

## Cuprins

|                                                                                            |        |
|--------------------------------------------------------------------------------------------|--------|
| ~ Seminar 1 ~ .....                                                                        | - 4 -  |
| Automat Finit Determinist .....                                                            | - 4 -  |
| Automat Finit Nedeterminist .....                                                          | - 4 -  |
| EXAMPLE: .....                                                                             | - 5 -  |
| $L1 = a^* = \{a^n, n \geq 0\}$ .....                                                       | - 5 -  |
| $L2 = a^+ = \{a^n, n \geq 1\}$ .....                                                       | - 5 -  |
| $L3 = \{a^{2n}, n \geq 0\}$ .....                                                          | - 5 -  |
| $L4 = \{a^{2n+1}, n \geq 0\}$ .....                                                        | - 5 -  |
| $L5 = \{a^{2n}, n \geq 1\}$ .....                                                          | - 5 -  |
| $L6 = \{a^{2n+1}, n \geq 1\}$ .....                                                        | - 6 -  |
| $L7 = \{a^{2n} b^{3m}, n \geq 0, m \geq 0\}$ .....                                         | - 6 -  |
| $L8 = \{a^{2n} b^{3m}, n \geq 1, m \geq 0\}$ .....                                         | - 7 -  |
| $L9 = \{a^{2n} b^{3m}, n \geq 0, m \geq 1\}$ .....                                         | - 7 -  |
| $L10 = \{a^{2n} b^{3m}, n \geq 1, m \geq 1\}$ .....                                        | - 7 -  |
| Operații cu limbaje .....                                                                  | - 8 -  |
| Reuniune $L = L_1 \cup L_2$ .....                                                          | - 8 -  |
| Concatenare $L = L_1 \cdot L_2$ .....                                                      | - 8 -  |
| Stelare $L = L_1^* = \bigcup_{n \geq 0} (L_1^n)$ .....                                     | - 8 -  |
| Ridicare la putere nenulă $L = L_1^+ = \bigcup_{n \geq 1} (L_1^n)$ .....                   | - 8 -  |
| “Compunerea” automatelor finite .....                                                      | - 9 -  |
| Reuniune .....                                                                             | - 9 -  |
| Concatenare .....                                                                          | - 9 -  |
| Stelare .....                                                                              | - 9 -  |
| Transformarea AFN $\rightarrow$ AFD .....                                                  | - 9 -  |
| EXEMPLU: .....                                                                             | - 10 - |
| Verificare acceptare cuvânt de către automat finit .....                                   | - 12 - |
| ~ Seminar 2 ~ .....                                                                        | - 14 - |
| Automat Finit Nedeterminist cu $\lambda$ -tranziții .....                                  | - 14 - |
| $\lambda$ -închiderea unei stări .....                                                     | - 14 - |
| Verificare acceptare cuvânt de către automat AFN- $\lambda$ .....                          | - 14 - |
| EXEMPLU: .....                                                                             | - 15 - |
| Transformarea AFN- $\lambda \rightarrow$ AFD .....                                         | - 18 - |
| EXEMPLU: .....                                                                             | - 19 - |
| Automatul minimal .....                                                                    | - 22 - |
| EXEMPLU: .....                                                                             | - 23 - |
| Condiții asupra numărului de apariții ale unui caracter într-un cuvânt .....               | - 25 - |
| EXAMPLE: .....                                                                             | - 25 - |
| $L_1 = \{w \in \{0,1\}^* a.i. \mid  w _0 = par,  w _1 = impar\}$ .....                     | - 25 - |
| $L_2 = \{w \in \{a,b\}^* a.i. \mid  w _a = 2n+1,  w _b = 3m+2, n \geq 0, m \geq 0\}$ ..... | - 26 - |

|                                                                              |        |
|------------------------------------------------------------------------------|--------|
| $L_3 = \{w \in \{a,b\}^* \mid a \text{ și } b \text{ nu apar în } w\}$ ..... | - 26 - |
| ~ Seminar 3 ~ .....                                                          | - 27 - |
| Expresii regulate .....                                                      | - 27 - |
| Precedența operațiilor *, ·, + .....                                         | - 27 - |
| Transformarea expresie_regulată → automat_finit .....                        | - 27 - |
| Transformarea automat_finit → expresie_regulată .....                        | - 28 - |
| Câteva formule utile.....                                                    | - 29 - |
| EXEMPLU:.....                                                                | - 29 - |
| Gramatici.....                                                               | - 31 - |
| Gramatici independente de context (GIC) / Context-free grammars (CFG) .....  | - 31 - |
| Gramatici liniare .....                                                      | - 31 - |
| Gramatici liniare la dreapta.....                                            | - 31 - |
| Gramatici liniare la stânga .....                                            | - 31 - |
| Gramatici regulate.....                                                      | - 31 - |
| EXEMPLE (gramatici regulate): .....                                          | - 32 - |
| $L1 = a^* = \{a^n, n \geq 0\}$ .....                                         | - 32 - |
| $L2 = a^+ = \{a^n, n \geq 1\}$ .....                                         | - 32 - |
| $L3 = \{a^{2n}, n \geq 0\}$ .....                                            | - 32 - |
| $L4 = \{a^{2n+1}, n \geq 0\}$ .....                                          | - 32 - |
| $L5 = \{a^{2n}, n \geq 1\}$ .....                                            | - 32 - |
| $L6 = \{a^{2n+1}, n \geq 1\}$ .....                                          | - 32 - |
| $L7 = \{a^{2n} b^{3m}, n \geq 0, m \geq 0\}$ .....                           | - 32 - |
| Analogia între automate finite și gramatici regulate.....                    | - 33 - |
| ~ Seminar 4 ~ .....                                                          | - 34 - |
| EXEMPLE (gramatici independente de context): .....                           | - 34 - |
| $L1 = \{a^n b^n, n \geq 0\}$ .....                                           | - 34 - |
| $L2 = \{a^n b^n, n \geq 1\}$ .....                                           | - 34 - |
| $L3 = \{a^n b^{2n}, n \geq 1\}$ .....                                        | - 35 - |
| $L4 = \{a^{2n} b^k c^{3n}, n \geq 0, k \geq 1\}$ .....                       | - 35 - |
| $L5 = \{a^{2n} b^k c^{3n}, n \geq 1, k \geq 1\}$ .....                       | - 35 - |
| $L6 = \{a^{2n} c^{3n} b^k, n \geq 1, k \geq 1\}$ .....                       | - 36 - |
| Verificare generare cuvânt de către gramatică.....                           | - 36 - |
| $L7 = \{a^n b^k c^{2k} d^n, n \geq 1, k \geq 0\}$ .....                      | - 38 - |
| $L8 = \{a^n b^k c^{2k} d^n, n \geq 0, k \geq 1\}$ .....                      | - 38 - |
| $L9 = \{a^n b^{2n} c^{2k} d^k, n \geq 0, k \geq 1\}$ .....                   | - 39 - |
| $L10 = \{a^n b^{2n} c^k d^m e^{3k}, n, k, m \geq 0\}$ .....                  | - 39 - |
| Transformarea GIC → gramatică în F.N. Chomsky .....                          | - 40 - |
| Automatul Push-Down (APD) / Automatul Stivă .....                            | - 43 - |
| Modalități de acceptare a unui cuvânt de către un APD .....                  | - 44 - |
| EXEMPLE (automate push-down):.....                                           | - 45 - |
| $L1 = \{a^n b^n, n \geq 1\}$ .....                                           | - 45 - |
| $L2 = \{a^n b^{2n}, n \geq 0\}$ .....                                        | - 46 - |
| $L3 = \{a^{2n} b^k c^{3k} d^n, n \geq 1, k \geq 1\}$ .....                   | - 47 - |
| Verificare acceptare cuvânt de către un APD .....                            | - 48 - |

$$L4 = \{a^{2n} b^{3n} c^{2k} d^k, n \geq 1, k \geq 1\} \dots\dots\dots - 48 -$$

## ~ Seminar 1 ~

(AFD; AFN; Operații cu limbaje; Compunere automate; Transformarea AFN  $\rightarrow$  AFD)

### ***Automat Finit Determinist***

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

$Q$  mulțimea de stări

$\Sigma$  alfabetul de intrare

$q_0 \in Q$  starea inițială

$F \subseteq Q$  mulțimea de stări finale

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

Pentru a verifica dacă un cuvânt este sau nu acceptat de un automat AFD:

- se pornește din starea inițială  $q_0$
- cât timp este posibil:
  - având starea curentă și caracterul curent din cuvânt, aplicăm funcția  $\delta$  pentru a determina noua stare
  - repetăm pentru noua stare și caracterul următor din cuvânt
- algoritmul se termină în 3 cazuri:
  - chiar dacă nu s-a citit complet tot cuvântul, dar automatul s-a blocat pentru că funcția de tranziție nu era definită pentru starea curentă și caracterul curent  $\rightarrow$  “cuvânt neacceptat”
  - dacă întreg cuvântul a fost citit, dar automatul a ajuns într-o stare care nu este finală (adică din mulțimea  $Q \setminus F$ )  $\rightarrow$  “cuvânt neacceptat”
  - dacă întreg cuvântul a fost citit, iar automatul a ajuns într-o stare care este finală (adică din mulțimea  $F$ )  $\rightarrow$  “cuvânt acceptat”

### ***Automat Finit Nedeterminist***

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

$Q$  mulțimea de stări

$\Sigma$  alfabetul de intrare

$q_0 \in Q$  starea inițială

$F \subseteq Q$  mulțimea de stări finale

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

Diferența dintre AFD și AFN constă în modul în care este definită funcția de tranziție.

La AFD: pentru orice pereche de o stare și un caracter, avem *cel mult* o stare de continuare (putem să nu avem niciuna, adică funcția să fie nedefinită pentru acele valori).

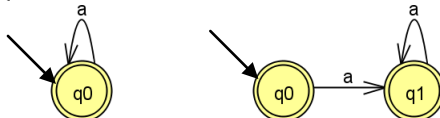
La AFN: pentru orice pereche de o stare și un caracter, avem ca posibilitate de continuare o submulțime a mulțimii de stări  $Q$  (putem să nu avem niciuna dacă funcția este nedefinită; putem să avem o singură stare de continuare; sau putem să avem mai multe stări de continuare, aici apare nedeterminismul).

Pentru a verifica dacă un cuvânt este sau nu acceptat de un automat AFN:

Se procedează analog ca în cazul AFD-ului, excepția fiind că la fiecare aplicare a funcției delta trebuie să reținem toate posibilitățile de stări de continuare, iar la pasul următor să reaplicăm delta pentru toate acestea împreună cu caracterul curent.

### EXEMPLE:

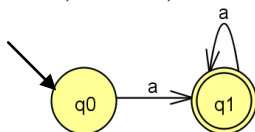
$$L1 = a^* = \{a^n, n \geq 0\} = \{\lambda = a^0, a = a^1, aa = a^2, aaa = a^3, aaaa = a^4, \dots\}$$



Ambele soluții sunt AFD-uri.

**Obs:** De fiecare dată când  $\lambda$  (cuvântul vid) trebuie să fie recunoscut de un AFD sau de un AFN (simplu, adică AFN fără  $\lambda$ -tranziiții), trebuie ca starea inițială  $q_0$  să fie și stare finală.

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



Diferența față de exemplul anterior?

Aici  $\lambda$  nu trebuie acceptat, deci starea inițială  $q_0$  trebuie să nu fie stare finală.

$$L3 = \{a^{2n}, n \geq 0\} = \{\lambda = a^0, aa = a^2, aaaa = a^4, aaaaaa = a^6, \dots\}$$

$$L4 = \{a^{2n+1}, n \geq 0\} = \{a = a^1, aaa = a^3, aaaaa = a^5, \dots\}$$



Cele două automate au aceleași stări și aceleași tranziții. Diferă doar mulțimea stărilor finale. Dacă setăm  $q_0$  finală, atunci vor fi acceptate cuvintele de lungime pară (inclusiv zero), iar dacă setăm în schimb  $q_1$  finală, atunci vor fi acceptate cuvintele de lungime impară.

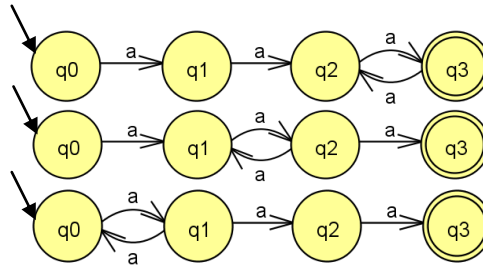
**Obs:** De fiecare dată când la putere apare un coeficient, înseamnă că vom avea un circuit de lungime egală cu acest coeficient (sau lungime proporțională cu el, în cazul în care puterea era aplicată unui cuvânt, nu doar unei litere).

$$L5 = \{a^{2n}, n \geq 1\} = \{aa = a^2, aaaa = a^4, aaaaaa = a^6, \dots\}$$



În primul rând trebuie să ne asigurăm că obligăm automatul să citească cei doi de “a” (cel mai mic cuvânt din limbaj) și că nu acceptă ceva mai scurt de atât. Pentru asta, adăugăm cele trei stări și tranzițiile de la  $q_0$  spre  $q_1$  și de la  $q_1$  spre  $q_2$ , iar apoi setăm  $q_2$  să fie stare finală. Apoi vedem că avem coeficientul 2 la putere, deci trebuie să avem un circuit de lungime 2. Avem două variante: fie adăugăm tranziție de la  $q_2$  spre  $q_1$ , fie adăugăm tranziție de la  $q_1$  spre  $q_0$ . Ambele automate recunosc același limbaj (dar primul este AFD, iar al doilea este AFN pentru că din  $q_1$  cu “a” putem ajunge și în  $q_0$  și în  $q_2$ ).

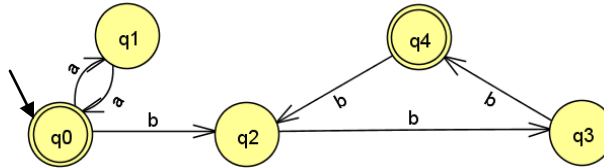
$$L6 = \{a^{2n+1}, n \geq 1\} = \{aaa=a^3, aaaaa=a^5, aaaaaaa=a^7, \dots\}$$



La fel ca la exemplul anterior, ne vom asigura întâi că automatul nu acceptă nimic mai scurt decât cel mai mic cuvânt din limbaj, adică “aaa”. Deci vom adăuga cele 4 stări și tranzițiile către dreapta și vom seta starea  $q_3$  finală. Apoi pentru a avea circuitul de lungime 2 (egală cu coeficientul lui  $n$ ), avem cele trei variante. Dacă adăugăm tranziția de la  $q_3$  spre  $q_2$  vom obține un AFD, iar dacă alegem una din celelalte două variante vom obține un AFN.

$$L7 = \{a^{2n}b^{3m}, n \geq 0, m \geq 0\}$$

$$= \{\lambda, a^2, a^4, a^6, a^8, \dots, b^3, b^6, b^9, b^{12}, \dots, a^2b^3, a^2b^6, a^4b^3, a^4b^6, a^6b^3, a^6b^6, \dots\}$$

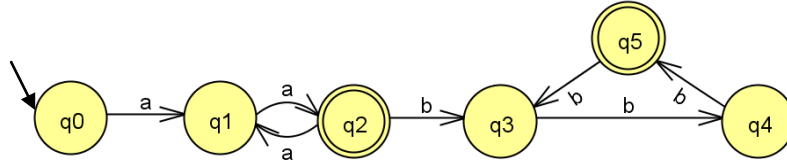


În primul rând vedem că cel mai mic cuvânt ce trebuie acceptat este  $\lambda$ , deci setăm starea inițială  $q_0$  să fie și stare finală. Apoi facem circuit de lungime 2 cu litera “a”. După ce citim un număr par de a-uri (inclusiv zero), ajungem în starea  $q_0$ , deci de aici trebuie să pornim cu citirea b-urilor. Adăugăm  $q_2$ ,  $q_3$  și cele două tranziții cu b. Știm că va trebui să avem un circuit de lungime 3 cu b-uri, dar trebuie să fim de asemenea atenți că literele de “a” și de „b” nu pot fi amestecate între ele, după ce am citit primul “b” nu mai avem voie să citim niciun „a”. Deci nu putem avea muchie de întoarcere de la  $q_3$  către  $q_0$  pentru că în  $q_0$  se pot citi a-uri. Înseamnă că avem nevoie de a nouă stare, așa că adăugăm  $q_4$  și tranziții de la  $q_3$  spre  $q_4$  și de la  $q_4$  spre  $q_2$  pentru a închide circuitul.

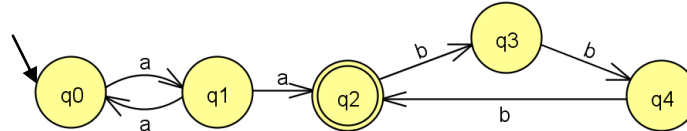
$$L8 = \{a^{2n}b^{3m}, n \geq 1, m \geq 0\}$$

$$= \{a^2, a^4, a^6, a^8, \dots, a^2b^3, a^2b^6, a^4b^3, a^4b^6, a^6b^3, a^6b^6, \dots\}$$

În acest caz, cuvintele conțin obligatoriu a-uri, b-urile putând să apară sau nu.



Trebuie să obligăm automatul să citească număr par de a-uri, dar cel puțin două, deci nu se mai poate întoarce din  $q_1$  în  $q_0$  cu al doilea “a”, ci trebuie să continue spre  $q_2$  și setăm  $q_2$  ca stare finală. Apoi începând din  $q_2$  citim doi de “b”, dar pentru al treilea nu putem închide circuitul întorcându-ne în  $q_2$  dacă am ales ca muchie de întoarcere pentru a-uri de la starea  $q_2$  spre starea  $q_1$ . Deci trebuie să adăugăm și starea  $q_5$  care va fi și ea finală și cele două tranziții de la  $q_4$  spre  $q_5$  și de la  $q_5$  spre  $q_3$ .



Dar dacă am fi ales ca muchie de întoarcere pentru a-uri de la  $q_1$  spre  $q_0$ ?

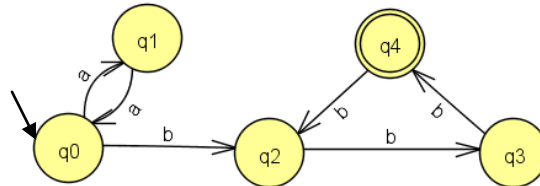
Atunci nu am mai fi avut nevoie de starea  $q_5$ , pentru că puteam să închidem circuitul b-urilor de la  $q_4$  spre  $q_2$  și am fi avut doar starea  $q_2$  finală.

Doar că în acest caz am fi obținut un AFN (din cauza lui  $q_1$ ).

$$L9 = \{a^{2n}b^{3m}, n \geq 0, m \geq 1\}$$

$$= \{b^3, b^6, b^9, b^{12}, \dots, a^2b^3, a^2b^6, a^4b^3, a^4b^6, a^6b^3, a^6b^6, \dots\}$$

În acest caz, cuvintele conțin obligatoriu b-uri, a-urile putând să apară sau nu.



