

~ Seminar 4 ~

➤ Lema de pompare pentru limbaje regulate (REG) [continuare]

• Exemple:

$L_4 = \{x \cdot x^R \mid x \in \{a, b\}^*\} \notin REG$ (discutat alegerea cuvântului α)

$L_5 = \{x \cdot x \mid x \in \{a, b\}^*\} \notin REG$ (discutat alegerea cuvântului α)

Obs: (**R = reversed**) Cuvântul w^R este oglindirea cuvântului w , de exemplu $(abc)^R = cba$.

➔ Pentru L_4 :

-- Cuvântul $\alpha = a^p a^p = a^{2p} = (aa)^p$ NU este o alegere bună (pentru acest α putem desena un AFD având un circuit de 2 de a).

- Dacă $v = a^{2k} \Rightarrow |\beta| = |u \cdot v^i \cdot w| = |u \cdot v \cdot w| + |v|^{(i-1)} = |a^{2p+(2k)*(i-1)}| = 2 * (p + k * (i - 1))$ este **par** (adică $\beta \in L_4$), $\forall i \geq 0$ (NU avem contradicție cu condiția 3 din lema).
- Dacă $v = a^{2k+1} \Rightarrow |\beta| = |a^{2p+(2k+1)*(i-1)}| = 2 * (p + k * (i - 1)) + (i - 1)$ este **impar**, adică $\beta \notin L_4$ (contradicție cu condiția 3 din lema) $\Leftrightarrow i$ este par (de exemplu alegem $i = 0$).

Concluzie: Dacă există descompuneri ale lui α (forme ale lui v) pentru care NU putem obține contradicție, înseamnă că demonstrația nu este corectă și trebuie ales un alt α .

-- Cuvântul $\alpha = a^p b b a^p$ este o alegere bună (pentru acest α nu putem desena un AFD).

- Dacă $v = a^k$ alegem $i = 2 \Rightarrow \beta = a^{p+k} b b a^p \notin L_4$ pentru că $1 \leq k \leq p$ (din primele două condiții din lema).

Concluzie: Avem un singur caz de descompunere a lui α (o singură formă a lui v), am obținut contradicție, deci demonstrația este corectă.

➔ Pentru L_5 :

-- Cuvântul $\alpha = a^p b a^p b$ este o alegere bună (pentru acest α nu putem desena un AFD).

Avem un singur caz de descompunere a lui α .

- Dacă $v = a^k$ alegem $i = 2 \Rightarrow \beta = a^{p+k} b a^p b \notin L_5$ pentru că $1 \leq k \leq p$ (din primele două condiții din lema). Deci avem contradicție și demonstrația este corectă.

-- Cuvântul $\alpha = b a^p b a^p$ este tot o alegere bună (pentru acest α nu putem desena un AFD), dar avem mai multe cazuri de descompunere a lui α .

- Dacă $v = b a^k$ (cu $0 \leq k \leq p - 1$) alegem $i = 0 \Rightarrow \beta = a^{p-k} b a^p \notin L_5$.
- Dacă $v = a^k$ (cu $1 \leq k \leq p$) alegem $i = 2 \Rightarrow \beta = b a^{p+k} b a^p \notin L_5$.

Concluzie: Avem contradicție pe fiecare caz posibil de descompunere, deci demonstrația este corectă.

• **Exemplu:**

$$L_6 = \{a^{n^2} | n \geq 1\} = \{a^1, a^4, a^9, a^{16}, a^{25}, a^{36}, \dots\} \notin REG$$

Obs: Proprietatea cuvintelor din limbajul L_6 este aceea că sunt formate doar din litere de "a" și lungimea lor este un număr pătrat perfect. Deci pentru a obține contradicția ($\beta \notin L_6$) trebuie să arătăm că lungimea lui β **nu** poate fi un pătrat perfect.

Demonstrație: Presupunem prin reducere la absurd că L_6 este limbaj regulat. Atunci $\exists p \in \mathbb{N}$ și putem aplica lema de pompare. (*În continuare negăm afirmația lemei.*)

Alegem cuvântul $\alpha = a^{p^2} \in L_6$, cu $|\alpha| = p^2 \geq p, \forall p \in \mathbb{N}$ (deci lungimea cuvântului respectă ipoteza lemei). Conform lemei, cuvântul poate fi scris sub forma $\alpha = u \cdot v \cdot w$.

