~ Seminar 2 ~

➢ *Algoritm:* Transformarea AFN → AFD

Se dă un automat AFN. Se cere să se construiască un automat AFD echivalent (care să accepte același limbaj).

Idee: Atunci când în AFN dintr-o stare cu un anumit simbol avem ramificare către mai multe stări-destinație, în AFD vom grupa toate aceste stări-destinație într-o singură stare pentru a elimina ramificarea.

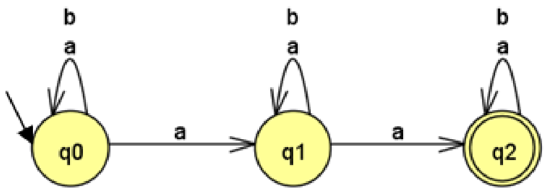
Algoritm: Pentru AFN-ul $(Q, \Sigma, q_0, F, \delta)$ construim AFD-ul $(Q', \Sigma, q_0, F', \delta')$ astfel:
- Stările AFD-ului vor reprezenta *submulțimi are mulțimii stărilor* AFN-ului ($Q' \subseteq \mathcal{P}(Q)$).
- Cele două automate vor avea același alfabet Σ și ***aceeași stare inițială*** q_0.
- Funcția de tranziție a AFD-ului este calculată astfel:

$$\delta'(R, x) = \bigcup_{q \in R} \delta(q, x), \qquad \forall R \in \mathcal{P}(Q), \forall x \in \Sigma$$

- Stările finale ale AFD-ului sunt mulțimile care ***conțin cel puțin o stare finală*** a AFN-ului $F' = \{R \mid R \in Q', R \cap F \neq \emptyset\}$.

Obs: La seminar vom folosi *metoda iterativă* de calcul pentru Q' (pornim din starea inițială și adăugăm stările AFD-ului pe măsură ce ajungem la ele calculând funcția de tranziție δ' pentru stările găsite anterior).

- ***Exemplu:*** Pentru AFN-ul următor construiți un AFD echivalent.

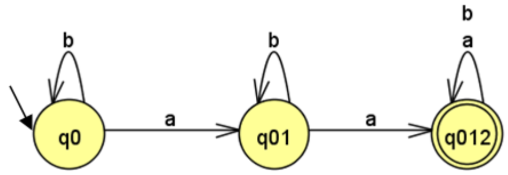


AFN

a) Completăm tabel_1 cu funcția de tranziție pentru AFN.
b) Completăm tabel_2 cu funcția de tranziție pentru AFD (pornim din starea inițială a AFN-ului și adăugăm pe rând stările obținute în interiorul tabel_2).
c) Desenăm graful pentru AFD conform tabel_2.

$\boldsymbol{\delta}$**_AFN**	**a**	**b**
$\boldsymbol{q_0}$ **init**	$\{q_0, q_1\} = q_{01}$	$\{q_0\}$
$\boldsymbol{q_1}$	$\{q_1, q_2\} = q_{12}$	$\{q_1\}$
$\boldsymbol{q_2 \in F}$	$\{q_2\}$	$\{q_2\}$

$\boldsymbol{\delta}$**_AFD**	**a**	**b**
$\boldsymbol{q_0}$ **init**	q_{01}	q_0
$\boldsymbol{q_{01}}$	q_{012}	q_{01}
$\boldsymbol{q_{012} \in F}$	q_{012}	q_{012}



AFD

➢ Verificare acceptare cuvânt de către AFD, AFN

Care este limbajul recunoscut de cele două automate din exemplul de mai sus?
Verificați (pentru AFD, apoi pentru AFN) dacă cuvintele **bba** și **babbaba** sunt acceptate sau respinse, folosind configurații.