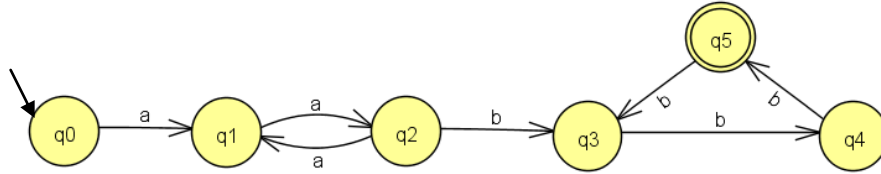
Observăm că automatul este foarte asemănător cu cel pentru limbajul L7.

Diferența este că acum starea  $q_0$  nu mai este stare finală, ceea ce înseamnă că putem sau nu să trecem prin circuitul de a-uri, dar sigur suntem obligați să citim b-uri înainte de a ajunge într-o stare finală.

$$L10 = \{a^{2n}b^{3m}, n \geq 1, m \geq 1\}$$

$$= \{a^2b^3, a^2b^6, a^4b^3, a^4b^6, a^6b^3, a^6b^6, \dots\}$$

În acest caz, cuvintele conțin obligatoriu și a-urile, și b-urile.



Acest automat seamănă cu cel pentru L8 (prima variantă), unde eram obligați să avem a-uri. În plus, pentru că acum trebuie să avem și b-uri, starea  $q_2$  nu va mai fi stare finală.

### 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**  $L = L_1 \cdot L_2$

$$L = \{w_i w_j \mid w_i \in L_1 \text{ și } w_j \in L_2\}$$

Exemplu:

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

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

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

**Stelare**  $L = L_1^* = \bigcup_{n \geq 0} (L_1^n)$

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

Exemplu:

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

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

**Obs:** Cuvântul vid  $\lambda$  va aparține limbajului stelat, indiferent dacă înainte aparținea sau nu limbajului inițial.

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

La fel ca la stelare, doar că nu mai există cuvântul vid.  $L^+ = L^* \setminus \{\lambda\}$



**Obs:** Atenție la ridicarea la putere a cuvintelor! (Să nu distribuți puterea ca la înmulțire, ci concatenați cuvântul cu el însuși de câte ori spune puterea.)  
 $(abc)^2 = abcabc$  ;  $(abc)^2 \neq aabbcc = a^2b^2c^2$

## “Compunerea” automatelor finite

### Reuniune

Dacă avem deja construite două automate finite (având stările inițiale  $q_{01}$  și  $q_{02}$  și mulțimile stărilor finale  $F_1$  și  $F_2$ ) care acceptă respectiv limbajele  $L_1$  și  $L_2$  și dorim să obținem un automat finit care să accepte limbajul  $L = L_1 \cup L_2$ :

- introducem o nouă stare  $q_0$  care va fi noua stare inițială
- adăugăm două  $\lambda$ -tranziții de la  $q_0$  spre  $q_{01}$  și de la  $q_0$  spre  $q_{02}$
- noua mulțime de stări finale va fi  $F = F_1 \cup F_2$

### Concatenare

Dacă avem deja construite două automate finite (având stările inițiale  $q_{01}$  și  $q_{02}$  și mulțimile stărilor finale  $F_1$  și  $F_2$ ) care acceptă respectiv limbajele  $L_1$  și  $L_2$  și dorim să obținem un automat finit care să accepte limbajul  $L = L_1 \cdot L_2$ :

- noua stare inițială va fi  $q_{01}$ , starea inițială a primului automat
- adăugăm  $\lambda$ -tranziții de la fiecare stare finală din  $F_1$  spre  $q_{02}$
- noua mulțime de stări finale va fi  $F = F_2$

### Stelare

Dacă avem deja construit un automat finit (având starea inițială  $q_{01}$  și mulțimea stărilor finale  $F_1$ ) care acceptă limbajul  $L_1$  și dorim să obținem un automat finit care să accepte limbajul  $L = L_1^*$ :

- introducem o nouă stare  $q_0$  care va fi noua stare inițială
- adăugăm o  $\lambda$ -tranziție de la  $q_0$  spre  $q_{01}$
- adăugăm  $\lambda$ -tranziții de la fiecare stare finală din  $F_1$  spre  $q_0$
- noua mulțime de stări finale va fi  $F = \{q_0\}$  (sau putem lăsa  $F = \{q_0\} \cup F_1$ )

## Transformarea AFN $\rightarrow$ AFD

Având dat un AFN (reprezentat sub formă de graf), vom calcula funcția delta completând un tabel astfel:

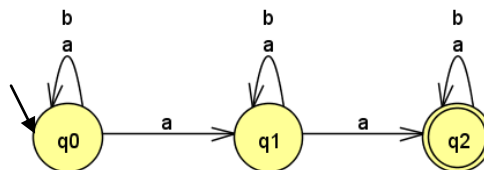
- La capetele de linie vom pune numele stărilor AFN-ului (elementele mulțimii  $Q$ ):  $q_0, q_1, \dots, q_{|Q|-1}$
- La capetele de coloană vom pune caracterele alfabetului AFN-ului (elementele mulțimii  $\Sigma$ )
- Calculăm conținutul primelor  $|Q|$  linii astfel: elementul aflat pe linia  $l_i$  și coloana  $c_j$  reprezintă mulțimea de stări în care se poate ajunge în AFN dacă pornim din starea de pe linia  $l_i$  (în cazul acesta:  $q_{i-1}$  dacă numerotăm liniile începând cu 1) și citim caracterul de pe coloana  $c_j$ . Pentru a afla această mulțime, ne uităm la graful

care reprezintă AFN-ul, căutăm starea de la linia  $l_i$  și pentru fiecare tranziție cu litera de la coloana  $c_j$  care pornește din această stare, adăugăm starea destinație a tranziției la mulțimea curentă din tabel.

- Apoi trebuie să calculăm această funcție delta și pentru noile mulțimi de stări obținute la pașii anteriori (aceste calcule se fac cel mai simplu folosind doar liniile din tabel calculate la pasul anterior, nu și graful automatului AFN)
  - (a) fie alegem să facem calculele pentru *toate noile mulțimi de stări*, și abia apoi să vedem care dintre ele vor apărea în AFD-ul obținut (tot făcând “parcurgerea” în adâncime descrisă mai jos).
  - (b) fie alegem să facem calculele *doar pentru noile mulțimi de stări care vor apărea* în AFD: pentru a decide care vor fi acestea, facem un fel de parcurgere în adâncime a grafului care va reprezenta AFD-ul. Adică pornim cu starea  $q_0$  care va fi starea inițială a AFD-ului, apoi parcurgem prima linie a tabelului (care conține calculele tranzițiilor ce pleacă din  $q_0$ ) și adăugăm la noul automat toate aceste tranziții împreună cu stările-destinație calculate în tabel. Apoi pentru toate aceste stări-destinație, dacă nu au deja în tabel un capăt de linie care să le corespundă, îl vom adăuga și vom calcula acea linie astfel:
    - fie noua stare-mulțime  $\{q_{k1}, q_{k2}, \dots, q_{kp}\}$ . Pentru fiecare coloană  $c_j$ , calculăm elementul din tabel făcând **reuniunea** elementelor aflate pe aceeași coloană  $c_j$  și pe liniile corespunzătoare stărilor  $q_{k1}, q_{k2}, \dots, q_{kp}$ . Apoi revenim la desenarea grafului AFD-ului, și adăugăm tranzițiile proaspăt calculate împreună cu stările-destinație. Reluăm procedeul până nu mai apar stări noi.
- Stările finale ale noului AFD obținut, vor fi acele stări-mulțime (pot avea și un singur element, nu neapărat mai multe) care conțin cel puțin una din stările care erau finale pentru AFN-ul inițial.

### EXEMPLU:

Se dă AFN-ul următor:



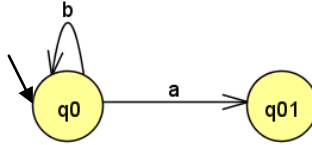
Calculăm tabelul pentru funcția delta: întâi primele 3 linii folosind graful AFN-ului, apoi următoarele linii folosind aceste prime 3 linii ale tabelului.

Vom avea două coloane, pentru că alfabetul automatului inițial este  $\Sigma = \{a, b\}$ .

| $\delta$ | a              | b     |
|----------|----------------|-------|
| $q_0$    | $\{q_0, q_1\}$ | $q_0$ |
| $q_1$    | $\{q_1, q_2\}$ | $q_1$ |
| $q_2$    | $q_2$          | $q_2$ |

Alegem varianta (b), adică să calculăm doar acele stări care vor apărea în AFD.

(1) Pornim cu starea  $q_0$  și îi adăugăm cele două tranziții și stările-destinație necesare, adică  $\{q_0, q_1\} = q_{01}$  (prin notație).



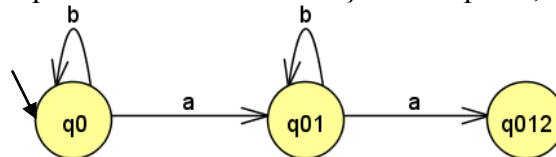
(2) Observăm că a apărut noua stare  $q_{01}$  pentru care nu avem o linie în tabel, așa că o adăugăm și o calculăm pentru cele două coloane.

$$\begin{aligned}\delta(q_{01}, a) &= \delta(\{q_0, q_1\}, a) \\ &= \{\delta(q_0, a)\} \cup \{\delta(q_1, a)\} \\ &= \{q_0, q_1\} \cup \{q_1, q_2\} = \{q_0, q_1, q_2\} = q_{012}\end{aligned}$$

$$\begin{aligned}\delta(q_{01}, b) &= \delta(\{q_0, q_1\}, b) \\ &= \{\delta(q_0, b)\} \cup \{\delta(q_1, b)\} \\ &= \{q_0\} \cup \{q_1\} = \{q_0, q_1\} = q_{01}\end{aligned}$$

| $\delta$ | a                             | b                       |
|----------|-------------------------------|-------------------------|
| $q_0$    | $\{q_0, q_1\} = q_{01}$       | $q_0$                   |
| $q_1$    | $\{q_1, q_2\} = q_{12}$       | $q_1$                   |
| $q_2$    | $q_2$                         | $q_2$                   |
| $q_{01}$ | $\{q_0, q_1, q_2\} = q_{012}$ | $\{q_0, q_1\} = q_{01}$ |

(3) Revenim la desenul AFD-ului și adăugăm tranzițiile care lipsesc, adică cele care pleacă din starea  $q_{01}$ , împreună cu stările-destinație care lipsesc, adică  $q_{012}$ .



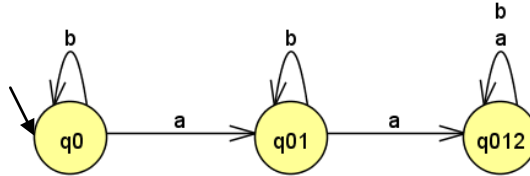
(4) Pentru noua stare apărută,  $q_{012}$ , îi adăugăm linia în tabel și o calculăm.

$$\begin{aligned}\delta(q_{012}, a) &= \delta(\{q_0, q_1, q_2\}, a) \\ &= \{\delta(q_0, a)\} \cup \{\delta(q_1, a)\} \cup \{\delta(q_2, a)\} \\ &= \{q_0, q_1\} \cup \{q_1, q_2\} \cup \{q_2\} = \{q_0, q_1, q_2\} = q_{012}\end{aligned}$$

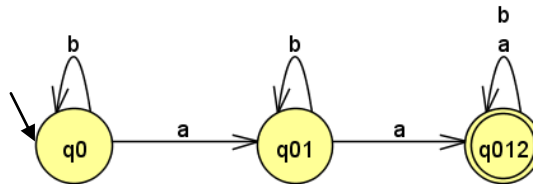
$$\begin{aligned}\delta(q_{012}, b) &= \delta(\{q_0, q_1, q_2\}, b) \\ &= \{\delta(q_0, b)\} \cup \{\delta(q_1, b)\} \cup \{\delta(q_2, b)\} \\ &= \{q_0\} \cup \{q_1\} \cup \{q_2\} = \{q_0, q_1, q_2\} = q_{012}\end{aligned}$$

| $\delta$  | a                             | b                             |
|-----------|-------------------------------|-------------------------------|
| $q_0$     | $\{q_0, q_1\} = q_{01}$       | $q_0$                         |
| $q_1$     | $\{q_1, q_2\} = q_{12}$       | $q_1$                         |
| $q_2$     | $q_2$                         | $q_2$                         |
| $q_{01}$  | $\{q_0, q_1, q_2\} = q_{012}$ | $\{q_0, q_1\} = q_{01}$       |
| $q_{012}$ | $\{q_0, q_1, q_2\} = q_{012}$ | $\{q_0, q_1, q_2\} = q_{012}$ |

(5) Revenim la graf și adăugăm tranzițiile care pleacă din starea  $q_{012}$ .



(6) Observăm că nu ne-a mai apărut nicio stare nouă, deci nu ne-a rămas decât să marcăm stările finale și am obținut AFD-ul cerut. În AFN-ul inițial stare finală era doar  $q_2$ . Orice stare-mulțime din AFD-ul nostru care îl conține pe  $q_2$  va fi stare finală. Există una singură:  $q_{012}$ .



**Obs:** Dacă am fi ales varianta (a), atunci am fi calculat în tabel și linia corespunzătoare stării  $q_{12}$ , însă observăm că nici ea, dar nici stările  $q_1$  și  $q_2$  nu apar în AFD.

### Verificare acceptare cuvânt de către automat finit

Folosim “configurații” (sau „descreri instantanee”) ale automatului, adică perechi formate din starea curentă și cât a mai rămas de citit din cuvântul de intrare, și aplicăm funcția delta cât timp este posibil (ne oprim fie când nu mai există tranziție definită, fie când se termină cuvântul). Dacă la final ajungem în stare finală, cuvântul va fi acceptat.

Pentru AFD:

$(q_0, bba) \vdash (q_0, ba) \vdash (q_0, a) \vdash (q_{01}, \lambda)$

Am terminat de citit cuvântul și am ajuns în starea  $q_{01}$ , care nu este finală, deci rezultă că automatul nu acceptă acest cuvânt.

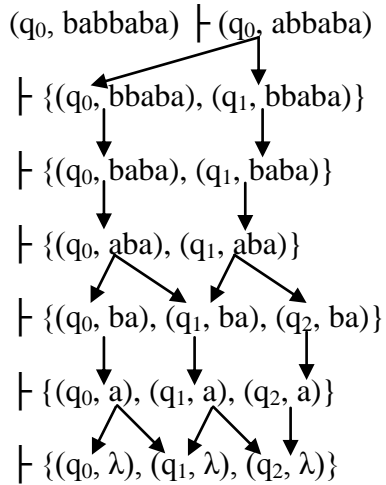
$(q_0, babbaba) \vdash (q_0, abbaba) \vdash (q_{01}, bbaba) \vdash (q_{01}, baba) \vdash (q_{01}, aba) \vdash (q_{012}, ba) \vdash (q_{012}, a) \vdash (q_{012}, \lambda)$

Am terminat de citit cuvântul și am ajuns în starea  $q_{012}$ , care este finală, deci rezultă că automatul acceptă acest cuvânt.

Pentru AFN:

$(q_0, bba) \vdash (q_0, ba) \vdash (q_0, a) \vdash \{(q_0, \lambda), (q_1, \lambda)\}$

Am terminat de citit cuvântul și am avut două drumuri posibile, care ne-au adus în stările  $q_0$  și  $q_1$ . Nici una din ele nu este stare finală, deci cuvântul nu este acceptat.



Am terminat de citit cuvântul și am avut șapte drumuri posibile, care ne-au adus în stările  $q_0$ ,  $q_1$  și  $q_2$ . Există printre ele o stare care este finală,  $q_2$ , deci cuvântul este acceptat.

## ~ Seminar 2 ~

(AFN- $\lambda$ ; Transformarea AFN- $\lambda \rightarrow$  AFD; Automatul minimal;  
Condiții asupra numărului de apariții ale unui caracter într-un cuvânt)

### ***Automat Finit Nedeterminist cu $\lambda$ -tranziții***

AFN- $\lambda = (Q, \Sigma, q_0, \delta, F)$

$Q$  mulțimea de stări

$\Sigma$  alfabetul de intrare

$q_0 \in Q$  starea inițială

$F \subseteq Q$  mulțimea de stări finale

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

Diferența față de AFN-ul simplu este că suntem într-o stare și putem citi fie un caracter din alfabetul  $\Sigma$ , fie cuvântul vid  $\lambda$ . Deci se poate întâmpla să ajungem într-o stare nouă fără să fi citit nicio literă nouă din cuvântul de intrare, ci doar aplicând  $\lambda$ -tranziții.

### ***$\lambda$ -închiderea unei stări***

Mulțimea de stări în care se poate ajunge plecând din starea  $q$  și aplicând zero sau mai multe  $\lambda$ -tranziții se numește „ $\lambda$ -închiderea stării  $q$ ” și se notează cu  $\langle q \rangle$ .

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

$$\langle q \rangle = \bigcup_{k \geq 0} \{r \mid r \in \delta(q, \lambda^k)\}$$

$$\langle q \rangle = \{q\} \cup \{q_i \mid q_i \in \delta(q, \lambda^1)\} \cup \{q_{ij} \mid q_{ij} \in \delta(q, \lambda^2)\} \cup \dots$$

Observăm că mulțimile se pot calcula inductiv după puterea lui  $\lambda$  :

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

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

### **Verificare acceptare cuvânt de către automat AFN- $\lambda$**

Pentru a verifica dacă un cuvânt este sau nu acceptat de un automat AFN- $\lambda$ :

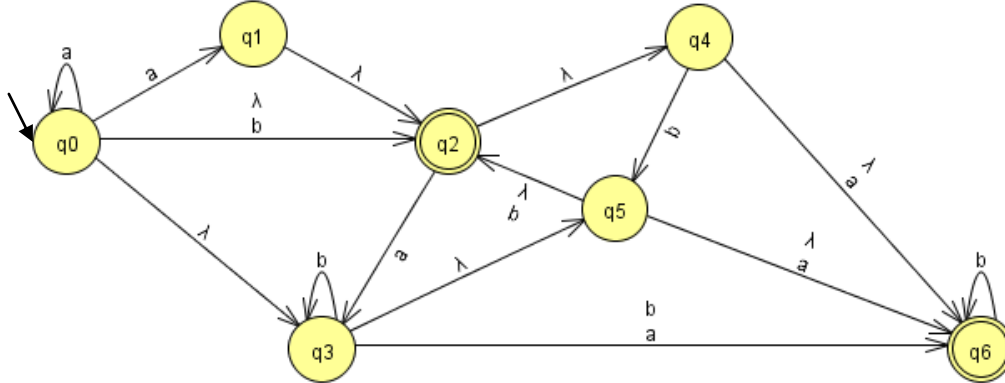
Se procedează analog ca în cazul AFN-ului, doar că înainte de a căuta toate stările posibile de continuare cu tranziții cu caracterul curent, trebuie să facem  $\lambda$ -închiderea mulțimii curente de stări. Iar după ce a fost citit tot cuvântul de intrare, trebuie să facem o ultimă  $\lambda$ -închidere a stărilor curente, pentru a obține mulțimea finală de stări în care poate ajunge automatul pentru cuvântul dat.

