

~ Seminar 1 ~

alfabet = mulțime finită și nevidă de simboluri

cuvânt peste un alfabet Σ = secvență finită de simboluri din Σ

limbaj formal peste un alfabet Σ = mulțime (finită sau infinită) de **cuvinte** (deci *lungimea* cuvintelor este finită, dar numărul de cuvinte din limbaj poate fi infinit)

$|w|$ = lungimea cuvântului w (numărul de simboluri) (**ex:** $|\lambda| = 0$, $|abac| = 4$)

$|w|_a$ = numărul de apariții ale simbolului a în cuvântul w (**ex:** $|01101|_0 = 2$, $|cbcb|_a = 0$)

➤ Operații cu limbaje

Reuniune $L = L_1 \cup L_2$

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

Exemplu:

$L_1 = \{a, ab, abc\}$

$L_2 = \{a, bd, cd\}$

$\rightarrow L = \{a, ab, abc, bd, cd\}$

Concatenare $L = L_1 \cdot L_2$

$L = \{w_i w_j \mid w_i \in L_1 \text{ și } w_j \in L_2\}$ (**concatenarea NU este comutativă !!**)

Exemplu:

$L_1 = \{a, ab, abc\}$

$L_2 = \{a, bd, cd\}$

$\rightarrow L = \{aa, abd, acd, aba, abbd, abcd, abca, abcbd, abccd\}$

Stelare $L = L_1^* = \bigcup_{n \geq 0} (L_1^n) = \{\lambda\} \cup L_1 \cup (L_1)^2 \cup (L_1)^3 \cup \dots$

$L_1^0 = \{\lambda\}$ (**λ este cuvântul vid, de lungime 0**)

$L_1^n = \{w_1 w_2 \dots w_n \mid w_i \in L_1 \text{ pentru orice } 1 \leq i \leq n\}$

Exemplu:

$L_1 = \{a, ab, abc\}$

$\rightarrow (L_1)^* = \{\lambda; a, ab, abc; aa, aab, aabc, aba, abab, ababc, abca, abcab, abcabc; \dots\}$

Obs: Cuvântul vid λ va aparține limbajului stelat, indiferent dacă înainte aparținea sau nu limbajului inițial.

Ridicare la putere nenulă $L = L_1^+ = \bigcup_{n \geq 1} (L_1^n)$

$$L^* = L^+ \cup \{\lambda\}$$

$$L^+ = L^* \setminus \{\lambda\} \Leftrightarrow \lambda \notin L$$

Obs: Atenție la ridicarea la putere a cuvintelor! (Să nu distribuiți puterea ca la înmulțire, ci concatenați cuvântul cu el însuși de câte ori spune puterea.)

$$(abc)^2 = abcabc \quad ; \quad (abc)^2 \neq aabbcc = a^2b^2c^2$$

Obs: Cuvântul vid este element neutru la concatenare: $w \cdot \lambda = \lambda \cdot w = w$

➤ Definiție DFA și mod de funcționare

Automat Finit Determinist (AFD) / Deterministic Finite Automaton (DFA)

$$\text{DFA} = (Q, \Sigma, q_0, F, \delta)$$

Q mulțimea de stări (mulțime finită și nevidă)

Σ alfabetul de intrare (mulțime finită de simboluri) („Sigma”)

$q_0 \in Q$ starea inițială

$F \subseteq Q$ mulțimea de stări finale

$\delta : Q \times \Sigma \rightarrow Q$ funcția de tranziție („delta”)

• Algoritm care verifică dacă un cuvânt este sau nu acceptat de un automat DFA:

- Pornim din starea inițială și avem cuvântul de intrare.
- Cât timp este posibil, la fiecare pas încercăm să procesăm câte un simbol din cuvântul de intrare (de la stânga la dreapta) astfel:
 - conform funcției de tranziție (dacă $\delta(q_i, x) = q_j$), din starea curentă (q_i) citind simbolul curent (x) din cuvântul de intrare ajungem într-o anumită stare (q_j) (care va deveni noua stare curentă din care vom citi următorul simbol din cuvânt).
- Algoritmul se termină în 3 cazuri:
 - a) dacă simbolul curent din cuvânt nu poate fi procesat (din cauză că nu există tranziție din starea curentă cu acel simbol), atunci cuvântul este **respins**.
 - b) dacă după procesarea întregului cuvânt s-a ajuns într-o stare nefinală (din mulțimea $Q - F$), atunci cuvântul este **respins**.
 - c) dacă după procesarea întregului cuvânt s-a ajuns într-o stare finală (din mulțimea F), atunci cuvântul este **acceptat**.

