Simplificarea gramaticilor independente de context. Gramatici regulate. Automate finite deterministe

1. Simplificarea gramaticilor independente de context

Gramaticile independent de context, deci şi gramaticile regulate, pot fi aduse la o formă mai simplă prin eliminarea a două tipuri de simboluri: **neobservabile** şi **inaccesibile**.

- **Neterminale neobservabile**

Definiția 1.1. Fie $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$ o gramatică independentă de context şi $X \in \boldsymbol{N}$. Spunem că X este **neterminal observabil** dacă există cel puţin o derivare de forma $X \overset{*}{\Rightarrow} w$, unde $w \in \boldsymbol{T}^*$. În caz contrar spunem că X este **neterminal neobservabil**.

Este evident faptul că numai neterminalele observabile sunt utile în generarea cuvintelor din $\boldsymbol{L}(\boldsymbol{G})$, deci producţiile care conţin în membrul stâng sau în cel drept neterminale neobservabile pot fi eliminate din $\boldsymbol{P}$.

Propoziția 1.1. Neterminalele observabile ale unei gramatici independente de context $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$ se pot obţine prin metoda şirului crescător de mulţimi, astfel:

$$\begin{cases} M_0 = \{X \in \boldsymbol{N} \mid \exists X \to \alpha \in \boldsymbol{P} \text{ cu } \alpha \in \boldsymbol{T}^*\} \\ M_{n+1} = M_n \cup \{X \in \boldsymbol{N} \mid \exists X \to \alpha \in \boldsymbol{P} \text{ cu } \alpha \in (\boldsymbol{T} \cup M_n)^*\} \end{cases}$$

Se observă că

$$M_0 \subset M_1 \subset \cdots \subset M_n \subset M_{n+1} \subset \boldsymbol{N}.$$

Deoarece mulţimea $\boldsymbol{N}$ este finită, rezultă că există un indice $k \geq 0$ pentru care şirul de mulţimi se stabilizează, respectiv

$$M_k = M_{k+1} = M_{k+2} = \cdots,$$

deci M_k este deci mulţimea neterminalelor productive. Putem elimina acum producţiile care conţin neterminale neobservabile, adică neterminalele din mulţimea $\boldsymbol{N} \backslash M_k$.

Exemplu 1.1. Se consideră gramatica independentă de context $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$, unde $\boldsymbol{N} = \{S, A, B, C, D\}$, $\boldsymbol{T} = \{a, b\}$ şi mulţimea producţiilor $\boldsymbol{P}$ este următoarea:

$$S \to AB|aBC \qquad (1)$$

$$A \to BA|a \qquad (2)$$

$$B \to b|AC \qquad (3)$$

Lect. univ. dr. TATIANA-CORINA DOSESCU
Facultatea de Informatică
Universitatea Titu Maiorescu

$$C \to AC|CB \qquad (4)$$

$$D \to AD|a \qquad (5)$$

$$\begin{aligned}
M_0 &= \{X \in \boldsymbol{N} \mid \exists X \to \alpha \in \boldsymbol{P} \text{ cu } \alpha \in \boldsymbol{T}^*\} = \{A, B, D\} \\
M_1 &= M_0 \cup \{X \in \boldsymbol{N} \mid \exists X \to \alpha \in \boldsymbol{P} \text{ cu } \alpha \in (\boldsymbol{T} \cup M_0)^*\} = \\
&= \{A, B, D\} \cup \{X \in \boldsymbol{N} \mid \exists X \to \alpha \in \boldsymbol{P} \text{ cu } \alpha \in \{A, B, D, a, b\}^*\} = \\
&= \{A, B, D\} \cup \{S, A, B, D\} = \{S, A, B, D\} \\
M_2 &= M_1 \cup \{X \in \boldsymbol{N} \mid \exists X \to \alpha \in \boldsymbol{P} \text{ cu } \alpha \in (\boldsymbol{T} \cup M_1)^*\} = \\
&= \{S, A, B, D\} \cup \{X \in \boldsymbol{N} \mid \exists X \to \alpha \in \boldsymbol{P} \text{ cu } \alpha \in \{S, A, B, D, a, b\}^*\} = \\
&= \{S, A, B, D\} \cup \{S, A, B, D\} = \{S, A, B, D\}
\end{aligned}$$