**Obs:**  $\lambda$ -închiderea unei mulțimi de stări este egală cu reuniunea  $\lambda$ -închiderilor acelor stări.

$$\langle \{q_{i1}, q_{i2}, \dots, q_{in}\} \rangle = \langle q_{i1} \rangle \cup \langle q_{i2} \rangle \cup \dots \cup \langle q_{in} \rangle$$

### EXEMPLU:

Se dă următorul AFN- $\lambda$ :



Calculăm  $\lambda$ -închiderile tuturor stărilor.

$$\delta(q_0, \lambda^0) = \{q_0\}$$

$$\delta(q_0, \lambda^1) = \{q_2, q_3\}$$

$$\delta(q_0, \lambda^2) = \{\delta(q_2, \lambda)\} \cup \{\delta(q_3, \lambda)\} = \{q_4\} \cup \{q_5\} = \{q_4, q_5\}$$

$$\delta(q_0, \lambda^3) = \{\delta(q_4, \lambda)\} \cup \{\delta(q_5, \lambda)\} = \{q_6\} \cup \{q_2, q_6\} = \{q_2, q_6\} \text{ dar } q_2 \text{ era deja în închidere}$$

$$\delta(q_0, \lambda^4) = \{\delta(q_6, \lambda)\} = \emptyset$$

$$\rightarrow \langle q_0 \rangle = \{q_0, q_2, q_3, q_4, q_5, q_6\}$$

$$\delta(q_1, \lambda^0) = \{q_1\}$$

$$\delta(q_1, \lambda^1) = \{q_2\}$$

$$\delta(q_1, \lambda^2) = \{\delta(q_2, \lambda)\} = \{q_4\}$$

$$\delta(q_1, \lambda^3) = \{\delta(q_4, \lambda)\} = \{q_6\}$$

$$\delta(q_1, \lambda^4) = \{\delta(q_6, \lambda)\} = \emptyset$$

$$\rightarrow \langle q_1 \rangle = \{q_1, q_2, q_4, q_6\}$$

$$\delta(q_2, \lambda^0) = \{q_2\}$$

$$\delta(q_2, \lambda^1) = \{q_4\}$$

$$\delta(q_2, \lambda^2) = \{\delta(q_4, \lambda)\} = \{q_6\}$$

$$\delta(q_2, \lambda^3) = \{\delta(q_6, \lambda)\} = \emptyset$$

$$\rightarrow \langle q_2 \rangle = \{q_2, q_4, q_6\}$$

$$\delta(q_3, \lambda^0) = \{q_3\}$$

$$\delta(q_3, \lambda^1) = \{q_5\}$$

$$\delta(q_3, \lambda^2) = \{\delta(q_5, \lambda)\} = \{q_2, q_6\}$$

$$\delta(q_3, \lambda^3) = \{\delta(q_2, \lambda)\} \cup \{\delta(q_6, \lambda)\} = \{q_4\} \cup \emptyset = \{q_4\}$$

$$\delta(q_3, \lambda^4) = \{\delta(q_4, \lambda)\} = \{q_6\} \text{ dar } q_6 \text{ era deja în închidere}$$

$$\rightarrow \langle q_3 \rangle = \{q_2, q_3, q_4, q_5, q_6\}$$

$$\delta(q_4, \lambda^0) = \{q_4\}$$

$$\delta(q_4, \lambda^1) = \{q_6\}$$

$$\delta(q_4, \lambda^2) = \{\delta(q_6, \lambda)\} = \emptyset$$

$$\rightarrow \langle q_4 \rangle = \{q_4, q_6\}$$

$$\delta(q_5, \lambda^0) = \{q_5\}$$

$$\delta(q_5, \lambda^1) = \{q_2, q_6\}$$

$$\delta(q_5, \lambda^2) = \{\delta(q_2, \lambda)\} \cup \{\delta(q_6, \lambda)\} = \{q_4\} \cup \emptyset = \{q_4\}$$

$$\delta(q_5, \lambda^3) = \{\delta(q_4, \lambda)\} = \{q_6\} \text{ dar } q_6 \text{ era deja în închidere}$$

$$\rightarrow \langle q_5 \rangle = \{q_2, q_4, q_5, q_6\}$$

$$\delta(q_6, \lambda^0) = \{q_6\}$$

$$\delta(q_6, \lambda^1) = \emptyset$$

$$\rightarrow \langle q_6 \rangle = \{q_6\}$$

Verificăm dacă un cuvânt este sau nu acceptat de automatul AFN- $\lambda$  dat.

pas 0: Pornim de la perechea  $(q_0, abbaa)$ .

pas 1a: Facem  $\lambda$ -închiderea mulțimii curente de stări pentru a obține noua mulțime de stări.

$$\langle q_0 \rangle = \{q_0, q_2, q_3, q_4, q_5, q_6\}$$

pas 1b: În fiecare stare din mulțimea curentă încercăm să citim simbolul curent din cuvânt și obținem o nouă mulțime de stări.

$$\delta(q_0, a) = \{q_0, q_1\}$$

$$\delta(q_2, a) = \{q_3\}$$

$$\delta(q_3, a) = \{q_6\}$$

$$\delta(q_4, a) = \{q_6\}$$

$$\delta(q_5, a) = \{q_6\}$$

$$\delta(q_6, a) = \emptyset$$

Deci noua mulțime de stări este  $\{q_0, q_1, q_3, q_6\}$ , iar cuvântul rămas este  $bbaa$ .



pas 2a: Facem  $\lambda$ -închiderea mulțimii curente de stări pentru a obține noua mulțime de stări.

$$\begin{aligned} \langle q_0, q_1, q_3, q_6 \rangle &= \langle q_0 \rangle \cup \langle q_1 \rangle \cup \langle q_3 \rangle \cup \langle q_6 \rangle \\ &= \{q_0, q_2, q_3, q_4, q_5, q_6\} \cup \{q_1, q_2, q_4, q_6\} \cup \{q_2, q_3, q_4, q_5, q_6\} \cup \{q_6\} \\ &= \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\} \end{aligned}$$

pas 2b: În fiecare stare din mulțimea curentă încercăm să citim simbolul curent din cuvânt și obținem o nouă mulțime de stări.

$$\delta(q_0, b) = \{q_2\}$$

$$\delta(q_1, b) = \emptyset$$

$$\delta(q_2, b) = \emptyset$$

$$\delta(q_3, b) = \{q_3, q_6\}$$

$$\delta(q_4, b) = \{q_5\}$$

$$\delta(q_5, b) = \{q_2\}$$

$$\delta(q_6, b) = \{q_6\}$$

Deci noua mulțime de stări este  $\{q_2, q_3, q_5, q_6\}$ , iar cuvântul rămas este *baa*.

pas 3a: Facem  $\lambda$ -închiderea mulțimii curente de stări pentru a obține noua mulțime de stări.

$$\begin{aligned} \langle q_2, q_3, q_5, q_6 \rangle &= \langle q_2 \rangle \cup \langle q_3 \rangle \cup \langle q_5 \rangle \cup \langle q_6 \rangle \\ &= \{q_2, q_4, q_6\} \cup \{q_2, q_3, q_4, q_5, q_6\} \cup \{q_2, q_4, q_5, q_6\} \cup \{q_6\} \\ &= \{q_2, q_3, q_4, q_5, q_6\} \end{aligned}$$

pas 3b: În fiecare stare din mulțimea curentă încercăm să citim simbolul curent din cuvânt și obținem o nouă mulțime de stări.

$$\delta(q_2, b) = \emptyset$$

$$\delta(q_3, b) = \{q_3, q_6\}$$

$$\delta(q_4, b) = \{q_5\}$$

$$\delta(q_5, b) = \{q_2\}$$

$$\delta(q_6, b) = \{q_6\}$$

Deci noua mulțime de stări este  $\{q_2, q_3, q_5, q_6\}$ , iar cuvântul rămas este *aa*.

pas 4a: Facem  $\lambda$ -închiderea mulțimii curente de stări pentru a obține noua mulțime de stări.

$$\begin{aligned} \langle q_2, q_3, q_5, q_6 \rangle &= \langle q_2 \rangle \cup \langle q_3 \rangle \cup \langle q_5 \rangle \cup \langle q_6 \rangle \\ &= \{q_2, q_4, q_6\} \cup \{q_2, q_3, q_4, q_5, q_6\} \cup \{q_2, q_4, q_5, q_6\} \cup \{q_6\} \\ &= \{q_2, q_3, q_4, q_5, q_6\} \end{aligned}$$

pas 4b: În fiecare stare din mulțimea curentă încercăm să citim simbolul curent din cuvânt și obținem o nouă mulțime de stări.

$$\delta(q_2, a) = \{q_3\}$$

$$\delta(q_3, a) = \{q_6\}$$

$$\delta(q_4, a) = \{q_6\}$$

$$\delta(q_5, a) = \{q_6\}$$

$$\delta(q_6, a) = \emptyset$$

Deci noua mulțime de stări este  $\{q_3, q_6\}$ , iar cuvântul rămas este  $a$ .

pas 5a: Facem  $\lambda$ -închiderea mulțimii curente de stări pentru a obține noua mulțime de stări.

$$\langle q_3, q_6 \rangle = \langle q_3 \rangle \cup \langle q_6 \rangle$$

$$= \{q_2, q_3, q_4, q_5, q_6\} \cup \{q_6\}$$

$$= \{q_2, q_3, q_4, q_5, q_6\}$$

pas 5b: În fiecare stare din mulțimea curentă încercăm să citim simbolul curent din cuvânt și obținem o nouă mulțime de stări.

$$\delta(q_2, a) = \{q_3\}$$

$$\delta(q_3, a) = \{q_6\}$$

$$\delta(q_4, a) = \{q_6\}$$

$$\delta(q_5, a) = \{q_6\}$$

$$\delta(q_6, a) = \emptyset$$

Deci noua mulțime de stări este  $\{q_3, q_6\}$ , iar cuvântul a fost citit în întregime.

pas 6: Facem ultima  $\lambda$ -închidere a mulțimii curente de stări pentru a obține mulțimea finală de stări.

$$\langle q_3, q_6 \rangle = \langle q_3 \rangle \cup \langle q_6 \rangle$$

$$= \{q_2, q_3, q_4, q_5, q_6\} \cup \{q_6\}$$

$$= \{q_2, q_3, q_4, q_5, q_6\}$$

pas 7: Verificăm dacă printre stările în care am ajuns este vreo una finală.

$$\{q_2, q_3, q_4, q_5, q_6\} \cap F = \{q_2, q_3, q_4, q_5, q_6\} \cap \{q_2, q_6\} = \{q_2, q_6\} \neq \emptyset$$

Deci cuvântul *abbaa* este acceptat de automatul AFN- $\lambda$  dat.

### **Transformarea AFN- $\lambda \rightarrow$ AFD**

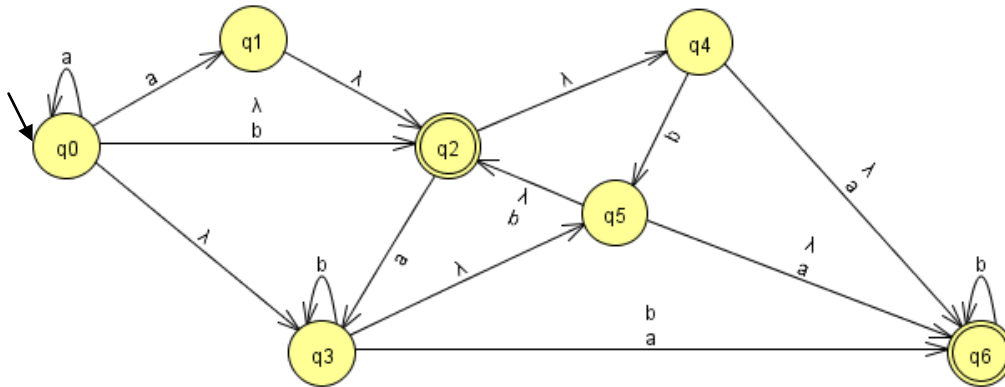
Având dat un AFN- $\lambda$  (reprezentat sub formă de graf), vom calcula funcția delta completând un tabel asemănător cu cel de la algoritmul de transformare AFN  $\rightarrow$  AFD.

- La acest algoritm, capetele de coloană vor fi caracterele din alfabet, dar concatenate la stânga și la dreapta cu  $\lambda^*$ .
- Pentru a calcula primele  $|Q|$  linii ale tabelului
  - facem  $\lambda$ -închiderea stării de la capătul de linie curent
  - apoi în toate stările obținute încercăm să citim caracterul de pe coloana curentă și obținem o nouă mulțime de stări
  - facem  $\lambda$ -închiderea acestei mulțimi și ceea ce obținem scriem în tabel

- Pentru a calcula următoarele linii (corespunzătoare noilor stări-mulțime care apar), procedăm la fel ca la celălalt algoritm, adică reunim ce găsim în tabel pe aceeași coloană și pe liniile potrivite stărilor componente din starea-mulțime.
- Stările finale ale noului AFD obținut, vor fi acele stări-mulțime (pot avea și un singur element, nu neapărat mai multe) care conțin cel puțin una din stările care erau finale pentru AFN-ul inițial. **În plus**, trebuie să verificăm dacă automatul inițial accepta cuvântul vid  $\lambda$  (adică dacă  $\langle q_0 \rangle \cap F_{AFN-\lambda} \neq \emptyset$ ). Dacă da, atunci starea inițială ( $\langle q_0 \rangle$ ) va fi și stare finală.

**EXEMPLU:**

Plecăm de la același automat de mai sus.



Întâi trebuie calculate  $\lambda$ -închiderile tuturor stărilor automatului inițial.

(Am făcut asta mai sus, deci doar rescriu ce am obținut.)

$$\langle q_0 \rangle = \{q_0, q_2, q_3, q_4, q_5, q_6\}$$

$$\langle q_1 \rangle = \{q_1, q_2, q_4, q_6\}$$

$$\langle q_2 \rangle = \{q_2, q_4, q_6\}$$

$$\langle q_3 \rangle = \{q_2, q_3, q_4, q_5, q_6\}$$

$$\langle q_4 \rangle = \{q_4, q_6\}$$

$$\langle q_5 \rangle = \{q_2, q_4, q_5, q_6\}$$

$$\langle q_6 \rangle = \{q_6\}$$

Avem tabelul pentru primele 7 linii obținut după cum urmează:

| $\delta$ | $\lambda^* \cdot a \cdot \lambda^*$ | $\lambda^* \cdot b \cdot \lambda^*$ |
|----------|-------------------------------------|-------------------------------------|
| q0       | {q0, q1, q2, q3, q4, q5, q6}        | {q2, q3, q4, q5, q6}                |
| q1       | {q2, q3, q4, q5, q6}                | {q2, q4, q5, q6}                    |
| q2       | {q2, q3, q4, q5, q6}                | {q2, q4, q5, q6}                    |
| q3       | {q2, q3, q4, q5, q6}                | {q2, q3, q4, q5, q6}                |
| q4       | {q6}                                | {q2, q4, q5, q6}                    |
| q5       | {q2, q3, q4, q5, q6}                | {q2, q4, q5, q6}                    |
| q6       | $\emptyset$                         | {q6}                                |