→ AFD, cuv bba:
$(q_0, bba) \vdash^b (q_0, ba) \vdash^b (q_0, a) \vdash^a (q_{01}, \lambda)$, $q_{01} \notin F$ => bba respins

→ AFN, cuv bba:
$(q_0, bba) \vdash^b (q_0, ba) \vdash^b (q_0, a) \vdash^a \{(q_0, \lambda), (q_1, \lambda)\}$, $\{q_0, q_1\} \cap F = \emptyset$ => bba respins

→ AFD, cuv babbaba:
$(q_0, babbaba) \vdash^b (q_0, abbaba) \vdash^a (q_{01}, bbaba) \vdash^b (q_{01}, baba) \vdash^b (q_{01}, aba)$
$\vdash^a (q_{012}, ba) \vdash^b (q_{012}, a) \vdash^a (q_{012}, \lambda)$, $q_{012} \in F$ => babbaba acceptat

→ AFN, cuv babbaba:
$(q_0, babbaba) \vdash^b (q_0, abbaba) \vdash^a \{(q_0, bbaba), (q_1, bbaba)\}$
$\vdash^b \{(q_0, baba), (q_0, baba)\} \vdash^b \{(q_0, aba), (q_1, aba)\}$
$\vdash^a \{(q_0, ba), (q_1, ba), (q_2, ba)\} \vdash^b \{(q_0, a), (q_1, a), (q_2, a)\} \vdash^a \{(q_0, \lambda), (q_1, \lambda), (q_2, \lambda)\}$,
$\{q_0, q_1, q_2\} \cap F = \{q_2\} \neq \emptyset$ => babbaba acceptat

➢ *Automat Finit Nedeterminist cu λ–tranziţii*

AFN–λ = (Q, Σ, q_0, δ, F)
Q mulţimea de stări
Σ alfabetul de intrare
$q_0 \in$ Q starea iniţială
F $\subseteq$ Q mulţimea de stări finale
$\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^Q$ funcţia de tranziţie („delta”)

<u>Diferenţa faţă de AFN-ul simplu</u> este că suntem într-o stare şi putem citi fie un caracter din alfabetul Σ, fie cuvântul vid λ. Deci se poate întâmpla să ajungem într-o stare nouă fără să fi citit nicio literă nouă din cuvântul de intrare, ci doar aplicând λ–tranziţii.

➢ λ–închiderea unei stări

Mulţimea de stări în care se poate ajunge plecând din starea q şi aplicând zero sau mai multe λ–tranziţii se numeşte „**λ–închiderea stării q**” şi se notează cu **<q>**.

Obs: Orice stare face parte din propria λ–închidere (pentru că $\delta(q,\lambda^0)=q$; practic putem presupune că orice stare are o λ–tranziţie *implicită* către ea însăşi).

$$<q> = \bigcup_{k\geq 0}\{r \mid r\in\hat{\delta}(q,\lambda^k)\}$$

$$<q> = \{q\}\cup\{q_i \mid q_i\in\hat{\delta}(q,\lambda^1)\}\cup\{q_{ij}\mid q_{ij}\in\hat{\delta}(q,\lambda^2)\}\cup\ldots$$

Observăm că mulţimile se pot calcula inductiv după puterea lui λ :

$\{q_{ij}\mid q_{ij}\in\hat{\delta}(q,\lambda^2)\}=\{q_{ij}\mid q_i\in\hat{\delta}(q,\lambda^1), q_{ij}\in\delta(q_i,\lambda^1)\}$.

Sau în general $\{r\mid r\in\delta(q,\lambda^k)\}=\{r\mid s\in\hat{\delta}(q,\lambda^{k-1}), r\in\delta(s,\lambda^1)\}$.

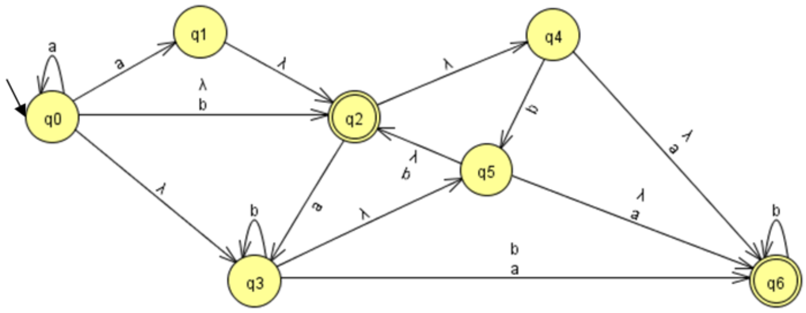
➢ Verificare acceptare cuvânt de către automat AFN–λ

<u>Pentru a verifica dacă un cuvânt este sau nu acceptat de un automat AFN–λ:</u>
Se procedează analog ca în cazul AFN-ului, doar că înainte de a căuta toate stările posibile de continuare cu tranziţii cu simbolul curent, trebuie să facem λ–închiderea mulţimii curente de stări. Iar după ce a fost citit tot cuvântul de intrare, trebuie să facem o ultimă λ–închidere a stărilor curente, pentru a obţine mulţimea finală de stări în care poate ajunge automatul pentru cuvântul dat.