- Pentru a scrie pas cu pas verificarea acceptării/respingerii unui cuvânt de un DFA folosim „**configurații**” (sau “descrieri instantanee”) ale automatului, adică perechi formate din starea curentă și ce a mai rămas de citit din cuvântul de intrare.

$$(r_0, a_1 a_2 a_3 \dots a_n) \vdash (r_1, a_2 a_3 \dots a_n) \vdash (r_2, a_3 \dots a_n) \vdash \dots \vdash (r_n, \lambda), r_0 = q_0, r_n \in F$$

- Limbajul acceptat de un DFA numit M:
$$L(M) = \{w \mid (q_0, w) \vdash^* (p, \lambda), p \in F\} \text{ sau } L(M) = \{w \mid \hat{\delta}(q_0, w) \in F\} \text{ (unde } \hat{\delta} : Q \times \Sigma^* \rightarrow Q \text{ este funcția de tranziție extinsă la cuvinte)}$$

➤ Definiție NFA și mod de funcționare

Automat Finit Nedeterminist (AFN) / Nondeterministic Finite Automaton (NFA)

$NFA = (Q, \Sigma, q_0, F, \delta)$

Q mulțimea de stări (mulțime finită și nevidă)

Σ alfabetul de intrare (mulțime finită de simboluri) („Sigma”)

$q_0 \in Q$ starea inițială

$F \subseteq Q$ mulțimea de stări finale

$\delta: Q \times \Sigma \rightarrow 2^Q$ funcția de tranziție („delta”)

- Limbajul acceptat de un NFA numit M :
 $L(M) = \{w \mid (q_0, w) \vdash^* \{(p_1, \lambda), \dots, (p_k, \lambda)\}, \{p_1, \dots, p_k\} \cap F \neq \emptyset\}$
sau $L(M) = \{w \mid \delta(q_0, w) \cap F \neq \emptyset\}$

• Algoritm care verifică dacă un cuvânt este sau nu acceptat de un automat NFA:

- Pornim din (mulțimea formată din) starea inițială și avem cuvântul de intrare.
- *Cât timp este posibil*, aplicăm aceeași idee ca la DFA, doar că în loc să avem o singură stare curentă, la fiecare pas avem o mulțime de stări curente. Din fiecare stare din această mulțime încercăm să citim simbolul curent din cuvânt și ajungem într-o nouă mulțime de stări (din care vom încerca să citim simbolul următor din cuvânt).
Formal, pentru mulțimea de stări curente R și simbolul curent x avem:

$$\delta(R, x) = \bigcup_{q \in R} \delta(q, x)$$

- Algoritmul se termină în 3 cazuri:
 - a) dacă simbolul curent din cuvânt *nu poate fi procesat* (din cauză că nu există tranziție din niciuna dintre stările curente cu acel simbol), atunci cuvântul este **respins**.
 - b) dacă după procesarea întregului cuvânt s-a ajuns într-o mulțime de stări R care nu conține nicio stare finală ($R \cap F = \emptyset$), atunci cuvântul este **respins**.
 - c) dacă după procesarea întregului cuvânt s-a ajuns într-o mulțime de stări R care conține cel puțin o stare finală ($R \cap F \neq \emptyset$), atunci cuvântul este **acceptat**.

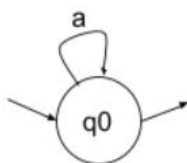
Obs: La DFA și NFA, $\lambda \in L \Leftrightarrow q_0 \in F$ (cuvântul vid este acceptat de automat dacă și numai dacă starea inițială este și stare finală).

➤ Exerciții: Desenați DFA / NFA pentru limbajele date

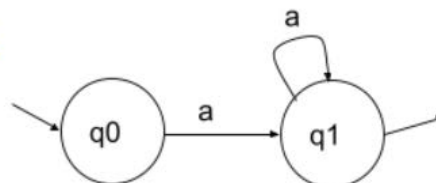
$L1 = a^* = \{a^n, n \geq 0\} = \{\lambda = a^0, a = a^1, aa = a^2, aaa = a^3, aaaa = a^4, \dots\}$

$L2 = a^+ = \{a^n, n \geq 1\} = \{a = a^1, aa = a^2, aaa = a^3, aaaa = a^4, \dots\}$

L1:



L2:



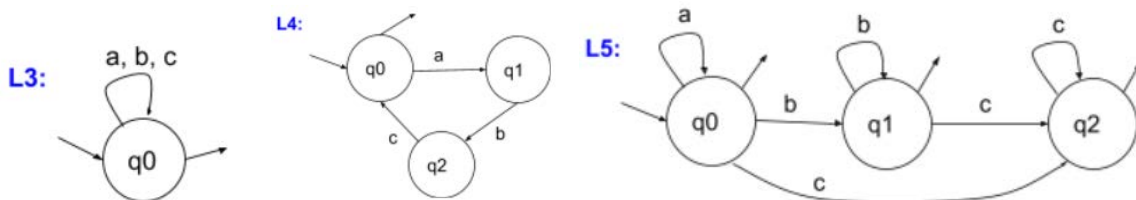
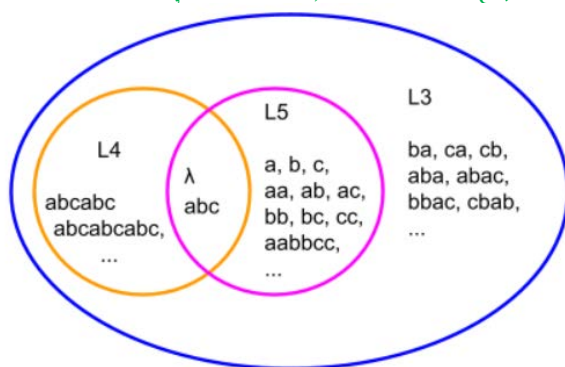
- Când folosim în automat: buclă / ciclu / stări noi ?

$L3 = \{a, b, c\}^* = \{\lambda, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, \dots, caa, cab, \dots\}$

$L4 = (abc)^* = \{\lambda, abc, abcabc, abcabcabc, \dots\}$

$L5 = \{a^n b^k c^p \mid n \geq 0, k \geq 0, p \geq 0\} = \{\lambda, a, a^2, a^3, \dots, b, b^2, b^3, \dots, c, c^2, c^3, \dots, ab, ab^2, a^2b, ab^3, a^3b, a^2b^2, \dots, ac, ac^2, a^2c, ac^3, a^3c, a^2c^2, \dots, bc, bc^2, b^2c, bc^3, b^3c, b^2c^2, \dots, abc, a^2bc, ab^2c, abc^2, \dots, a^2b^2c^2, \dots\}$

Există vreo relație de incluziune / intersecție între unele dintre limbajele L3, L4, L5 ?
(Da: $L4 \subset L3$ și $L5 \subset L3$; $L4 \cap L5 = \{\lambda, abc\}$)



Observații:

→ L3 conține toate cuvintele posibile peste alfabetul $\{a, b, c\}$, cuvinte de lungime ≥ 0 , în care literele pot fi în orice ordine. De aceea, punem toate simbolurile pe **bucla aceleiași stări** pentru a permite amestecarea lor.

→ L4 conține cuvinte obținute prin concatenarea cuvântului “abc” cu el însuși de $n \geq 0$ ori. De aceea, folosim un **ciclu** în care avem câte o muchie pentru fiecare literă din cuvântul “abc”, exact în această ordine. Setăm stare finală doar pe cea în care ajungem după ce citim c, ultima literă din cuvânt.

→ L5 conține cuvinte de forma $a^n b^k c^p$ pe care le obținem dând orice valori numere naturale pentru n, k și p. Literele trebuie să fie **neamestecate** și **neapărat în această ordine**: întâi a-uri, apoi b-uri, apoi c-uri. Pentru fiecare literă (a, b, c) trebuie să folosim **câte o buclă pe o stare diferită** pentru a nu amesteca literele; când începem să citim o literă nouă (primul b sau c) **trebuie să mergem într-o stare nouă** și nu avem voie să avem muchii de întoarcere spre stări plecând din care citeam fostele litere, pentru a asigura ordinea corectă (niciun a după b și niciun a sau b după c).

Pentru $k=0$ (lipsă b-uri) adăugăm muchia cu c care sare din q0 direct în q2. Setăm **stările finale** q0 (pt a accepta cuvântul vid / cuvinte doar cu a), q1 (pt cuvinte doar cu b / cu a și b) și q2 (pt cuvinte doar cu c / cu a și c / cu b și c / cu a, b și c).

• **Temă de gândire:**

$L6 = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$ (același număr de a-uri și b-uri, amestecate oricum)

$L7 = \{a^n b^n \mid n \geq 0\} \subset L6$ (același număr de a-uri și b-uri, obligatoriu întâi a-uri, apoi b-uri)

versus

$L8 = \{w \in \{a, b\}^* \mid |w|_a \bmod 2 = |w|_b \bmod 2\}$ (aceeași paritate pt numărul de a-uri și b-uri)

- Putem să desenăm un DFA? Dar un NFA?

- Puteți să dați un exemplu de limbaj pentru care putem desena un NFA, dar nu un DFA?

Observații:

→ $L6$ și $L7$ nu sunt limbaje regulate (nu există niciun automat finit care să le accepte).

Argument intuitiv: nu putem să verificăm egalitatea dintre **numărul total** de a-uri și cel de b-uri pentru o **infinitate de valori posibile** ale lui n (orice număr natural) deoarece DFA / NFA au la dispoziție doar **un număr finit de stări** în care putem “memora” o cantitate finită de informații.

→ În schimb $L8$ este limbaj regulat pentru că există **un număr finit de cazuri** pentru **paritatea numărului** de a-uri și cel de b-uri.

Idee: Memorăm în indicii stărilor care sunt resturile împărțirii la 2 pentru numărul de a-uri și cel de b-uri citite (începând din starea inițială) până în acel moment; primul indice este pentru a-uri și al doilea pentru b-uri.