$$\begin{aligned}
<\delta(<q_0>,a)> &= <\delta(\{q_0,q_2,q_3,q_4,q_5,q_6\},a)> \\
&= <\{\delta(q_0,a)\} \cup \{\delta(q_2,a)\} \cup \{\delta(q_3,a)\} \cup \{\delta(q_4,a)\} \cup \{\delta(q_5,a)\} \cup \{\delta(q_6,a)\}> \\
&= <\{q_0,q_1\} \cup \{q_3\} \cup \{q_6\} \cup \{q_6\} \cup \{q_6\} \cup \emptyset> = <\{q_0,q_1,q_3,q_6\}> \\
&= <q_0> \cup <q_1> \cup <q_3> \cup <q_6> \\
&= \{q_0,q_2,q_3,q_4,q_5,q_6\} \cup \{q_1,q_2,q_4,q_6\} \cup \{q_2,q_3,q_4,q_5,q_6\} \cup \{q_6\} = \{q_0,q_1,q_2,q_3,q_4,q_5,q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_0>,b)> &= <\delta(\{q_0,q_2,q_3,q_4,q_5,q_6\},b)> \\
&= <\{\delta(q_0,b)\} \cup \{\delta(q_2,b)\} \cup \{\delta(q_3,b)\} \cup \{\delta(q_4,b)\} \cup \{\delta(q_5,b)\} \cup \{\delta(q_6,b)\}> \\
&= <\{q_2\} \cup \emptyset \cup \{q_3,q_6\} \cup \{q_5\} \cup \{q_2\} \cup \{q_6\}> = <\{q_2,q_3,q_5,q_6\}> \\
&= <q_2> \cup <q_3> \cup <q_5> \cup <q_6> \\
&= \{q_2,q_4,q_6\} \cup \{q_2,q_3,q_4,q_5,q_6\} \cup \{q_2,q_4,q_5,q_6\} \cup \{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_1>,a)> &= <\delta(\{q_1,q_2,q_4,q_6\},a)> \\
&= <\{\delta(q_1,a)\} \cup \{\delta(q_2,a)\} \cup \{\delta(q_4,a)\} \cup \{\delta(q_6,a)\}> \\
&= <\emptyset \cup \{q_3\} \cup \{q_6\} \cup \emptyset> = <\{q_3,q_6\}> \\
&= <q_3> \cup <q_6> \\
&= \{q_2,q_3,q_4,q_5,q_6\} \cup \{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_1>,b)> &= <\delta(\{q_1,q_2,q_4,q_6\},b)> \\
&= <\{\delta(q_1,b)\} \cup \{\delta(q_2,b)\} \cup \{\delta(q_4,b)\} \cup \{\delta(q_6,b)\}> \\
&= <\emptyset \cup \emptyset \cup \{q_5\} \cup \{q_6\}> = <\{q_5,q_6\}> \\
&= <q_5> \cup <q_6> \\
&= \{q_2,q_4,q_5,q_6\} \cup \{q_6\} = \{q_2,q_4,q_5,q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_2>,a)> &= <\delta(\{q_2,q_4,q_6\},a)> \\
&= <\{\delta(q_2,a)\} \cup \{\delta(q_4,a)\} \cup \{\delta(q_6,a)\}> \\
&= <\{q_3\} \cup \{q_6\} \cup \emptyset> = <\{q_3,q_6\}> \\
&= <q_3> \cup <q_6> \\
&= \{q_2,q_3,q_4,q_5,q_6\} \cup \{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_2>,b)> &= <\delta(\{q_2,q_4,q_6\},b)> \\
&= <\{\delta(q_2,b)\} \cup \{\delta(q_4,b)\} \cup \{\delta(q_6,b)\}> \\
&= <\emptyset \cup \{q_5\} \cup \{q_6\}> = <\{q_5,q_6\}> \\
&= <q_5> \cup <q_6> \\
&= \{q_2,q_4,q_5,q_6\} \cup \{q_6\} = \{q_2,q_4,q_5,q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_3>,a)> &= <\delta(\{q_2,q_3,q_4,q_5,q_6\},a)> \\
&= <\{\delta(q_2,a)\} \cup \{\delta(q_3,a)\} \cup \{\delta(q_4,a)\} \cup \{\delta(q_5,a)\} \cup \{\delta(q_6,a)\}> \\
&= <\{q_3\} \cup \{q_6\} \cup \{q_6\} \cup \{q_6\} \cup \emptyset> = <\{q_3,q_6\}> \\
&= <q_3> \cup <q_6> \\
&= \{q_2,q_3,q_4,q_5,q_6\} \cup \{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_3>,b)> &= <\delta(\{q_2,q_3,q_4,q_5,q_6\},b)> \\
&= <\{\delta(q_2,b)\} \cup \{\delta(q_3,b)\} \cup \{\delta(q_4,b)\} \cup \{\delta(q_5,b)\} \cup \{\delta(q_6,b)\}> \\
&= <\emptyset \cup \{q_3,q_6\} \cup \{q_5\} \cup \{q_2\} \cup \{q_6\}> = <\{q_2,q_3,q_5,q_6\}> \\
&= <q_2> \cup <q_3> \cup <q_5> \cup <q_6> \\
&= \{q_2,q_4,q_6\} \cup \{q_2,q_3,q_4,q_5,q_6\} \cup \{q_2,q_4,q_5,q_6\} \cup \{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_4>,a)> &= <\delta(\{q_4,q_6\},a)> = <\{\delta(q_4,a)\} \cup \{\delta(q_6,a)\}> \\
&= <\{q_6\} \cup \emptyset> = <\{q_6\}> = <q_6> = \{q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_4>,b)> &= <\delta(\{q_4,q_6\},b)> = <\{\delta(q_4,b)\} \cup \{\delta(q_6,b)\}> \\
&= <\{q_5\} \cup \{q_6\}> = <\{q_5,q_6\}> = <q_5> \cup <q_6> = \{q_2,q_4,q_5,q_6\} \cup \{q_6\} = \{q_2,q_4,q_5,q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_5>,a)> &= <\delta(\{q_2,q_4,q_5,q_6\},a)> \\
&= <\{\delta(q_2,a)\} \cup \{\delta(q_4,a)\} \cup \{\delta(q_5,a)\} \cup \{\delta(q_6,a)\}> \\
&= <\{q_3\} \cup \{q_6\} \cup \{q_6\} \cup \emptyset> = <\{q_3,q_6\}> \\
&= <q_3> \cup <q_6> \\
&= \{q_2,q_3,q_4,q_5,q_6\} \cup \{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}
\end{aligned}$$

$$\begin{aligned}
<\delta(<q_5>,b)> &= <\delta(\{q_2,q_4,q_5,q_6\},b)> \\
&= <\{\delta(q_2,b)\} \cup \{\delta(q_4,b)\} \cup \{\delta(q_5,b)\} \cup \{\delta(q_6,b)\}> \\
&= <\emptyset \cup \{q_5\} \cup \{q_2\} \cup \{q_6\}> = <\{q_2,q_5,q_6\}> \\
&= <q_2> \cup <q_5> \cup <q_6> \\
&= \{q_2,q_4,q_6\} \cup \{q_2,q_4,q_5,q_6\} \cup \{q_6\} = \{q_2,q_4,q_5,q_6\}
\end{aligned}$$

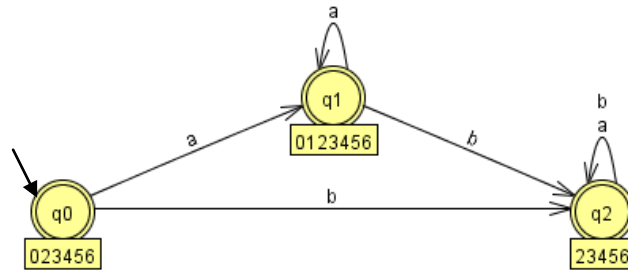
$$<\delta(<q_6>,a)> = <\delta(\{q_6\},a)> = <\{\delta(q_6,a)\}> = <\emptyset> = \emptyset$$

$$<\delta(<q_6>,b)> = <\delta(\{q_6\},b)> = <\{\delta(q_6,b)\}> = <\{q_6\}> = <q_6> = \{q_6\}$$

Apoi calculăm liniile următoare, corespunzătoare noilor stări-mulțime, pe măsură ce ele apar în automat. Începem cu starea inițială,  $\langle q_0 \rangle = q_{023456}$ .

| $\delta$      | $\lambda^* \cdot a \cdot \lambda^*$     | $\lambda^* \cdot b \cdot \lambda^*$ |
|---------------|-----------------------------------------|-------------------------------------|
| $q_0$         | $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$ | $\{q_2, q_3, q_4, q_5, q_6\}$       |
| $q_1$         | $\{q_2, q_3, q_4, q_5, q_6\}$           | $\{q_2, q_4, q_5, q_6\}$            |
| $q_2$         | $\{q_2, q_3, q_4, q_5, q_6\}$           | $\{q_2, q_4, q_5, q_6\}$            |
| $q_3$         | $\{q_2, q_3, q_4, q_5, q_6\}$           | $\{q_2, q_3, q_4, q_5, q_6\}$       |
| $q_4$         | $\{q_6\}$                               | $\{q_2, q_4, q_5, q_6\}$            |
| $q_5$         | $\{q_2, q_3, q_4, q_5, q_6\}$           | $\{q_2, q_4, q_5, q_6\}$            |
| $q_6$         | $\emptyset$                             | $\{q_6\}$                           |
| $q_{023456}$  | $q_{0123456}$                           | $q_{23456}$                         |
| $q_{0123456}$ | $q_{0123456}$                           | $q_{23456}$                         |
| $q_{23456}$   | $q_{23456}$                             | $q_{23456}$                         |

Vom obține în final automatul AFD următor.



Automatul inițial avea stările finale  $F_{AFN-\lambda} = \{q_2, q_6\}$ , deci în noul automat vom avea stările finale  $F_{AFD} = \{q_{023456}, q_{0123456}, q_{23456}\}$ .

**Obs:** Limbajul acceptat de acest automat este  $L = \{a,b\}^*$ , adică orice cuvinte formate cu literele  $a$  și  $b$  în orice ordine, cuvinte de orice lungime, inclusiv zero.

**Obs:** Cred că la seminar am uitat să punem și tranziția de la  $q_5$  la  $q_2$  cu  $b$ , așa că pot exista mici diferențe la calculele intermediare, dar rezultatul final este același.

### Automatul minimal

Se pornește de la un AFD complet definit (adică pentru orice stare din mulțimea  $Q$  și pentru orice caracter din alfabetul  $\Sigma$ , funcția  $\delta$  este definită).

Ideea algoritmului este de a găsi acele stări care au comportament echivalent, pentru a le grupa și a obține o unică stare nouă în locul acestora.

- Două stări sunt „**echivalente**” dacă pentru orice cuvânt am alege, plecând din cele două stări, fie ajungem în două stări finale, fie ajungem în două stări nefinale.  

$$\forall p, q \in Q, p \equiv q \Leftrightarrow [\forall w \in \Sigma^*, \delta(p, w) \in F \leftrightarrow \delta(q, w) \in F]$$
- Două stări sunt „**separabile**” dacă există un cuvânt pentru care plecând din cele două stări ajungem într-o stare finală și într-una nefinală.  

$$\forall p, q \in Q, p \not\equiv_w q \Leftrightarrow [\exists w \in \Sigma^*, \delta(p, w) \in F \leftrightarrow \delta(q, w) \notin F]$$

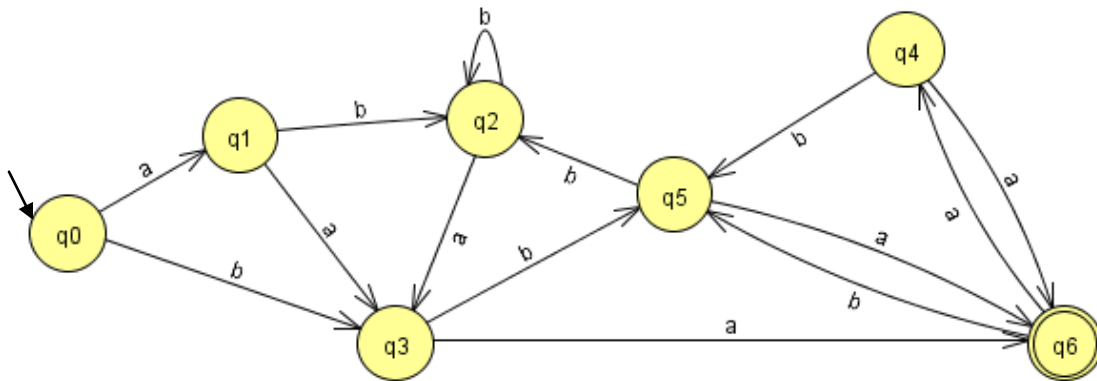
Vom construi un tabel, pe linii și pe coloane având stările automatului inițial. Vom completa tabelul pentru fiecare pereche de stări  $(q_i, q_j)$ , având  $i > j$ , cu un cuvânt prin care cele două stări sunt separabile. Dacă nu vom găsi un astfel de cuvânt atunci acele stări sunt echivalente.

Vom completa tabelul căutând cuvintele recursiv, în ordinea crescătoare a lungimii lor. Cuvintele de lungime  $k$  le vom obține cu ajutorul celor de lungime  $k-1$  calculate anterior.

**Obs:** Două stări sunt separabile prin cuvântul vid  $\lambda$  (de lungime zero), dacă una din ele este finală și cealaltă este nefinală în automatul inițial.

### EXEMPLU:

Fie automatul AFD complet definit:



Pas 0: Căutăm perechile de stări separabile prin cuvântul  $\lambda$  de lungime zero. Avem o singură stare finală în acest exemplu, deci ea va fi separabilă prin  $\lambda$  de toate celelalte stări, care sunt nefinale. Completăm în tabel, pentru toate perechile  $(q_6, q_i)$ ,  $0 \leq i \leq 5$ .

|       | $q_0$     | $q_1$     | $q_2$     | $q_3$     | $q_4$     | $q_5$     | $q_6$ |
|-------|-----------|-----------|-----------|-----------|-----------|-----------|-------|
| $q_0$ |           |           |           |           |           |           |       |
| $q_1$ |           |           |           |           |           |           |       |
| $q_2$ |           |           |           |           |           |           |       |
| $q_3$ |           |           |           |           |           |           |       |
| $q_4$ |           |           |           |           |           |           |       |
| $q_5$ |           |           |           |           |           |           |       |
| $q_6$ | $\lambda$ | $\lambda$ | $\lambda$ | $\lambda$ | $\lambda$ | $\lambda$ |       |

Pas 1: Căutăm cuvinte de lungime 1.

Avem  $\delta(q_0, a) = q_1 \notin F$ ;  $\delta(q_1, a) = q_3 \notin F$ ;  $\delta(q_2, a) = q_3 \notin F$ ;  
 $\delta(q_3, a) = q_6 \in F$ ;  $\delta(q_4, a) = q_6 \in F$ ;  $\delta(q_5, a) = q_6 \in F$

Rezultă că orice stare din mulțimea  $\{q_0, q_1, q_2\}$  va fi separabilă de orice stare din mulțimea  $\{q_3, q_4, q_5\}$  prin cuvântul "a".

(Observăm că toate tranzițiile cu "b" merg către stări nefinale, deci nu va exista nicio pereche de stări care să fie separabile prin cuvântul "b".)

**Obs:** E posibil să fie separabile și prin altceva, dar este suficient să găsim un singur cuvânt.

|                | q <sub>0</sub> | q <sub>1</sub> | q <sub>2</sub> | q <sub>3</sub> | q <sub>4</sub> | q <sub>5</sub> | q <sub>6</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| q <sub>0</sub> |                |                |                |                |                |                |                |
| q <sub>1</sub> |                |                |                |                |                |                |                |
| q <sub>2</sub> |                |                |                |                |                |                |                |
| q <sub>3</sub> | a              | a              | a              |                |                |                |                |
| q <sub>4</sub> | a              | a              | a              |                |                |                |                |
| q <sub>5</sub> | a              | a              | a              |                |                |                |                |
| q <sub>6</sub> | λ              | λ              | λ              | λ              | λ              | λ              |                |

Pas 2: Căutăm cuvinte de lungime 2.

- Perechea (q<sub>1</sub>, q<sub>0</sub>) cu caracterul "a" ajunge în perechea (q<sub>3</sub>, q<sub>1</sub>). Dar q<sub>3</sub> și q<sub>1</sub> erau separabile prin "a". Rezultă că stările q<sub>1</sub> și q<sub>0</sub> vor fi separabile prin cuvântul "aa" (*obținut prin concatenarea caracterului încercat și a fostului cuvânt calculat în tabel*).
- Perechea (q<sub>2</sub>, q<sub>0</sub>) cu caracterul "a" ajunge în perechea (q<sub>3</sub>, q<sub>1</sub>). Dar q<sub>3</sub> și q<sub>1</sub> erau separabile prin "a". Rezultă că stările q<sub>1</sub> și q<sub>0</sub> vor fi separabile prin cuvântul "aa".
- Perechea (q<sub>2</sub>, q<sub>1</sub>) cu caracterul "a" ajunge în perechea (q<sub>3</sub>, q<sub>3</sub>), iar cu caracterul "b" ajunge în perechea (q<sub>2</sub>, q<sub>2</sub>). *Observăm* că dacă după citirea primului caracter (indiferent dacă este "a" sau "b"), ambele drumuri pornesc din același loc, nu au cum să se mai ramifice în continuare. Rezultă că aceste două stări, q<sub>2</sub> și q<sub>1</sub>, sunt echivalente.
- Perechea (q<sub>4</sub>, q<sub>3</sub>) cu caracterul "a" ajunge în perechea (q<sub>6</sub>, q<sub>6</sub>), iar cu caracterul "b" ajunge în perechea (q<sub>5</sub>, q<sub>5</sub>). Cu aceeași observație ca mai devreme, rezultă că stările q<sub>4</sub> și q<sub>3</sub> sunt echivalente.
- Perechea (q<sub>5</sub>, q<sub>3</sub>) cu caracterul "a" ajunge în perechea (q<sub>6</sub>, q<sub>6</sub>), dar cu caracterul "b" ajunge în perechea (q<sub>2</sub>, q<sub>5</sub>). Perechea (q<sub>2</sub>, q<sub>5</sub>) era separabilă prin "a". Rezultă, prin concatenare, că stările q<sub>5</sub> și q<sub>3</sub> sunt separabile prin "ba".
- Perechea (q<sub>5</sub>, q<sub>4</sub>) cu caracterul "a" ajunge în perechea (q<sub>6</sub>, q<sub>6</sub>), dar cu caracterul "b" ajunge în perechea (q<sub>2</sub>, q<sub>5</sub>). Perechea (q<sub>2</sub>, q<sub>5</sub>) era separabilă prin "a". Rezultă, prin concatenare, că stările q<sub>5</sub> și q<sub>4</sub> sunt separabile prin "ba".

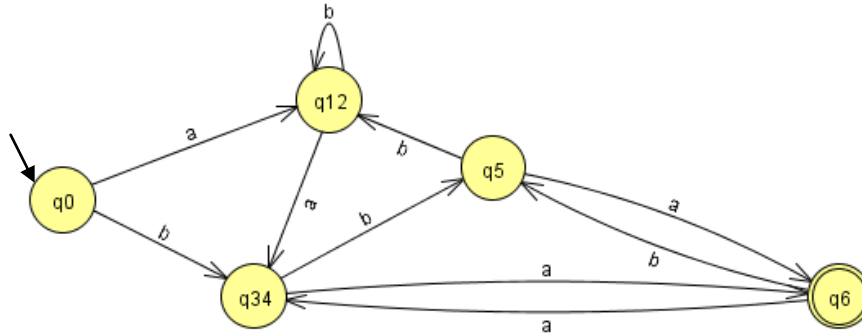
|                | q <sub>0</sub> | q <sub>1</sub> | q <sub>2</sub> | q <sub>3</sub> | q <sub>4</sub> | q <sub>5</sub> | q <sub>6</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| q <sub>0</sub> |                |                |                |                |                |                |                |
| q <sub>1</sub> | aa             |                |                |                |                |                |                |
| q <sub>2</sub> | aa             | ∅              |                |                |                |                |                |
| q <sub>3</sub> | a              | a              | a              |                |                |                |                |
| q <sub>4</sub> | a              | a              | a              | ∅              |                |                |                |
| q <sub>5</sub> | a              | a              | a              | ba             | ba             |                |                |
| q <sub>6</sub> | λ              | λ              | λ              | λ              | λ              | λ              |                |

Am terminat de completat tabelul și am obținut stări echivalente:  $q_1 \equiv q_2$  și  $q_3 \equiv q_4$ .



Automatul AFD minimal obținut va avea  $Q = \{q_0, q_{12}, q_{34}, q_5, q_6\}$  și  $F = \{q_6\}$ .

Tranzițiile le desenăm conform automatului inițial, dar ținând cont de această grupare a stărilor.



### Condiții asupra numărului de apariții ale unui caracter într-un cuvânt

Notăm cu  $|w|_a$  numărul de apariții ale caracterului „a” în cuvântul „w”.

**Obs:** În general, când avem limbaje definite cu astfel de condiții asupra numărului de apariții ale unui caracter, sugestia pentru a putea construi mai ușor automatul este să folosim niște indici la stări pentru o codificare cât mai intuitivă a cuvintelor din limbaj.

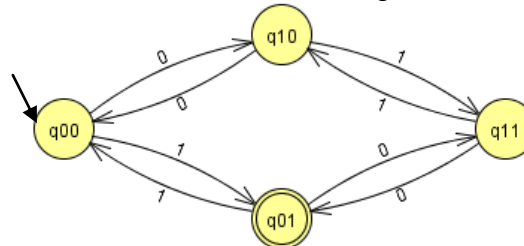
**Atenție!** Nu confundați exemplele următoare cu cele din primul seminar, în sensul de a încerca să descrieți limbajele acestea cu expresii asemănătoare cu acelea cu puteri. Acolo existau în plus restricții asupra poziției literelor în cuvânt (de exemplu toate a-urile la început și apoi toate b-urile). Aici nu contează ordinea literelor, ci doar numărul lor.