Se observă că în acest moment şirul de mulţimi s-a stabilizat deoarece $M_2 = M_1$, deci neterminalele observabile sunt cele din $M_2 = \{S, A, B, D\}$. Rezultă că singurul neterminal neobservabil este C. Putem simplifica gramatica $\boldsymbol{G}$, îl eliminăm pe *C* din ***N*** şi eliminăm din ***P*** producţiile în care acesta apare, obținând astfel o gramatică independentă de context

$\boldsymbol{G}' = (\boldsymbol{N}', \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P}')$ echivalentă cu $\boldsymbol{G}$ şi care nu conține neterminale neobservabile, unde

$\boldsymbol{N}' = \{S, A, B, D\}$, iar mulţimea producţiilor $\boldsymbol{P}'$ este următoarea:

$$S \to AB \qquad (1)$$

$$A \to BA|a \qquad (2)$$

$$B \to b|AC \qquad (3)$$

$$D \to AD|a \qquad (4)$$

Teorema 1.1. Fie $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$ o gramatică independentă de context. Există algoritm pentru a verifica dacă ***L(G)*** este o mulțime vidă sau nu.

Ca urmare, este suficient să se verifice dacă simbolul de start *S* este observabil adică:

L(G) ≠ ∅ ⟺ *S* este observabil.

- **Simboluri inaccesibile**

Definiția 1.2. Fie $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$ o gramatică independentă de context şi $X \in \boldsymbol{N} \cup \boldsymbol{T}$. Spunem că $X \in \boldsymbol{N} \cup \boldsymbol{T}$ este **simbol accesibil** dacă există cel puţin o derivare de forma

$S \overset{*}{\Rightarrow} \alpha X \beta$, unde $\alpha, \beta \in (\boldsymbol{N} \cup \boldsymbol{T})^*$.

În caz contrar spunem că X este **simbol inaccesibil**.

Lect. univ. dr. TATIANA-CORINA DOSESCU
Facultatea de Informatică
Universitatea Titu Maiorescu

Este evident faptul că numai simbolurile accesibile sunt utile în generarea cuvintelor din $\boldsymbol{L(G)}$, deci producţiile care conţin în membrul stâng sau în cel drept simboluri inaccesibile pot fi eliminate din $\boldsymbol{P}$.

Simbolurile accesibile ale unei gramatici independente de context $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$ se pot obţine prin metoda şirului crescător de mulţimi, astfel:

$$\begin{cases} M_0 = \{S\} \\ M_{n+1} = M_n \cup \{X \in \boldsymbol{N} \cup \boldsymbol{T} \mid \exists\, Y \in M_n \cap \boldsymbol{N} \text{ astfel încât } Y \to \alpha X \beta \in \boldsymbol{P} \text{ cu } \alpha, \beta \in (\boldsymbol{N} \cup \boldsymbol{T})^*\} \end{cases}$$

Se observă că $M_0 \subset M_1 \subset \cdots \subset M_n \subset M_{n+1} \subset \boldsymbol{N}$. Deoarece mulţimea $\boldsymbol{N}$ este finită, rezultă că există un indice $k \geq 0$ pentru care şirul de mulţimi se stabilizează, respectiv $M_k = M_{k+1} = M_{k+2} = \cdots$, deci M_k este deci mulţimea simbolurilor accesibile. Putem elimina acum producţiile care conţin simboluri inaccesibile, adică simbolurile din mulţimea $(\boldsymbol{N} \cup \boldsymbol{T}) \setminus M_k$.

Exemplul 1.2. Se consideră gramatica independentă de context $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$, unde $\boldsymbol{N} = \{S, A, B, D\}$, $\boldsymbol{T} = \{a, b, c\}$ şi mulţimea producţiilor $\boldsymbol{P}$ este următoarea:

$$S \to AB \qquad (1)$$

$$A \to BA \mid a \qquad (2)$$

$$B \to b \qquad (3)$$

$$D \to AD \mid c \qquad (4)\ .$$

Simbolurile accesibile ale gramaticii independente de context se pot obţine prin metoda şirului crescător de mulţimi, astfel:

$M_0 = \{S\}$

$M_1 = M_0 \cup \{A, B\} = \{S, A, B\}$

$M_2 = M_1 \cup \{a, b\} = \{S, A, B, a, b\}$

$M_3 = M_2.$

Se observă că în acest moment şirul de mulţimi s-a stabilizat deoarece $M_3 = M_2$, deci simbolurile accesibile sunt cele din $M_3 = \{S, A, B, a, b\}$. Rezultă că simbolurile inaccesibile sunt D şi *c.* Eliminăm pe D din $\boldsymbol{N}$ şi pe *c* din $\boldsymbol{T}$. Eliminăm din $\boldsymbol{P}$ producţiile în care acestea apar, obţinând astfel o gramatică independentă de context $\boldsymbol{G}' = (\boldsymbol{N}', \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P}')$ echivalentă cu $\boldsymbol{G}$ şi care nu conţine neterminale inaccesibile, unde $\boldsymbol{N}' = \{S, A, B\}$, $\boldsymbol{T}' = \{a, b\}$ iar şi mulţimea producţiilor $\boldsymbol{P}'$ este următoarea:

$$S \to AB \qquad (1)$$

$$A \to BA|a \qquad (2)$$

$$B \to b|AC \qquad (3)\ .$$

2. Gramatici regulate. Automate finite deterministe

2.1. Gramatici regulate

Definiția 2.1.1. O gramatică $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$ este **gramatică regulată** dacă orice producție a sa este fie de forma $A \to aB$, fie de forma $A \to a$ cu $A, B \in \boldsymbol{N}$ și $a \in \boldsymbol{T}$.

Exemplul 2.1.1. Fie gramatica regulată $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$, unde $\boldsymbol{N}$ = $\{S\}$, $\boldsymbol{T}$ = $\{a\}$, iar mulțimea producțiilor este $\boldsymbol{P}$ = { $S \to a|aS$}. Se observă foarte ușor că limbajul generat de gramatica $\boldsymbol{G}$ este $\boldsymbol{L(G)}$ = $\{a^n \ |n \geq 1\}$.

Exemplul 2.1.2. Fie gramatica regulată $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$, unde $\boldsymbol{N}$ = $\{\mathrm{S}, \mathrm{A}\}$, $\boldsymbol{T} = \{a, b, c\}$, iar mulțimea producțiilor $\boldsymbol{P}$ este următoarea:

$S \to aS$ (1)

$S \to bA$ (2)

$A \to cA$ (3)

$A \to c$ (4)

Cuvântul $w_1 = abc \in \boldsymbol{L}(\boldsymbol{G})$ deoarece:

$$\boldsymbol{S} \underset{(1)}{\overset{*}{\Rightarrow}} a\boldsymbol{S} \underset{(2)}{\overset{*}{\Rightarrow}} ab\boldsymbol{A} \underset{(4)}{\overset{*}{\Rightarrow}} abc = w_1 \in \boldsymbol{L}(\boldsymbol{G}).$$

Cuvântul $w_2 = aaabcccc \in \boldsymbol{L}(\boldsymbol{G})$ deoarece:

$$\boldsymbol{S} \underset{(1)}{\overset{*}{\Rightarrow}} a\boldsymbol{S} \underset{(1)}{\overset{*}{\Rightarrow}} aa\boldsymbol{S} \underset{(1)}{\overset{*}{\Rightarrow}} aaa\boldsymbol{S} \underset{(2)}{\overset{*}{\Rightarrow}} aaab\boldsymbol{A} \underset{(3)}{\overset{*}{\Rightarrow}} aaabc\boldsymbol{A} \underset{(3)}{\overset{*}{\Rightarrow}} aaabcc\boldsymbol{A} \underset{(3)}{\overset{*}{\Rightarrow}} aaabccc\boldsymbol{A} \underset{(4)}{\overset{*}{\Rightarrow}}$$

$$\underset{(4)}{\overset{*}{\Rightarrow}} aaabcccc = w_2 \in \boldsymbol{L}(\boldsymbol{G}).$$

Deoarece după aplicarea producției (2) se poate aplica producția (3) cel puţin o dată, iar producţia (4) doar o singură dată, rezultă că limbajul generat de gramatica $\boldsymbol{G}$ este:

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

Exemplul 2.1.3. Fie gramatica regulată $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$, unde $\boldsymbol{N}$ = $\{\mathrm{S}, \mathrm{A}\}$, $\boldsymbol{T} = \{a, b, c\}$, iar mulțimea producțiilor $\boldsymbol{P}$ este următoarea:

$S \to aA$ (1)

$A \to aA|aB$ (2.1| 2.2)

$B \to bC$ (3)

$C \to cB|c$ (4.1| 4.2)