Observăm că toate cuvintele din L_6 sunt formate doar din litere de "a". Atunci notăm $v = a^k$. Din condițiile (1) și (2) ale lemei avem $1 \leq |v| \leq p$. Deci $1 \leq |a^k| \leq p$, adică $1 \leq k \leq p$ (*).

De asemenea, condiția (3) din lema spune că $u \cdot v^i \cdot w \in L_6, \forall i \geq 0$.

Alegem $i = 2$ și avem cuvântul $\beta = u \cdot v^2 \cdot w$ de lungime

$$|\beta| = |u \cdot v^2 \cdot w| = |u \cdot v \cdot w| + |v| = |\alpha| + |v| = p^2 + |v| = p^2 + k.$$

Conform (*), avem $1 \leq k \leq p$ (adunăm peste tot p^2) $\Leftrightarrow p^2 + 1 \leq p^2 + k \leq p^2 + p$.

Dar $p^2 < p^2 + 1$ și $p^2 + p < (p+1)^2$. Rezultă că $p^2 < p^2 + k < (p+1)^2$, adică $p^2 < |\beta| < (p+1)^2$. Deci $|\beta|$ nu poate fi pătrat perfect (pentru că este inclus strict între două pătrate perfecte consecutive) $\Rightarrow \beta \notin L_6$, contradicție cu condiția (3) din lema, deci presupunerea făcută este falsă și L_6 nu este limbaj regulat. ■

➤ **Expresii regulate (Regex)**

Definiție: Se numește familia expresiilor regulate peste Σ și se notează $RegEx(\Sigma)$ mulțimea de cuvinte peste alfabetul $\Sigma \cup \{ (,), +, \cdot, *, \emptyset, \lambda \}$ definită recursiv astfel:

- i) $\emptyset, \lambda \in RegEx$ și $a \in RegEx, \forall a \in \Sigma$.
- ii) Dacă $e_1, e_2 \in RegEx$, atunci $(e_1 + e_2) \in RegEx$. (reuniune)
- iii) Dacă $e_1, e_2 \in RegEx$, atunci $(e_1 \cdot e_2) \in RegEx$. (concatenare)
- iv) Dacă $e \in RegEx$, atunci $(e^*) \in RegEx$. (stelare)

• **Precedența operațiilor:** $() > * > \cdot > +$ (paranteze > stelare > concatenare > reuniune)

Obs: În evaluarea unei expresii regulate se ține cont în primul rând de paranteze, iar apoi ordinea în care se evaluează operațiile este: stelare, apoi concatenare, apoi reuniune.

(Dacă vrei să fii siguri că nu le încurcați, puteți să faceți o analogie cu operațiile aritmetice, unde se evaluează întâi ridicarea la putere, apoi înmulțirea și apoi adunarea.)

• Reamintim din seminarul 1:

$$L = L_1 \cup L_2 = \{w \mid w \in L_1 \text{ sau } w \in L_2\}$$



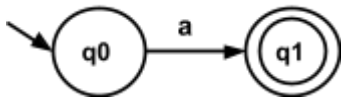
$$L = L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \text{ și } w_2 \in L_2\}$$

$$L = (L_1)^* = \{\lambda\} \cup \bigcup_{n \geq 1} \{w_1 w_2 \dots w_n \mid w_i \in L_1, \forall 1 \leq i \leq n\}$$

➤ *Algoritm: Transformarea RegEx → AFN-λ*

Pentru fiecare caz din definiția RegEx vom construi câte un automat finit echivalent.

