automat liniar marginit care sa accepte* L .

Demonstratie. Demonstratia este similara cu cea a teoremei 4.2.5.1. Daca $L \in L_1$, exista o gramatica $G = (V_N,V_T,x_0,P)$ cu $L = L(G)$. Un cuvânt $w \in L$ daca si numai daca $x_0 \overset{*}{\underset{G}{\Rightarrow}} w$.

Noi vom considera un automat liniar marginit, cu banda de intrare cu doua piste, pe prima pista vom plasa cuvântul w si pe a doua pista vom simula derivatiile din G , pornind cu x_0 .

Automatul liniar marginit face urmatoarele operatii:

1. Daca continutul pistei a doua este α , alege nedeterminist o pozitie i în α , $1 \le i \le |\alpha|$.
2. Se selecteaza o regula $u \to v \in P$.
3. Daca u apare în α începând cu pozitia i , înlocuim u prin v , eventual deplasând simboli la dreapta daca $|u| < |v|$. Daca rezultatul înlocuirii lui u cu v este mai lung decât w , se respinge acest cuvânt si se începe cu x_0 pe pista a doua si se trece la pas 1.
4. Comparam cuvântul rezultat pe pista a doua, cu cuvântul w de pe pista unu. Daca continutul celor doua piste este

identic, se acceptă w, dacă nu, se trece la pasul 1.

Dacă nu se mai poate aplica nici o regulă şi conţinutul celor două piste diferă, cuvântul w se respinge.
Din algoritmul de mai sus, se vede că $w \in L(M)$ dacă şi numai dacă există o derivare $x_0 \overset{*}{\underset{G}{\Rightarrow}} w$, deci $w \in L(G)$, ceea ce ne arata ca $L(G) = L(M)$.
Să demonstrăm acum că orice limbaj din $\mathcal{L}_{LBA}$ este un limbaj din $\mathcal{L}_1$.
Teorema 5.4.3. *Fie $L = L(M)$ cu M un automat liniar mărginit, $M = (S, \Sigma, \Gamma, \delta, s_0, \mu, ª, F)$, atunci $L - \{\lambda\}$ este un limbaj din $\mathcal{L}_1$.*

Exerciţii rezolvate:

1)Fie maşina Turing
$M = \left(\{s_0, s_1, s_2, s_3, s_4\}, \{0,1\}, \{0,1,x,yB\}, \delta, s_0, B, \{s_4\}\right)$ cu
δ definit pentru:

δ	0	1	x	y	B
s_0	(s_1, x, R)	$\varnothing$	$\varnothing$	(s_3, y, R)	$\varnothing$
s_1	$(s_{1,} 0, R)$	(s_2, y, L)	$\varnothing$	(s_1, y, R)	$\varnothing$
s_2	$(s_2, 0, L)$	$\varnothing$	(s_0, x, R)	(s_2, y, L)	$\varnothing$
s_3	$\varnothing$	$\varnothing$	$\varnothing$	(s_3, y, R)	(s_4, B, R)
s_4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Să se arate că $L(M) = \{0^n 1^n \mid n \geq 1\}$.
Rezolvare:

Să demonstrăm că $\{w\mathring{w} \mid w \in \{0,1\}^*\} \subseteq L(M)$. Să considerăm câteva cazuri particulare:

1) $w=\lambda$, atunci $\tilde{w}=\lambda$. Deci avem $s_0 \quad Bs_7$. Starea s_7 este starea finala¿ rezulta¿ ca¿ $\lambda\lambda$ este acceptat.