Se poate afirma că limbajul generat de gramatica $\boldsymbol{G}$ conține cuvântul w = *aabc,* într-adevăr are loc următoarea derivare:

$$\boldsymbol{S} \underset{(1)}{\overset{*}{\Rightarrow}} a\boldsymbol{A} \underset{(2.2)}{\overset{*}{\Rightarrow}} aa\boldsymbol{B} \underset{(3)}{\overset{*}{\Rightarrow}} aab\boldsymbol{C} \underset{(4.2)}{\overset{*}{\Rightarrow}} aabc = w \epsilon \boldsymbol{L}(\boldsymbol{G}).$$

Deoarece producția (2.1) introduce cel puțin două simboluri *a*, iar producția (2.2) urmată de producţia (3), la rândul ei urmată de (4.1) de oricâte ori și la final urmată de (4.2), ce introduc perechea *bc* de oricâte ori, rezultă că limbajul generat de gramatica $\boldsymbol{G}$ este:

$$\boldsymbol{L}(\boldsymbol{G}) = \{a^n(bc)^m |\, n \geq 2, m \geq 1\}.$$

2.2. Automate finite deterministe

Definiția 2.2.1. Se numește **automat finit determinist** un cvintuplu $\boldsymbol{A}$ = ($\boldsymbol{\Sigma}$, $\boldsymbol{Q}$, $\boldsymbol{\delta}$, $\boldsymbol{q_0}$, $\boldsymbol{F}$), unde:

1. $\boldsymbol{\Sigma}$ se numește **alfabetul de intrare**;
2. $\boldsymbol{Q}$ se numește **mulțimea stărilor** și este o mulțime finită nevidă;
3. $\delta : \boldsymbol{Q} \times \boldsymbol{\Sigma} \to \boldsymbol{Q}$ se numește **funcția de tranziție**;
4. $\boldsymbol{q_0} \in \boldsymbol{Q}$ reprezintă **starea inițială**;
5. $\boldsymbol{F} \subseteq \boldsymbol{Q}$ se numește **mulțimea stărilor finale**.

Observația 2.2.1. Funcția de tranziție $\boldsymbol{\delta}$ se poate prelungi pe mulțimea $\boldsymbol{Q} \times \boldsymbol{\Sigma}^*$ obţinându-se funcția $\overline{\boldsymbol{\delta}} : \boldsymbol{Q} \times \boldsymbol{\Sigma}^* \to \boldsymbol{Q}$, care poate fi aplicată și unui cuvânt, nu numai unui simbol, definită astfel:

$$\begin{cases} \overline{\boldsymbol{\delta}}(q, \varepsilon) = q, \forall q \in \boldsymbol{Q} \\ \overline{\boldsymbol{\delta}}(q, wa) = \overline{\boldsymbol{\delta}}(\overline{\boldsymbol{\delta}}(q, w), a), \forall q \in \boldsymbol{Q}, w \in \boldsymbol{\Sigma}^*, a \in \boldsymbol{\Sigma} \end{cases}.$$

Observația 2.2.2. Pentru a se simplifica scrierea, în continuare se va nota $\overline{\boldsymbol{\delta}}$ tot prin $\boldsymbol{\delta}$.

Observația 2.2.3. (*S. Marcus*) Se poate verifica că, pentru *x*, *y* $\in \boldsymbol{\Sigma}^*$ și oricare $q \in \boldsymbol{Q}$ avem:

$$\boldsymbol{\delta}(q, xy) = \boldsymbol{\delta}(\boldsymbol{\delta}(q, x), y),$$

și în particular, pentru *x* = *a* $\in \boldsymbol{\Sigma}$ avem:

$$\boldsymbol{\delta}(q, ay) = \boldsymbol{\delta}(\boldsymbol{\delta}(q, a), y). \qquad (**)$$

Definiția 2.2.2. Cuvântul $w \in \boldsymbol{\Sigma}^*$este **admis/ acceptat de automatul *A*** dacă

$$\boldsymbol{\delta}(q_0, w) \in \boldsymbol{F}.$$

Mulțimea tuturor cuvintelor acceptate de automatul $\boldsymbol{A}$ se notează prin $\mathcal{T}(A)$ și se numește **limbajul acceptat de automatul *A***, adică:

$$\mathcal{T}(A) = \{w \in \boldsymbol{\Sigma}^* \mid \boldsymbol{\delta}(q_0, w) \in \boldsymbol{F}\}.$$