Caz i)

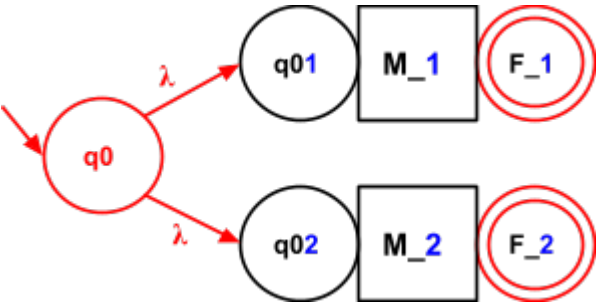
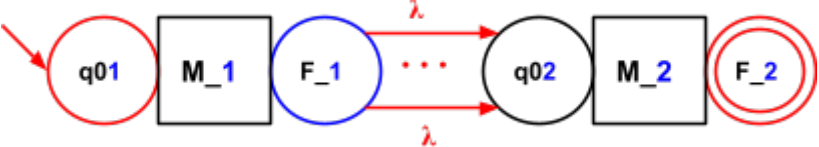
RegEx	$e = \emptyset$	$e = \lambda$	$e = a$, unde $a \in \Sigma$
Limбай	$L = \emptyset$	$L = \{\lambda\}$	$L = \{a\}$
Automat Finit			

În **cazurile ii), iii) și iv)** presupunem că pentru expresia regulată e_k și limbajul $L(e_k)$, $k \in \{1, 2\}$ avem deja automate finite $AF(L(e_k)) = (Q_k, \Sigma_k, q_{0k}, F_k, \delta_k)$, cu $Q_1 \cap Q_2 = \emptyset$ (stări disjuncte).

Desenăm schema unui automat punând în evidență starea inițială q_{0k} și mulțimea stărilor finale F_k . Dreptunghiul M_k include toate celelalte stări și tranzițiile automatului.

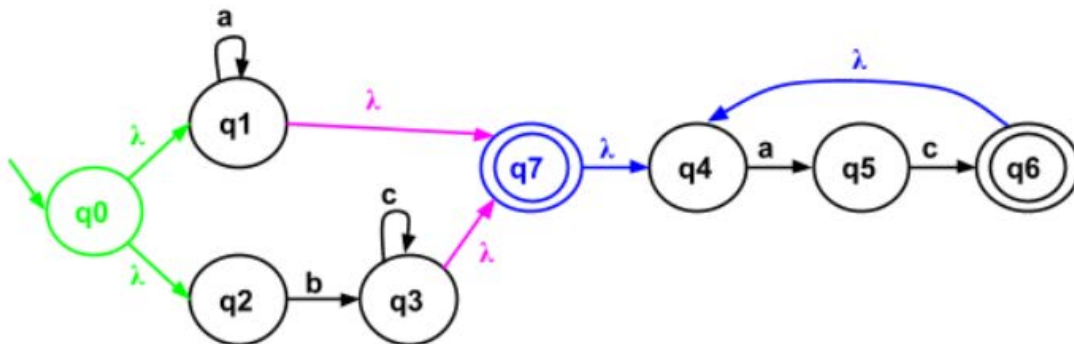


Vom construi automatele pentru operațiile de reuniune, concatenare și stelare.

<p>Caz ii) <i>(reuniune)</i></p> <p>RegEx $e = e_1 + e_2$</p> <p>Limбай $L(e) = L(e_1) \cup L(e_2)$</p>	<p>Automat Finit $AF(L(e_1 + e_2)) = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma_1 \cup \Sigma_2, q_0, F_1 \cup F_2, \delta_1 \cup \delta_2 \cup \{\delta(q_0, \lambda) = \{q_{01}, q_{02}\}\})$</p> 
<p>Caz iii) <i>(concatenare)</i></p> <p>RegEx $e = e_1 \cdot e_2$</p> <p>Limбай $L(e) = L(e_1) \cdot L(e_2)$</p>	<p>Automat Finit $AF(L(e_1 \cdot e_2)) = (Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, q_{01}, F_2, \delta_1 \cup \delta_2 \cup \{\delta(q_{f1}, \lambda) \ni q_{02} \mid \forall f_1 \in F_1\})$</p> 

<p>Caz iv) (stelare)</p> <p>RegEx $e = (e_1)^*$</p> <p>Limбай $L(e) = (L(e_1))^*$</p>	<p>Automat Finit $AF(L((e_1)^*)) = (Q_1 \cup \{q_0\}, \Sigma_1, q_0, F_1 \cup \{q_0\},$ $\delta_1 \cup \{\delta(q_0, \lambda) = q_{01}\} \cup \{\delta(q_{f_1}, \lambda) \ni q_{01} \mid \forall f_1 \in F_1\})$</p>
--	---