2) $w=01$, atunci $w=10$ si̊ avem:

$s_0 0110 \quad s_1 110 \quad {}^{*} 110 s_1 \quad 11 s_3 0 \quad 1 s_5 1 \quad {}^{*} s_5 B11 \quad s_0 11 \quad s_2 1 \quad 1 s_2 \quad s_4 1 \quad s_6 \quad s_0 \quad s_7$

3) Sa¿ consideram acum cazul general $w=i_1 ... i_k, k \geq 1$. Atunci $\tilde{w}=i_k i_{k-1} ... i_1$. Avem urmatoarele succesiuni de tranzitii.

i) Cazul $i_1=0$

$s_0 0 i_2 - i_k i_k ... i_2 0 \quad s_1 i_2 ... i_k i_k ... i_2 0 \quad {}^{*} i_2 ... i_k i_k ... i_2 0 s_1 \quad i_2 ... i_k i_k ... i_2 s_3 0 \quad i_2 ... i_k i_k ... i_3 s_5 i_2 \quad {}^{*}$

$s_5 B i_2 ... i_k i_k ... i_2 \quad s_0 i_2 ... i_k i_k ... i_2$.

ii) Cazuk $i_1=1$ se trateazac analog.

$s_0 1 i_2 - i_k i_k ... i_2 1 \quad s_1 i_2 ... i_k i_k ... i_2 1 \quad {}^{*} i_2 ... i_k i_k ... i_2 1 s_1 \quad i_2 ... i_k i_k ... i_2 s_3 1$

$i_2 ... i_k i_k ... i_3 s_5 i_2 \quad {}^{*} s_5 B i_2 ... i_k i_k ... i_2 \quad s_0 i_2 ... i_k i_k ... i_2$.

DacaR avem: $s_0 i_1 ... i_k i_k ... i_1 \quad {}^{*} s_0 i_2 ... i_k i_k ... i_2 \quad .... \quad s_0 i_k i_k \quad s_0 \quad s_7$. Deci orice cuvânt de forma $i_1 ... i_k i_k ... i_1$ este acceptat de mas2ina Turing M.

Sa“demonstram acum ca“ $L(M) \subseteq \left\{ w\tilde{w} \mid w \in \{0,1\} \right\}$ Fie un cuvânt $p \in L(M)$. Fie $p=i_1 ... i_n, n \geq 0$ si̊ $i_j \in \{0,1\}$. Din $p \in L(M)$ avem $s_0 p \quad {}^{*} \alpha_1 s_7 \alpha_2$. Sa, vedem cum poate evolua masina M din s_0 în s_7. ConsideraS m $s_0 i_1 ... i_n$. Avem douaS cazuri: $i_1=0$ sau $i_1=1$. Ambele cazuri se trateaza3 analog as3 ca3 vom considera numai cazul $i_1=0$:

$s_0 0 i_2 ... i_n \quad s_1 i_2 ... i_n \quad {}^{*} i_2 ... i_n s_1 \quad i_2 ... i_{n-1} s_3 i_n$.

Daca[a] $i_n=1$ atunci $\delta(s_3,1)=\varnothing$, deci se blocheaza[a] Trebuie ca $i_n=0$. În acest caz avem: $i_2 ... i_{n-1} s_3 0 \quad i_2 ... i_{n-2} s_5 i_{n-1} \quad {}^{*} s_5 B i_2 ... i_{n-1} \quad s_0 i_2 ... i_{n-1}$.

Procedând în acelsÒ fel avem:

$i_1 = i_n$

$i_2 = i_{n-1}$

.
.
.
$i_j = i_{n-j+1}$

Daca n ar fi impar, $n = 2k+1$ atunci am obtine: $s_0 i_{k+1}$. Avem iar doua cazuri $i_{k+1} = 0$ si $i_{k+1} = 1$.

$s_0 0 \quad s_1 \quad s_2$ si ne-am blocat.

Analog cazul $i_{k+1} = 1$. $s_0 1 \quad s_2 \quad s_4$ si se blocheaza

Deci n trebuie sa fie par, $n = 2k$ sa avem $i_1 = 2k, i_2 = 2k-1, ..., i_k = i_{k+1}$.

Deci $p = i_1 ... i_k i_k ... i_1 = w\tilde{w}$.