La un automat ne interesează limbajul **acceptat** de acesta, în contrast,

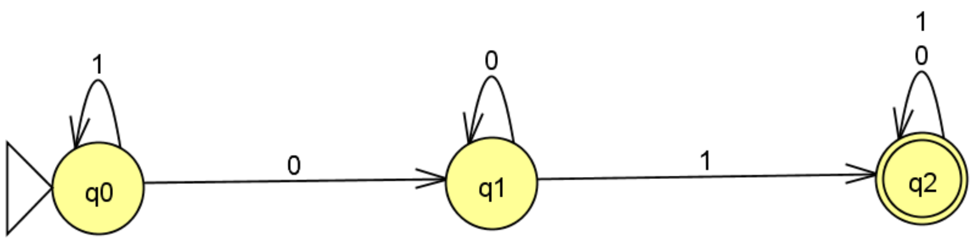
la o gramatică ne interesa limbajul **generat** de ea!

Un automat finit determinist poate fi reprezentat prin două metode:

1. **analitic**, prin precizarea mulțimilor **Σ** și ***Q***, funcția de tranziție **δ** fiind dată cu ajutorul unui tabel, format din linii și coloane, în care capetele liniilor conțin stările automatului, capetele de coloane conțin simbolurile din alfabetul de intrare, astfel încât starea inițială este marcată cu → (săgeată), iar stările finale sunt marcate cu *.

2. **grafic**, folosind un graf orientat în care fiecare nod reprezintă o stare a automatului, un arc între două noduri ce reprezintă stările $\boldsymbol{s_i}$ și $\boldsymbol{s_j}$ este etichetat cu simbolul *a* dacă are loc tranziția $\boldsymbol{\delta}(\boldsymbol{s_i}, a) = \boldsymbol{s_j}$. Starea inițială este marcată cu o săgeată, iar o stare finală este reprezentată de un nod dublu încercuit.

Exemplul 2.2.1. Graful finit orientat de mai jos este reprezentarea grafică a automatului finit determinist $\boldsymbol{A} = (\Sigma,\ Q,\ \delta,\ \boldsymbol{q_0},\ F)$ care acceptă toate șirurile binare care conțin subșirul 01.



Se pot verifica următoarele afirmații: w_1= 110110 ∈ $\mathcal{T}(A)$, iar w_2 = 1100 ∉ $\mathcal{T}(A)$.

Mai întâi, se construiește reprezentarea analitică a automatului ***A***, astfel:

- **Σ** = $\{0,1\}$;
- ***Q*** = $\{q_0, q_1, q_2\}$;
- ***F*** = $\{q_2\}$;
- funcția de tranziție **δ** este definită în următorul tabel:

Lect. univ. dr. TATIANA-CORINA DOSESCU
Facultatea de Informatică
Universitatea Titu Maiorescu

	δ	0	1
→	$\boldsymbol{q_0}$	q_1	q_0
	$\boldsymbol{q_1}$	q_1	q_2
*	$\boldsymbol{q_2}$	q_2	q_2

Folosind relația (**) de mai sus se verifică relația w_1 = 110110 ∈ $\mathcal{T}(A)$:

$$\boldsymbol{\delta}(q_0, 110110) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 1), 10110) = \boldsymbol{\delta}(q_0, 10110) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 1), 0110) =$$

$$= \boldsymbol{\delta}(q_0, 0110) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 0), 110) = \boldsymbol{\delta}(q_1, 110) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_1, 1), 10) = \boldsymbol{\delta}(q_2, 10) =$$

$$= \boldsymbol{\delta}(\boldsymbol{\delta}(q_1, 1), 0) = \boldsymbol{\delta}(q_2, 0) = q_2 \in \boldsymbol{F}.$$

Deci w_1 = 110110 ∈ $\mathcal{T}(A)$.