Obs: λ–închiderea unei mulţimi de stări este egală cu reuniunea λ–închiderilor acelor stări.
$<\{q_{i1}, q_{i2}, \ldots, q_{in}\}> = <q_{i1}>\cup<q_{i2}>\cup\ldots\cup<q_{in}>$

Obs: La AFN–λ, $\lambda\in L \Leftrightarrow <q_0>\cap F\neq\emptyset$ (cuvântul vid este acceptat de automat dacă și numai dacă λ-închiderea stării inițiale conține cel puțin o stare finală).

- ***Exemplu:*** Se dă următorul AFN- λ.



a) Calculăm λ–închiderile tuturor stărilor.

$<q_0> = \{q_0, q_2, q_3, q_4, q_5, q_6\}$
$<q_1> = \{q_1, q_2, q_4, q_6\}$
$<q_2> = \{q_2, q_4, q_6\}$
$<q_3> = \{q_3, q_5, q_2, q_6, q_4\}$
$<q_4> = \{q_4, q_6\}$
$<q_5> = \{q_5, q_2, q_6, q_4\}$
$<q_6> = \{q_6\}$

b) Verificăm dacă cuvântul **abbaa** este acceptat sau respins de acest AFN-λ, folosind configurații.

$(q_0, abbaa) \vdash^{\lambda^*} (q_{023456}, abbaa) \vdash^{a} (q_{0136}, bbaa) \vdash^{\lambda^*} (q_{0123456}, bbaa) \vdash^{b} (q_{2365}, baa)$

$\vdash^{\lambda^*} (q_{23456}, baa) \vdash^{b} (q_{3652}, aa) \vdash^{\lambda^*} (q_{23456}, aa) \vdash^{a} (q_{36}, a) \vdash^{\lambda^*} (q_{23456}, a)$

$\vdash^{a} (q_{36}, \lambda) \vdash^{\lambda^*} (q_{23456}, \lambda)$, $\{q_2, q_3, q_4, q_5, q_6\} \cap F = \{q_2, q_6\} \neq \emptyset$ => abbaa acceptat

- Desenați câte un AFN / AFD pentru limbajele următoare.

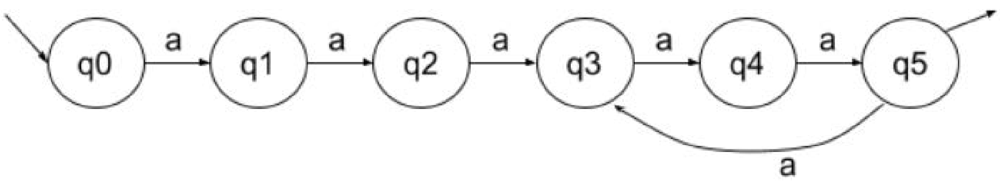
$L1 = \{a^{3k+2} \mid k \geq 1\}$ (AFD)

$L2 = \{w \in \{a, b, c\}^* \mid (|w|_b \; mod \; 5) \text{ este impar}\}$ (AFD)

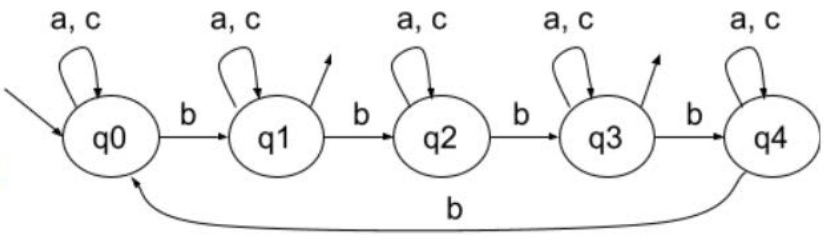
$L3 = \{waaa \mid w \in \{a, b\}^*\}$ (AFN și AFD)