- Exemplu:** Desenați 3 automate finite pentru $L_1 = a^*$, $L_2 = bc^*$, $L_3 = ac$, apoi folosind algoritmi pentru reuniune, concatenare, stelare și ținând cont de paranteze și de ordinea operațiilor, desenați automatul pentru $L_4 = (a^* + bc^*) \cdot (ac)^* = (L_1 + L_2) \cdot (L_3)^*$.



➤ **Algoritm:** Transformarea AFN- $\lambda \rightarrow$ RegEx

Definiție: Se numește **AFE (automat finit extins)**, $M = (Q, \Sigma, et, q_0, F)$, unde, la fel ca la celelalte automate finite, Q este mulțimea stărilor, Σ este alfabetul, q_0 este starea inițială, F este mulțimea stărilor finale. Aici (în locul funcției de tranziție) avem **funcția de etichetare** $et: Q \times Q \rightarrow \mathbf{RegEx}(\Sigma)$.

Notăm $et(p, q)$ prin e_{pq} (expresia regulată asociată săgeții de la starea p la starea q)

Ideea algoritmului este de a transforma automatul finit într-un automat finit extins și apoi a elimina una câte una stările până ajungem la o expresie regulată echivalentă cu automatul inițial.

- Algoritm:**

Pas 1: Transformăm automatul finit dat într-un AFE astfel: dacă de la starea q_x către starea q_y există mai multe tranziții, atunci le înlocuim cu expresia regulată obținută prin reunirea (operatorul “+”) simbolurilor de pe acele tranziții.

$$et(q_x, q_y) = \{w \in \mathbf{RegEx}(\Sigma) \mid w = a_1 + a_2 + \dots + a_n ;$$

$$q_y \in \delta(q_x, a_i), a_i \in (\Sigma \cup \{\lambda\}), \forall i \in \{1, \dots, n\}\}$$

Pas 2: Dacă starea inițială este și finală sau dacă există săgeți care vin către starea inițială, atunci se adaugă la automat o nouă stare care va fi inițială și va avea o săgeată cu expresia λ către fosta stare inițială.

Pas 3: Dacă există mai multe stări finale sau dacă există săgeți care pleacă din vreo stare finală, atunci se adaugă la automat o nouă stare care va fi unica finală și va avea săgeți cu expresia λ din toate fostele stări finale către ea.

Pas 4: În orice ordine, se elimină pe rând, una câte una, toate stările în afară de cea inițială și cea finală, astfel:

→ Presupunem că vrem să eliminăm starea q și că există săgeți cu etichetele (expresiile regulate) $et(p, q)$, $et(q, s)$ și eventual bucla cu $et(q, q)$.

→ Atunci obținem noua etichetă (expresie regulată) de pe săgeata de la starea p la starea s :

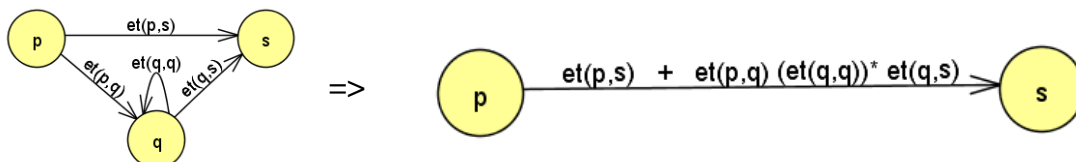
- [(fosta etichetă directă de la p la s) sau (\emptyset dacă nu există săgeată directă)]
reunită cu
- [(eticheta de la p la q) concatenată cu
(stelarea etichetei buclei de la q la q , sau λ dacă bucla nu există) concatenată cu
(eticheta de la q la s)]. (Vezi desenul de mai jos.)

Pas 5: Atunci când rămân doar două stări, expresia obținută între starea inițială și cea finală este răspunsul final (o expresie regulată echivalentă cu automatul finit dat).