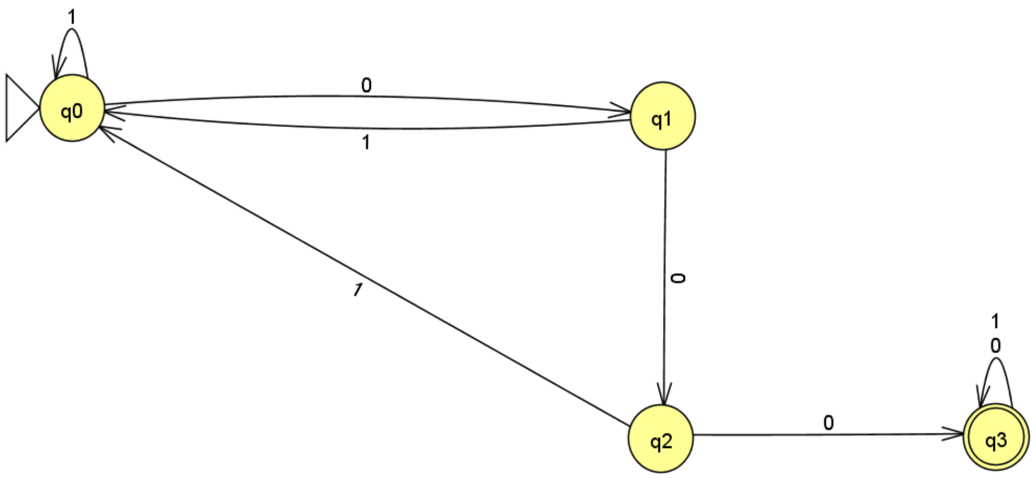
Cuvântul w_2 nu este acceptat de automatul ***A***. Într-adevăr, are loc:

$$\boldsymbol{\delta}(q_0, 1100) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 1), 100) = \boldsymbol{\delta}(q_0, 100) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 1), 00) =$$

$$= \boldsymbol{\delta}(q_0, 00) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 0), 0) = \boldsymbol{\delta}(q_1, 0) = q_1 \notin \boldsymbol{F}.$$

Deci $w_2 \notin \mathcal{T}(A)$, adică w_2 nu este acceptat de automatul ***A***.

Exemplul 2.2.2. Graful finit orientat de mai jos este reprezentarea grafică a automatul finit determinist $A = (\Sigma, Q, \delta, \boldsymbol{q_0}, F)$ care acceptă toate șirurile binare care conțin subșirul 000.



Se pot verifica următoarele afirmații: w_1 = 11000 ∈ $\mathcal{T}(A)$, iar w_2 = 1001001 ∉ $\mathcal{T}(A)$.

Mai întâi, se construiește reprezentarea analitică a automatului ***A***, astfel:

- **Σ** = $\{0, 1\}$;
- ***Q*** = $\{q_0, q_1, q_2, q_3\}$;
- ***F*** = $\{q_3\}$;
- funcția de tranziție **δ** este definită în următorul tabel:

Lect. univ. dr. TATIANA-CORINA DOSESCU
Facultatea de Informatică
Universitatea Titu Maiorescu

	δ	0	1
→	$\boldsymbol{q_0}$	q_1	q_0
	$\boldsymbol{q_1}$	q_2	q_0
	$\boldsymbol{q_2}$	q_3	q_0
*	$\boldsymbol{q_3}$	q_3	q_3

Folosind relația (**) de mai sus se verifică relația w_1 = 11000 ∈ $\mathcal{T}(\boldsymbol{A})$:

$$\boldsymbol{\delta}(q_0, 11000) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 1), 1000) = \boldsymbol{\delta}(q_0, 1000) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 1), 000) =$$

$$= \boldsymbol{\delta}(q_0, 000) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 0), 00) = \boldsymbol{\delta}(q_1, 00) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_1, 0), 0) = \boldsymbol{\delta}(q_2, 0) = q_3 \in \boldsymbol{F}.$$

Deci w_1 = 11000 ∈ $\mathcal{T}(\boldsymbol{A})$.

Cuvântul w_2 nu este acceptat de automatul ***A***. Într-adevăr, are loc:

$$\boldsymbol{\delta}(q_0, 1001001) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 1), 001001) = \boldsymbol{\delta}(q_0, 001001) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 0), 01001) =$$

$$= \boldsymbol{\delta}(q_1, 01001) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_1, 0), 1001) = \boldsymbol{\delta}(q_2, 1001) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_1, 1), 001) =$$

$$= \boldsymbol{\delta}(q_0, 001) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_0, 0), 01) = \boldsymbol{\delta}(q_1, 01) = \boldsymbol{\delta}(\boldsymbol{\delta}(q_1, 0), 1) = \boldsymbol{\delta}(q_2, 1) = q_0 \notin \boldsymbol{F}.$$

Deci $w_2 \notin \mathcal{T}(\boldsymbol{A})$, adică w_2 nu este acceptat de automatul ***A***.

Lect. univ. dr. TATIANA-CORINA DOSESCU
Facultatea de Informatică
Universitatea Titu Maiorescu