### EXEMPLE:

$$L_1 = \{w \in \{0,1\}^* \text{ a.î. } |w|_0 = \text{par}, |w|_1 = \text{impar}\}$$

Ideea de rezolvare este ca fiecare stare să aibă doi indici, egali cu restul împărțirii la 2 a numărului de apariții ale caracterului “0”, respectiv ale caracterului “1”.

Starea finală va fi  $q_{01}$  pentru că ea codifică numărul par de 0-uri și impar de 1-uri.



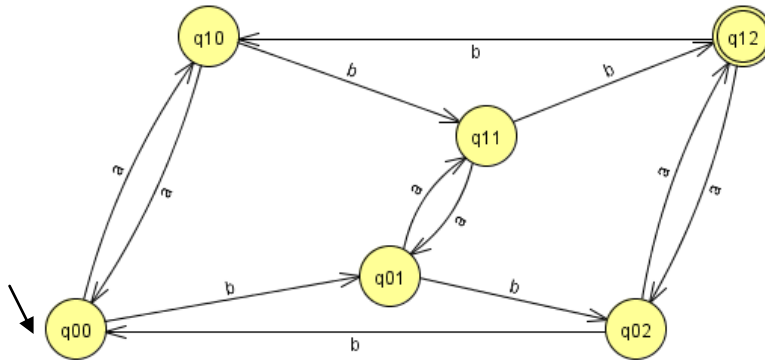
**Obs:** Pentru că nu există nicio altă restricție asupra formei cuvântului, automatul va fi *complet definit*, adică în orice stare am fi trebuie să avem tranziții cu toate caracterele din alfabet.

$$L_2 = \{w \in \{a,b\}^* a \hat{.} \mid |w|_a = 2n+1, |w|_b = 3m+2, n \geq 0, m \geq 0\}$$

**Obs:**

- Fiecare stare va avea doi indici pentru că alfabetul are două elemente și ambele au condiții atașate.
- Primul indice, corespunzător caracterului “a”, va lua valori în mulțimea resturilor împărțirii la 2, deoarece coeficientul necunoscutei “n” este 2. Al doilea indice, corespunzător caracterului “b”, va lua valori în mulțimea resturilor împărțirii la 3, deoarece coeficientul necunoscutei “m” este 3.
- Numărul de stări va fi egal cu produsul numerelor ce reprezintă cardinalele mulțimilor în care iau valori indicii, în cazul acesta  $2 \cdot 3 = 6$ .

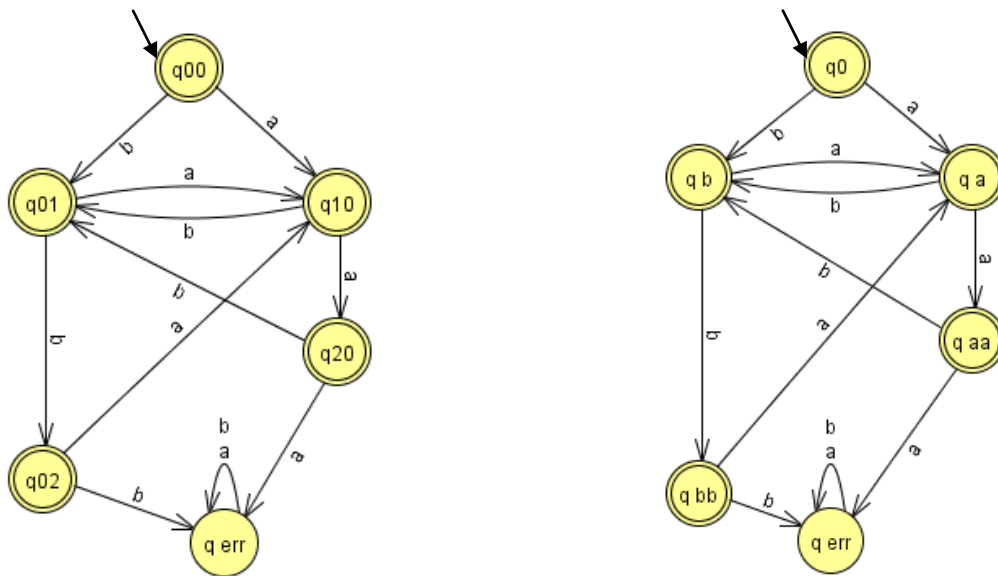
Starea finală va fi  $q_{12}$ , deoarece termenii liberi sunt “+ 1” și “+ 2”.



$$L_3 = \{w \in \{a,b\}^* a \hat{.} \mid aaa \text{ și } bbb \text{ nu apar în } w\}$$

**Obs:** Acest limbaj ar mai putea fi scris și  $L_3 = \{\lambda, b, bb\} \cdot (\{a, aa\} \cdot \{b, bb\})^* \cdot \{\lambda, a, aa\}$ .

Sunteți de acord că e adevărat? Știți să explicați cum am ajuns la acea expresie ?



### ~ Seminar 3 ~

(Expresii regulate; Transformarea expresie\_regulată  $\rightarrow$  automat\_finit

Transformarea automat\_finit  $\rightarrow$  expresie\_regulată ; Gramatici ; Gramatici regulate)

#### Expresii regulate

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

i)  $\emptyset, \lambda \in REX$  ;  $a \in REX$  pentru  $\forall a \in \Sigma$

ii)  $e_1, e_2 \in REX \Rightarrow (e_1 + e_2) \in REX$

iii)  $e_1, e_2 \in REX \Rightarrow (e_1 \cdot e_2) \in REX$

iv)  $e \in REX \Rightarrow (e^*) \in REX$

#### Precedența operațiilor $*, \cdot, +$

**Obs:** Atenție la ordinea în care se evaluează operațiile: întâi stelarea, apoi concatenarea și apoi reuniunea. (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.)

#### Transformarea expresie\_regulată $\rightarrow$ automat\_finit

| Expresie regulată | Limbaj                                        | Automat finit                                                                                                                                                                             |
|-------------------|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\emptyset$       | $\emptyset$                                   | $Q = \{q_0\}, F = \emptyset, \delta = \emptyset$                                                                                                                                          |
| $\lambda$         | $\{\lambda\}$                                 | $Q = \{q_0\}, F = \{q_0\}, \delta = \emptyset$                                                                                                                                            |
| $a$               | $\{a\}, a \in \Sigma$                         | $Q = \{q_0, q_1\}, F = \{q_1\}, \delta(q_0, a) = q_1$                                                                                                                                     |
| $e_1 + e_2$       | $L(e_1 + e_2)$<br>$= L(e_1) \cup L(e_2)$      | $Q = Q_1 \cup Q_2 \cup \{q_0\}; F = F_1 \cup F_2; V = V_1 \cup V_2;$<br>$\delta = \delta_1 \cup \delta_2 \cup \{\delta(q_0, \lambda) = \{q_{01}, q_{02}\}\}$                              |
| $e_1 \cdot e_2$   | $L(e_1 \cdot e_2)$<br>$= L(e_1) \cdot L(e_2)$ | $Q = Q_1 \cup Q_2, q_0 = q_{01}; F = F_2; V = V_1 \cup V_2;$<br>$\delta = \delta_1 \cup \delta_2 \cup \{\delta(q_{f1}, \lambda) = q_{02}, \text{pentru } \forall q_{f1} \in F_1\}$        |
| $(e_1)^*$         | $L(e_1^*) = (L(e_1))^*$                       | $Q = Q_1 \cup \{q_0\}; F = \{q_0\}; V = V_1;$<br>$\delta = \delta_1 \cup \{\delta(q_0, \lambda) = q_{01}\} \cup \{\delta(q_{f1}, \lambda) = q_0, \text{pentru } \forall q_{f1} \in F_1\}$ |

Descrierea în cuvinte a ultimelor trei linii ale tabelului o găsiți în acest document la pagina 9 ([compunerea automatelor finite](#)).

**Obs:** La construirea automatelor compuse se ține cont de precedența operațiilor (întâi stelarea, apoi concatenarea, apoi reuniunea) și de parantezele din expresie.

### ***Transformarea automat\_finit → expresie\_regulată***

Se numește *EFA* (*automat finit extins*),  $M = (Q, \Sigma, et, q_0, F)$ , unde, la fel ca la celelalte automate finite,  $Q$  este mulțimea stărilor,  $\Sigma$  este alfabetul,  $q_0$  este starea inițială,  $F$  este mulțimea stărilor finale. Aici avem *funcția de etichetare*  $et : Q \times Q \rightarrow REX(\Sigma)$ .

Notăm  $et(p, q)$  prin  $e_{pq}$ .

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.

- Dacă există tranziții care ajung în starea inițială, atunci se adaugă la automat o nouă stare care va fi inițială și va avea o  $\lambda$ -tranzitie către fosta stare inițială.
- Dacă există mai multe stări finale, atunci se adaugă la automat o nouă stare care va fi singura finală și va avea  $\lambda$ -tranziii din toate fostele stări finale către ea.
- Se elimină pe rând, una câte una, toate stările în afară de cea inițială și cea finală. Presupunem că vrem să eliminăm starea  $q_k$  și că există etichetele  $et(q_i, q_k)$ ,  $et(q_k, q_j)$  și eventual bucla  $et(q_k, q_k)$ . Atunci obținem noua etichetă între stările  $q_i$  și  $q_j$  reunind (fosta etichetă directă de la  $q_i$  la  $q_j$ ) cu (“eticheta de la  $q_i$  la  $q_k$ ” concatenată cu “stelarea etichetei buclei de la  $q_k$  la  $q_k$ ” concatenată cu “eticheta de la  $q_k$  la  $q_j$ ”).
- Atunci când rămân doar două stări, se elimină bucla stării finale, dacă există. Concatenând (expresia obținută între starea inițială și cea finală) cu (stelarea expresiei de pe bucla stării finale) obținem răspunsul final.

În practică, vom calcula etichetele folosind un tabel cu numărul de coloane complete egal cu numărul de stări ale automatului, și cu numărul de linii egal cu pătratul numărului de stări ale automatului.

**Obs:** Atenție, stările automatului vor fi numerotate începând de la  $q_1$  (care va fi starea inițială).

Notăm cu  $R_{ij}^k$  expresia regulată ce se obține plecând din starea  $q_i$ , ajungând în starea  $q_j$  și având ca stări intermediare primele  $k$  stări (adică  $q_1, q_2, \dots, q_k$ ).

Tabelul va conține la capete de linii toate perechile posibile  $(q_i, q_j)$  cu  $i$  și  $j$  între 1 și  $|Q|$ , iar la capete de coloană toți  $k$ , cu  $k$  de la 0 la  $|Q| - 1$ .

Tabelul va fi calculat coloană cu coloană, pentru că valorile coloanei curente se calculează în funcție de valori aflate pe coloana precedentă.

Prima coloană se calculează astfel:

$$R_{ij}^0 = \begin{cases} \{a \in V \mid \delta(q_i, a) = q_j\}, & \text{dacă } i \neq j \\ \{\lambda\} \cup \{a \in V \mid \delta(q_i, a) = q_j\}, & \text{dacă } i = j \end{cases}$$

Următoarele coloane se calculează astfel:

(coloana curentă  $k+1$  în funcție de coloana precedentă  $k$ )

$$R_{ij}^{k+1} = R_{ij}^k \cup R_{ik+1}^k \cdot (R_{k+1,k+1}^k)^* \cdot (R_{k+1,j}^k)$$

Răspunsul final, expresia regulată căutată, va fi obținută calculând  $\bigcup_{f \in F} R_{1f}^{|\mathcal{Q}|}$ , adică

reuniunea etichetelor de la starea inițială la toate stările finale, având ca stări intermediare toate stările automatului.

### Câteva formule utile

(A)  $e \cdot \emptyset = \emptyset$  ;  $\emptyset \cdot e = \emptyset$  ( $\emptyset$  este pentru concatenare cum este 0 pentru înmulțire)

(B)  $e \cdot \lambda = e$  ;  $\lambda \cdot e = e$  ( $\lambda$  este pentru concatenare cum este 1 pentru înmulțire)

(C)  $e^* \cdot e = e^+$  ;  $e \cdot e^* = e^+$

(D)  $\{e_1, e_2\}^* = (e_1 + e_2)^* = (e_1^* \cdot e_2^*)^*$  (formulă valabilă pentru oricâte expresii, nu doar 2)

(E)  $e_1 \cdot (e_2 + e_3) = (e_1 \cdot e_2) + (e_1 \cdot e_3)$  ;  $(e_1 + e_2) \cdot e_3 = (e_1 \cdot e_3) + (e_2 \cdot e_3)$

### EXEMPLU:

Se dă automatul finit următor și se dorește obținerea expresiei regulate care îi corespunde, folosind tabelul descris anterior.



Avem două stări, deci tabelul va avea  $2^2 = 4$  linii și 2 coloane complete (plus eventual una incompletă pentru răspunsul final).

| $R_{ij}^{k+1}$    | $k+1 = 0$     | $k+1 = 1$ | $k+1 = 2$ |
|-------------------|---------------|-----------|-----------|
| $r_{ij} = r_{11}$ | $1 + \lambda$ |           |           |
| $r_{ij} = r_{12}$ | $0$           |           |           |
| $r_{ij} = r_{21}$ | $\emptyset$   |           |           |
| $r_{ij} = r_{22}$ | $\lambda$     |           |           |

(a) Calculăm prima coloană, pentru  $k+1 = 0$ .

- Dacă indicii  $i$  și  $j$  sunt diferiți, atunci căutăm tranzițiile de la  $q_i$  la  $q_j$  și scriem în tabel (pe linia corespunzătoare lui  $r_{ij}$ ) expresia obținută prin reuniunea acelor caractere de pe tranziții.

Deci  $R_{12}^0 = 0$ ,  $R_{21}^0 = \emptyset$ .

- Dacă indicii  $i$  și  $j$  sunt egali, căutăm bucele stării  $q_i = q_j$  și scriem în tabel expresia obținută prin reuniunea acelor caractere de pe bucle și a caracterului  $\lambda$ .

Deci  $R_{11}^0 = 1 + \lambda$ ,  $R_{22}^0 = \lambda$ .

(b) Calculăm a doua coloană, pentru  $k+1 = 1$ .

$$R_{11}^1 = R_{11}^0 \cup R_{11}^0 \cdot (R_{11}^0)^* \cdot R_{11}^0 = (1 + \lambda) \cup (1 + \lambda) \cdot (1 + \lambda)^* \cdot (1 + \lambda)$$

Dacă notăm  $(1 + \lambda)$  cu  $b$ ,

$$\text{avem } b \cup b \cdot b^* \cdot b = b^1 \cup \{b^n \mid n \geq 2\} = \{b^n \mid n \geq 1\} = b^+ = (1 + \lambda)^+$$

Putem aplica formula (C) și avem  $(1 + \lambda)^+ = (1 + \lambda)^* \cdot (1 + \lambda)$

Apoi putem aplica formula (D) pentru prima parte și obținem  $(1^* \cdot \lambda^*)^* \cdot (1 + \lambda)$

Dar  $\lambda^* = \lambda$  și conform formulei (B) prima paranteză este egală cu  $1^*$ , deci vom avea

$(1^*)^* \cdot (1 + \lambda) = 1^* \cdot (1 + \lambda)$ . Aplicăm formula (E) (și apoi (C) și (B)) și obținem

$(1^* \cdot 1) + (1^* \cdot \lambda) = 1^+ + 1^* = \{1^n \mid n \geq 1\} \cup \{1^n \mid n \geq 0\} = \{1^n \mid n \geq 0\} = 1^*$  pe care îl trecem în tabel.

$$R_{12}^1 = R_{12}^0 \cup R_{11}^0 \cdot (R_{11}^0)^* \cdot R_{12}^0 = 0 \cup (1 + \lambda) \cdot (1 + \lambda)^* \cdot 0$$

Aplicăm formula (C) și obținem  $0 \cup (1 + \lambda)^+ \cdot 0$ , care este echivalent după cum am aflat

$$\text{mai sus cu } 0 \cup 1^* \cdot 0 = \lambda \cdot 0 \cup 1^* \cdot 0 = \lambda \cdot 0 + 1^* \cdot 0 = (\lambda + 1^*) \cdot 0 = 1^* \cdot 0$$

$$R_{21}^1 = R_{21}^0 \cup R_{21}^0 \cdot (R_{11}^0)^* \cdot R_{11}^0 = \emptyset \cup \emptyset \cdot (1 + \lambda)^* \cdot (1 + \lambda)$$

Aplicăm formula (A) și avem  $\emptyset \cup \emptyset = \emptyset$

$$R_{22}^1 = R_{22}^0 \cup R_{21}^0 \cdot (R_{11}^0)^* \cdot R_{12}^0 = \lambda \cup \emptyset \cdot (1 + \lambda)^* \cdot 0 = \lambda \cup \emptyset = \lambda$$

| $R_{ij}^{k+1}$    | $k+1 = 0$     | $k+1 = 1$     | $k+1 = 2$     |
|-------------------|---------------|---------------|---------------|
| $r_{ij} = r_{11}$ | $1 + \lambda$ | $1^*$         |               |
| $r_{ij} = r_{12}$ | $0$           | $1^* \cdot 0$ | $1^* \cdot 0$ |
| $r_{ij} = r_{21}$ | $\emptyset$   | $\emptyset$   |               |
| $r_{ij} = r_{22}$ | $\lambda$     | $\lambda$     |               |

(c) Calculăm ultima coloană, cea incompletă, pentru  $k+1 = 2$ .

Singura stare finală este  $q_2$ , deci avem de calculat doar eticheta de la starea  $q_1$  (cea inițială) la starea  $q_2$ . (Dacă aveam mai multe stări finale, calculam pentru fiecare câte o expresie și apoi le reunem.)

$$R_{12}^2 = R_{12}^1 \cup R_{12}^1 \cdot (R_{22}^1)^* \cdot R_{22}^1 = (1^* \cdot 0) \cup (1^* \cdot 0) \cdot \lambda^* \cdot \lambda = (1^* \cdot 0) \cup (1^* \cdot 0) = 1^* \cdot 0$$

Deci expresia regulată corespunzătoare automatului dat este  $1^* \cdot 0$ .

### **Gramatici**

$G = (N, T, S, P)$

$N = \{A, B, C, \dots\}$  mulțimea de simboluri neterminale

$T = \{a, b, c, \dots\}$  mulțimea de simboluri terminale

$S \in N$  simbolul de start

$P$  mulțimea de producții (sau reguli de producție)

Modul în care este definită funcția  $P$  determină diferite tipuri de gramatici.