• **Observații:**

(1) La pas 4, pentru starea q pe care dorim să o eliminăm (împreună cu toate săgețile lipite de ea), trebuie să găsim orice “predecesor” $p \neq q$ (adică există o săgeată de la p la q) și orice “succesor” $s \neq q$ (adică există săgeată de la q la s). Deci făcând abstracție de eventuala buclă a lui q , căutăm și *grupăm orice săgeată care intră spre q cu orice săgeată care iese din q* și astfel obținem expresia regulată de pe săgeata de la p la s cu formula explicată mai sus. Atenție, dacă $p = s$, înseamnă că vom obține o buclă.

→ Vrem să eliminăm q și avem un p (“predecesor”) și un s („succesor”).



(2) Dacă una din expresii conține reuniune (“+”), atunci o *incluăm între paranteze*, pentru a se executa întâi acea reuniune și abia apoi concatenarea cu expresiile de pe alte săgeți. Fiecare expresie obținută între p și s încercăm să o *simplificăm* cât mai mult folosind formulele de mai jos.

(3) În funcție de *ordinea* în care alegem să eliminăm stările la pasul 4, vom obține o anumită expresie, dar toate sunt echivalente între ele. **Sfat:** În general, eliminăm starea care are momentan cele mai puține săgeți pentru a calcula cât mai puține drumuri.

Atenție să nu confundați semnul “+” dintre expresii (folosit pentru *reuniunea* lor) cu semnul “+” pus la putere (folosit pentru *concatenare repetată*, cel puțin puterea 1).

Obs: Algoritmul de mai sus descoperă și *reunește expresiile regulate corespunzătoare tuturor drumurilor de la starea inițială la o stare finală*. Puteți verifica asta pe exemplele următoare, comparând automatul finit dat cu expresia regulată obținută la finalul algoritmului.

• **Câteva formule utile**

- (A) $e \cdot \emptyset = \emptyset$ și $\emptyset \cdot e = \emptyset$ (\emptyset este pentru concatenare cum este 0 pentru înmulțire)
 (B) $e \cdot \lambda = e$ și $\lambda \cdot e = e$ (λ este pentru concatenare cum este 1 pentru înmulțire)
 (C) $e^* \cdot e = e^+$ și $e \cdot e^* = e^+$ (Dar e^+ nu va fi folosită în RegEx pt că nu respectă definiția lor.)
 (D) $\{e_1, e_2\}^* = (e_1 + e_2)^* = (e_1^* \cdot e_2^*)^*$ (Formulă valabilă pentru oricâte expresii, nu doar 2.)
 (E) $e_1 \cdot (e_2 + e_3) = (e_1 \cdot e_2) + (e_1 \cdot e_3)$ și $(e_1 + e_2) \cdot e_3 = (e_1 \cdot e_3) + (e_2 \cdot e_3)$
 (F) $e + \emptyset = \emptyset + e = e$ (\emptyset este pentru reuniune cum este 0 pentru adunare)
 (G) $\emptyset^* = \{\lambda\}$ (conform definiției stelării) și $\lambda^* = \lambda$ (conform formulei B de mai sus)
 (H) Dacă $e_1 \supseteq e_2$, atunci $e_1 + e_2 = e_2 + e_1 = e_1$. (De exemplu: $a + ab^* = ab^*$)
 (I) În loc de $\lambda + (e)^+ = \lambda + e \cdot e^*$ scriem e^* .

• **Exemplu rezolvat:**

Să se transforme următorul automat finit într-o expresie regulată echivalentă.

<p>Pas 1 (AF \rightarrow AFE): \Rightarrow Reunim tranzițiile aflate pe aceeași săgeată.</p>	
<p>Pas 2 (Verificăm <i>starea inițială</i>: - să nu fie stare finală și - să nu vină săgeți către ea) \Rightarrow adăugăm o nouă stare inițială cu λ către fosta stare inițială.</p>	
<p>Pas 3 (Verificăm <i>starea finală</i>: - să fie unica finală și - să nu plece săgeți din ea) \Rightarrow adăugăm o nouă unică stare finală spre care vin λ din fostele stări finale.</p>	
<p>Pas 4 (eliminăm q_2): $\Rightarrow et(q_1, q_3) = et(q_1, q_3) + et(q_1, q_2) \cdot (et(q_2, q_2))^* \cdot et(q_2, q_3)$</p>	
<p>Pas 4 (eliminăm q_1): $\Rightarrow et(q_0, q_3) = et(q_0, q_3) + et(q_0, q_1) \cdot (et(q_1, q_1))^* \cdot et(q_1, q_3)$</p>	

• **Exemplu:** [discutat la seminar]

Să se transforme următorul automat finit într-o expresie regulată echivalentă.