$L4 = \{xababy \mid x, y \in \{a, b\}^*\}$ (AFN și AFD)

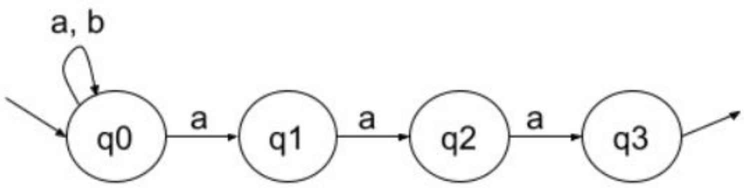
L1:



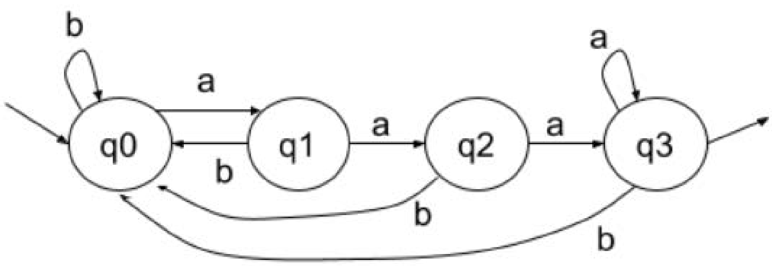
L2:

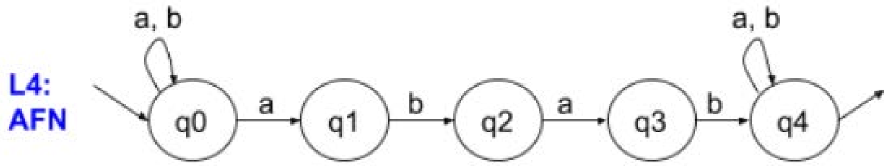


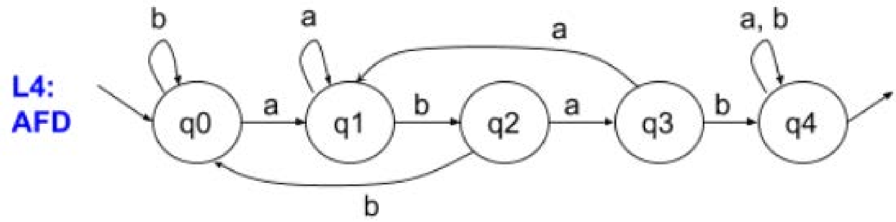
L3: AFN



L3: AFD





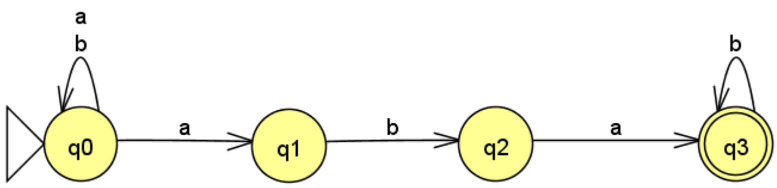


~ Temă ~

EX_1: Desenați câte un AFD pentru fiecare limbaj dat.

$L5 = \{a^{2n}b^{3k} \mid n \geq 0, k \geq 0\}$ Ce se schimbă dacă avem k ≥ 1 ?
$L6 = \{a^{2n}b^{3k} \mid n \geq 1, k \geq 1\}$ Ce se schimbă dacă avem k ≥ 0 ?
$L7 = \{w \in \{0,1\}^* \mid w \text{ începe cu 1 și se termină cu } 0\}$
$L8 = \{w \in \{0,1\}^* \mid w \text{ conține cel puțin trei de } 1\}$
$L9 = \{w \in \{0,1\}^* \mid |w| \geq 3 \text{ și al treilea simbol din w este un } 0\}$

EX_2: a) Pentru următorul AFN construiți un AFD echivalent.
b) Verificați pe AFD și AFN dacă cuvântul **bababb** este acceptat sau nu.



EX_3: Pentru următorul AFN–λ: **(a)** calculaţi λ-închiderile tuturor stărilor şi **(b)** verificaţi dacă cuvântul **aababb** este acceptat sau nu.