### **Gramatici independente de context (GIC) / Context-free grammars (CFG)**

$$P \subseteq N \times (N \cup T)^*$$

Cazuri particulare de gramatici independente de context:

#### **Gramatici liniare**

Au producții de tipurile  $A \rightarrow x, \quad \text{cu } x \in T^* \text{ și } A \in N$   
 $A \rightarrow xBy, \quad \text{cu } x, y \in T^* \text{ și } A, B \in N$

#### **Gramatici liniare la dreapta**

Au producții de tipurile  $A \rightarrow x, \quad \text{cu } x \in T^* \text{ și } A \in N$   
 $A \rightarrow xB, \quad \text{cu } x \in T^* \text{ și } A, B \in N$

#### **Gramatici liniare la stânga**

Au producții de tipurile  $A \rightarrow x, \quad \text{cu } x \in T^* \text{ și } A \in N$   
 $A \rightarrow Bx, \quad \text{cu } x \in T^* \text{ și } A, B \in N$

#### **Gramatici regulate**

O gramatică se numește regulată dacă este liniară la dreapta sau liniară la stânga.

Dar pentru că gramaticile liniare la dreapta și cele liniare la stânga definesc același set de limbaje, atunci în general ne referim la cele liniare la dreapta.

$$A \rightarrow \lambda$$

Deci gramaticile regulate au producții de tipurile  $A \rightarrow a$

$$A \rightarrow aB, \quad \text{cu } a \in T \text{ și } A, B \in N$$

**Obs:** Putem grupa mai multe producții care au același membru stâng și putem scrie pe scurt  $A \rightarrow \lambda | a | aB, \quad \text{cu } a \in T \text{ și } A, B \in N$

**EXEMPLE (gramatici regulate):**

Vom relua acele exemple din primul seminar pentru care am construit automate finite (pagina 5 din acest document), pentru a vedea corespondența strânsă dintre automate finite și gramatici regulate (dintre stări și neterminale; tranziții și producții).

**Obs:** Vom scrie pe scurt, doar producțiile, acolo unde nu există risc de confuzii (adică se înțelege clar că literele mari sunt neterminale, literele mici sunt terminale, iar simbolul de start este S).

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

$$S \rightarrow aS \mid \lambda$$

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

$$S \rightarrow aS \mid a$$

$$\mathbf{L3} = \{\mathbf{a^{2n}}, \mathbf{n \geq 0}\} = \{\lambda = a^0, aa = a^2, aaaa = a^4, aaaaaa = a^6, \dots\}$$

$$S \rightarrow aA \mid \lambda$$

$$A \rightarrow aS$$

$$\mathbf{L4} = \{\mathbf{a^{2n+1}}, \mathbf{n \geq 0}\} = \{a = a^1, aaa = a^3, aaaaa = a^5, \dots\}$$

$$S \rightarrow aA \mid a \quad \text{sau} \quad S \rightarrow aA$$

$$A \rightarrow aS \quad \text{sau} \quad A \rightarrow aS \mid \lambda$$

$$\mathbf{L5} = \{\mathbf{a^{2n}}, \mathbf{n \geq 1}\} = \{aa = a^2, aaaa = a^4, aaaaaa = a^6, \dots\}$$

$$S \rightarrow aA \quad \text{sau} \quad S \rightarrow aA$$

$$A \rightarrow aS \mid a \quad \text{sau} \quad A \rightarrow aB \mid a \quad \text{sau} \quad A \rightarrow aB$$

$$B \rightarrow aA \quad \text{sau} \quad B \rightarrow aA \mid \lambda$$

$$\mathbf{L6} = \{\mathbf{a^{2n+1}}, \mathbf{n \geq 1}\} = \{aaa = a^3, aaaaa = a^5, aaaaaa = a^7, \dots\}$$

$$S \rightarrow aA \quad \text{sau} \quad S \rightarrow aA$$

$$A \rightarrow aB \quad \text{sau} \quad A \rightarrow aS \mid aB$$

$$B \rightarrow aA \mid a \quad \text{sau} \quad B \rightarrow a$$

$$\mathbf{L7} = \{\mathbf{a^{2n}b^{3m}}, \mathbf{n \geq 0, m \geq 0}\}$$

$$= \{\lambda, a^2, a^4, a^6, a^8, \dots, b^3, b^6, b^9, b^{12}, \dots, a^2b^3, a^2b^6, a^4b^3, a^4b^6, a^6b^3, a^6b^6, \dots\}$$

$$S \rightarrow aA \mid bB \mid \lambda \quad \text{sau} \quad S \rightarrow aA \mid bB \mid \lambda$$

$$A \rightarrow aS \quad \text{sau} \quad A \rightarrow aS$$

$$B \rightarrow bC \quad \text{sau} \quad B \rightarrow bC$$

$$C \rightarrow bD \quad \text{sau} \quad C \rightarrow bD \mid b$$

$$D \rightarrow bB \mid \lambda \quad \text{sau} \quad D \rightarrow bB$$



### **Analogia între automate finite și gramatici regulate**

- i) La limbajele  $L1$  și  $L2$ , unde avem cuvinte de forma  $a^n$ :
  - la automate avem buclă (adică tranziție de la o stare către ea însăși)
  - la gramatici avem o producție în care același neterminal apare și în membrul stâng și în cel drept al producției
- ii) La limbajele  $L3 - L10$ , unde la putere avem și un coeficient număr natural (2 pentru a-uri și 3 pentru b-uri):
  - la automate avem circuit de lungime egală cu coeficientul între stări diferite
  - la gramatici avem circuit de lungime egală cu coeficientul între neterminale diferite
- iii) Dar ce se întâmplă dacă avem limbaje definite folosind condiții asupra numărului de apariții ale caracterelor în cuvinte ?
  - la automate ideea era să folosim indici la stări, câte unul pentru fiecare condiție
  - la gramatici ideea este să folosim indici la neterminale, câte unul pentru fiecare condiție

## ~ Seminar 4 ~

(Exemple cu gramatici independente de context GIC ;  
Transformarea GIC  $\rightarrow$  gramatică în F.N. Chomsky ;  
Automatul Push-Down / Automatul Stivă)

### **EXEMPLE (gramatici independente de context):**

În general, trebuie să lucrăm cu gramatici independente de context, pentru că nu putem cu gramatici regulate (respectiv, trebuie să lucrăm cu automate push-down, pentru că nu putem cu automate finite), atunci când limbajul conține cuvinte în care numărul de litere dintr-o parte a cuvântului depinde de numărul de litere din altă parte a cuvântului și deci avem nevoie să „controlăm” cumva acest număr de litere.

$$L1 = \{a^n b^n, n \geq 0\}$$

$$= \{\lambda = a^0 b^0, ab = a^1 b^1, aabb = a^2 b^2, aaabbb = a^3 b^3, aaaabbbb = a^4 b^4, \dots\}$$

$$S \rightarrow a^{(1)} S b^{(2)} | \lambda \quad \text{sau} \quad \begin{array}{l} S \rightarrow a^{(1)} A b^{(2)} | \lambda \\ A \rightarrow a^{(3)} A b^{(4)} | \lambda \end{array}$$

Ideea acestei gramatici este de a genera cuvântul dinspre margini spre mijlocul lui, astfel pentru fiecare aplicare a producției (1) vor apărea un „a” și un „b”, iar între ele va rămâne S-ul pentru a continua generarea. La final, se aplică producția (2) pentru a nu mai avea neterminat în cuvântul obținut.

Cel mai mic cuvânt este într-adevăr cuvântul vid, pentru că putem aplica direct a doua producție fără să o aplicăm deloc pe prima.

$$L2 = \{a^n b^n, n \geq 1\}$$

$$= \{ab = a^1 b^1, aabb = a^2 b^2, aaabbb = a^3 b^3, aaaabbbb = a^4 b^4, \dots\}$$

$$S \rightarrow a^{(1)} S b^{(2)} | ab \quad \text{sau} \quad \begin{array}{l} S \rightarrow a^{(1)} A b^{(2)} \\ A \rightarrow a^{(2)} A b^{(3)} | \lambda \end{array}$$

Aici, cel mai mic cuvânt trebuie să fie „ab”, deci observăm că din ambele gramatici de la exemplul anterior au dispărut producțiile „ $S \rightarrow \lambda$ ”.

În prima soluție avem ca pas se oprire cuvântul „ab”, iar în a doua soluție suntem obligați să generăm cel puțin un „a” și un „b” (prin prima producție) înainte de a ne opri (cu a treia producție).

$$L3 = \{a^n b^{2n}, n \geq 1\}$$

$$= \{abb=a^1 b^2, aabbbb=a^2 b^4, aaabbbbb=a^3 b^6, \dots\}$$

$$S \xrightarrow{(1)} aSbb \mid \xrightarrow{(2)} abb$$

Pentru fiecare „a” din prima parte a cuvântului trebuie să avem doi de „b” în a doua parte a cuvântului.

Deoarece avem „ $n \geq 1$ ”, pasul de oprire, a doua producție este cu „abb” în membrul drept.

Dacă era „ $n \geq 0$ ”, atunci a doua producție ar fi avut ca membru drept cuvântul vid.

$$L4 = \{a^{2n} b^k c^{3n}, n \geq 0, k \geq 1\}$$

$$= \{b=b^1, bb=b^2, bbb=b^3, bbbb=b^4, \dots, \\ aabccc=a^2 b^1 c^3, aabbccc=a^2 b^2 c^3, aabbbccc=a^2 b^3 c^3, aabbbbccc=a^2 b^4 c^3, \dots, \\ aaaabcccccc=a^4 b^1 c^6, aaaabbcccccc=a^4 b^2 c^6, aaaabbbcccccc=a^4 b^3 c^6, \dots\}$$

$$S \xrightarrow{(1)} aaSccc \mid \xrightarrow{(2)} bA$$

$$A \xrightarrow{(3)} bA \mid \xrightarrow{(4)} \lambda$$

Pentru fiecare „aa” de la începutul cuvântului trebuie să avem „ccc” la finalul lui, iar între ele vom avea „b”-uri. Deci aici vom folosi întâi neterminalul S pentru a genera „a”-urile și „c”-urile, iar apoi vom trece în alt neterminal cu ajutorul căruia vom genera „b”-urile. Observăm că avem „ $n \geq 0$ ”, deci nu trebuie să-l obligăm să aibă „a”-uri și „c”-uri, ci trebuie să îl lăsăm să treacă direct la „b”-uri.

Cel mai mic cuvânt este „b”: aplicăm producția (2) și apoi pe (4).

$$L5 = \{a^{2n} b^k c^{3n}, n \geq 1, k \geq 1\}$$

$$= \{aabccc=a^2 b^1 c^3, aabbccc=a^2 b^2 c^3, aabbbccc=a^2 b^3 c^3, aabbbbccc=a^2 b^4 c^3, \dots, \\ aaaabcccccc=a^4 b^1 c^6, aaaabbcccccc=a^4 b^2 c^6, aaaabbbcccccc=a^4 b^3 c^6, \dots\}$$

$$S \xrightarrow{(1)} aaSccc \mid \xrightarrow{(2)} aaAccc \quad \text{sau} \quad S \xrightarrow{(1)} aaSccc \mid \xrightarrow{(2)} aaAbccc \quad \text{sau} \quad S \xrightarrow{(1)} aaSccc \mid \xrightarrow{(2)} aabAccc$$

$$A \xrightarrow{(3)} bA \mid \xrightarrow{(4)} b \quad \text{sau} \quad A \xrightarrow{(3)} bA \mid \xrightarrow{(4)} \lambda \quad \text{sau} \quad A \xrightarrow{(3)} bA \mid \xrightarrow{(4)} \lambda$$

Diferența față de exemplul anterior este că aici și n este strict pozitiv, de aceea trebuie ca înainte de a trece la neterminalul A să generăm obligatoriu minim doi de „a” și trei de „c” (producția a doua). Apoi în neterminalul A să generăm cel puțin un „b” (de aceea observăm că s-a modificat și producția a patra).

Cel mai mic cuvânt este „aabccc”: aplicăm producția (2) și apoi (4).

$$L6 = \{a^{2n}c^{3n}b^k, n \geq 1, k \geq 1\}$$

$$= \{ aaccb = a^2c^3b^1, aaccbb = a^2c^3b^2, aaccbbb = a^2c^3b^3, aaccbbbbb = a^2c^3b^4, \dots, \\ aaaabcccccb = a^4c^6b^1, aaaacccccbb = a^4c^6b^2, aaaacccccbbb = a^4c^6b^3, \dots \}$$

$$S \xrightarrow{(1)} AB$$

$$A \xrightarrow{(2)} aaAccc \mid aaccc \xrightarrow{(3)}$$

$$B \xrightarrow{(4)} bB \mid b \xrightarrow{(5)}$$

Observăm că nu întotdeauna putem genera cuvintele dinspre exterior spre centru, având maxim un singur neterminal în membrul drept al producțiilor. Uneori este necesar să generăm cuvintele pe bucăți, folosind pentru fiecare parte un neterminal separat. Aici, vom folosi neterminalul A pentru a genera „a”-urile și „c”-urile spre centru simultan câte doi de „a” și câte trei de „c”, iar din neterminalul B vom genera „b”-urile. Dacă am fi avut condiția „ $n \geq 0$ ”, atunci producția (3) ar fi fost „ $A \rightarrow \lambda$ ”, iar dacă am fi avut condiția „ $k \geq 0$ ”, atunci producția (5) ar fi fost „ $B \rightarrow \lambda$ ”.

## Verificare generare cuvânt de către gramatică

**Obs:** Atenție la terminologia folosită:

- automatul *acceptă* un (cuvânt din) limbaj
- cuvântul / limbajul *este acceptat de* automat
- gramatica *generează* un (cuvânt din) limbaj
- cuvântul / limbajul *este generat de* gramatică

Pentru a verifica dacă un cuvânt este sau nu generat de o gramatică, se folosesc „**arborii de derivare**”, care au ca rădăcină simbolul de start al gramaticii (adică S, dacă nu se specifică altul).

Pentru a trece de la un nod la altul al arborelui, ne alegem un neterminal care face parte din cuvântul curent, căutăm acele producții care îl au ca membru stâng, alegem una dintre ele și obținem noul cuvânt curent prin înlocuirea neterminalului cu membrul drept al acelei producții.

- Fie ne oprim când am găsit un drum prin care am ajuns să generăm cuvântul verificat (deci suntem într-o frunză a arborelui, nu mai avem neterminale în cuvânt).
- Fie ne oprim când toate drumurile posibile au creat cuvinte mai lungi decât cel verificat, dar fără să întâlnim frunza corectă (deci nu are rost să continuăm pentru că vom găsi cuvinte din ce în ce mai lungi, deci diferite de cel verificat)

(a) Fie îi reprezentăm chiar sub formă de arbore, de sus în jos începând cu rădăcina, etichetând nodurile cu cuvântul curent, iar muchiile cu numărul producției pe care am aplicat-o pentru a ajunge din nodul-tată în nodul-fiu.

(b) Fie îi reprezentăm de la stânga la dreapta cu aceeași convenție pentru etichetele nodurilor și muchiilor. În acest caz, ori decidem să urmăm toate ramificațiile posibile, ori

scriem direct doar drumul care ne duce la rezultat (fiind cuvinte scurte și gramatici destul de mici, putem „vedea” drumul fără prea mare greutate). Preferabil alegem a doua variantă când avem de verificat un singur cuvânt.

**Obs:** Uneori se poate întâmpla ca *pentru un cuvânt să existe mai mulți arbori de derivare distincți*.

- (i) Dacă *gramatica este regulată*, asta poate însemna că în gramatică există unele producții redundante (care ar putea fi eliminate fără să apară probleme, adică după eliminare să se obțină o gramatică echivalentă cu cea inițială).
- (ii) Iar dacă *gramatica este independentă de context* și are cel puțin o producție în al cărei membru drept apar mai multe neterminale, atunci apar mai mulți arbori pentru un cuvânt care conține la un moment dat mai multe neterminale, pentru că nu este obligatoriu să încercăm să rescriem doar neterminalele cel mai din stânga/dreapta, ci putem să le alegem în orice ordine.

#### EXEMPLU:

Pentru gramatica anterioară, vom verifica dacă generează cuvântul  $a^4c^6b^3 \in L6$ .

$$S \xrightarrow{(1)} AB$$

$$A \xrightarrow{(2)} aaAccc \mid \xrightarrow{(3)} aaccc$$

$$B \xrightarrow{(4)} bB \mid \xrightarrow{(5)} b$$

(a) Aveți aici ([http://bit.ly/arbore\\_derivare](http://bit.ly/arbore_derivare)) poza arborelui de derivare pentru a găsi toate drumurile posibile prin care poate fi obținut cuvântul dat.

În desen, muchiile sunt etichetate cu numărul producției care a fost aplicată, iar *nodurile interioare* cu neterminalele care a fost rescris (în interiorul cercului) și cu noul cuvânt obținut (în dreptunghiul de sub cerc).

Avem mai multe tipuri de *noduri fără fii*:

- Cele marcate cu „--” reprezintă începutul unor subarbori care mai pot fi extinși, dar în care nu vom mai găsi soluții pentru cuvântul dat. Acest lucru se poate întâmpla din două motive: fie am aplicat de prea multe ori una dintre producțiile recursive (2 sau 4), fie am aplicat producția pentru pasul de oprire (3 sau 5) înainte de a o aplica de suficiente ori pe cea recursivă.
- Cele marcate cu „cerc dublu” reprezintă frunze ale arborelui, adică sfârșitul drumurilor în urma cărora am obținut cuvinte fără neterminale, adică unele dintre cuvintele care sunt generate de gramatică.
- Cel marcate cu „cerc dublu în care este scris OK” reprezintă toate drumurile prin care putem genera cuvântul dat.

Observăm (direct din cuvânt și din structura gramaticii) că producțiile pe care trebuie să le aplicăm sunt: o dată (1), apoi o dată (2), o dată (3) (dar numai după ce a fost aplicată 2), de două ori (4), o dată (5) (dar numai după ce a fost aplicată 4 de două ori). Dar ordinea pentru 2,3,4,5 poate fi oricare în afara restricțiilor menționate. De aceea vom obține cele 10 soluții: 23445, 24345, 24435, 24453, 42345, 42435, 42453, 44235, 44253, 44523.

(b) A doua variantă, scriem direct pe scurt un drum prin care știm că vom obține cuvântul dat. Îl vom alege pe primul, în care vom rescrie mereu cel mai din stânga neterminal.

$$\overset{(1)}{S} \Rightarrow \overset{(2)}{AB} \Rightarrow aa\overset{(3)}{Accc}B \Rightarrow a^4c^6\overset{(4)}{B} \Rightarrow a^4c^6b\overset{(4)}{B} \Rightarrow a^4c^6b^2\overset{(5)}{B} \Rightarrow a^4c^6b^3$$

**Obs:** Dacă cerința este doar de a verifica pentru un anumit cuvânt dacă este sau nu generat de gramatică, atunci este preferabilă varianta (b). Dar dacă cerința este să se afle toate cuvintele generate de o gramatică, pentru o lungime de cuvânt dată sau pentru un număr maxim de aplicări ale producțiilor, atunci trebuie folosită varianta (a) cu extinderea mai multor drumuri.