<p>Pas 1 (AF \rightarrow AFE):</p> <p>\Rightarrow Reunim tranzițiile aflate pe aceeași săgeată.</p>	
<p>Pas 2 (Verificăm <i>starea inițială</i>):</p> <ul style="list-style-type: none"> - să nu fie stare finală și - să nu vină săgeți către ea) <p>\Rightarrow adăugăm o nouă stare inițială cu λ către fosta stare inițială.</p>	
<p>Pas 3 (Verificăm <i>starea finală</i>):</p> <ul style="list-style-type: none"> - să fie unica finală și - să nu plece săgeți din ea) <p>\Rightarrow adăugăm o nouă unică stare finală spre care vin λ din fostele stări finale.</p>	
<p>Pas 4 (eliminăm <i>q3</i>):</p> <p>$\Rightarrow et(q_2, q_4) = et(q_2, q_4) + et(q_2, q_3) \cdot (et(q_3, q_3))^* \cdot et(q_3, q_4)$</p> <p>$\Rightarrow et(q_1, q_4) = et(q_1, q_4) + et(q_1, q_3) \cdot (et(q_3, q_3))^* \cdot et(q_3, q_4)$</p>	
<p>Pas 4 (eliminăm <i>q2</i>):</p> <p>$\Rightarrow et(q_1, q_4) = et(q_1, q_4) + et(q_1, q_2) \cdot (et(q_2, q_2))^* \cdot et(q_2, q_4)$</p>	
<p>Pas 4 (eliminăm <i>q1</i>):</p> <p>$\Rightarrow et(q_0, q_4) = et(q_0, q_4) + et(q_0, q_1) \cdot (et(q_1, q_1))^* \cdot et(q_1, q_4)$</p>	

~ Temă ~

EX_1: Demonstrați că următoarele limbaje nu sunt regulate, folosind lema de pompare.

$$L7 = \{a^{2^n} \mid n \geq 0\} = \{a^1, a^2, a^4, a^8, a^{16}, a^{32}, a^{64}, \dots\} \notin REG$$

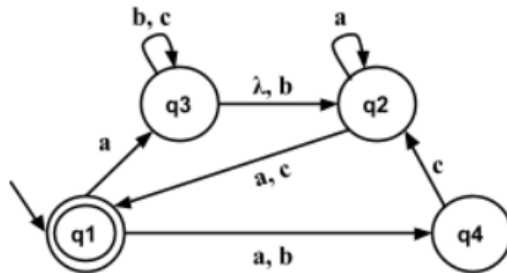
$$L8 = \{a^n \mid n \text{ număr prim}\} = \{a^2, a^3, a^5, a^7, a^{11}, a^{13}, a^{17}, a^{19}, a^{23}, \dots\} \notin REG$$

(Încercați să redactați demonstrațiile fără să aveți sub ochi alte exemple, să verificați dacă ați înțeles și reținut structura acestor demonstrații cu lema de pompare.)

EX_2: Desenați 5 automate finite (cu stări disjuncte) pentru $L_1 = b^* \cdot a \cdot (bc)^*$, $L_2 = \lambda + c$, $L_3 = (a + c)^*$, $L_4 = b^* \cdot c \cdot (a + b)$, $L_5 = (c + a) \cdot (c + b) \cdot a^*$, apoi folosind algoritmi pentru reuniune, concatenare, stelare și ținând cont de paranteze și de ordinea operațiilor, desenați automatul pentru $L_6 = ((L_1 + L_2)^* + (L_3 \cdot L_4)^*) \cdot L_5$.

EX_3: Să se transforme următorul automat finit într-o expresie regulată echivalentă.

(Pentru pașii 1+2+3 puteți să desenați un singur graf, apoi la pasul 4 pentru eliminarea fiecărei stări desenați câte un graf separat.)



• (Exerciții recapitulative)

→ **Închiderea limbajelor regulate la operații** (complement; intersecție, diferență, reuniune)

Obs: Dacă limbajul regulat L este acceptat de un AFD complet definit (fără tranziții lipsă) $AFD(L) = (Q, \Sigma, \delta, q_0, F)$, atunci putem construi un AFD care să accepte **complementul lui L** ($\Sigma^* \setminus L$) prin **interschimbarea stărilor finale cu cele nefinale** $AFD(\bar{L}) = (Q, \Sigma, \delta, q_0, Q \setminus F)$.

Obs: Dacă limbajele regulate L_1 și L_2 sunt acceptate de 2 automate AFD complet definite $AFD(L_1) = (Q_1 = \{q_0, q_1, \dots\}, \Sigma, \delta_1, q_0, F_1)$ și $AFD(L_2) = (Q_2 = \{r_0, r_1, \dots\}, \Sigma, \delta_2, r_0, F_2)$, atunci putem construi un AFD cu stări obținute prin **produs cartezian între mulțimile de stări** ale celor 2 automate: $AFD(L) = (Q, \Sigma, \delta, (q_0, r_0), F)$ având

- stările $Q = Q_1 \times Q_2 = \{(q_i, r_j) \mid q_i \in Q_1 \text{ și } r_j \in Q_2\}$,
- tranzițiile $\delta((q_i, r_j), x) = (\delta_1(q_i, x), \delta_2(r_j, x))$, $\forall (q_i, r_j) \in Q, \forall x \in \Sigma$,
- starea inițială (q_0, r_0) (perechea formată din cele două stări inițiale),
- stările finale F depind dacă automatul acceptă limbajul:

- ✓ (intersecție) $L = L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ și } w \in L_2\} \Rightarrow F = F_1 \times F_2$
- ✓ (diferență) $L = L_1 \setminus L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ și } w \notin L_2\} \Rightarrow F = F_1 \times (Q_2 \setminus F_2)$
sau $L = L_2 \setminus L_1 = \{w \in \Sigma^* \mid w \notin L_1 \text{ și } w \in L_2\} \Rightarrow F = (Q_1 \setminus F_1) \times F_2$
- ✓ (reuniune) $L = L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ sau } w \in L_2\} \Rightarrow F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

Obs: Toate limbajele de la EX_4 și EX_5 sunt definite peste alfabetul $\Sigma = \{a, b\}$.

EX_4: Pentru fiecare limbaj L dat, desenați un AFD complet definit care să accepte L (scrieți alături care ar fi stările finale F), dar pe graf setați stările finale F' astfel încât să accepte \bar{L} (complementul limbajului L dat).

(a) $L = \{w \mid w \in a^*b^*\}$

(b) $L = \{w \mid w \in (ab^+)^*\}$

(c) $L = \{w \mid w \in a^* \cup b^*\}$

EX_5: Pentru $L1$ și $L2$ limbaje regulate date, desenați două AFD complet definite (peste alfabetul $\Sigma = \{a, b\}$) având stări disjuncte. Desenați AFD-ul cu stări obținute prin produs cartezian între stările automatelor pentru $L1$ și $L2$ (în această ordine), apoi scrieți alături care ar fi stările finale pentru a accepta limbajele:

$$L3 = L1 \cap L2, \quad L4 = L1 \setminus L2, \quad L5 = L2 \setminus L1, \quad L6 = L1 \cup L2.$$

(a) $L1 = \{w \mid |w|_a \text{ este par}\}$ și $L2 = \{w \mid w \text{ conține 1 sau 2 de } b\}$

(b) $L1 = \{w \mid w \text{ începe cu un } a\}$ și $L2 = \{w \mid w \text{ conține cel mult un } b\}$

(c) $L1 = \{w \mid |w|_a \text{ este impar}\}$ și $L2 = \{w \mid w \text{ se termină cu un } b\}$