Tema de control:

Fie masina Turing

$M = (\{s_0, s_1, s_2, s_3, s_4, s_5 s_6, s_7, s_8\}, \{a,b,c\}, \{a,b,c,x,y,B\}, \delta, s_0, B, \{s_8\})$

cu δ definit prin:

δ	a	b	c	x	y	B
s_0	(s_1, x, R)	(s_4, x, R)	(s_7, L, R)	$\varnothing$	$\varnothing$	$\varnothing$
s_1	(s_1, a, R)	(s_1, b, R)	(s_3, c, R)	$\varnothing$	$\varnothing$	$\varnothing$
s_2	(s_2, a, R)	(s_2, b, R)	(s_4, c, R)	$\varnothing$	$\varnothing$	$\varnothing$
s_3	$(s_5 y, L)$	$\varnothing$	$\varnothing$	$\varnothing$	(s_3, y, R)	$\varnothing$
s_4	$\varnothing$	$(s_6 y, L)$	$\varnothing$	$\varnothing$	(s_4, y, R)	$\varnothing$
s_5	$(s_5 a, L)$	(s_5, b, L)	(s_5, c, L)	(s_0, x, R)	$(s_5 y, L)$	$\varnothing$
s_6	$(s_6 a, L)$	(s_6, b, L)	(s_6, c, L)	(s_0, x, R)	$(s_6 y, L)$	$\varnothing$
s_7	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	(s_7, y, R)	(s_8, B, R)
s_8	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Sã se arate cã $L(M) = \{wcw \mid w \in \{a,b\}^*\}$.

Bibliografie

1. **Toader Jucan** - *Limbaje formale şi automate*, Editura Matrix Rom, Bucureşti, 1999, 162 p.
2. **Toader Jucan, Ştefan Andrei** – *Limbaje formale şi teoria automatelor.Teorie şi practicã*, Editura Universitãţii "Al. I. Cuza", Iaşi, 2002, 327p.
3. **Gheorghe Grigoras** - *Limbaje formale si tehnici de compilare*, Editura Universitãţii "Al. I. Cuza", Iaşi, 1985, 256p.
4. **Virgil Cãzãnescu** – *Introducere în teoria limbajelor formale*, Editura Academiei, Bucureşti, 1983.

Manualul de "Limbaje formale şi automate" pentru secţia ID

– Erată –

1. Lista corecţiilor de la **Tema 1**:
   - pag. 7, rândul 3 de sus, se va citi: ...neterminalii
   - pag. 17, rândul 1 de jos, se va citi: $G = (\{x_0, x_1, x_2\}, \ldots)$
   - pag. 18, rândul 3 de sus, se va citi: 3) $x_1x_2 \rightarrow x_2x_1$
   - pag. 18, rândul 9 de sus, se va citi: $G = (\ldots, \{a, b, c\}, A, P)$
   - pag. 19, rândul 4 de sus, se va citi: 6) $B \rightarrow \lambda$

2. La pag. 59, rândurile 16 şi 17 dispar (de la Teorema 3.18)

3. Lista corecţiilor de la **Tema 3**:
   - pag. 70, rândul 13 de sus, se va citi: $(\ldots, \{a, z_0\}, \delta, s_0, z_0, \{s_3\})$ cu ...
   - pag. 70, rândul 16 de sus, se va citi: 3) $\delta(s_1, b, a) = \{(s_2, \lambda)\}$
   - pag. 71, rândul 7 de sus, se va citi: 4) $\delta(s_2, b, b) = \{(s_2, bb)\}$

   - pag. 95, rândul 6 de jos, se va citi: $x_1 \ldots x_{i-1}sx_ix_{i+1} \ldots x_n \vdash x_1 \ldots x_{i-2}s'x_{i-1}yx_{i+1} \ldots x_n$
   - pag. 96, rândul 1 de jos, se va citi: $L(M) = \{w \mid w \in \Sigma^*, s_0w \vdash_M^* \alpha_1 s\, \alpha_2, s \in F, \alpha_1, \alpha_2 \in \Gamma^*\}$