Continuăm cu exemplele de gramatici independente de context.

Dacă avem cuvinte formate din 4 porțiuni (a, b, c, d) și vrem ca lungimile porțiunilor (puterile la care sunt literele) să depindă unele de altele grupate două câte două atunci avem doar două posibilități de grupare: *prima cu a patra* (și a doua cu a treia) sau *prima cu a doua* (și a treia cu a patra).

Deci nu vom putea să avem grupe intersectate (prima cu a treia; a doua cu a patra)!

$$L7 = \{a^n b^k c^{2k} d^n, n \geq 1, k \geq 0\}$$

$$= \{ad, a^2d^2, a^3d^3, a^4d^4, \dots, abc^2d, ab^2c^4d, ab^3c^6d, \dots, a^2bc^2d^2, a^2b^2c^4d^2, a^2b^3c^6d^2, \dots\}$$

$$\begin{array}{lcl} S \rightarrow \overset{(1)}{aSd} \mid \overset{(2)}{aAd} & \text{sau} & S \rightarrow \overset{(1)}{aSd} \mid \overset{(2)}{ad} \mid \overset{(3)}{aAd} \\ A \rightarrow \overset{(3)}{bAcc} \mid \overset{(4)}{\lambda} & & A \rightarrow \overset{(4)}{bAcc} \mid \overset{(5)}{\lambda} \end{array}$$

(în a doua soluție, producția (2) este redundantă pentru că poate fi obținută aplicând (3) și apoi (5))

Aici avem grupate porțiunile 1-4 și 2-3, deci vom genera cuvintele dinspre exterior spre centru, întâi „a”-urile simultan cu „d”-urile, iar apoi „b”-urile simultan cu „c”-urile.

Atenție, trebuie să folosim *neterminale diferite*, pentru că altfel s-ar genera și cuvinte cu literele amestecate („a” și „b” între ele și „c” și „d” între ele).

Avem „ $k \geq 0$ ”, deci „b”-urile și „c”-urile pot lipsi, de aceea avem producția (4) pe care o aplicăm pentru cazurile când avem  $k = 0$ .

Avem „ $n \geq 1$ ”, deci producția (2), pentru că trebuie să ne asigurăm că avem cel puțin un „a” și un „d” înainte de a trece la neterminalul A pentru a genera mijlocul cuvântului.

Dacă am fi avut „ $k \geq 1$ ”, atunci producția (4) s-ar fi transformat în „ $A \rightarrow bcc$ ”.

$$L8 = \{a^n b^k c^{2k} d^n, n \geq 0, k \geq 1\}$$

$$= \{bc^2, b^2c^4, b^3c^6, \dots, abc^2d, ab^2c^4d, ab^3c^6d, \dots, a^2bc^2d^2, a^2b^2c^4d^2, a^2b^3c^6d^2, \dots\}$$

$$S \xrightarrow{(1)} aSd \mid \xrightarrow{(2)} bAcc$$

$$A \xrightarrow{(3)} bAcc \mid \xrightarrow{(4)} \lambda$$

Structura cuvintelor este aceeași ca la exemplul anterior, s-au schimbat doar condițiile. Avem „ $n \geq 0$ ”, deci „a”-urile și „d”-urile pot lipsi, dar mijlocul nu. De aceea avem producția (2) în care folosim direct „bcc”-ul pentru a trece la neterminalul A. Avem „ $k \geq 1$ ”, dar deja am citit un „bcc” cu neterminalul S, deci pasul de oprire pentru A va fi cu cuvântul vid.

Dacă am fi avut „ $k \geq 0$ ”, înseamnă că și mijlocul („b”-urile și „c”-urile) ar fi putut lipsi, deci am fi avut în plus producția „ $S \rightarrow \lambda$ ”.

$$L9 = \{a^n b^{2n} c^{2k} d^k, n \geq 0, k \geq 1\}$$

$$= \{c^2d, c^4d^2, c^6d^3, \dots, ab^2c^2d, ab^2c^4d^2, ab^2c^6d^3, \dots, a^2b^4c^2d, a^2b^4c^4d^2, a^2b^4c^6d^3, \dots\}$$

$$S \xrightarrow{(1)} AB$$

$$A \xrightarrow{(2)} aAbb \mid \xrightarrow{(3)} \lambda$$

$$B \xrightarrow{(4)} ccBd \mid \xrightarrow{(5)} ccd$$

Aici avem grupate porțiunile 1-2 și 3-4, deci din simbolul de start S ne vom ramifica în două părți disjuncte: vom folosi un neterminal A pentru a genera dinspre exterior spre interior porțiunea cu „a”-uri și „b”-uri, iar separat vom folosi un neterminal B pentru a genera dinspre exterior spre interior porțiunea cu „c”-uri și „d”-uri. Avem „ $n \geq 0$ ”, deci pasul de oprire pentru A va fi cuvântul vid. Avem „ $k \geq 1$ ”, deci pasul de oprire pentru B va fi „ccd”.

$$L10 = \{a^n b^{2n} c^k d^m e^{3k}, n, k, m \geq 0\}$$

$$S \xrightarrow{(1)} AB \mid \xrightarrow{(2)} \lambda$$

$$A \xrightarrow{(3)} aAbb \mid \xrightarrow{(4)} \lambda$$

$$B \xrightarrow{(5)} cBeee \mid \xrightarrow{(6)} \lambda \mid \xrightarrow{(7)} dC$$

$$C \xrightarrow{(8)} dC \mid \xrightarrow{(9)} \lambda$$

(producția (2) este redundantă pentru că poate fi obținută din 1, 4, 6)

Vom folosi neterminaul A pentru a genera spre interior „a”-urile și „b”-urile. Separat, vom folosi neterminalul B pentru a genera spre interior „c”-urile și „e”-urile, iar apoi între ele „d”-urile.

Pentru „ $n \geq 0$ ” avem producția (4).

Pentru „ $m \geq 0$ ” avem producția (6).

Pentru „ $k \geq 0$ ” avem producția (7) (în care trecem direct cu „d” mai departe fără să fi citit „ceee”) sau producția (6) (atunci când avem și  $m = 0$ ).

### ***Transformarea GIC $\rightarrow$ gramatică în F.N. Chomsky***

O gramatică este (scrisă) în **forma normală Chomsky** dacă are doar producții care au în membrul stâng un neterminal, iar în membrul drept fie un terminal, fie două neterminale.

$A \rightarrow a$

$A \rightarrow BC$

EXEMPLU:

Avem următoarea gramatică independentă de context:

$S \rightarrow a^{(1)}B^{(2)}c^{(3)}DeF \mid HF \mid HBc$

$B \rightarrow b^{(4)} \mid \lambda^{(5)}$

$D \rightarrow d^{(6)} \mid \lambda^{(7)}$

$F \rightarrow G^{(8)}$

$G \rightarrow f^{(9)} \mid g^{(10)}$

$H \rightarrow \lambda^{(11)}$

#### Pasul 1 – Eliminarea $\lambda$ -producțiilor

Vrem să nu mai avem în gramatică producții care au ca membru drept cuvântul vid.

Pot exista două situații:

- (i) Dacă neterminalul care are o  $\lambda$ -producție NU are și alte producții, atunci
- va fi eliminată producția lui
  - toate producțiile care au ca membru drept un cuvânt de lungime minim 2 în care apare acest neterminal  $\rightarrow$  vor fi înlocuite prin eliminarea neterminalului din cuvinte
  - dacă membrul drept avea lungime 1 (adică era doar acest neterminal), atunci se înlocuiește cu  $\lambda$  (dar și această producție va fi ulterior eliminată tot la „pasul 1”)

*În EXEMPLU:*

Neterminalul H este singurul de acest gen, având doar  $\lambda$ -producția (11), care va fi eliminată. În plus, producția (2) va fi înlocuită cu „ $S \rightarrow F$ ” (12), iar producția (3) va fi înlocuită cu „ $S \rightarrow Bc$ ” (13).



$$S \rightarrow a^{(1)}B^{(12)}c^{(13)}DeF \mid F \mid Bc$$

$$B \rightarrow b^{(4)} \mid \lambda^{(5)}$$

$$D \rightarrow d^{(6)} \mid \lambda^{(7)}$$

$$F \rightarrow G^{(8)}$$

$$G \rightarrow f^{(9)} \mid g^{(10)}$$

- (ii) Dacă neterminalul care are o  $\lambda$ -producție are și alte producții, atunci
- va fi eliminată doar  $\lambda$ -producția lui
  - toate producțiile care au ca membru drept un cuvânt de lungime minim 2 în care apare acest neterminal  $\rightarrow$  vor fi înlocuite atât de varianta în care cuvântul conține neterminalul, cât și de varianta în care neterminalul este eliminat din cuvânt
  - dacă membrul drept avea lungime 1 (adică era doar acest neterminal), atunci această producție este o redenumire și va fi eliminată la „pasul 2”

În *EXEMPLU*:

Neterminalele B și D au mai multe producții printre care și câte o  $\lambda$ -producție. Deci (5) și (7) vor fi eliminate.

În producția (13) apare doar B, deci va fi înlocuită cu „ $S \rightarrow Bc$ ” (14) și „ $S \rightarrow c$ ” (15).

În producția (1) apar și B și D, deci va fi înlocuită de toate variațiile posibile:

-- B și D prezente: „ $S \rightarrow aBcDeF$ ” (16)

-- B lipsă, D prezent: „ $S \rightarrow acDeF$ ” (17)

-- B prezent, D lipsă: „ $S \rightarrow aBceF$ ” (18)

-- B și D lipsă: „ $S \rightarrow aceF$ ” (19)

$$S \rightarrow a^{(16)}B^{(17)}c^{(18)}DeF \mid ac^{(17)}DeF \mid a^{(18)}B^{(19)}ceF \mid ace^{(19)}F \mid F \mid Bc \mid c$$

$$B \rightarrow b^{(4)} \quad ; \quad D \rightarrow d^{(6)} \quad ; \quad F \rightarrow G^{(8)} \quad ; \quad G \rightarrow f^{(9)} \mid g^{(10)}$$

## Pasul 2 – Eliminarea redenumirilor

Vrem să nu mai avem în gramatică producții care au ca membru drept un neterminal.

Înlocuim neterminalul din dreapta cu toate cuvintele care sunt membru drept în producțiile sale.

Verificăm dacă acest neterminal din dreapta mai apare în dreapta în cadrul altor producții

- Dacă nu, atunci eliminăm toate producțiile pe care le avea.
- Dacă mai apare în cuvinte de lungime minim 2, arunci trebuie să i le păstrăm
- Dacă apare în cuvânt de lungime 1, înseamnă că este tot o redenumire și va fi eliminată tot la „pasul 2”

În *EXEMPLU*:

Avem două șiruri de redenumiri:

$$F \rightarrow G^{(8)} \rightarrow f^{(9)} \mid g^{(10)}$$

Producția (8) va fi înlocuită cu „ $F \rightarrow f$ ” (20) și „ $F \rightarrow g$ ” (21).

Neterminalul G nu mai apare nicăieri în dreapta, deci vom elimina (9) și (10).  
 $S \rightarrow F$  (11)  $\rightarrow f$  (20) |  $g$  (21)

Producția (11) va fi înlocuită cu „ $S \rightarrow f$ ” (22) și „ $S \rightarrow g$ ” (23).

Dar neterminalul F mai apare în dreapta în producțiile 16–19, deci NU vom elimina producțiile lui F.

$$S \rightarrow \overset{(16)}{a} \overset{(17)}{B} \overset{(18)}{c} \overset{(19)}{D} \overset{(22)}{e} \overset{(23)}{F} \mid \overset{(22)}{f} \mid \overset{(14)}{g} \mid \overset{(15)}{B} \overset{(14)}{c} \overset{(15)}{c}$$

$$B \rightarrow \overset{(4)}{b} \quad ; \quad D \rightarrow \overset{(6)}{d} \quad ; \quad F \rightarrow \overset{(20)}{f} \mid \overset{(21)}{g}$$

### Pasul 3 – Adăugare neterminale noi pentru terminalele din cuvinte

Vrem ca terminalele să apară doar singure în membrul drept. De aceea, peste tot unde apar în componența unui cuvânt de lungime minim 2, le vom înlocui cu un neterminal nou și vom adăuga producția de la neterminalul nou la terminalul pe care l-a înlocuit.

*În EXEMPLU:*

Producțiile 22, 23, 15, 4, 6, 20, 21 au deja forma dorită.

Producția (16) va fi înlocuită cu „ $S \rightarrow X_a B X_c D X_e F$ ” (24).

Producția (17) va fi înlocuită cu „ $S \rightarrow X_a X_c D X_e F$ ” (25).

Producția (18) va fi înlocuită cu „ $S \rightarrow X_a B X_c X_e F$ ” (26).

Producția (19) va fi înlocuită cu „ $S \rightarrow X_a X_c X_e F$ ” (27).

Producția (14) va fi înlocuită cu „ $S \rightarrow B X_c$ ” (28).

În plus, vom avea „ $X_a \rightarrow a$ ” (29); „ $X_c \rightarrow c$ ” (30); „ $X_e \rightarrow e$ ” (31).

$$S \rightarrow \overset{(24)}{X_a} \overset{(24)}{B} \overset{(24)}{X_c} \overset{(24)}{D} \overset{(24)}{X_e} \overset{(24)}{F} \mid \overset{(25)}{X_a} \overset{(25)}{X_c} \overset{(25)}{D} \overset{(25)}{X_e} \overset{(25)}{F} \mid \overset{(26)}{X_a} \overset{(26)}{B} \overset{(26)}{X_c} \overset{(26)}{X_e} \overset{(26)}{F} \mid \overset{(27)}{X_a} \overset{(27)}{X_c} \overset{(27)}{X_e} \overset{(27)}{F} \mid \overset{(22)}{f} \mid \overset{(23)}{g} \mid \overset{(28)}{B} \overset{(28)}{X_c} \mid \overset{(15)}{c}$$

$$B \rightarrow \overset{(4)}{b} \quad ; \quad D \rightarrow \overset{(6)}{d} \quad ; \quad F \rightarrow \overset{(20)}{f} \mid \overset{(21)}{g}$$

$$X_a \rightarrow \overset{(29)}{a} \quad ; \quad X_c \rightarrow \overset{(30)}{c} \quad ; \quad X_e \rightarrow \overset{(31)}{e}$$

### Pasul 4 – Adăugare neterminale noi pentru „spargerea” cuvintelor lungi (>2)

Vrem ca în dreapta să avem cuvinte formate din exact două neterminale. De aceea, unde avem cuvinte mai lungi, păstrăm doar primul neterminal din cuvânt și îi alipim un neterminal nou, iar noul neterminal va avea o producție cu membrul drept cuvântul pe care l-a înlocuit. Reluăm procedeul până când toate cuvintele ajung la lungimea 2.

**Obs:** Fiecare producție care avea un cuvânt de lungime k va fi înlocuită de k–1 producții cu cuvinte de lungime 2.

*În EXEMPLU:*

Producția (24) va deveni „ $S \rightarrow X_a Y_1$ ” (32), iar  $Y_1 \sim B X_c D X_e F$ .

Apoi „ $Y_1 \rightarrow B Y_2$ ” (33), iar  $Y_2 \sim X_c D X_e F$ .

Apoi „ $Y_2 \rightarrow X_c Y_3$ ” (34), iar  $Y_3 \sim D X_e F$ .

Apoi „ $Y_3 \rightarrow D Y_4$ ” (35), iar  $Y_4 \sim X_e F$ .

În final „ $Y_4 \rightarrow X_e F$ ” (36).

Observăm că putem avea sufixe comune ale cuvintelor și atunci am putea folosi neterminale deja adăugate anterior.

Producția (25) va deveni „ $S \rightarrow X_a Y_2$ ” (37), pentru că aveam  $Y_2 \sim X_c D X_e F$ .

Producția (26) va deveni „ $S \rightarrow X_a Y_5$ ” (38), iar  $Y_5 \sim B X_c X_e F$ .

Apoi „ $Y_5 \rightarrow B Y_6$ ” (39), iar  $Y_6 \sim X_c X_e F$ .

Apoi „ $Y_6 \rightarrow X_c Y_4$ ” (40), pentru că aveam  $Y_4 \sim X_e F$ .

Producția (27) va deveni „ $S \rightarrow X_a Y_6$ ” (41), pentru că aveam  $Y_6 \sim X_c X_e F$ .

$$\begin{aligned}
 S &\rightarrow X_a^{(32)} Y_1^{(37)} \mid X_a^{(38)} Y_2^{(41)} \mid X_a^{(22)} Y_5^{(23)} \mid X_a^{(28)} Y_6^{(15)} \mid f^{(20)} \mid g^{(21)} \mid B X_c \mid c \\
 B &\rightarrow b^{(4)} ; \quad D \rightarrow d^{(6)} ; \quad F \rightarrow f^{(20)} \mid g^{(21)} \\
 X_a &\rightarrow a^{(29)} ; \quad X_c \rightarrow c^{(30)} ; \quad X_e \rightarrow e^{(31)} \\
 Y_1 &\rightarrow B Y_2^{(33)} ; \quad Y_2 \rightarrow X_c Y_3^{(34)} ; \quad Y_3 \rightarrow D Y_4^{(35)} ; \quad Y_4 \rightarrow X_e F^{(36)} ; \quad Y_5 \rightarrow B Y_6^{(39)} ; \quad Y_6 \rightarrow X_c Y_4^{(40)}
 \end{aligned}$$

### ***Automatul Push-Down (APD) / Automatul Stivă***

APD = (Q,  $\Sigma$ ,  $q_0$ , F,  $\delta$ ,  $\Gamma$ ,  $Z_0$ )

Q mulțimea de stări

$\Sigma$  alfabetul de intrare

$q_0 \in Q$  starea inițială

F  $\subseteq$  Q mulțimea de stări finale

$\Gamma$  alfabetul stivei („gamma”)

$Z_0 \in \Gamma$  simbolul inițial al stivei

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

Modul în care funcționează tranzițiile pentru un APD

- La intrare avem nevoie de 3 parametri:

- starea curentă (*element din mulțimea Q*)
- caracterul curent din cuvântul de intrare (*element din mulțimea  $\Sigma$* )
- **simbolul** aflat în vârful stivei (*element din mulțimea  $\Gamma$* )

- La ieșire vom avea 2 parametri:

- starea în care ajungem (*element din mulțimea Q*)
- **cuvântul** cu care înlocuim **simbolul** din vârful stivei (*element din mulțimea  $\Gamma^*$* )

**Atenție!** Simbolul din vârful stivei este mereu *eliminat* din stivă, înainte de a se adăuga noul cuvânt în locul lui.

Deci avem mai multe variante pentru această înlocuire:

- Dacă scriem  $\lambda$  în stivă, înseamnă că doar „am șters” simbolul din vârful stivei fără a adăuga nimic în locul lui.
- Dacă scriem același caracter în stivă (cel care fusese eliminat), înseamnă că dorim să „păstrăm stiva nemodificată”.
- Dacă scriem în stivă un cuvânt de lungime minim 2 avem două cazuri:

- dacă dorim să păstrăm și caracterul care era deja în stivă, atunci cuvântul introdus va avea ultima literă exact cea care fusese eliminată (*deci partea de început, din stânga, a cuvântului reprezintă vârful stivei*).
- dacă nu dorim să păstrăm caracterul care era în vârful stivei, ci chiar să-l înlocuim, atunci cuvântul se poate termina cu alt caracter decât cel eliminat.

**Obs:** La fel ca automatele finite, și automatele push-down pot fi:

- „*deterministe*” (definite anterior, dar sunt acceptate și  $\lambda$ -tranzițiile! Dar dacă între două stări avem  $\lambda$ -tranziție, nu putem avea tranziție și cu un caracter din  $\Sigma$ .)
- „*nedeterministe*” (când avem mai multe posibilități de continuare: fie ajungând în stări diferite, fie scriind pe stivă cuvinte diferite, fie ambele (ajungând în stări diferite și scriind pe stivă cuvinte diferite))
- „*nedeterministe cu  $\lambda$ -tranziții*” (când între două stări avem voie să avem tranziții și cu  $\lambda$  și cu caractere din  $\Sigma$  și în plus avem nedeterminismul descris anterior)

### Modalități de acceptare a unui cuvânt de către un APD

(a) cu stări finale

$$L_F(APD) = \{w \mid (q_0, w, Z_0) \xrightarrow{*} (p, \underline{\lambda}, \alpha), p \in F, \alpha \in \Gamma^*\}$$

*(Obs: semnul cu steluță deasupra ar trebui să fie doar T-ul întors, fără săgeată în dreapta, dar nu am găsit simbolul potrivit...)*

Citim: Cuvântul  $w$  este acceptat de automat dacă pornind din configurația având starea  $q_0$ , cuvântul de intrare  $w$  și pe stivă simbolul inițial  $Z_0$ , după ce aplicăm un număr oarecare de tranziții ajungem în configurația finală.

În acest caz (a) ne interesează:

- prima componentă să fie o stare finală;
- a doua componentă să fie cuvântul vid (adică să fi terminat de parcurs tot cuvântul de intrare)

În acest caz (a) NU ne interesează a treia componentă (conținutul stivei): poate să fie vidă sau poate să conțină un cuvânt cu simboluri din alfabetul  $\Gamma$ .

(b) cu vidarea stivei

$$L_\lambda(APD) = \{w \mid (q_0, w, Z_0) \xrightarrow{*} (p, \underline{\lambda}, \underline{\lambda}), p \in Q\}$$

În acest caz (b) ne interesează:

- a doua componentă să fie cuvântul vid (adică să fi terminat de parcurs tot cuvântul de intrare)
- a treia componentă să fie cuvântul vid (adică stiva să fie vidă)

În acest caz (b) NU ne interesează prima componentă: starea în care ajungem poate să fie finală sau poate să fie nefinală.

**Obs:** Atenție, dacă ștergeți tot conținutul stivei, *automatul își oprește funcționarea*, pentru că nu poate aplica funcția delta dacă nu mai are parametrul al treilea (simbolul din vârful stivei). Deci asigurați-vă că nu se va întâmpla asta înainte de a fi citit tot cuvântul de intrare. În general e bine să-l păstrăm pe  $Z_0$  la început și să-l eliminăm abia la final (eventual folosind o  $\lambda$ -tranziție).

(c) cu stări finale și cu vidarea stivei

*În unele cărți apare și acest al treilea caz, care este de fapt intersecția primelor două.*

$$L_{F \& \lambda}^*(APD) = \{w \mid (q_0, w, Z_0) \mapsto^* (\underline{p}, \underline{\lambda}, \underline{\lambda}), p \in F\}$$

În acest caz (c) ne interesează:

- prima componentă să fie o stare finală;
- a doua componentă să fie cuvântul vid (adică să fi terminat de parcurs tot cuvântul de intrare)
- a treia componentă să fie cuvântul vid (adică stiva să fie vidă)

### **EXEMPLE (automate push-down):**

De fiecare dată vom scrie în cuvinte modul în care vom folosi stiva.

Adică pentru un caracter anume citit din cuvântul de intrare ce dorim să se întâmple pe stivă:

- să eliminăm de tot vârful stivei sau să-l păstrăm?
- să scriem și ceva în plus sau să lăsăm stiva exact cum era înainte?

Putem reprezenta automatul push-down în două moduri:

(i) Fie alegem să scriem lista cu cazurile în care definim funcția de tranziție și menționăm dacă avem stări finale care sunt ele.

(ii) Fie alegem să facem ca la automatele finite, să desenăm sub formă de graf cu nodurile stări și muchiile tranziții.

**Obs:** Avantajul alegerii variantei (i) ar fi faptul că putem pune ”comentarii” în dreptul rândurilor pentru a urmări mai ușor logica automatului, iar dacă greșim ceva când scriem pe foaie e mult mai ușor de corectat prin tăierea/adăugarea unui rând (mai ales pentru că nu contează ordinea în care sunt scrise rândurile) decât modificarea pe desen.

Totuși, pentru unele limbaje se poate să vi se pară mai intuitivă varianta cu graful și atunci o puteți folosi.

$$L1 = \{a^n b^n, n \geq 1\}$$

Vrem ca numărul de „a”-uri să fie egal cu numărul de „b”-uri, deci vom reține acest număr cu ajutorul stivei astfel:

- pentru fiecare „a” citit din cuvântul de intrare, vom adăuga un „A” în stivă;
- pentru fiecare „b” citit vom șterge un „A” din stivă;

Trebuie să ne asigurăm că citim cel puțin un „a” și un „b” (pentru că  $n \geq 1$ ) și că literele nu sunt amestecate. De aceea, după ce citim primul „b” va trebui să schimbăm starea.

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$  // citim primul a, mereu inițial îl găsim pe  $Z_0$  în stivă

// vrem să scriem A, dar să-l păstrăm și pe  $Z_0$ , deci scriem „AZ<sub>0</sub>”

$\delta(q_0, a, A) = (q_0, AA)$  // citim oricare alt a și avem A în stivă (introdus anterior),

// vrem să scriem A, dar fără să-l ștergem pe anteriorul, deci scriem „AA”

$\delta(q_0, b, A) = (q_1, \lambda)$  // citim primul b, găsim A în stivă

// schimbăm starea, vrem să-l ștergem pe A, deci scriem în locul lui „λ”

$\delta(q_1, b, A) = (q_1, \lambda)$  // citim oricare alt b, găsim sigur A în stivă; scriem „λ” pe stivă

Pentru a termina execuția și a accepta cuvintele prin cele 3 metode:

(a) cu stări finale:

Adăugăm  $\delta(q_1, \lambda, Z_0) = (q_2, Z_0)$  sau  $\delta(q_1, \lambda, Z_0) = (q_2, \lambda)$ .

Avem  $F = \{q_2\}$  stare finală. Putem să-l păstrăm pe  $Z_0$  pe stivă sau să-l ștergem.

(b) cu vidarea stivei:

Adăugăm  $\delta(q_1, \lambda, Z_0) = (q_1, \lambda)$  sau  $\delta(q_1, \lambda, Z_0) = (q_2, \lambda)$ .

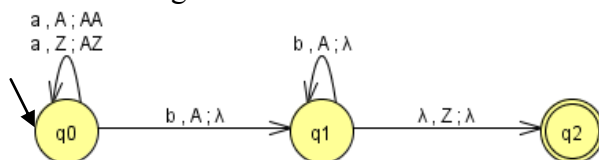
Putem avea  $q_1 \notin F, q_2 \in F$ . Ne interesează doar să avem λ pe stivă.

(c) cu stări finale și cu vidarea stivei:

Adăugăm  $\delta(q_1, \lambda, Z_0) = (q_2, \lambda)$ .

Avem  $F = \{q_2\}$  stare finală și stiva vidă.

Aceeași soluție (c), sub formă de graf:



**Obs:** Tranzițiile se reprezintă cu structura următoare: caracterul citit la intrare, apoi virgulă, apoi caracterul care va fi eliminat din vârful stivei, apoi punct și virgulă (sau slash „/”), apoi cuvântul care va fi scris în stivă (cu aceeași convenție, prima literă din cuvânt este cea care va fi în vârful stivei după adăugarea cuvântului).

$L2 = \{a^n b^{2n}, n \geq 0\}$

Pentru fiecare „a” citit, vom adăuga doi de A în stivă (adică AA).

Pentru fiecare „b” citit, vom șterge un A din stivă.

Astfel ne asigurăm că numărul de „b”-uri este dublul celui de „a”-uri.

Schimbăm starea după ce citim primul „b”, pentru a nu amesteca literele.

Trebuie să ne asigurăm că acceptăm și cuvântul vid (pentru că avem „ $n \geq 0$ ”).

$\delta(q_0, a, Z_0) = (q_0, AAZ_0)$  // adaugam AA pentru primul "a"

$\delta(q_0, a, A) = (q_0, AAA)$  // adaugam AA pentru oricare "a"

$\delta(q_0, b, A) = (q_1, \lambda)$  // stergem un A pentru primul "b"

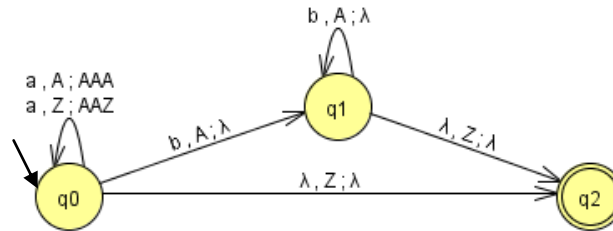
$\delta(q_1, b, A) = (q_1, \lambda)$  // stergem un A pentru oricare "b"

$\delta(q_1, \lambda, Z_0) = (q_2, \lambda)$  // gata cuvântul, stare finala si stiva vida

$\delta(q_0, \lambda, Z_0) = (q_2, \lambda)$  // acceptam cuvântul vid, stare finala si stiva vida

$F = \{q_2\}$

Aceeași soluție, sub formă de graf:



$L3 = \{a^{2n}b^k c^{3k}d^n, n \geq 1, k \geq 1\}$

Pentru fiecare „aa” de la începutul cuvântului trebuie să avem „d” la finalul cuvântului.

(Obs: Dacă am vrea să adăugăm câte un A pentru fiecare „a”, atunci ar trebui ca pentru fiecare „d” să dispară doi de A; nu putem face asta direct, ci ar trebui un circuit de lungime 2 între două stări, cu o muchie se elimină un A pentru „d”-ul citit, iar cu muchia de întoarcere se elimină un A făcând o λ-tranziție)

Cel mai simplu, pentru pentru fiecare „aa” citiți, vom scrie în stivă un singur A (pentru primul „a” din pereche se adaugă A, pentru al doilea „a” nu se modifică stiva). Avem nevoie de circuit de lungime 2 pentru a controla ca numărul de „a”-uri să fie par.

La mijlocul cuvântului, pentru fiecare „b” din stânga trebuie să avem „ccc” în dreapta.

Deasupra A-urilor din stivă, pentru fiecare „b” vom adăuga câte 3 de B.

Apoi pentru fiecare „c” vom șterge câte un B din stivă.

(Obs: Pentru „c”-uri nu vom avea nevoie de circuit de lungime 3 între stări pentru că în stivă avem deja un număr de B-uri multiplu de 3.)

La final, pentru fiecare „d” vom șterge câte un „A”.

$\delta(q_0, a, Z_0) = (q_1, AZ_0)$  // pentru primul „a”, adăugăm un A

$\delta(q_1, a, A) = (q_0, A)$  // pentru „a”-ul al 2-lea (4-lea, 6-lea, ...) nu modificăm stiva

$\delta(q_0, a, A) = (q_1, AA)$  // pentru „a”-ul al 3-lea (5-lea, 7-lea, ...) adăugăm un A

$\delta(q_0, b, A) = (q_2, BBBA)$  // pentru primul „b” schimbăm starea, adăugăm BBB

$\delta(q_2, b, B) = (q_2, BBBB)$  // pentru orice alt „b”, adăugăm BBB

$\delta(q_2, c, B) = (q_3, \lambda)$  // pentru primul „c” schimbăm starea, ștergem un B

$\delta(q_3, c, B) = (q_3, \lambda)$  // pentru orice alt „c”, ștergem un B

$\delta(q_3, d, A) = (q_3, \lambda)$  // pentru orice „d”, ștergem un A

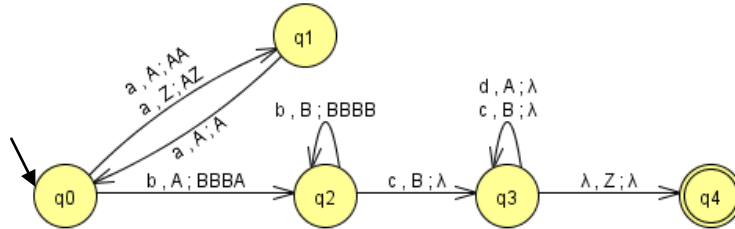
$\delta(q_3, \lambda, Z_0) = (q_4, \lambda)$  // am terminat cuvântul, stare finală și stivă vidă

$F = \{q_4\}$

**Obs:** Când am început să citim „d”-urile nu a fost neapărat necesar să schimbăm starea față de cea în care am citit „c”-urile, deoarece nu există problema amestecării lor.

Putem citi „c”-uri doar atât timp cât găsim B-uri în stivă, iar apoi putem citi „d”-uri doar pentru A-urile din stivă.

Aceeași soluție, sub formă de graf:



### Verificare acceptare cuvânt de către un APD

Folosim „configurații” (sau „descriseri instantanee”) având 3 componente: starea curentă, cât a mai rămas de citit din cuvântul de intrare și (întreg!) conținutul stivei (partea stângă fiind vârful stivei, unde se fac ștergerile și scrierile).

Fie  $n = 3$  și  $k = 2$ , deci avem cuvântul  $a^6b^2c^6d^3$ .

Scriu de fiecare dată numărul tranziției pe care am aplicat-o (a câta era în lista scrisă anterior).

$(q_0, a^6b^2c^6d^3, Z_0) \xrightarrow{(1)} (q_1, a^5b^2c^6d^3, AZ_0) \xrightarrow{(2)} (q_0, a^4b^2c^6d^3, AZ_0)$   
 $\xrightarrow{(3)} (q_1, a^3b^2c^6d^3, AZ_0) \xrightarrow{(2)} (q_0, a^2b^2c^6d^3, AAZ_0)$   
 $\xrightarrow{(3)} (q_1, ab^2c^6d^3, AAAZ_0) \xrightarrow{(2)} (q_0, b^2c^6d^3, AAAZ_0)$   
 $\xrightarrow{(4)} (q_2, bc^6d^3, BBBAAAZ_0) \xrightarrow{(5)} (q_2, c^6d^3, B^6A^3Z_0)$   
 $\xrightarrow{(6)} (q_3, c^5d^3, B^5A^3Z_0) \xrightarrow{(7)} (q_3, c^4d^3, B^4A^3Z_0) \xrightarrow{(7)} (q_3, c^3d^3, B^3A^3Z_0)$   
 $\xrightarrow{(7)} (q_3, c^2d^3, B^2A^3Z_0) \xrightarrow{(7)} (q_3, cd^3, BA^3Z_0) \xrightarrow{(7)} (q_3, d^3, A^3Z_0)$   
 $\xrightarrow{(8)} (q_3, d^2, A^2Z_0) \xrightarrow{(8)} (q_3, d, AZ_0) \xrightarrow{(8)} (q_3, \lambda, Z_0)$   
 $\xrightarrow{(9)} (q_4, \lambda, \lambda), q_4 \in F$ , deci cuvântul este acceptat.

$L4 = \{a^{2n}b^{3n}c^{2k}d^k, n \geq 1, k \geq 1\}$

Acum avem grupate primele două părți și ultimele două părți.

Pentru fiecare „aa” trebuie să avem apoi „bbb”. Pentru primul „a” din fiecare pereche vom adăuga A în stivă, iar pentru al doilea „a” din fiecare pereche vom adăuga AA în stivă. Deci pentru fiecare „b” vom șterge câte un A din stivă.

(Obs: Am fi putut la fel de bine pentru unul dintre „a” să scriem AAA iar pentru celălalt nimic. Orice variantă am alege, important e ca după citirea „aa” să apară AAA în stivă.)

Apoi, după ce toți A au fost șterși, trebuie să avem pentru fiecare „cc” câte un „d”. Pentru primul „c” din fiecare pereche vom adăuga un D în stivă, iar pentru al doilea „c” din fiecare pereche vom lăsa stiva nemodificată. Apoi pentru fiecare „d” citit vom șterge un D din stivă.



Atât pentru „a”-uri cât și pentru „c”-uri vom avea nevoie de câte un circuit de lungime 2 între stări.

$\delta(q_0, a, Z_0) = (q_1, AZ_0)$  // pentru primul „a”, adăugăm un A

$\delta(q_1, a, A) = (q_0, AAA)$  // pentru „a”-ul al 2-lea (4-lea, 6-lea, ...) adăugăm AA

$\delta(q_0, a, A) = (q_1, AA)$  // pentru „a”-ul al 3-lea (5-lea, 7-lea, ...) adăugăm un A

$\delta(q_0, b, A) = (q_2, \lambda)$  // pentru primul „b” schimbăm starea, ștergem un A

$\delta(q_2, b, A) = (q_2, \lambda)$  // pentru orice alt „b”, ștergem un A

$\delta(q_2, c, Z_0) = (q_3, DZ_0)$  // pentru primul „c” schimbăm starea, adăugăm un D

$\delta(q_3, c, D) = (q_2, D)$  // pentru „c”-ul al 2-lea (4-lea, 6-lea, ...) nu modificăm stiva

$\delta(q_2, c, D) = (q_3, DD)$  // pentru „c”-ul al 3-lea (5-lea, 7-lea, ...) adăugăm un D

$\delta(q_2, d, D) = (q_4, \lambda)$  // pentru primul „d” schimbăm starea, ștergem un D

$\delta(q_4, d, D) = (q_4, \lambda)$  // pentru orice alt „d”, ștergem un D

$\delta(q_4, \lambda, Z_0) = (q_5, \lambda)$  // am terminat cuvântul, stare finală și stivă vidă

$F = \{q_5\}$

*Obs: Când am vrut să închidem circuitul pentru „c”-uri nu a trebuit neapărat să schimbăm starea, am putut folosi  $q_2$  pentru că aveam „b” cu A și „c” cu D (sau  $Z_0$ ), deci nu exista risc de amestecare. Dar când am început citirea „d”-urilor a trebuit să o schimbăm pentru că aveam „c” cu D și „d” cu D, deci s-ar fi putut amesteca adăugările și ștergerile D-urilor (adică „c”-urile cu „d”-urile).*

Aceeași soluție, sub formă de graf:

