Cuprins

~ Seminar 1 ~

(AFD; AFN; Operaţii cu limbaje; Compunere automate; Transformarea AFN→AFD)

Automat Finit Determinist

AFD = (Q, Σ, q_0, δ, F)
Q mulţimea de stări
Σ alfabetul de intrare
$q_0 \in Q$ starea iniţială
$F \subseteq Q$ mulţimea de stări finale
$\delta : Q \times \Sigma \to Q$ funcţia de tranziţie („delta")

Pentru a verifica dacă un cuvânt este sau nu acceptat de un automat AFD:
- se porneşte din starea iniţială q_0
- cât timp este posibil:
  - având starea curentă şi caracterul curent din cuvânt, aplicăm funcţia δ pentru a determina noua stare
  - repetăm pentru noua stare şi 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 funcţia de tranziţie nu era definită pentru starea curentă şi caracterul curent → *"cuvânt neacceptat"*
  - dacă întreg cuvântul a fost citit, dar automatul a ajuns într-o stare care nu este finală (adică din mulţimea Q\F) → *"cuvânt neacceptat"*
  - dacă întreg cuvântul a fost citit, iar automatul a ajuns într-o stare care este finală (adică din mulţimea F) → *"cuvânt acceptat"*

Automat Finit Nedeterminist

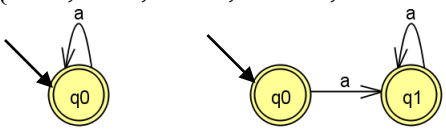
AFN = (Q, Σ, q_0, δ, F)
Q mulţimea de stări
Σ alfabetul de intrare
$q_0 \in Q$ starea iniţială
$F \subseteq Q$ mulţimea de stări finale
$\delta : Q \times \Sigma \to 2^Q$ funcţia de tranziţie („delta")

Diferenţa dintre AFD şi AFN constă în modul în care este definită funcţia de tranziţie.
La AFD: pentru orice pereche de o stare şi un caracter, avem *cel mult* o stare de continuare (putem să nu avem niciuna, adică funcţia să fie nedefinită pentru acele valori).
La AFN: pentru orice pereche de o stare şi un caracter, avem ca posibilitate de continuare o submulţime a mulţimii de stări Q (putem să nu avem niciuna dacă funcţia 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, excepţia fiind că la fiecare aplicare a funcţiei delta trebuie să reţinem toate posibilităţile de stări de continuare, iar la pasul următor să reaplicăm delta pentru toate acestea împreună cu caracterul curent.

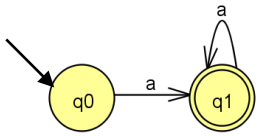
EXEMPLE:

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



Ambele soluţii sunt AFD-uri.
Obs: De fiecare dată când λ (cuvântul vid) trebuie să fie recunoscut de un AFD sau de un AFN (simplu, adică AFN fără λ-tranziţii), trebuie ca starea iniţială q_0 să fie şi stare finală.

L2 = a^+ = {a^n, n ≥ 1} = {$a=a^1$, $aa=a^2$, $aaa=a^3$, $aaaa=a^4$, …}



Diferenţa faţă de exemplul anterior?
Aici λ nu trebuie acceptat, deci starea iniţială q_0 trebuie să nu fie stare finală.

L3 = {a^{2n}, n ≥ 0} = {$\lambda=a^0$, $aa=a^2$, $aaaa=a^4$, $aaaaaa=a^6$, …}
L4 = {a^{2n+1}, n ≥ 0} = {$a=a^1$, $aaa=a^3$, $aaaaa=a^5$, …}

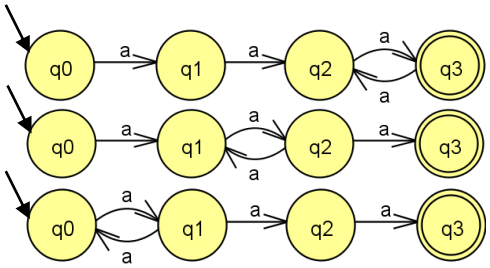
Cele două automate au aceleaşi stări şi aceleaşi tranziţii. Diferă doar mulţimea stărilor finale. Dacă setăm q_0 finală, atunci vor fi acceptate cuvintele de lungime pară (inclusiv zero), iar dacă setăm în schimb q_1 finală, atunci vor fi acceptate cuvintele de lungime impară.

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

L5 = {a^{2n}, n ≥ 1} = {$aa=a^2$, $aaaa=a^4$, $aaaaaa=a^6$, …}

În primul rând trebuie să ne asigurăm că obligăm automatul să citească cei doi de "a" (cel mai mic cuvânt din limbaj) şi că nu acceptă ceva mai scurt de atât. Pentru asta, adăugăm cele trei stări şi tranziţiile de la q_0 spre q_1 şi 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 tranziţie de la q_2 spre q_1, fie adăugăm tranziţie de la q_1 spre q_0. Ambele automate recunosc acelaşi limbaj (dar primul este AFD, iar al doilea este AFN pentru că din q_1 cu "a" putem ajunge şi în q_0 şi în q_2).

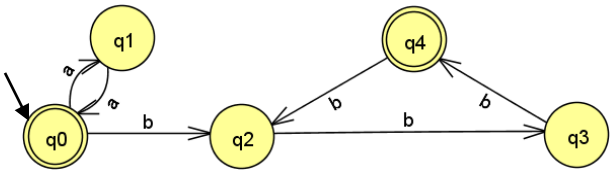
L6 = {a^{2n+1}, n ≥ 1} = {aaa=a^3, aaaaa=a^5, aaaaaaa=a^7, …}



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 şi tranziţiile către dreapta şi 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 tranziţia de la q_3 spre q_2 vom obţine un AFD, iar dacă alegem una din celelalte două variante vom obţine un AFN.

L7 = {$a^{2n}b^{3m}$, n ≥ 0, m ≥ 0}

= {λ, a^2, a^4, a^6, a^8, …, b^3, b^6, b^9, b^{12}, …, a^2b^3, a^2b^6, a^4b^3,a^4b^6, a^6b^3, a^6b^6, …}

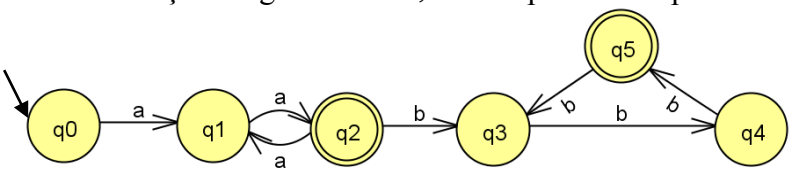


În primul rând vedem că cel mai mic cuvânt ce trebuie acceptat este λ, deci setăm starea iniţială q_0 să fie şi stare finală. Apoi facem circuit de lungime 2 cu litera "a". După ce citim un număr par de a-uri (inclusiv zero), ajungem în starea q_0, deci de aici trebuie să pornim cu citirea b-urilor. Adăugăm q_2, q_3 şi cele două tranziţii cu b. Ştim că va trebui să avem un circuit de lungime 3 cu b-uri, dar trebuie să fim de asemenea atenţi că literele de "a" şi 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, aşa că adăugăm q_4 şi tranziţii de la q_3 spre q_4 şi de la q_4 spre q_2 pentru a închide circuitul.

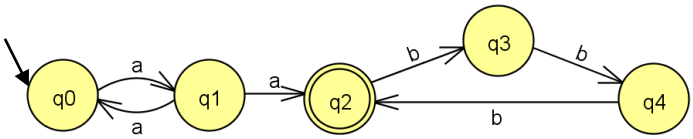
L8 = {$a^{2n} b^{3m}$, n ≥ 1, m ≥ 0}

= {a^2, a^4, a^6, a^8, …, a^2b^3, a^2b^6, a^4b^3,a^4b^6, a^6b^3, a^6b^6, …}

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



Trebuie să obligăm automatul să citească număr par de a-uri, dar cel puţin două, deci nu se mai poate întoarce din q_1 în q_0 cu al doilea "a", ci trebuie să continue spre q_2 şi 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 şi starea q5 care va fi şi ea finală şi cele două tranziţii de la q_4 spre q_5 şi de la q_5 spre q_3.

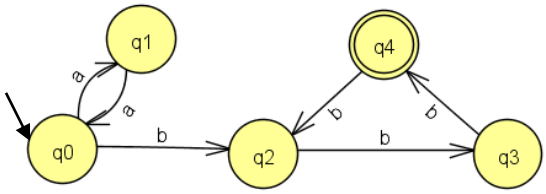


Dar dacă am fi ales ca muchie de întoarcere pentru a-uri de la q_1 spre q_0?
Atunci nu am mai fi avut nevoie de starea q_5, pentru că puteam să închidem circuitul b-urilor de la q_4 spre q_2 şi am fi avut doar starea q_2 finală.
Doar că în acest caz am fi obţinut un AFN (din cauza lui q_1).

L9 = {$a^{2n} b^{3m}$, n ≥ 0, m ≥ 1}

= {b^3, b^6, b^9, b^{12}, …, a^2b^3, a^2b^6, a^4b^3,a^4b^6, a^6b^3, a^6b^6, …}

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

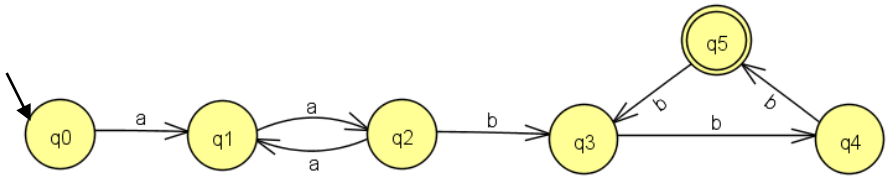


Observăm că automatul este foarte asemănător cu cel pentru limbajul L7.
Diferenţa este că acum starea q_0 nu mai este stare finală, ceea ce înseamnă că putem sau nu să trecem prin circuitul de a-uri, dar sigur suntem obligaţi să citim b-uri înainte de a ajunge într-o stare finală.

L10 = {$a^{2n} b^{3m}$, n ≥ 1, m ≥ 1}

= {a^2b^3, a^2b^6, a^4b^3,a^4b^6, a^6b^3, a^6b^6, …}

În acest caz, cuvintele conţin obligatoriu şi a-urile, şi b-urile.



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

Operaţii cu limbaje

Reuniune $L = L_1 \bigcup L_2$

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

Exemplu:
L_1 = {a, ab, abc}
L_2 = {a, bd, cd}
→ L = {a, ab, abc, bd, cd}

Concatenare $L = L_1 \cdot L_2$

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

Exemplu:
L_1 = {a, ab, abc}
L_2 = {a, bd, cd}
→ L = {aa, abd, acd, aba, abbd, abcd, abca, abcbd, abccd}

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

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

Exemplu:
L_1 = {a, ab, abc}
→ $(L_1)^*$ = {λ ; a, ab, abc ; aa, aab, aabc, aba, abab, ababc, abca, abcab, abcabc ; …}

Obs: Cuvântul vid λ va aparţine limbajului stelat, indiferent dacă înainte aparţinea sau nu limbajului iniţial.

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

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

Obs: Atenţie la ridicarea la putere a cuvintelor! *(Să nu distribuiţi puterea ca la înmulţire, ci concatenaţi cuvântul cu el însuşi 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 iniţiale q_{01} şi q_{02} şi mulţimile stărilor finale F1 şi F2) care acceptă respectiv limbajele L1 şi L2 şi dorim să obţinem un automat finit care să accepte limbajul $L = L_1 \bigcup L_2$:

- introducem o nouă stare q_0 care va fi noua stare iniţială
- adăugăm două λ-tranziţii de la q_0 spre q_{01} şi de la q_0 spre q_{02}
- noua mulţime de stări finale va fi $F = F_1 \bigcup F_2$

Concatenare

Dacă avem deja construite două automate finite (având stările iniţiale q_{01} şi q_{02} şi mulţimile stărilor finale F1 şi F2) care acceptă respectiv limbajele L1 şi L2 şi dorim să obţinem un automat finit care să accepte limbajul $L = L_1 \cdot L_2$:

- noua stare iniţială va fi q_{01}, starea iniţială a primului automat
- adăugăm λ-tranziţii de la fiecare stare finală din F1 spre q_{02}
- noua mulţime de stări finale va fi F = F2

Stelare

Dacă avem deja construit un automat finit (având starea iniţială q_{01} şi mulţimea stărilor finale F1) care acceptă limbajul L1 şi dorim să obţinem un automat finit care să accepte limbajul $L = L_1^*$:

- introducem o nouă stare q_0 care va fi noua stare iniţială
- adăugăm o λ-tranziţie de la q_0 spre q_{01}
- adăugăm λ-tranziţii de la fiecare stare finală din F1 spre q_0
- noua mulţime de stări finale va fi $F = \{q_0\}$ *(sau putem lăsa* $F = \{q_0\} \bigcup F_1$ *)*

Transformarea AFN→AFD

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

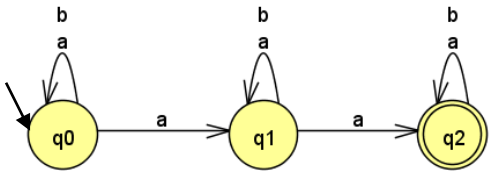
- La capetele de linie vom pune numele stărilor AFN-ului (elementele mulţimii Q): $q_0, q_1, ..., q_{|Q|-1}$
- La capetele de coloană vom pune caracterele alfabetului AFN-ului (elementele mulţimii Σ)
- Calculăm conţinutul primelor |Q| linii astfel: elementul aflat pe linia l_i şi coloana c_j reprezintă mulţimea 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) şi citim caracterul de pe coloana c_j. Pentru a afla această mulţime, ne uităm la graful

care reprezintă AFN-ul, căutăm starea de la linia l_i şi pentru fiecare tranziţie cu litera de la coloana c_j care porneşte din această stare, adăugăm starea destinaţie a tranziţiei la mulţimea curentă din tabel.

- Apoi trebuie să calculăm această funcţie delta şi pentru noile mulţimi de stări obţinute la paşii anteriori (aceste calcule se fac cel mai simplu folosind doar liniile din tabel calculate la pasul anterior, nu şi graful automatului AFN)
  (a) fie alegem să facem calculele pentru *toate noile mulţimi de stări*, şi abia apoi să vedem care dintre ele vor apărea în AFD-ul obţinut (tot făcând "parcurgerea" în adâncime descrisă mai jos).
  (b) fie alegem să facem calculele *doar pentru noile mulţimi 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 iniţială a AFD-ului, apoi parcurgem prima linie a tabelului (care conţine calculele tranziţiilor ce pleacă din q_0) şi adăugăm la noul automat toate aceste tranziţii împreună cu stările-destinaţie calculate în tabel. Apoi pentru toate aceste stări-destinaţie, dacă nu au deja în tabel un capăt de linie care să le corespundă, îl vom adăuga şi vom calcula acea linie astfel:
    - fie noua stare-mulţime $\{q_{k1}, q_{k2}, \ldots, q_{kp}\}$. Pentru fiecare coloană c_j, calculăm elementul din tabel făcând **reuniunea** elementelor aflate pe aceeaşi coloană c_j şi pe liniile corespunzătoare stărilor $q_{k1}, q_{k2}, \ldots, q_{kp}$.
  Apoi revenim la desenarea grafului AFD-ului, şi adăugăm tranziţiile proaspăt calculate împreună cu stările-destinaţie. Reluăm procedeul până nu mai apar stări noi.
- Stările finale ale noului AFD obţinut, vor fi acele stări-mulţime (pot avea şi un singur element, nu neapărat mai multe) care conţin cel puţin una din stările care erau finale pentru AFN-ul iniţial.

EXEMPLU:

Se dă AFN-ul următor:



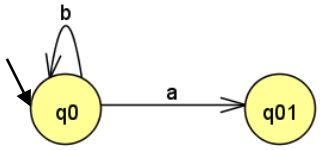
Calculăm tabelul pentru funcţia delta: întâi primele 3 linii folosind graful AFN-ului, apoi următoarele linii folosind aceste prime 3 linii ale tabelului.

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

δ	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 şi îi adăugăm cele două tranziţii şi stările-destinaţie necesare, adică $\{q_0, q_1\} = q_{01}$ (prin notaţie).



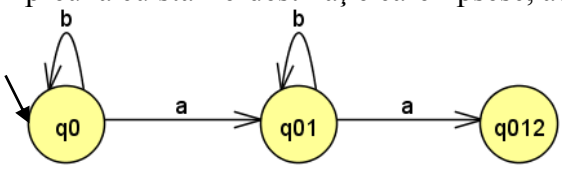
(2) Observăm că a apărut noua stare q_{01} pentru care nu avem o linie în tabel, aşa că o adăugăm şi o calculăm pentru cele două coloane.

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

$$\delta(q_{01}, b) = \delta(\{q_0, q_1\}, b)$$
$$= \{\delta(q_0, b)\} \cup \{\delta(q_1, b)\}$$
$$= \{q_0\} \cup \{q_1\} = \{q_0, q_1\} = q_{01}$$

δ	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 şi adăugăm tranziţiile care lipsesc, adică cele care pleacă din starea q_{01}, împreună cu stările-destinaţie care lipsesc, adică q_{012}.



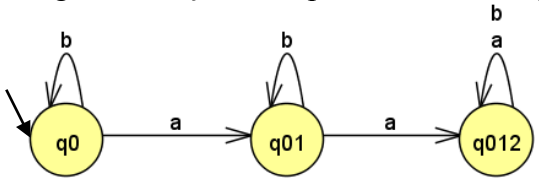
(4) Pentru noua stare apărută, q_{012}, îi adăugăm linia în tabel şi o calculăm.

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

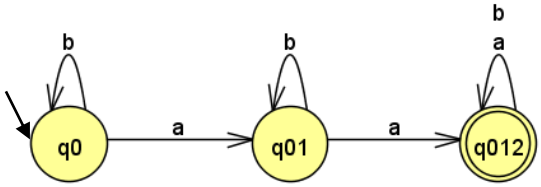
$$\delta(q_{012}, b) = \delta(\{q_0, q_1, q_2\}, b)$$
$$= \{\delta(q_0, b)\} \cup \{\delta(q_1, b)\} \cup \{\delta(q_2, b)\}$$
$$= \{q_0\} \cup \{q_1\} \cup \{q_2\} = \{q_0, q_1, q_2\} = q_{012}$$

δ	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 şi adăugăm tranziţiile 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 şi am obţinul AFD-ul cerut. În AFN-ul iniţial stare finală era doar q_2. Orice stare-mulţime din AFD-ul nostru care îl conţine 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 şi linia corespunzătoare stării q_{12}, însă observăm că nici ea, dar nici stările q_1 şi q_2 nu apar în AFD.

Verificare acceptare cuvânt de către automat finit

Folosim "configuraţii" (sau „descrieri instantanee") ale automatului, adică perechi formate din starea curentă şi cât a mai rămas de citit din cuvântul de intrare, şi aplicăm funcţia delta cât timp este posibil (ne oprim fie când nu mai există tranziţie 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 şi am ajuns în starea q_{01}, care nu este finală, deci rezultă că automatul nu acceptă acest cuvânt.

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

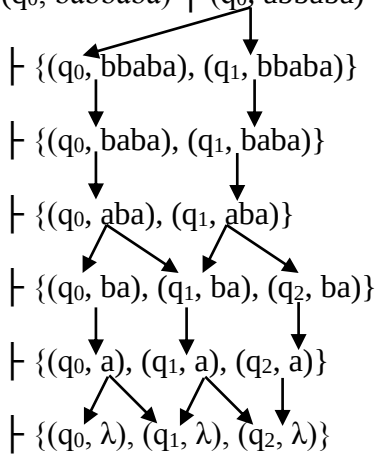
Am terminat de citit cuvântul şi am ajuns în starea q_{012}, care este finală, deci rezultă că automatul acceptă acest cuvânt.

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 şi am avut două drumuri posibile, care ne-au adus în stările q_0 şi q_1. Nici una din ele nu este stare finală, deci cuvântul nu este acceptat.

$(q_0, babbaba) \vdash (q_0, abbaba)$



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

~ Seminar 2 ~

(AFN–λ; Transformarea AFN–λ → AFD; Automatul minimal;
Condiţii asupra numărului de apariţii ale unui caracter într-un cuvânt)

Automat Finit Nedeterminist cu λ–tranziţii

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

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

λ–închiderea unei stări

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

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

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

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

Observăm că mulţimile se pot calcula inductiv după puterea lui λ :
$\{q_{ij} \mid q_{ij} \in \delta(q, \lambda^2)\} = \{q_{ij} \mid q_i \in \delta(q, \lambda^1), q_{ij} \in \delta(q_i, \lambda^1)\}$.
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–λ

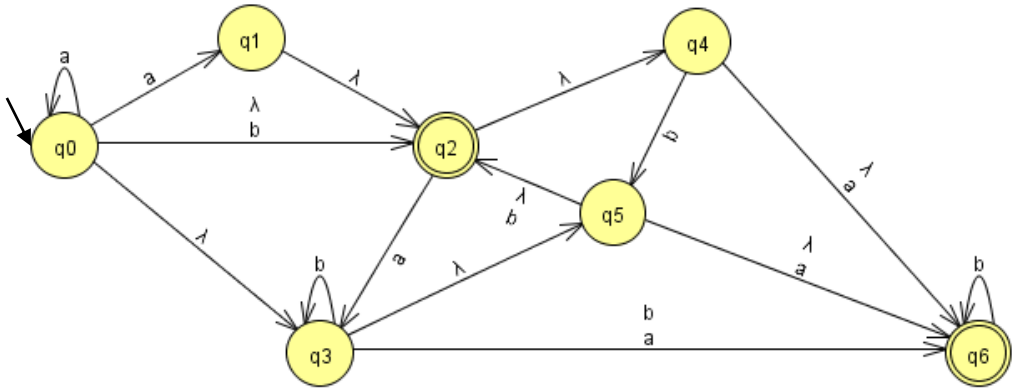
Pentru a verifica dacă un cuvânt este sau nu acceptat de un automat AFN–λ:
Se procedează analog ca în cazul AFN-ului, doar că înainte de a căuta toate stările posibile de continuare cu tranziţii cu caracterul curent, trebuie să facem λ–închiderea mulţimii curente de stări. Iar după ce a fost citit tot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

Obs: λ–închiderea unei mulţimi de stări este egală cu reuniunea λ–închiderilor acelor stări.

$<\{q_{i1}, q_{i2}, ..., q_{in}\}> = <q_{i1}> \cup <q_{i2}> \cup ... \cup <q_{in}>$

EXEMPLU:

Se dă următorul AFN–λ:



Calculăm λ–închiderile tuturor stărilor.

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

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

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

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

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

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

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

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

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

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

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

$\rightarrow <q_1> = \{q_1, q_2, q_4, q_6\}$

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

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

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

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

$\rightarrow <q_2> = \{q_2, q_4, q_6\}$

$\delta(q_3, \lambda^0) = \{q_3\}$
$\delta(q_3, \lambda^1) = \{q_5\}$
$\delta(q_3, \lambda^2) = \{\delta(q_5, \lambda)\} = \{q_2, q_6\}$
$\delta(q_3, \lambda^3) = \{\delta(q_2, \lambda)\} \cup \{\delta(q_6, \lambda)\} = \{q_4\} \cup \varnothing = \{q_4\}$
$\delta(q_3, \lambda^4) = \{\delta(q_4, \lambda)\} = \{q_6\}$ *dar* q_6 *era deja în închidere*
$\rightarrow < q_3 > = \{q_2, q_3, q_4, q_5, q_6\}$

$\delta(q_4, \lambda^0) = \{q_4\}$
$\delta(q_4, \lambda^1) = \{q_6\}$
$\delta(q_4, \lambda^2) = \{\delta(q_6, \lambda)\} = \varnothing$
$\rightarrow < q_4 > = \{q_4, q_6\}$

$\delta(q_5, \lambda^0) = \{q_5\}$
$\delta(q_5, \lambda^1) = \{q_2, q_6\}$
$\delta(q_5, \lambda^2) = \{\delta(q_2, \lambda)\} \cup \{\delta(q_6, \lambda)\} = \{q_4\} \cup \varnothing = \{q_4\}$
$\delta(q_5, \lambda^3) = \{\delta(q_4, \lambda)\} = \{q_6\}$ *dar* q_6 *era deja în închidere*
$\rightarrow < q_5 > = \{q_2, q_4, q_5, q_6\}$

$\delta(q_6, \lambda^0) = \{q_6\}$
$\delta(q_6, \lambda^1) = \varnothing$
$\rightarrow < q_6 > = \{q_6\}$

Verificăm dacă un cuvânt este sau nu acceptat de automatul AFN–λ dat.
pas 0: Pornim de la perechea (q_0, abbaa).

pas 1a: Facem λ–închiderea mulţimii curente de stări pentru a obţine noua mulţime de stări.
$<q_0>$ = {q_0, q_2, q_3, q_4, q_5, q_6}
pas 1b: În fiecare stare din mulţimea curentă încercăm să citim simbolul curent din cuvânt şi obţinem o nouă mulţime de stări.
$\delta(q_0, a) = \{q_0, q_1\}$
$\delta(q_2, a) = \{q_3\}$
$\delta(q_3, a) = \{q_6\}$
$\delta(q_4, a) = \{q_6\}$
$\delta(q_5, a) = \{q_6\}$
$\delta(q_6, a) = \varnothing$
Deci noua mulţime de stări este {q_0, q_1, q_3, q_6}, iar cuvântul rămas este *bbaa*.

pas 2a: Facem λ–închiderea mulţimii curente de stări pentru a obţine noua mulţime de stări.

$< q_0, q_1, q_3, q_6 > = < q_0 > \cup < q_1 > \cup < q_3 > \cup < q_6 >$

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

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

pas 2b: În fiecare stare din mulţimea curentă încercăm să citim simbolul curent din cuvânt şi obţinem o nouă mulţime de stări.

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

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

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

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

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

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

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

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

pas 3a: Facem λ–închiderea mulţimii curente de stări pentru a obţine noua mulţime de stări.

$< q_2, q_3, q_5, q_6 > = < q_2 > \cup < q_3 > \cup < q_5 > \cup < q_6 >$

$= \{q_2, q_4, q_6\} \cup \{q_2, q_3, q_4, q_5, q_6\} \cup \{q_2, q_4, q_5, q_6\} \cup \{6_6\}$

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

pas 3b: În fiecare stare din mulţimea curentă încercăm să citim simbolul curent din cuvânt şi obţinem o nouă mulţime de stări.

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

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

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

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

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

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

pas 4a: Facem λ–închiderea mulţimii curente de stări pentru a obţine noua mulţime de stări.

$< q_2, q_3, q_5, q_6 > = < q_2 > \cup < q_3 > \cup < q_5 > \cup < q_6 >$

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

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

pas 4b: În fiecare stare din mulţimea curentă încercăm să citim simbolul curent din cuvânt şi obţinem o nouă mulţime de stări.

$\delta(q_2, a) = \{q_3\}$
$\delta(q_3, a) = \{q_6\}$
$\delta(q_4, a) = \{q_6\}$
$\delta(q_5, a) = \{q_6\}$
$\delta(q_6, a) = \varnothing$
Deci noua mulţime de stări este $\{q_3, q_6\}$, iar cuvântul rămas este *a*.

pas 5a: Facem λ–închiderea mulţimii curente de stări pentru a obţine noua mulţime de stări.
$< q_3, q_6 > = < q_3 > \cup < q_6 >$
$= \{q_2, q_3, q_4, q_5, q_6\} \cup \{q_6\}$
$= \{q_2, q_3, q_4, q_5, q_6\}$
pas 5b: În fiecare stare din mulţimea curentă încercăm să citim simbolul curent din cuvânt şi obţinem o nouă mulţime de stări.
$\delta(q_2, a) = \{q_3\}$
$\delta(q_3, a) = \{q_6\}$
$\delta(q_4, a) = \{q_6\}$
$\delta(q_5, a) = \{q_6\}$
$\delta(q_6, a) = \varnothing$
Deci noua mulţime de stări este $\{q_3, q_6\}$, iar cuvântul a fost citit în întregime.

pas 6: Facem ultima λ–închidere a mulţimii curente de stări pentru a obţine mulţimea finală de stări.
$< q_3, q_6 > = < q_3 > \cup < q_6 >$
$= \{q_2, q_3, q_4, q_5, q_6\} \cup \{q_6\}$
$= \{q_2, q_3, q_4, q_5, q_6\}$
pas 7: Verificăm dacă printre stările în care am ajuns este vreo una finală.
$\{q_2, q_3, q_4, q_5, q_6\} \cap F = \{q_2, q_3, q_4, q_5, q_6\} \cap \{q_2, q_6\} = \{q_2, q_6\} \neq \varnothing$
Deci cuvântul *abbaa* este acceptat de automatul AFN–λ dat.

Transformarea AFN–λ → AFD

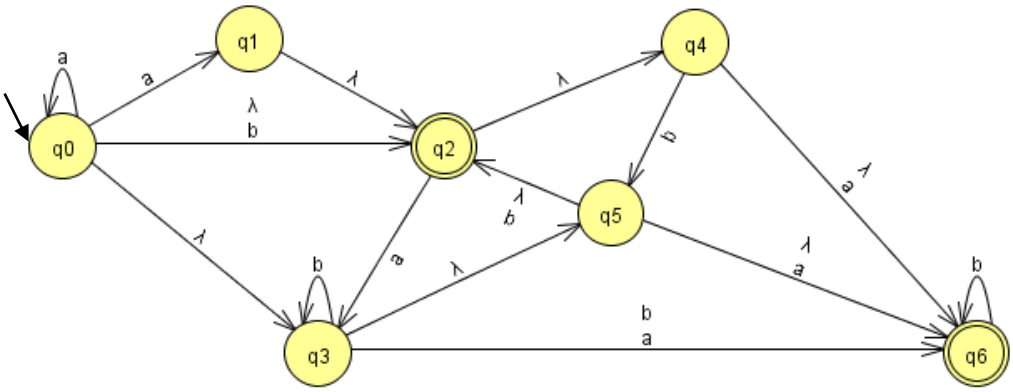
Având dat un AFN–λ (reprezentat sub formă de graf), vom calcula funcţia delta completând un tabel asemănător cu cel de la algoritmul de transformare AFN →AFD.

- La acest algoritm, capetele de coloană vor fi caracterele din alfabet, dar concatenate la stânga şi la dreapta cu λ^*.
- Pentru a calcula primele |Q| linii ale tabelului
  - facem λ–închiderea stării de la capătul de linie curent
  - apoi în toate stările obţinute încercăm să citim caracterul de pe coloana curentă şi obţinem o nouă mulţime de stări
  - facem λ–închiderea acestei mulţimi şi ceea ce obţinem scriem în tabel

- Pentru a calcula următoarele linii (corespunzătoare noilor stări-mulţime care apar), procedăm la fel ca la celălalt algoritm, adică reunim ce găsim în tabel pe aceeaşi coloană şi pe liniile potrivite stărilor componente din starea-mulţime.
- Stările finale ale noului AFD obţinut, vor fi acele stări-mulţime (pot avea şi un singur element, nu neapărat mai multe) care conţin cel puţin una din stările care erau finale pentru AFN-ul iniţial. **În plus**, trebuie să verificăm daca automatul iniţial accepta cuvântul vid λ (adică dacă $< q_0 > \cap F_{AFN-\lambda} \neq \varnothing$). Dacă da, atunci starea iniţială (<q0>) va fi şi stare finală.

EXEMPLU:

Plecăm de la acelaşi automat de mai sus.



Întâi trebuie calculate λ–închiderile tuturor stărilor automatului iniţial.
(Am făcut asta mai sus, deci doar rescriu ce am obţinut.)

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

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

$< q_2 > = \{q_2, q_4, q_6\}$

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

$< q_4 > = \{q_4, q_6\}$

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

$< q_6 > = \{q_6\}$

Avem tabelul pentru primele 7 linii obţinut după cum urmează:

δ	$\lambda^* \cdot a \cdot \lambda^*$	$\lambda^* \cdot b \cdot \lambda^*$
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	$\varnothing$	$\{q_6\}$

$$<\delta(<q_0>,a)> = <\delta(\{q_0,q_2,q_3,q_4,q_5,q_6\},a)>$$
$$= <\{\delta(q_0,a)\}\cup\{\delta(q_2,a)\}\cup\{\delta(q_3,a)\}\cup\{\delta(q_4,a)\}\cup\{\delta(q_5,a)\}\cup\{\delta(q_6,a)\}>$$
$$= <\{q_0,q_1\}\cup\{q_3\}\cup\{q_6\}\cup\{q_6\}\cup\{q_6\}\cup\varnothing> = <\{q_0,q_1,q_3,q_6\}>$$
$$= <q_0>\cup<q_1>\cup<q_3>\cup<q_6>$$
$$= \{q_0,q_2,q_3,q_4,q_5,q_6\}\cup\{q_1,q_2,q_4,q_6\}\cup\{q_2,q_3,q_4,q_5,q_6\}\cup\{q_6\} = \{q_0,q_1,q_2,q_3,q_4,q_5,q_6\}$$

$$<\delta(<q_0>,b)> = <\delta(\{q_0,q_2,q_3,q_4,q_5,q_6\},b)>$$
$$= <\{\delta(q_0,b)\}\cup\{\delta(q_2,b)\}\cup\{\delta(q_3,b)\}\cup\{\delta(q_4,b)\}\cup\{\delta(q_5,b)\}\cup\{\delta(q_6,b)\}>$$
$$= <\{q_2\}\cup\varnothing\cup\{q_3,q_6\}\cup\{q_5\}\cup\{q_2\}\cup\{q_6\}> = <\{q_2,q_3,q_5,q_6\}>$$
$$= <q_2>\cup<q_3>\cup<q_5>\cup<q_6>$$
$$= \{q_2,q_4,q_6\}\cup\{q_2,q_3,q_4,q_5,q_6\}\cup\{q_2,q_4,q_5,q_6\}\cup\{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}$$

$$<\delta(<q_1>,a)> = <\delta(\{q_1,q_2,q_4,q_6\},a)>$$
$$= <\{\delta(q_1,a)\}\cup\{\delta(q_2,a)\}\cup\{\delta(q_4,a)\}\cup\{\delta(q_6,a)\}>$$
$$= <\varnothing\cup\{q_3\}\cup\{q_6\}\cup\varnothing> = <\{q_3,q_6\}>$$
$$= <q_3>\cup<q_6>$$
$$= \{q_2,q_3,q_4,q_5,q_6\}\cup\{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}$$

$$<\delta(<q_1>,b)> = <\delta(\{q_1,q_2,q_4,q_6\},b)>$$
$$= <\{\delta(q_1,b)\}\cup\{\delta(q_2,b)\}\cup\{\delta(q_4,b)\}\cup\{\delta(q_6,b)\}>$$
$$= <\varnothing\cup\varnothing\cup\{q_5\}\cup\{q_6\}> = <\{q_5,q_6\}>$$
$$= <q_5>\cup<q_6>$$
$$= \{q_2,q_4,q_5,q_6\}\cup\{q_6\} = \{q_2,q_4,q_5,q_6\}$$

$$<\delta(<q_2>,a)> = <\delta(\{q_2,q_4,q_6\},a)>$$
$$= <\{\delta(q_2,a)\}\cup\{\delta(q_4,a)\}\cup\{\delta(q_6,a)\}>$$
$$= <\{q_3\}\cup\{q_6\}\cup\varnothing> = <\{q_3,q_6\}>$$
$$= <q_3>\cup<q_6>$$
$$= \{q_2,q_3,q_4,q_5,q_6\}\cup\{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}$$

$$<\delta(<q_2>,b)> = <\delta(\{q_2,q_4,q_6\},b)>$$
$$= <\{\delta(q_2,b)\}\cup\{\delta(q_4,b)\}\cup\{\delta(q_6,b)\}>$$
$$= <\varnothing\cup\{q_5\}\cup\{q_6\}> = <\{q_5,q_6\}>$$
$$= <q_5>\cup<q_6>$$
$$= \{q_2,q_4,q_5,q_6\}\cup\{q_6\} = \{q_2,q_4,q_5,q_6\}$$

$$<\delta(<q_3>,a)> = <\delta(\{q_2,q_3,q_4,q_5,q_6\},a)>$$
$$= <\{\delta(q_2,a)\}\cup\{\delta(q_3,a)\}\cup\{\delta(q_4,a)\}\cup\{\delta(q_5,a)\}\cup\{\delta(q_6,a)\}>$$
$$= <\{q_3\}\cup\{q_6\}\cup\{q_6\}\cup\{q_6\}\cup\varnothing> = <\{q_3,q_6\}>$$
$$= <q_3>\cup<q_6>$$
$$= \{q_2,q_3,q_4,q_5,q_6\}\cup\{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}$$

$$<\delta(<q_3>,b)> = <\delta(\{q_2,q_3,q_4,q_5,q_6\},b)>$$
$$= <\{\delta(q_2,b)\}\cup\{\delta(q_3,b)\}\cup\{\delta(q_4,b)\}\cup\{\delta(q_5,b)\}\cup\{\delta(q_6,b)\}>$$
$$= <\varnothing\cup\{q_3,q_6\}\cup\{q_5\}\cup\{q_2\}\cup\{q_6\}> = <\{q_2,q_3,q_5,q_6\}>$$
$$= <q_2>\cup<q_3>\cup<q_5>\cup<q_6>$$
$$= \{q_2,q_4,q_6\}\cup\{q_2,q_3,q_4,q_5,q_6\}\cup\{q_2,q_4,q_5,q_6\}\cup\{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}$$

$$<\delta(<q_4>,a)> = <\delta(\{q_4,q_6\},a)> = <\{\delta(q_4,a)\}\cup\{\delta(q_6,a)\}>$$
$$= <\{q_6\}\cup\varnothing> = <\{q_6\}> = <q_6> = \{q_6\}$$

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

$$<\delta(<q_5>,a)> = <\delta(\{q_2,q_4,q_5,q_6\},a)>$$
$$= <\{\delta(q_2,a)\}\cup\{\delta(q_4,a)\}\cup\{\delta(q_5,a)\}\cup\{\delta(q_6,a)\}>$$
$$= <\{q_3\}\cup\{q_6\}\cup\{q_6\}\cup\varnothing> = <\{q_3,q_6\}>$$
$$= <q_3>\cup<q_6>$$
$$= \{q_2,q_3,q_4,q_5,q_6\}\cup\{q_6\} = \{q_2,q_3,q_4,q_5,q_6\}$$

$$<\delta(<q_5>,b)> = <\delta(\{q_2,q_4,q_5,q_6\},b)>$$
$$= <\{\delta(q_2,b)\}\cup\{\delta(q_4,b)\}\cup\{\delta(q_5,b)\}\cup\{\delta(q_6,b)\}>$$
$$= <\varnothing\cup\{q_5\}\cup\{q_2\}\cup\{q_6\}> = <\{q_2,q_5,q_6\}>$$
$$= <q_2>\cup<q_5>\cup<q_6>$$
$$= \{q_2,q_4,q_6\}\cup\{q_2,q_4,q_5,q_6\}\cup\{q_6\} = \{q_2,q_4,q_5,q_6\}$$

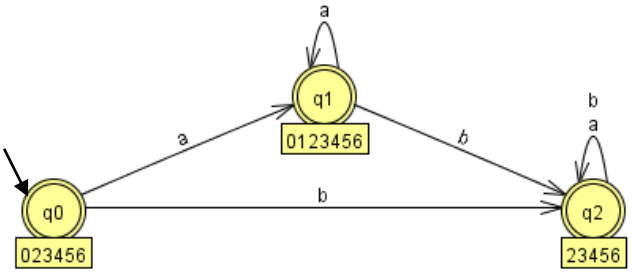
$$<\delta(<q_6>,a)> = <\delta(\{q_6\},a)> = <\{\delta(q_6,a)\}> = <\varnothing> = \varnothing$$

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

Apoi calculăm liniile următoare, corespunzătoare noilor stări-mulţime, pe măsură ce ele apar în automat. Începem cu starea iniţială, $<q_0> = q_{023456}$.

δ	$\lambda^* \cdot a \cdot \lambda^*$	$\lambda^* \cdot b \cdot \lambda^*$
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	$\varnothing$	$\{q_6\}$
q_{023456}	$q_{0123456}$	q_{23456}
$q_{0123456}$	$q_{0123456}$	q_{23456}
q_{23456}	q_{23456}	q_{23456}

Vom obţine în final automatul AFD următor.



Automatul iniţial 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* şi *b* în orice ordine, cuvinte de orice lungime, inclusiv zero.

Obs: Cred că la seminar am uitat să punem şi tranziţia de la q_5 la q_2 cu b, aşa că pot exista mici diferenţe la calculele intermediare, dar rezultatul final este acelaşi.

Automatul minimal

Se porneşte de la un AFD complet definit (adică pentru orice stare din mulţimea Q şi pentru orice caracter din alfabetul Σ, funcţia δ este definită).

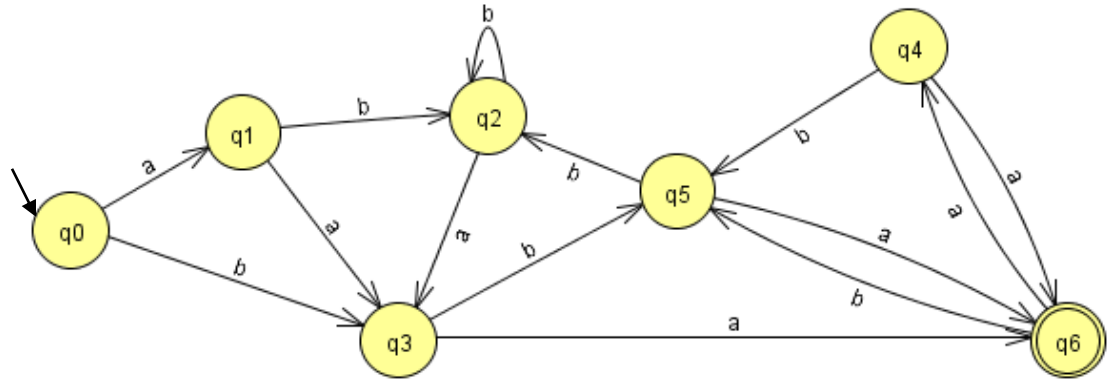
Ideea algoritmului este de a găsi acele stări care au comportament echivalent, pentru a le grupa şi a obţine o unică stare nouă în locul acestora.

- 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 şi 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 şi pe coloane având stările automatului iniţial. 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 obţine cu ajutorul celor de lungime *k*–1 calculate anterior.
Obs: Două stări sunt separabile prin cuvântul vid λ (de lungime zero), dacă una din ele este finală şi cealaltă este nefinală în automatul iniţial.

EXEMPLU:
Fie automatul AFD complet definit:



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

	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	λ	λ	λ	λ	λ	λ	

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 mulţimea {q_0, q_1, q_2} va fi separabilă de orice stare din mulţimea {q_3, q_4, q_5} prin cuvântul "a".
(Observăm că toate tranziţiile 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 şi prin altceva, dar este suficient să găsim un singur cuvânt.

	q_0	q_1	q_2	q_3	q_4	q_5	q_6
q_0							
q_1							
q_2							
q_3	a	a	a				
q_4	a	a	a				
q_5	a	a	a				
q_6	λ	λ	λ	λ	λ	λ	

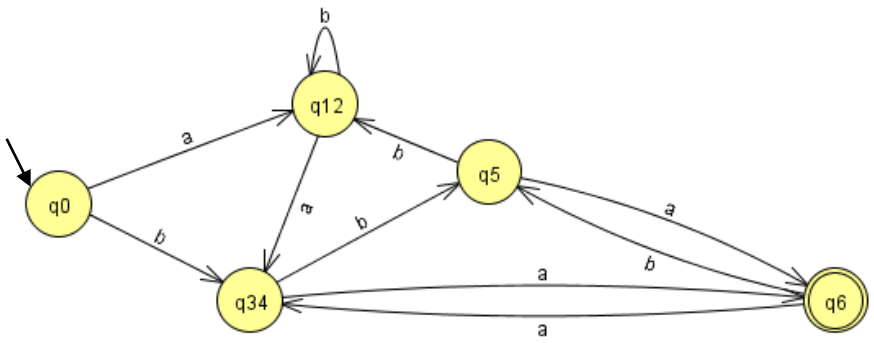
Pas 2: Căutăm cuvinte de lungime 2.

- Perechea (q_1, q_0) cu caracterul "a" ajunge în perechea (q_3, q_1). Dar q_3 şi q_1 erau separabile prin "a". Rezultă că stările q_1 şi q_0 vor fi separabile prin cuvântul "aa" (*obţinut prin concatenarea caracterului încercat şi a fostului cuvânt calculat în tabel*).
- Perechea (q_2, q_0) cu caracterul "a" ajunge în perechea (q_3, q_1). Dar q_3 şi q_1 erau separabile prin "a". Rezultă că stările q_1 şi q_0 vor fi separabile prin cuvântul "aa".
- Perechea (q_2, q_1) cu caracterul "a" ajunge în perechea (q_3, q_3), iar cu caracterul "b" ajunge în perechea (q_2, q_2). *Observăm* că dacă după citirea primului caracter (indiferent dacă este "a" sau "b"), ambele drumuri pornesc din acelaşi loc, nu au cum să se mai ramifice în continuare. Rezultă că aceste două stări, q_2 şi q_1, sunt echivalente.
- Perechea (q_4, q_3) cu caracterul "a" ajunge în perechea (q_6, q_6), iar cu caracterul "b" ajunge în perechea (q_5, q_5). Cu aceeaşi observaţie ca mai devreme, rezultă că stările q_4 şi q_3 sunt echivalente.
- Perechea (q_5, q_3) cu caracterul "a" ajunge în perechea (q_6, q_6), dar cu caracterul "b" ajunge în perechea (q_2, q_5). Perechea (q_2, q_5) era separabilă prin "a". Rezultă, prin concatenare, că stările q_5 şi q_3 sunt separabile prin "ba".
- Perechea (q_5, q_4) cu caracterul "a" ajunge în perechea (q_6, q_6), dar cu caracterul "b" ajunge în perechea (q_2, q_5). Perechea (q_2, q_5) era separabilă prin "a". Rezultă, prin concatenare, că stările q_5 şi q_4 sunt separabile prin "ba".

	q_0	q_1	q_2	q_3	q_4	q_5	q_6
q_0							
q_1	aa						
q_2	aa	∅					
q_3	a	a	a				
q_4	a	a	a	∅			
q_5	a	a	a	ba	ba		
q_6	λ	λ	λ	λ	λ	λ	

Am terminat de completat tabelul şi am obţinut stări echivalente: $q_1 \equiv q_2$ *şi* $q_3 \equiv q_4$.

Automatul AFD minimal obţinut va avea Q = {q_0, q_{12}, q_{34}, q_5, q_6} şi F = {q_6}.
Tranziţiile le desenăm conform automatului iniţial, dar ţinând cont de această grupare a stărilor.



Condiţii asupra numărului de apariţii ale unui caracter într-un cuvânt

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

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

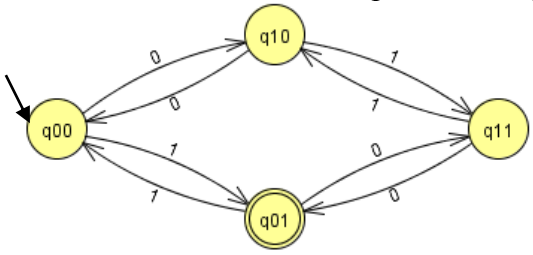
Atenţie! Nu confundaţi exemplele următoare cu cele din primul seminar, în sensul de a încerca să descrieţi limbajele acestea cu expresii asemănătoare cu acelea cu puteri. Acolo existau în plus restricţii asupra poziţiei literelor în cuvânt (de exemplu toate a-urile la început şi apoi toate b-urile). Aici nu contează ordinea literelor, ci doar numărul lor.

EXEMPLE:

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

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

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



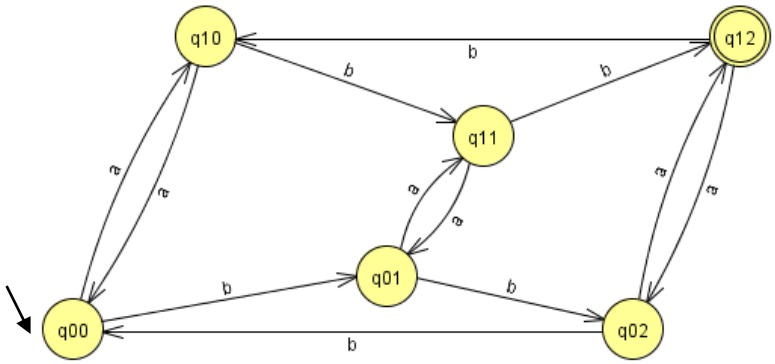
Obs: Pentru că nu există nicio altă restricţie asupra formei cuvântului, automatul va fi *complet definit*, adică în orice stare am fi trebuie să avem tranziţii cu toate caracterele din alfabet.

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

Obs:

- Fiecare stare va avea doi indici pentru că alfabetul are două elemente şi ambele au condiţii ataşate.
- Primul indice, corespunzător caracterului "a", va lua valori în mulţimea resturilor împărţirii la 2, deoarece coeficientul necunoscutei "n" este 2. Al doilea indice, corespunzător caracterului "b", va lua valori în mulţimea resturilor împărţirii la 3, deoarece coeficientul necunoscutei "m" este 3.
- Numărul de stări va fi egal cu produsul numerelor ce reprezintă cardinalele mulţimilor în care iau valori indicii, în cazul acesta 2·3=6.

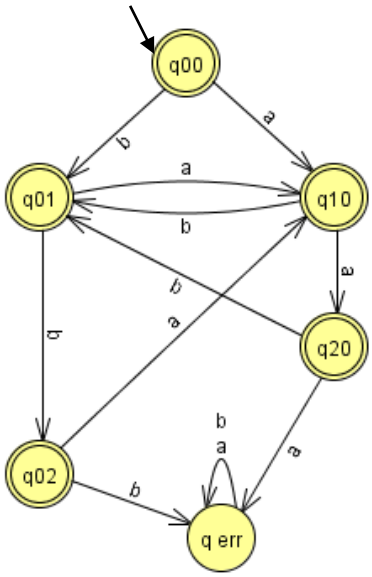
Starea finală va fi q_{12}, deoarece termenii liberi sunt "+ 1" şi "+ 2".

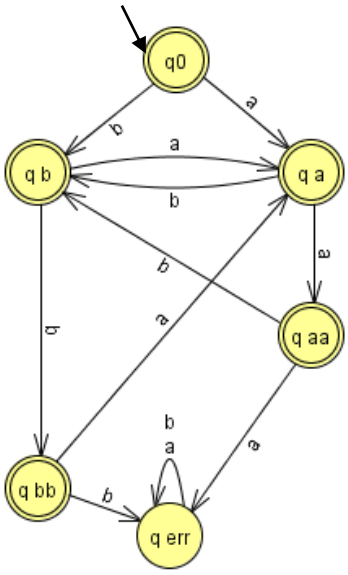


$L_3 = \{w \in \{a,b\}^* \; a.î. \; aaa \; şi \; bbb \; nu \; apar \; în \; w\}$

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

Sunteţi de acord că e adevărat? Ştiţi să explicaţi cum am ajuns la acea expresie ?





~ Seminar 3 ~

(Expresii regulate; Transformarea expresie_regulată → automat_finit
Transformarea automat_finit → expresie_regulată ; Gramatici ; Gramatici regulate)

Expresii regulate

Se numeşte familia expresiilor regulate peste Σ şi se notează REX(Σ) mulţimea de cuvinte peste alfabetul $\Sigma \cup \{(,),+,\cdot,*,\varnothing,\lambda\}$ definită recursiv astfel:

i) $\varnothing, \lambda \in REX$; $a \in REX$ *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) $e \in REX \Rightarrow (e^*) \in REX$

Precedenţa operaţiilor $*, \cdot, +$

Obs: Atenţie la ordinea în care se evaluează operaţiile: întâi stelarea, apoi concatenarea şi apoi reuniunea. *(Dacă vreţi să fiţi siguri că nu le încurcaţi, puteţi să faceţi o analogie cu operaţiile aritmetice, unde se evaluează întâi ridicarea la putere, apoi înmulţirea şi apoi adunarea.)*

Transformarea expresie_regulată → automat_finit

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

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

Obs: La construirea automatelor compuse se ţine cont de precedenţa operaţiilor (întâi stelarea, apoi concatenarea, apoi reuniunea) şi de parantezele din expresie.

Transformarea automat_finit → expresie_regulată

Se numeşte *EFA (automat finit extins),* $M = (Q, \Sigma, et, q_0, F)$, unde, la fel ca la celelalte automate finite, Q este mulţimea stărilor, Σ este alfabetul, q_0 este starea iniţială, F este mulţimea stărilor finale. Aici avem *funcţia de etichetare* $et : Q \times Q \to REX(\Sigma)$.
Notăm $et(p,q)$ prin e_{pq}.

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

(a) Dacă există tranziţii care ajung în starea iniţială, atunci se adaugă la automat o nouă stare care va fi iniţială şi va avea o λ–tranziţie către fosta stare iniţială.
(b) Dacă există mai multe stări finale, atunci se adaugă la automat o nouă stare care va fi singura finală şi va avea λ–tranziţii din toate fostele stări finale către ea.
(c) Se elimină pe rând, una câte una, toate stările în afară de cea iniţială şi cea finală. Presupunem că vrem să eliminăm starea q_k şi că există etichetele $et(q_i, q_k), et(q_k, q_j)$ şi eventual bucla $et(q_k, q_k)$. Atunci obţinem noua etichetă între stările q_i şi 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").
(d) Atunci când rămân doar două stări, se elimină bucla stării finale, dacă există. Concatenând (expresia obţinută între starea iniţială şi cea finală) cu (stelarea expresiei de pe bucla stării finale) obţinem răspunsul final.

În practică, vom calcula etichetele folosind un tabel cu numărul de coloane complete egal cu numărul de stări ale automatului, şi cu numărul de linii egal cu pătratul numărului de stări ale automatului.
Obs: Atenţie, stările automatului vor fi numerotate începând de la q_1 (care va fi starea iniţială).
Notăm cu R^k_{ij} expresia regulată ce se obţine plecând din starea q_i, ajungând în starea q_j şi având ca stări intermediare primele k stări (adică q_1, q_2, ..., q_k).

Tabelul va conţine la capete de linii toate perechile posibile (q_i, q_j) cu i şi j între 1 şi |Q|, iar la capete de coloană toţi k, cu k de la 0 la |Q| – 1.
Tabelul va fi calculat coloană cu coloană, pentru că valorile coloanei curente se calculează în funcţie de valori aflate pe coloana precedentă.
Prima coloană se calculează astfel:

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

Următoarele coloane se calculează astfel:
(coloana curentă k+1 în funcţie de coloana precedentă k)
$R_{i\,j}^{k+1} = R_{i\,j}^{k} \cup R_{i\,k+1}^{k} \cdot (R_{k+1\,k+1}^{k})^{*} \cdot (R_{k+1\,j}^{k})$

Răspunsul final, expresia regulată căutată, va fi obţinută calculând $\bigcup_{f \in F} R_{1\,f}^{|Q|}$, adică reuniunea etichetelor de la starea iniţială la toate stările finale, având ca stări intermediare toate stările automatului.

Câteva formule utile

(A) $e \cdot \varnothing = \varnothing$; $\varnothing \cdot e = \varnothing$ ($\varnothing$ este pentru concatenare cum este 0 pentru înmulţire)
(B) $e \cdot \lambda = e$; $\lambda \cdot e = e$ (λ este pentru concatenare cum este 1 pentru înmulţire)
(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 şi se doreşte obţinerea expresiei regulate care îi corespunde, folosind tabelul descris anterior.

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

$R_{i\,j}^{k+1}$	k+1 = 0	k+1 = 1	k+1 = 2
$r_{ij} = r_{11}$	$1 + \lambda$		
$r_{ij} = r_{12}$	0		
$r_{ij} = r_{21}$	$\varnothing$		
$r_{ij} = r_{22}$	λ		

(a) Calculăm prima coloană, pentru k+1 = 0.
- Dacă indicii i şi j sunt diferiţi, atunci căutăm tranziţiile de la q_i la q_j şi scriem în tabel (pe linia corespunzătoare lui r_{ij}) expresia obţinută prin reuniunea acelor caractere de pe tranziţii.
Deci $R_{12}^{0} = 0$, $R_{21}^{0} = \varnothing$.
- Dacă indicii i şi j sunt egali, căutăm buclele stării q_i=q_j şi scriem în tabel expresia obţinută prin reuniunea acelor caractere de pe bucle şi a caracterului λ.
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,
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) şi avem $(1+\lambda)^+ = (1+\lambda)^* \cdot (1+\lambda)$
Apoi putem aplica formula (D) pentru prima parte şi obţinem $(1^* \cdot \lambda^*)^* \cdot (1+\lambda)$
Dar $\lambda^* = \lambda$ şi conform formulei (B) prima paranteză este egală cu 1^*, deci vom avea $(1^*)^* \cdot (1+\lambda) = 1^* \cdot (1+\lambda)$. Aplicăm formula (E) (si apoi (C) şi (B)) şi obţinem $(1^* \cdot 1) + (1^* \cdot \lambda) = 1^+ + 1^* = \{1^n \mid n \geq 1\} \cup \{1^n \mid n \geq 0\} = \{1^n \mid n \geq 0\} = 1^*$ 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) şi obţinem $0 \cup (1+\lambda)^+ \cdot 0$, care este echivalent după cum am aflat mai sus cu $0 \cup 1^* \cdot 0 = \lambda \cdot 0 \cup 1^* \cdot 0 = \lambda \cdot 0 + 1^* \cdot 0 = (\lambda + 1^*) \cdot 0 = 1^* \cdot 0$

$R_{21}^1 = R_{21}^0 \cup R_{21}^0 \cdot (R_{11}^0)^* \cdot R_{11}^0 = \varnothing \cup \varnothing \cdot (1+\lambda)^* \cdot (1+\lambda)$
Aplicăm formula (A) şi avem $\varnothing \cup \varnothing = \varnothing$

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

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	$1^* \cdot 0$	$1^* \cdot 0$
$r_{ij} = r_{21}$	$\varnothing$	$\varnothing$	
$r_{ij} = r_{22}$	λ	λ	

(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 iniţiala) la starea q_2. *(Dacă aveam mai multe stări finale, calculam pentru fiecare câte o expresie şi apoi le reuneam.)*

$R_{12}^2 = R_{12}^1 \cup R_{12}^1 \cdot (R_{22}^1)^* \cdot R_{22}^1 = (1^* \cdot 0) \cup (1^* \cdot 0) \cdot \lambda^* \cdot \lambda = (1^* \cdot 0) \cup (1^* \cdot 0) = 1^* \cdot 0$
Deci expresia regulată corespunzătoare automatului dat este $1^* \cdot 0$.

Gramatici

G = (N, T, S, P)
N = {A, B, C, …} mulţimea de simboluri neterminale
T = {a, b, c, ...} mulţimea de simboluri terminale
$S \in N$ simbolul de start
P mulţimea de producţii (sau reguli de producţie)

Modul în care este definită funcţia 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 producţii de tipurile $A \to x, \quad cu\ x \in T^*\ şi\ A \in N$
$A \to xBy, \quad cu\ x, y \in T^*\ şi\ A, B \in N$

Gramatici liniare la dreapta

Au producţii de tipurile $A \to x, \quad cu\ x \in T^*\ şi\ A \in N$
$A \to xB, \quad cu\ x \in T^*\ şi\ A, B \in N$

Gramatici liniare la stânga

Au producţii de tipurile $A \to x, \quad cu\ x \in T^*\ şi\ A \in N$
$A \to Bx, \quad cu\ x \in T^*\ şi\ A, B \in N$

Gramatici regulate

O gramatică se numește regulată dacă este liniară la dreapta sau liniară la stânga.
Dar pentru că gramaticile liniare la dreapta şi cele liniare la stânga definesc acelaşi set de limbaje, atunci în general ne referim la cele liniare la dreapta.

Deci gramaticile regulate au producţii de tipurile $A \to \lambda$
$A \to a$
$A \to aB, \quad cu\ a \in T\ şi\ A, B \in N$

Obs: Putem grupa mai multe producţii care au acelaşi membru stâng şi putem scrie pe scurt $A \to \lambda \mid a \mid aB, \quad cu\ a \in T\ şi\ A, B \in N$

EXEMPLE (gramatici regulate):

Vom relua acele exemple din primul seminar pentru care am construit automate finite *(pagina 5 din acest document)*, pentru a vedea corespondenţa strânsă dintre automate finite si gramatici regulate (dintre stări şi neterminale; tranziţii şi producţii).

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

L1 = a^* = {a^n, n ≥ 0} = {λ=a^0, a=a^1, aa=a^2, aaa=a^3, aaaa=a^4, …}

$S \to aS \mid \lambda$

L2 = a^+ = {a^n, n ≥ 1} = {a=a^1, aa=a^2, aaa=a^3, aaaa=a^4, …}

$S \to aS \mid a$

L3 = {a^{2n}, n ≥ 0} = {λ=a^0, aa=a^2, aaaa=a^4, aaaaaa=a^6, …}

$S \to aA \mid \lambda$

$A \to aS$

L4 = {a^{2n+1}, n ≥ 0} = {a=a^1, aaa=a^3, aaaaa=a^5, …}

$S \to aA \mid a$ sau $S \to aA$

$A \to aS$ $A \to aS \mid \lambda$

L5 = {a^{2n}, n ≥ 1} = {aa=a^2, aaaa=a^4, aaaaaa=a^6, …}

$S \to aA$ sau $S \to aA$ sau $S \to aA$

$A \to aS \mid a$ $A \to aB \mid a$ $A \to aB$

$B \to aA$ $B \to aA \mid \lambda$

L6 = {a^{2n+1}, n ≥ 1} = {aaa=a^3, aaaaa=a^5, aaaaaaa=a^7, …}

$S \to aA$ $S \to aA$

$A \to aB$ sau $A \to aS \mid aB$

$B \to aA \mid a$ $B \to a$

L7 = {$a^{2n}b^{3m}$, n ≥ 0, m ≥ 0}

= {λ, a^2, a^4, a^6, a^8, …, b^3, b^6, b^9, b^{12}, …, a^2b^3, a^2b^6, a^4b^3,a^4b^6, a^6b^3, a^6b^6, …}

$S \to aA \mid bB \mid \lambda$ $S \to aA \mid bB \mid \lambda$

$A \to aS$ $A \to aS$

$B \to bC$ sau $B \to bC$

$C \to bD$ $C \to bD \mid b$

$D \to bB \mid \lambda$ $D \to bB$

Analogia între automate finite şi gramatici regulate

i) La limbajele L1 şi L2, unde avem cuvinte de forma a^n:
- la automate avem buclă (adică tranziţie de la o stare către ea însăşi)
- la gramatici avem o producţie în care acelaşi neterminal apare şi în membrul stâng şi în cel drept al producţiei

ii) La limbajele L3 – L10, unde la putere avem şi un coeficient număr natural (2 pentru a-uri şi 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 condiţii asupra numărului de apariţii ale caracterelor în cuvinte ?
- la automate ideea era să folosim indici la stări, câte unul pentru fiecare condiţie
- la gramatici ideea este să folosim indici la neterminale, câte unul pentru fiecare condiţie

~ Seminar 4 ~

(Exemple cu gramatici independente de context GIC ;
Transformarea GIC → 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 conţine cuvinte în care numărul de litere dintr-o parte a cuvântului depinde de numărul de litere din altă parte a cuvântului şi deci avem nevoie să „controlăm" cumva acest număr de litere.

L1 = {$a^n b^n$, n ≥ 0}
= {$\lambda=a^0b^0$, $ab=a^1b^1$, $aabb=a^2b^2$, $aaabbb=a^3b^3$, $aaaabbbb=a^4b^4$, …}

$S \to \overset{(1)}{aSb} \mid \overset{(2)}{\lambda}$ sau $S \to \overset{(1)}{aAb} \mid \overset{(2)}{\lambda}$, $A \to \overset{(3)}{aAb} \mid \overset{(4)}{\lambda}$

Ideea acestei gramatici este de a genera cuvântul dinspre margini spre mijlocui lui, astfel pentru fiecare aplicare a producţiei (1) vor apărea un „a" şi un „b", iar între ele va rămâne S-ul pentru a continua generarea. La final, se aplică producţia (2) pentru a nu mai avea neterminal în cuvântul obţinut.
Cel mai mic cuvând este într-adevăr cuvântul vid, pentru că putem aplica direct a doua producţie fără să o aplicăm deloc pe prima.

L2 = {$a^n b^n$, n ≥ 1}
= {$ab=a^1b^1$, $aabb=a^2b^2$, $aaabbb=a^3b^3$, $aaaabbbb=a^4b^4$, …}

$S \to \overset{(1)}{aSb} \mid \overset{(2)}{ab}$ sau $S \to \overset{(1)}{aAb}$, $A \to \overset{(2)}{aAb} \mid \overset{(3)}{\lambda}$

Aici, cel mai mic cuvânt trebuie să fie „ab", deci observăm că din ambele gramatici de la exemplul anterior au dispărut producţiile „S→λ".
În prima soluţie avem ca pas se oprire cuvântul „ab", iar în a doua soluţie suntem obligaţi să generăm cel puţin un „a" şi un „b" (prin prima producţie) înainte de a ne opri (cu a treia producţie).

L3 = {$a^n b^{2n}$, n ≥ 1}
= {abb=a^1b^2, aabbbb=a^2b^4, aaabbbbbb=a^3b^6, …}

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

Pentru fiecare „a” din prima parte a cuvântului trebuie să avem doi de „b” în a doua parte a cuvântului.
Deoarece avem „n ≥ 1”, pasul de oprire, a doua producţie este cu „abb” în membrul drept.
Dacă era „n ≥ 0”, atunci a doua producţie ar fi avut ca membru drept cuvântul vid.

L4 = {$a^{2n} b^k c^{3n}$, n ≥ 0, k ≥ 1}
= { b=b^1, bb=b^2, bbb=b^3, bbbb=b^4, …,
aabccc=$a^2b^1c^3$, aabbccc=$a^2b^2c^3$, aabbbccc=$a^2b^3c^3$, aabbbbccc=a^2b^4c3, …,
aaaabccccccc=$a^4b^1c^6$, aaaabbcccccc=$a^4b^2c^6$, aaaabbbccccccc=$a^4b^3c^6$, ... }

$$S \xrightarrow{(1)} aaSccc \mid \xrightarrow{(2)} bA$$
$$A \xrightarrow{(3)} bA \mid \xrightarrow{(4)} \lambda$$

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 şi „c”-urile, iar apoi vom trece în alt neterminal cu ajutorul căruia vom genera „b”-urile.
Observăm că avem „n ≥ 0”, deci nu trebuie să-l obligăm să aibă „a”-uri şi „c”-uri, ci trebuie să îl lăsăm să treacă direct la „b”-uri.
Cel mai mic cuvânt este „b”: aplicăm producţia (2) şi apoi pe (4).

L5 = {$a^{2n} b^k c^{3n}$, n ≥ 1, k ≥ 1}
= { aabccc=$a^2b^1c^3$, aabbccc=$a^2b^2c^3$, aabbbccc=$a^2b^3c^3$, aabbbbccc=$a^2b^4c^3$, …,
aaaabccccccc=$a^4b^1c^6$, aaaabbcccccc=$a^4b^2c^6$, aaaabbbccccccc=$a^4b^3c^6$, ... }

$$S \xrightarrow{(1)} aaSccc \mid \xrightarrow{(2)} aaAccc \quad A \xrightarrow{(3)} bA \mid \xrightarrow{(4)} b$$

sau

$$S \xrightarrow{(1)} aaSccc \mid \xrightarrow{(2)} aaAbccc \quad A \xrightarrow{(3)} bA \mid \xrightarrow{(4)} \lambda$$

sau

$$S \xrightarrow{(1)} aaSccc \mid \xrightarrow{(2)} aabAccc \quad A \xrightarrow{(3)} bA \mid \xrightarrow{(4)} \lambda$$

Diferenţa faţă de exemplul anterior este că aici şi n este strict pozitiv, de aceea trebuie ca înainte de a trece la neterminalul A să generăm obligatoriu minim doi de „a” şi trei de „c” (producţia a doua). Apoi în neterminalul A să generăm cel puţin un „b” (de aceea observăm că s-a modificat şi producţia a patra).
Cel mai mic cuvânt este „aabccc”: aplicăm producţia (2) şi apoi (4).

L6 = {$a^{2n}c^{3n}b^{k}$, n ≥ 1, k ≥ 1}
= { aacccb=$a^2c^3b^1$, aacccbb=$a^2c^3b^2$, aacccbbb=$a^2c^3b^3$, aacccbbbb=$a^2c^3b^4$, …,
aaaabccccccb=$a^4c^6b^1$, aaaacccccbb=$a^4c^6b^2$, aaaacccccbbb= $a^4c^6b^3$, ... }

$S \xrightarrow{(1)} AB$

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

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

Observăm că nu întotdeauna putem genera cuvintele dinspre exterior spre centru, având maxim un singur neterminal în membrul drept al producţiilor. Uneori este necesar să generăm cuvintele pe bucăţi, folosind pentru fiecare parte un neterminal separat.
Aici, vom folosi neterminalul A pentru a genera „a”-urile şi „c”-urile spre centru simultan câte doi de „a” şi câte trei de „c”, iar din neterminalul B vom genera „b”-urile.
Dacă am fi avut condiţia „n ≥ 0”, atunci producţia (3) ar fi fost „A→λ”, iar dacă am fi avut condiţia „k ≥ 0”, atunci producţia (5) ar fi fost „B→λ”.

Verificare generare cuvânt de către gramatică

Obs: Atenţie 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 producţii care îl au ca membru stâng, alegem una dintre ele şi obţinem noul cuvânt curent prin înlocuirea neterminalului cu membrul drept al acelei producţii.
- 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 producţiei pe care am aplicat-o pentru a ajunge din nodul-tată în nodul-fiu.

(b) Fie îi reprezentăm de la stânga la dreapta cu aceeaşi convenţie pentru etichetele nodurilor şi muchiilor. În acest caz, ori decidem să urmăm toate ramificaţiile posibile, ori

scriem direct doar drumul care ne duce la rezultat (fiind cuvinte scurte şi gramatici destul de mici, putem „vedea” drumul fară 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 mulţi arbori de derivare distincţi*.
(i) Dacă *gramatica este regulată*, asta poate însemna că în gramatică există unele producţii redundante (care ar putea fi eliminate fără să apară probleme, adică după eliminare să se obţină o gramatică echivalentă cu cea iniţială).
(ii) Iar dacă *gramatica este independentă de context* şi are cel puţin o producţie în al cărei membru drept apar mai multe neterminale, atunci apar mai mulţi arbori pentru un cuvânt care conţine la un moment dat mai multe neterminale, pentru că nu este obligatoriu să încercăm să rescriem doar neterminalul cel mai din stânga/dreapta, ci putem să le alegem în orice ordine.

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

$S \to \overset{(1)}{AB}$

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

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

(a) Aveţi aici (http://bit.ly/arbore_derivare) poza arborelui de derivare pentru a găsi toate drumurile posibile prin care poate fi obţinut cuvântul dat.
În desen, muchiile sunt etichetate cu numărul producţiei care a fost aplicată, iar *nodurile interioare* cu neterminalul care a fost rescris (în interiorul cercului) şi cu noul cuvânt obţinut (în dreptunghiul de sub cerc).
Avem mai multe tipuri de *noduri fără fii*:

- Cele marcate cu „--” reprezintă începutul unor subarbori care mai pot fi extinşi, dar în care nu vom mai găsi soluţii pentru cuvântul dat. Acest lucru se poate întâmpla din două motive: fie am aplicat de prea multe ori una dintre producţiile recursive (2 sau 4), fie am aplicat producţia 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 sfârşitul drumurilor în urma cărora am obţinut 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 şi din structura gramaticii) că producţiile 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 restricţiilor menţionate. De aceea vom obţine cele 10 soluţii: 23445, 24345, 24435, 24453, 42345, 42435, 42453, 44235, 44253, 44523.

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

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

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

Continuăm cu exemplele de gramatici independente de context.

Dacă avem cuvinte formate din 4 porţiuni (a, b, c, d) şi vrem ca lungimile porţiunilor (puterile la care sunt literele) să depindă unele de altele grupate două câte două atunci avem doar două posibilităţi de grupare: *prima cu a patra* (şi a doua cu a treia) sau *prima cu a doua* (şi 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 ≥ 1, k ≥ 0}
= { ad, a^2d^2, a^3d^3, a^4d^4, ..., abc^2d, ab^2c^4d, ab^3c^6d, ..., $a^2bc^2d^2$, $a^2b^2c^4d^2$, $a^2b^3c^6d^2$, ... }

$S \to \overset{(1)}{aSd} \mid \overset{(2)}{aAd}$
$A \to \overset{(3)}{bAcc} \mid \overset{(4)}{\lambda}$

sau

$S \to \overset{(1)}{aSd} \mid \overset{(2)}{ad} \mid \overset{(3)}{aAd}$
$A \to \overset{(4)}{bAcc} \mid \overset{(5)}{\lambda}$

(în a doua soluţie, producţia (2) este redundantă pentru că poate fi obţinută aplicând (3) şi apoi (5))

Aici avem grupate porţiunile 1-4 şi 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.
Atenţie, trebuie să folosim *neterminale diferite*, pentru că altfel s-ar genera şi cuvinte cu literele amestecate („a” şi „b” între ele şi „c” şi „d” între ele).
Avem „k ≥ 0”, deci „b”-urile şi „c”-urile pot lipsi, de aceea avem producţia (4) pe care o aplicăm pentru cazurile când avem k = 0.
Avem „n ≥ 1”, deci producţia (2), pentru că trebuie să ne asigurăm că avem cel puţin un „a” şi un „d” înainte de a trece la neterminalul A pentru a genera mijlocul cuvântului.

Dacă am fi avut „k ≥ 1”, atunci producţia (4) s-ar fi transformat în „A→bcc”.

L8 = {$a^n b^k c^{2k} d^n$, n ≥ 0, k ≥ 1}
= { bc^2, b^2c^4, b^3c^6, ..., abc^2d, ab^2c^4d, ab^3c^6d, ..., $a^2bc^2d^2$, $a^2b^2c^4d^2$, $a^2b^3c^6d^2$, ... }

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

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

Structura cuvintelor este aceeaşi ca la exemplul anterior, s-au schimbat doar condiţiile. Avem „n ≥ 0”, deci „a”-urile şi „d”-urile pot lipsi, dar mijlocul nu. De aceea avem producţia (2) în care folosim direct „bcc”-ul pentru a trece la neterminalul A.
Avem „k ≥ 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 ≥ 0”, înseamnă că şi mijlocul („b”-urile şi „c”-urile) ar fi putut lipsi, deci am fi avut în plus producţia „S→λ”.

L9 = {$a^n b^{2n} c^{2k} d^k$, n ≥ 0, k ≥ 1}
= { c^2d, c^4d^2, c^6d^3, ..., ab^2c^2d, $ab^2c^4d^2$, $ab^2c^6d^3$, ..., $a^2b^4c^2d$, $a^2b^4c^4d^2$, $a^2b^4c^6d^3$, ... }

$S \to \overset{(1)}{AB}$

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

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

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

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

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

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

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

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

(producţia (2) este reduntantă pentru că poate fi obţinută din 1, 4, 6)

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

Pentru „n ≥ 0” avem producţia (4).
Pentru „m ≥ 0” avem producţia (6).
Pentru „k ≥ 0” avem producţia (7) (în care trecem direct cu „d” mai departe fără să fi citit „ceee”) sau producţia (6) (atunci când avem şi m = 0).

Transformarea GIC → gramatică în F.N. Chomsky

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

$A \to a$

$A \to BC$

EXEMPLU:
Avem următoarea gramatică independentă de context:

$S \to \overset{(1)}{aBcDeF} \mid \overset{(2)}{HF} \mid \overset{(3)}{HBc}$

$B \to \overset{(4)}{b} \mid \overset{(5)}{\lambda}$

$D \to \overset{(6)}{d} \mid \overset{(7)}{\lambda}$

$F \to \overset{(8)}{G}$

$G \to \overset{(9)}{f} \mid \overset{(10)}{g}$

$H \to \overset{(11)}{\lambda}$

Pasul 1 – *Eliminarea λ–producţiilor*
Vrem să nu mai avem în gramatică producţii care au ca membru drept cuvântul vid.
Pot exista două situaţii:

(i) Dacă neterminalul care are o λ–producţie NU are şi alte producţii, atunci
- va fi eliminată producţia lui
- toate producţiile care au ca membru drept un cuvânt de lungime minim 2 în care apare acest neterminal → vor fi înlocuite prin eliminarea neterminalului din cuvinte
- dacă membrul drept avea lungime 1 (adică era doar acest neterminal), atunci se înlocuieşte cu λ (dar şi această producţie va fi ulterior eliminată tot la „pasul 1”)

În EXEMPLU:
Neterminalul H este singurul de acest gen, având doar λ–producţia (11), care va fi eliminată. În plus, producţia (2) va fi înlocuită cu „S→F” (12), iar producţia (3) va fi înlocuită cu „S→Bc” (13).

$S \xrightarrow{(1)} aBcDeF \mid \overset{(12)}{F} \mid \overset{(13)}{Bc}$

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

$D \xrightarrow{(6)} d \mid \overset{(7)}{\lambda}$

$F \xrightarrow{(8)} G$

$G \xrightarrow{(9)} f \mid \overset{(10)}{g}$

(ii) Dacă neterminalul care are o λ–producţie are şi alte producţii, atunci
- va fi eliminată doar λ–producţia lui
- toate producţiile care au ca membru drept un cuvânt de lungime minim 2 în care apare acest neterminal → vor fi înlocuite atât de varianta în care cuvântul conţine neterminalul, cât şi de varianta în care neterminalul este eliminat din cuvânt
- dacă membrul drept avea lungime 1 (adică era doar acest neterminal), atunci această producţie este o redenumire şi va fi eliminată la „pasul 2"

În EXEMPLU:
Neterminalele B şi D au mai multe producţii printre care şi câte o λ–producţie. Deci (5) şi (7) vor fi eliminate.
În producţia (13) apare doar B, deci va fi înlocuită cu „S→Bc" (14) şi „S→c" (15).
În producţia (1) apar şi B şi D, deci va fi înlocuită de toate variaţiile posibile:
-- B şi D prezente: „S→aBcDeF" (16)
-- B lipsă, D prezent: „S→acDeF" (17)
-- B prezent, D lipsă: „S→aBceF" (18)
-- B şi D lipsă: „S→aceF" (19)

$S \xrightarrow{(16)} aBcDeF \mid \overset{(17)}{acDeF} \mid \overset{(18)}{aBceF} \mid \overset{(19)}{aceF} \mid \overset{(11)}{F} \mid \overset{(14)}{Bc} \mid \overset{(15)}{c}$

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

Pasul 2 – *Eliminarea redenumirilor*
Vrem să nu mai avem în gramatică producţii care au ca membru drept un neterminal.

Înlocuim neterminalul din dreapta cu toate cuvintele care sunt membru drept în producţiile sale.
Verificăm dacă acest neterminal din dreapta mai apare în dreapta în cadrul altor producţii
- Daca nu, atunci eliminăm toate producţiile 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 şi va fi eliminată tot la „pasul 2"

În EXEMPLU:
Avem două şiruri de redenumiri:
F → G (8) → f (9) | g (10)
Producţia (8) va fi înlocuită cu „F→f " (20) şi „F→g" (21).

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

Producţia (11) va fi înlocuită cu „S→f ” (22) şi „S→g” (23).

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

$$S \to \overset{(16)}{aBcDeF} \mid \overset{(17)}{acDeF} \mid \overset{(18)}{aBceF} \mid \overset{(19)}{aceF} \mid \overset{(22)}{f} \mid \overset{(23)}{g} \mid \overset{(14)}{Bc} \mid \overset{(15)}{c}$$

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

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 componenţa unui cuvânt de lungime minim 2, le vom înlocui cu un neterminal nou şi vom adăuga producţia de la neterminalul nou la terminalul pe care l-a înlocuit.

În EXEMPLU:
Producţiile 22, 23, 15, 4, 6, 20, 21 au deja forma dorită.
Producţia (16) va fi înlocuită cu „S → X_a B X_c D X_e F” (24).
Producţia (17) va fi înlocuită cu „S → X_a X_c D X_e F” (25).
Producţia (18) va fi înlocuită cu „S → X_a B X_c X_e F” (26).
Producţia (19) va fi înlocuită cu „S → X_a X_c X_e F” (27).
Producţia (14) va fi înlocuită cu „S → B X_c” (28).
În plus, vom avea „X_a→a” (29); „X_c→c” (30); „X_e→e” (31).

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

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

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

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 şi îi alipim un neterminal nou, iar noul neterminal va avea o producţie cu membrul drept cuvântul pe care l-a înlocuit. Reluăm procedeul până când toate cuvintele ajung la lungimea 2.

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

În EXEMPLU:
Producţia (24) va deveni „S→X_a Y_1” (32), iar Y_1 ~ B X_c D X_e F.
Apoi „Y_1→B Y_2” (33), iar Y_2 ~ X_c D X_e F.
Apoi „Y_2→X_c Y_3” (34), iar Y_3 ~ D X_e F.
Apoi „Y_3→D Y_4” (35), iar Y_4~ X_e F.
În final „Y_4→ X_e F” (36).
Observăm că putem avea sufixe comune ale cuvintelor şi atunci am putea folosi neterminale deja adăugate anterior.

Producţia (25) va deveni „S→$X_a\,Y_2$” (37), pentru că aveam $Y_2 \sim X_c\,D\,X_e\,F$.
Producţia (26) va deveni „S→$X_a\,Y_5$” (38), iar $Y_5 \sim B\,X_c\,X_e\,F$.
Apoi „Y_5→$B\,Y_6$” (39), iar $Y_6 \sim X_c\,X_e\,F$.
Apoi „Y_6→$X_c\,Y_4$” (40), pentru că aveam $Y_4 \sim X_e\,F$.
Producţia (27) va deveni „S→$X_a\,Y_6$” (41), pentru că aveam $Y_6 \sim X_c\,X_e\,F$.

$$S \to \overset{(32)}{X_a Y_1} \mid \overset{(37)}{X_a Y_2} \mid \overset{(38)}{X_a Y_5} \mid \overset{(41)}{X_a Y_6} \mid \overset{(22)}{f} \mid \overset{(23)}{g} \mid \overset{(28)}{B X_c} \mid \overset{(15)}{c}$$

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

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

$$Y_1 \to \overset{(33)}{B Y_2} \quad ; \quad Y_2 \to \overset{(34)}{X_c Y_3} \quad ; \quad Y_3 \to \overset{(35)}{D Y_4} \quad ; \quad Y_4 \to \overset{(36)}{X_e F} \quad ; \quad Y_5 \to \overset{(39)}{B Y_6} \quad ; \quad Y_6 \to \overset{(40)}{X_c Y_4}$$

Automatul Push-Down (APD) / Automatul Stivă

APD = (Q, Σ, q_0, F, δ, Γ, Z_0)
Q mulţimea de stări
Σ alfabetul de intrare
$q_0 \in Q$ starea iniţială
$F \subseteq Q$ mulţimea de stări finale
Γ alfabetul stivei („gamma”)
$Z_0 \in \Gamma$ simbolul iniţial al stivei
$\delta : Q \times \Sigma \times \Gamma \to Q \times \Gamma^*$ funcţia de tranziţie („delta”)

Modul în care funcţionează tranziţiile pentru un APD
- La intrare avem nevoie de 3 parametri:
  - starea curentă *(element din mulţimea Q)*
  - caracterul curent din cuvântul de intrare *(element din mulţimea Σ)*
  - **simbolul** aflat în vârful stivei *(element din mulţimea Γ)*
- La ieşire vom avea 2 parametri:
  - starea în care ajungem *(element din mulţimea Q)*
  - **cuvântul** cu care înlocuim **simbolul** din vârful stivei *(element din mulţimea Γ^*)*

Atenţie! 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 λ în stivă, înseamnă că doar „am şters” simbolul din vârful stivei fără a adăuga nimic în locul lui.
- Dacă scriem acelaşi 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 şi 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, şi automatele push-down pot fi:
- *„deterministe"* (definite anterior, dar sunt acceptate şi λ–tranziţiile! Dar dacă între două stări avem λ–tranziţie, nu putem avea tranziţie şi cu un caracter din Σ.)
- *„nedeterministe"* (când avem mai multe posibilităţi de continuare: fie ajungând în stări diferite, fie scriind pe stivă cuvinte diferite, fie ambele (ajungând în stări diferite şi scriind pe stivă cuvinte diferite))
- *„nedeterministe cu λ–tranziţii"* (când între două stări avem voie să avem tranziţii şi cu λ şi cu caractere din Σ şi în plus avem nedeterminismul descris anterior)

Modalităţi de acceptare a unui cuvânt de către un APD

(a) cu stări finale

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

(Obs: semnul cu steluţă 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 configuraţia având starea q_0, cuvântul de intrare w şi pe stivă simbolul iniţial Z_0, după ce aplicăm un număr oarecare de tranziţii ajungem în configuraţia 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 (conţinutul stivei): poate să fie vidă sau poate să conţină un cuvânt cu simboluri din alfabetul Γ.

(b) cu vidarea stivei

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

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: Atenţie, dacă ştergeţi tot conţinutul stivei, *automatul îşi opreşte funcţionarea*, pentru că nu poate aplica funcţia delta dacă nu mai are parametrul al treilea (simbolul din vârful stivei). Deci asiguraţi-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 şi să-l eliminăm abia la final (eventual folosind o λ–tranziţie).

(c) cu stări finale şi cu vidarea stivei
În unele cărţi apare şi acest al treilea caz, care este de fapt intersecţia primelor două.

$$L_{F\&\lambda}(APD) = \{w \mid (q_0, w, Z_0) \overset{*}{\vdash\!\!\rightarrow} (\underline{p}, \underline{\lambda}, \underline{\lambda}),\ p \in F\}$$

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 <u>în cuvinte</u> *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 şi 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 funcţia de tranziţie şi menţionăm 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 şi muchiile tranziţii.

Obs: *Avantajul* alegerii variantei (i) ar fi faptul că putem pune ”comentarii” în dreptul rândurilor pentru a urmări mai uşor logica automatului, iar dacă greşim ceva când scriem pe foaie e mult mai uşor de corectat prin tăierea/adăugarea unui rând (mai ales pentru că nu contează ordinea în care sunt scrise rândurile) decât modificarea pe desen.
Totuşi, pentru unele limbaje se poate să vi se pară mai intuitivă varianta cu graful şi atunci o puteţi folosi.

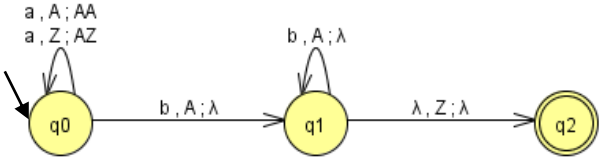
L1 = {$a^n b^n$, n ≥ 1}
Vrem ca numărul de „a”-uri să fie egal cu numărul de „b”-uri, deci vom reţine acest număr cu ajutorul stivei astfel:
- pentru fiecare „a” citit din cuvântul de intrare, vom adăuga un „A” în stivă;
- pentru fiecare „b” citit vom şterge un „A” din stivă;

Trebuie să ne asigurăm că citim cel puţin un „a” şi un „b” (pentru că n ≥ 1) şi 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 iniţial îl găsim pe Z_0 în stivă*
// vrem să scriem A, dar să-l păstrăm şi pe Z_0, deci scriem „AZ_0”
$\delta(q_0, a, A) = (q_0, AA)$ // citim oricare alt a şi avem A în stivă (introdus anterior),
// vrem să scriem A, dar fără să-l ştergem 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 ştergem 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 execuţia şi 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 ştergem.
(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 şi cu vidarea stivei:
Adăugăm $\delta(q_1, \lambda, Z_0) = (q_2, \lambda)$.
Avem $F = \{q_2\}$ stare finală şi stiva vidă.

Aceeaşi soluţie (c), sub formă de graf:



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

L2 = {$a^n b^{2n}$, n ≥ 0}
Pentru fiecare „a” citit, vom adăuga doi de A în stivă (adică AA).
Pentru fiecare „b” citit, vom şterge 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.
Trebui să ne asigurăm că acceptăm şi cuvântul vid (pentru că avem „n ≥ 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"*

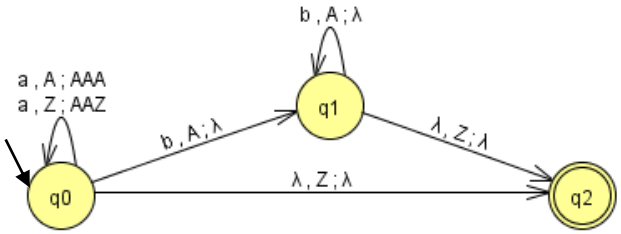
$\delta(q_1, b, A) = (q_1, \lambda)$ // *stergem un A pentru oricare "b"*

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

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

$F = \{q_2\}$

Aceeaşi soluţie, sub formă de graf:



L3 = {$a^{2n} b^k c^{3k} d^n$, n ≥ 1, k ≥ 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 întoarcele se elimină un A făcând o λ–tranziţie)*
Cel mai simplu, pentru pentru fiecare „aa” citiţi, 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 şterge 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 şterge 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, şterge un B

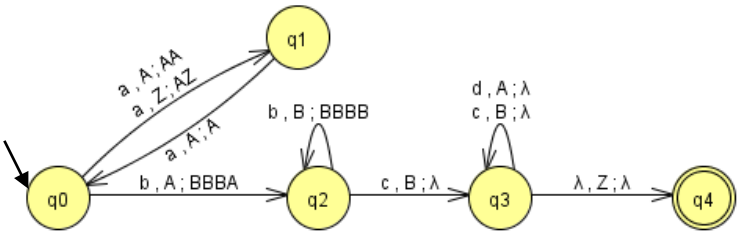
$\delta(q_3, c, B) = (q_3, \lambda)$ // pentru orice alt „c”, şterge un B

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

$\delta(q_3, \lambda, Z_0) = (q_4, \lambda)$ // am terminat cuvântul, stare finală şi stivă vidă
$F = \{q_4\}$

Obs: *Când am început să citim „d”-urile nu a fost neapărat necesar să schimbăm starea faţă 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.*

Aceeaşi soluţie, sub formă de graf:



Verificare acceptare cuvânt de către un APD

Folosim „configuraţii” (sau „descrieri instantanee”) având 3 componente: starea curentă, cât a mai rămas de citit din cuvântul de intrare şi (întreg!) conţinutul stivei (partea stângă fiind vârful stivei, unde se fac ştergerile şi scrierile).

Fie n = 3 şi k = 2, deci avem cuvântul $a^6b^2c^6d^3$.
Scriu de fiecare dată numărul tranziţiei pe care am aplicat-o (a câta era în lista scrisă anterior).
$(q_0, a^6b^2c^6d^3, Z_0) \vdash^{(1)} (q_1, a^5b^2c^6d^3, AZ_0) \vdash^{(2)} (q_0, a^4b^2c^6d^3, AAZ_0)$
$\vdash^{(3)} (q_1, a^3b^2c^6d^3, AAZ_0) \vdash^{(2)} (q_0, a^2b^2c^6d^3, AAAZ_0)$
$\vdash^{(3)} (q_1, ab^2c^6d^3, AAAAZ_0) \vdash^{(2)} (q_0, b^2c^6d^3, AAAAZ_0)$
$\vdash^{(4)} (q_2, bc^6d^3, BBBAAAAZ_0) \vdash^{(5)} (q_2, c^6d^3, B^6A^3Z_0)$
$\vdash^{(6)} (q_3, c^5d^3, B^5A^3Z_0) \vdash^{(7)} (q_3, c^4d^3, B^4A^3Z_0) \vdash^{(7)} (q_3, c^3d^3, B^3A^3Z_0)$
$\vdash^{(7)} (q_3, c^2d^3, B^2A^3Z_0) \vdash^{(7)} (q_3, cd^3, BA^3Z_0) \vdash^{(7)} (q_3, d^3, A^3Z_0)$
$\vdash^{(8)} (q_3, d^2, A^2Z_0) \vdash^{(8)} (q_3, d, AZ_0) \vdash^{(8)} (q_3, \lambda, Z_0)$
$\vdash^{(9)} (q_4, \lambda, \lambda)$, $q_4 \in F$, deci cuvântul este acceptat.

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

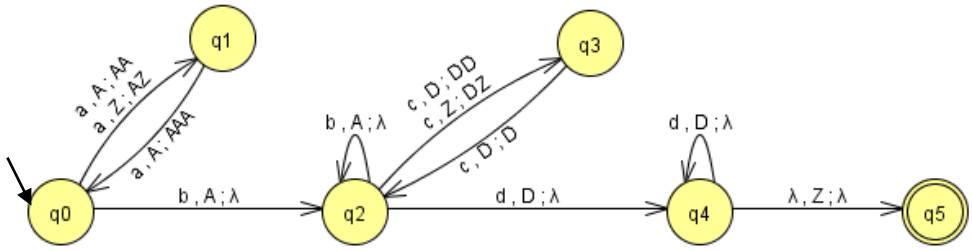
Acum avem grupate primele două părţi şi ultimele două părţi.
Pentru fiecare „aa” trebuie să avem apoi „bbb”. Pentru primul „a” din fiecare pereche vom adăuga A în stivă, iar pentru al doilea „a” din fiecare pereche vom adăuga AA în stivă. Deci pentru fiecare „b” vom şterge câte un A din stivă.
(Obs: 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 toţi A au fost şterşi, 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 şterge un D din stivă.

Atât pentru „a”-uri cât şi 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, ştergem un A
$\delta(q_2, b, A) = (q_2, \lambda)$ // pentru orice alt „b”, ştergem 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, ştergem un D
$\delta(q_4, d, D) = (q_4, \lambda)$ // pentru orice alt „d”, ştergem un D
$\delta(q_4, \lambda, Z_0) = (q_5, \lambda)$ // am terminat cuvântul, stare finală şi 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 şi „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 şi „d” cu D, deci s-ar fi putut amesteca adăugările şi ştergerile D-urilor (adică „c”-urile cu „d”-urile).

Aceeaşi soluţie, sub formă de graf:



~ Seminar 5 ~

(Exemple cu APD (continuare) ;
Algoritmul CYK: Verificare generare cuvânt de către gramatică în F.N.Chomsky)

EXEMPLE (automate push-down) [continuare]:

Notăm cu **w^R** ***cuvântul oglindit*** al lui w. *(R vine de la reverse)*
De exemplu: $(abcde)^R$ = edcba

L5 = { w w^R | w $\in$ {a,b,c}* \ {λ} } ***(palindroame de lungime pară)***
= {aa, bb, cc, abba, acca, baab, bccb, caac, cbbc,
aaaaaa, aabbaa, aaccaa, abaaba, abbbba, abccba, acaaca, acbbca, acccca, ...}

Pentru prima jumătate de cuvânt am dori ca pentru fiecare „a" să adăugăm un A în stivă, pentru fiecare „b" să adăugăm un B, iar pentru fiecare „c" că adăugăm un C. Vrem ca literele să fie amestecate deci trebuie să folosim aceeaşi stare pentru toate citirile.
Pentru a doua jumătate de cuvânt am dori ca pentru fiecare „a" să ştergem un A, pentru fiecare „b" să ştergem un B, iar pentru fiecare „c" să ştergem un C. La fel, folosind aceeaşi stare pentru toate ştergerile.

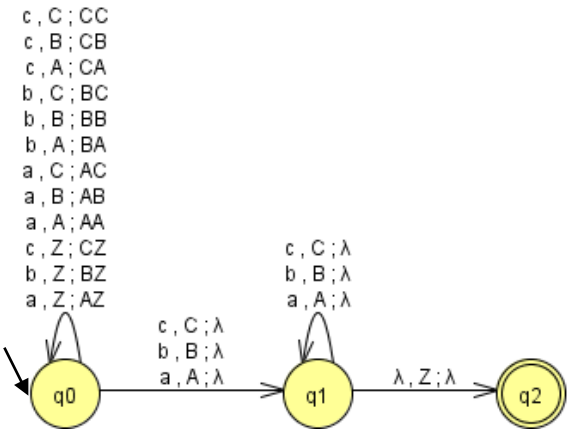
*Dar putem ca **exact** după prima literă din a doua jumătate a cuvântului să schimbăm starea? NU, pentru că nu avem de unde şti unde este jumătatea cuvântului!*
Deci dacă cuvântul conţine undeva „aa", nu vom şti dacă exact acolo este mijlocul (deci ar trebui să trecem în starea pentru ştergeri) sau dacă suntem cumva tot în prima jumătate (deci trebuie să scriem şi pentru al doilea).
De aceea, acesta va fi un ***APD nedeterminist*** !

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$ // cuvântul începe cu „a"
$\delta(q_0, b, Z_0) = (q_0, BZ_0)$ // sau cuvântul începe cu „b"
$\delta(q_0, c, Z_0) = (q_0, CZ_0)$ // sau cuvântul începe cu „c"
$\delta(q_0, a, A) = \{(q_0, AA), (q_1, \lambda)\}$ // fie scriem; fie am ajuns la mijloc, altă stare şi ştergem
$\delta(q_0, a, B) = (q_0, AB)$ // păstrăm B-ul, scriem A
$\delta(q_0, a, C) = (q_0, AC)$ // păstrăm C-ul, scriem A
$\delta(q_0, b, B) = \{(q_0, BB), (q_1, \lambda)\}$ // fie scriem; fie am ajuns la mijloc, altă stare şi ştergem
$\delta(q_0, b, A) = (q_0, BA)$ // păstrăm A-ul, scriem B
$\delta(q_0, b, C) = (q_0, BC)$ // păstrăm C-ul, scriem B
$\delta(q_0, c, C) = \{(q_0, CC), (q_1, \lambda)\}$ // fie scriem; fie am ajuns la mijloc, altă stare şi ştergem
$\delta(q_0, c, A) = (q_0, CA)$ // păstrăm A-ul, scriem C

$\delta(q_0, c, B) = (q_0, CB)$ // păstrăm B-ul, scriem C
$\delta(q_1, a, A) = (q_1, \lambda)$ // ştergem A când citim „a” din a doua jumătate
$\delta(q_1, b, B) = (q_1, \lambda)$ // ştergem B când citim „b” din a doua jumătate
$\delta(q_1, c, C) = (q_1, \lambda)$ // ştergem C când citim „c” din a doua jumătate
$\delta(q_1, \lambda, Z_0) = (q_2, \lambda)$ // gata cuvântul, stare finală, stivă vidă
$F = \{q_2\}$

Înainte de a decide că nu acceptă un cuvânt, APD-ul nedeterminist va încerca toate drumurile posibile.
De exemplu, dacă avem cuvântul „aaaa”.
Plecăm din configuraţia (q_0, aaaa, Z_0), aplicăm tranziţia (1) şi ajungem în (q_0, aaa, AZ_0).
Apoi avem două variante de continuare:
(i) Dacă aplicăm tranziţia (4), ajungem în (q_0, aa, AAZ_0).
Acum avem din nou două variante de continuare:
(i.i) Dacă aplicăm tranziţia (4), ajungem în (q_0, a, $AAAZ_0$).
Din nou, avem cele două variante de continuare:
(i.i.i) Dacă aplicăm tranziţia (4), ajungem în (q_0, λ, $AAAAZ_0$).
De aici, nu avem nicio posibilitate de continuare, deci cuvântul nu va fi acceptat folosind acest drum.
(i.i.ii) Dacă aplicăm tranziţia (5), ajungem în (q_1, λ, AAZ_0).
Am terminat de citit cuvântul, dar avem A în vârful stivei, deci cuvântul nu va fi acceptat folosind acest drum.
(i.ii) Dacă aplicăm tranziţia (5), ajungem în (q_1, a, AZ_0).
Aplicăm tranziţia (16), ajungem în (q_1, λ, Z_0).
Aplicăm (19), ajungem în (q_2, λ, λ) şi *acceptăm* cuvântul.
(ii) Dacă aplicăm tranziţia (5), ajungem în (q_1, aa, Z_0).
De aici, nu avem nicio posibilitate de continuare. Nu am terminat de citit tot cuvântul de intrare, deci nu va fi acceptat folosind acest drum.

Aceeaşi soluţie, sub formă de graf:

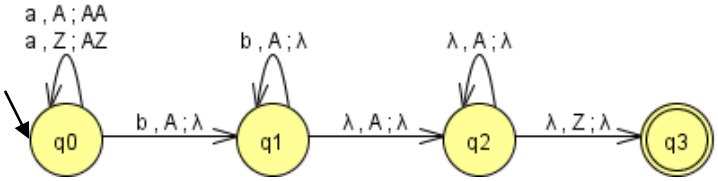


Obs: Până acum am avut doar condiţii cu egalităţi între puteri (sau divizori ai puterilor), dar putem avea şi ***inegalităţi***.

L6 = {$a^n b^k$, n > k ≥ 1}
Pentru fiecare „a” vom adăuga un A în stivă, pentru fiecare „b” vom şterge un A din stivă.
Numărul de „a”-uri trebuie să fie strict mai mare decât numărul de „b”-uri. De aceea, după ce am terminat de citit cuvântul de intrare, trebuie să mai existe cel puţin un A în stivă. Schimbăm starea, ştergem toată stiva şi acceptăm cuvântul.

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$
$\delta(q_0, a, A) = (q_0, AA)$
$\delta(q_0, b, A) = (q_1, \lambda)$
$\delta(q_1, b, A) = (q_1, \lambda)$
$\delta(q_1, \lambda, A) = (q_2, \lambda)$ // obligatoriu mai găseşte cel puţin un A în stivă
$\delta(q_2, \lambda, A) = (q_2, \lambda)$ // altfel nu poate ajunge din q_1 în q_3
$\delta(q_2, \lambda, Z_0) = (q_3, \lambda),\ F = \{q_3\}$

Aceeaşi soluţie, sub formă de graf:



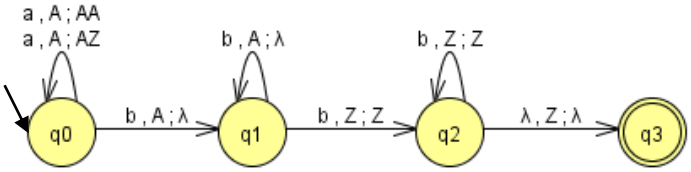
Obs: Dacă am fi avut în schimb condiţia „**n ≥ k ≥ 1**”, am fi avut în plus tranziţia $\delta(q_1, \lambda, Z_0) = (q_3, \lambda)$ pentru cazul de egalitate (n=k).
Pentru condiţia „**n > k ≥ 0**”, am fi avut şi $\delta(q_0, \lambda, A) = (q_2, \lambda)$ (în plus faţă de cele iniţiale), pentru k=0.
Pentru (pentru „**n ≥ k ≥ 0**”) am fi avut $\delta(q_1, \lambda, Z_0) = (q_3, \lambda)$ (pentru n=k) şi $\delta(q_0, \lambda, Z_0) = (q_2, Z_0)$ (pentru k=0).

L7 = {$a^n b^k$, k > n ≥ 1}
Pentru fiecare „a” vom adăuga un A în stivă, pentru fiecare „b” vom şterge un A din stivă.
Numărul de „a”-uri trebuie să fie strict mai mic decât numărul de „b”-uri. De aceea, după ce vom elimina toţi de A din stivă va trebui să mai citim cel puţin un „b” din cuvânt şi să lăsăm stiva nemodificată. Când am terminat de citit cuvântul, schimbăm starea, îl ştergem şi pe Z_0 din stivă şi acceptăm cuvântul.

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$

$\delta(q_0, a, A) = (q_0, AA)$
$\delta(q_0, b, A) = (q_1, \lambda)$
$\delta(q_1, b, A) = (q_1, \lambda)$
$\delta(q_1, b, Z_0) = (q_2, Z_0)$ // obligatoriu mai citeşte cel puţin un „b”
$\delta(q_2, b, Z_0) = (q_2, Z_0)$ // dar nu uităm că trebuie să-l păstrăm pe Z_0 până la final
$\delta(q_2, \lambda, Z_0) = (q_3, \lambda), F = \{q_3\}$

Aceeaşi soluţie, sub formă de graf:

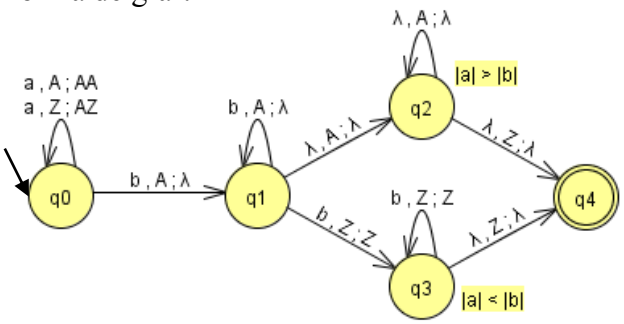


Obs: Dacă era condiţia „k ≥ n ≥ 1” aveam şi $\delta(q_1, \lambda, Z_0) = (q_3, \lambda)$ (pentru k=n).
Dacă era „k > n ≥ 0” aveam şi $\delta(q_0, b, Z_0) = (q_2, Z_0)$ (pentru n=0).
Dacă era „k ≥ n ≥ 0” aveam $\delta(q_1, \lambda, Z_0) = (q_3, \lambda)$ (pentru k=n) şi
$\delta(q_0, \lambda, Z_0) = (q_2, Z_0)$ (pentru n=0).

L8 = {$a^n b^k$, n ≥ 1, k ≥ 1, n ≠ k}
Putem avea fie numărul de „a”-uri strict mai mare decât numărul de „b”-uri fie invers (numărul de „b”-uri strict mai mare decât numărul de „a”-uri).
Pentru fiecare „a” vom adăuga un A, iar pentru fiecare „b” vom şterge un A. Fie ne vor rămâne A-uri în stivă după ce terminăm de citit cuvântul de intrare (dacă n > k), fie vom întâlni Z_0 dar vom mai citi cel puţin un „b” din cuvânt (dacă n < k).

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$
$\delta(q_0, a, A) = (q_0, AA)$
$\delta(q_0, b, A) = (q_1, \lambda)$
$\delta(q_1, b, A) = (q_1, \lambda)$
// n > k
$\delta(q_1, \lambda, A) = (q_2, \lambda)$ // obligatoriu mai găseşte cel puţin un A în stivă
$\delta(q_2, \lambda, A) = (q_2, \lambda)$ // altfel nu poate ajunge din q_1 în q_4
$\delta(q_2, \lambda, Z_0) = (q_4, \lambda)$
// n < k
$\delta(q_1, b, Z_0) = (q_3, Z_0)$ // obligatoriu mai citeşte cel puţin un „b”
$\delta(q_3, b, Z_0) = (q_3, Z_0)$ // dar nu uităm că trebuie să-l păstrăm pe Z_0 până la final
$\delta(q_3, \lambda, Z_0) = (q_4, \lambda)$
$F = \{q_4\}$

Aceeaşi soluţie, sub formă de graf:



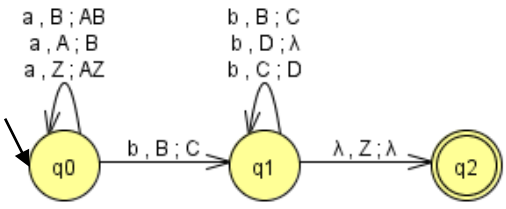
L9 = {$a^{2n} b^{3k}$, n ≥ 1, k ≥ 1, n ≠ k}

Obs: Există şi *o altă modalitate de a avea circuite de o anumită lungime*. În loc să folosim stări diferite, putem folosi simboluri diferite în stivă pe care le înlocuim unul cu altul.

Trebuie să avem grupuri de câte doi „a”. Când îl citim pe primul din grup adăugăm un A în stivă. Când îl citim pe al doilea din grup vom înlocui A-ul cu un B. La final vom avea n de B în stivă.
Trebuie să avem grupuri de câte trei „b”. Când citim primul „b” din grup înlocuim un B cu C în stivă. Când citim al doilea „b” din grup înlocuim C-ul cu un D. Iar când citim al treilea „b” din grup ştergem D-ul din stivă.
Deci avem circuit între A-B-A pentru 2n şi circuit între B-C-D-B pentru 3k.

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$ // pentru primul ”a”, adăugăm A
$\delta(q_0, a, A) = (q_0, B)$ // pentru al 2-lea (4-lea, 6-lea, ...) ”a”, înlocuim A cu B
$\delta(q_0, a, B) = (q_0, AB)$ // pentru al 3-lea (5-lea, 7-lea, ...) ”a”, adăugăm A
// *schimbăm starea când începem citirea ”b”-urilor*
$\delta(q_0, b, B) = (q_1, C)$ // pentru primul ”b”, înlocuim B cu C
$\delta(q_1, b, C) = (q_1, D)$ // pentru al doilea (al 3m+2 -lea) ”b”, înlocuim C cu D
$\delta(q_1, b, D) = (q_1, \lambda)$ // pentru al treilea (al 3m+3 -lea) ”b”, ştergem D-ul
$\delta(q_1, b, B) = (q_1, C)$ // pentru al patrulea (al 3m+1 -lea) ”b”, înlocuim B cu C
$\delta(q_1, \lambda, Z_0) = (q_2, \lambda), F = \{q_2\}$

Aceeaşi soluţie, sub formă de graf:



Algoritmul CYK (verificare generare cuvânt de către gramatică în F.N.Chomsky)

Am văzut că dacă gramatica avea producţii specifice uneia independente de context, atunci foloseam arbori de derivare şi porneam de la simbolul de start S şi încercam să ajungem la cuvântul dat. *(pagina 36 în acest document)*

Dar dacă gramatica are producţiile scrise în forma normală Chomsky, atunci vom folosi acest algoritm CYK, şi vom porni de la cuvântul dat încercând să ajungem la simbolul de start S.

$V_{ij} = \{A \in N \mid \exists B \in V_{ik}, \exists C \in V_{i+k,\, j-k}, (A \to BC) \in P, k \in \{1,2,..., j-1\}\}$

Notăm cu $V_{i\,j}$ mulţimea de neterminale care pot genera acea porţiune din cuvântul de intrare care începe de pe poziţia i şi are lungimea j.

Ideea este de a împărţi cuvântul în două părţi în toate modurile posibile. Dacă prima parte poate fi generată de neterminalul B, a doua parte poate fi generată de neterminalul C şi avem în gramatică producţia "A→BC", atunci înseamnă că întreg cuvântul poate fi generat de neterminalul A.
Observăm că avem împărţiri posibile pentru k=1 (prima literă în prima parte, iar restul în a doua), k=2 (primele două litere în prima parte, iar restul în a doua), ... până la k=j–1 (doar ultima literă în a doua parte, iar restul în prima).

Obs: *Nu reţineţi indicii pe dinafară, ci încercaţi să îi deduceţi logic!*
Valoare pe care o calculăm este $V_{i\,j}$ pentru cuvântul care începe la poziţia i şi are lungimea j.
Prima parte (generată de neterminalul B) conţine primele k litere din cuvânt. Deci va începe la poziţia i (la fel ca întreg cuvântul) şi va avea lungime k, deci $V_{i\,k}$.
A doua parte (generată de neterminalul C) conţine restul de litere, adică j–k, şi începe cu k poziţii mai în dreapta faţă de cuvântul total, adică la i+k, deci avem $V_{i+k,\,j-k}$.

Răspunsul final pe care îl căutăm este $V_{1,\,|w|}$ adică mulţimea de neterminale din care putem genera cuvântul dat începând de pe prima poziţie şi având lungimea egală cu întreg cuvântul. Dacă în această mulţime se va găsi şi simbolul de start S, înseamnă că cuvântul este generat de gramatică. Dacă nu, atunci nu este generat.

La pasul iniţial vom căuta neterminalele care generează fiecare literă mică (terminal) din cuvânt: $V_{i1} = \{A \in N \mid (A \to x_i) \in P\}$, unde x_i este litera de pe poziţia i din cuvântul dat.

EXEMPLU:

Avem o gramatică în F.N.Chomsky: $\begin{aligned} S &\to BA \\ A &\to a \mid BA \mid CA \\ B &\to b \mid AA \mid AB \\ C &\to c \mid BC \mid BB \end{aligned}$ şi un cuvânt: baabca.

Avem un tabel cu i pe coloane şi j pe linii în care vom completa neterminalele din mulţimile $V_{i\,j}$. Variabilele i şi j iau valori între 1 şi 6 (lungimea cuvântului dat). Observăm că putem avea **maxim i + j – 1 = |w|** = 6. Deci de exemplu V_{62} nu va fi calculat (6 + 2 – 1 = 7 > 6) pentru că dacă începem pe poziţia 6 (ultima) nu vom putea avea lungimea 2 în cuvânt.

Pentru prima linie, j = 1, vom căuta în gramatică pentru fiecare literă mică de ce neterminale poate fi generată.
Vom avea $V_{21} = V_{31} = V_{61} = \{A\}$ pentru că litera ”a” se află în cuvânt pe poziţiile 2, 3, 6, iar neterminalul A este singurul care generează ”a”.
Vom avea $V_{11} = V_{41} = \{B\}$ pentru că litera ”b” se află în cuvânt pe poziţiile 1, 4, iar neterminalul B este singurul care generează ”b”.
Vom avea $V_{51} = \{C\}$ pentru că litera ”c” se află în cuvânt pe poziţia 5, iar neterminalul C este singurul care generează ”c”.

$V_{i\,j}$	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6
j = 1	{B}	{A}	{A}	{B}	{C}	{A}
j = 2						
j = 3						
j = 4						
j = 5						
j = 6						

Pentru a doua linie, j = 2, avem cuvinte de lungime 2, deci nu le putem separa decât într-un singur mod, pentru k = 1.
$V_{12} = V_{11} \times V_{21} = \{B\} \times \{A\} = \{BA\}$
În gramatică apare BA ca membru drept pentru neterminalele S şi A, deci în tabel vom completa cu mulţimea {S, A}.
$V_{22} = V_{21} \times V_{31} = \{A\} \times \{A\} = \{AA\}$
Singurul neterminal care se rescrie în AA este B, deci îl trecem în tabel.
$V_{32} = V_{31} \times V_{41} = \{A\} \times \{B\} = \{AB\}$ rezultă B în tabel.
$V_{42} = V_{41} \times V_{51} = \{B\} \times \{C\} = \{BC\}$ rezultă C în tabel.
$V_{52} = V_{51} \times V_{61} = \{C\} \times \{A\} = \{CA\}$ rezultă A în tabel.

Obs: Pentru a calcula orice valoare de pe linia j=2 se face produsul cartezian între mulţimile aflate la nord şi la nord-est de acea valoare.

V_{ij}	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6
j = 1	{B}	{A}	{A}	{B}	{C}	{A}
j = 2	{S, A}	{B}	{B}	{C}	{A}	
j = 3						
j = 4						
j = 5						
j = 6						

Pentru a treia linie, j = 3, avem cuvinte de lungime 3, deci putem avea k=1 sau k=2 şi reunim cele două cazuri.

$V_{13} = V_{11} \times V_{22} \cup V_{12} \times V_{31} = \{B\} \times \{B\} \cup \{S, A\} \times \{A\} = \{BB, SA, AA\}$

Nu avem SA ca membru drept în gramatică, dar avem BB generat de C, iar AA de B, deci în tabel completăm {B, C}.

$V_{23} = V_{21} \times V_{32} \cup V_{22} \times V_{41} = \{A\} \times \{B\} \cup \{B\} \times \{B\} = \{AB, BB\}$ rezultă {B, C} în tabel.

$V_{33} = V_{31} \times V_{42} \cup V_{32} \times V_{51} = \{A\} \times \{C\} \cup \{B\} \times \{C\} = \{AC, BC\}$ rezultă {C} în tabel.

$V_{43} = V_{41} \times V_{52} \cup V_{42} \times V_{61} = \{B\} \times \{A\} \cup \{C\} \times \{A\} = \{BA, CA\}$ rezultă {S, A} în tabel.

V_{ij}	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6
j = 1	{B}	{A}	{A}	{B}	{C}	{A}
j = 2	{S, A}	{B}	{B}	{C}	{A}	
j = 3	{B, C}	{B, C}	{C}	{S, A}		
j = 4						
j = 5						
j = 6						

Pentru a patra linie, j = 4, avem cuvinte de lungime 4, deci putem avea k=1, k=2 sau k=3.

$V_{14} = (V_{11} \times V_{23}) \cup (V_{12} \times V_{32}) \cup (V_{13} \times V_{41})$

$= \{B\} \times \{B, C\} \cup \{S, A\} \times \{B\} \cup \{B, C\} \times \{B\} = \{BB, BC, SB, AB, CB\}$

Nu avem SB sau CB ca membru drept în gramatică, dar pentru restul avem {B,C} în tabel.

$V_{24} = (V_{21} \times V_{33}) \cup (V_{22} \times V_{42}) \cup (V_{23} \times V_{51})$

$= \{A\} \times \{C\} \cup \{B\} \times \{C\} \cup \{B, C\} \times \{C\} = \{AC, BC, CC\}$

Nu avem AC sau CC, dar pentru BC rezultă {C} în tabel.

$V_{34} = (V_{31} \times V_{43}) \cup (V_{32} \times V_{52}) \cup (V_{33} \times V_{61})$

$= \{A\} \times \{S, A\} \cup \{B\} \times \{A\} \cup \{C\} \times \{A\} = \{AS, AA, BA, CA\}$

Nu avem AS, dar pentru celelalte rezultă {S,A,B} în tabel.

V_{ij}	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6
j = 1	{B}	{A}	{A}	{B}	{C}	{A}
j = 2	{S, A}	{B}	{B}	{C}	{A}	
j = 3	{B, C}	{B, C}	{C}	{S, A}		
j = 4	{B, C}	{C}	{S, A, B}			
j = 5						
j = 6						

Pentru a cincea linie, avem cuvinte de lungime j = 5, iar k ia valori între 1 şi 4.

$V_{15} = (V_{11} \times V_{24}) \cup (V_{12} \times V_{33}) \cup (V_{13} \times V_{42}) \cup (V_{14} \times V_{51})$

$= \{B\} \times \{C\} \cup \{S, A\} \times \{C\} \cup \{B, C\} \times \{C\} \cup \{B, C\} \times \{C\} = \{BC, SC, AC, CC\}$

Nu avem SC, AC sau CC, dar pentru BC rezultă {C} în tabel.

$V_{25} = (V_{21} \times V_{34}) \cup (V_{22} \times V_{43}) \cup (V_{23} \times V_{52}) \cup (V_{24} \times V_{61})$

$= \{A\} \times \{S, A, B\} \cup \{B\} \times \{S, A\} \cup \{B, C\} \times \{A\} \cup \{C\} \times \{A\} = \{AS, AA, AB, BS, BA, CA\}$

Nu avem AS, BS dar pentru AA, AB, BA, CA rezultă {S,A,B} în tabel.

V_{ij}	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6
j = 1	{B}	{A}	{A}	{B}	{C}	{A}
j = 2	{S, A}	{B}	{B}	{C}	{A}	
j = 3	{B, C}	{B, C}	{C}	{S, A}		
j = 4	{B, C}	{C}	{S, A, B}			
j = 5	{C}	{S, A, B}				
j = 6						

Pentru linia a şasea, avem cuvântul total de lungime j=6, variabila k ia valori între 1 şi 5.

$V_{16} = (V_{11} \times V_{25}) \cup (V_{12} \times V_{34}) \cup (V_{13} \times V_{43}) \cup (V_{14} \times V_{52}) \cup (V_{15} \times V_{61})$

$= \{B\} \times \{S, A, B\} \cup \{S, A\} \times \{S, A, B\} \cup \{B, C\} \times \{S, A\} \cup \{B, C\} \times \{A\} \cup \{C\} \times \{A\}$

$= \{BS, BA, BB, SS, SA, SB, AS, AA, AB, CS, CA\}$

Pentru BA, BB, AA, AB, CA avem în tabel {S,A,B,C}.

V_{ij}	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6
j = 1	{B}	{A}	{A}	{B}	{C}	{A}
j = 2	{S, A}	{B}	{B}	{C}	{A}	
j = 3	{B, C}	{B, C}	{C}	{S, A}		
j = 4	{B, C}	{C}	{S, A, B}			
j = 5	{C}	{S, A, B}				
j = 6	{S,A,B,C}					

În mulţimea finală obţinută pentru V_{16} există S, deci cuvântul este generat de gramatică.

Obs: Dacă mulţimea finală conţine şi alte elemente, înseamnă că acel cuvânt dat ar putea fi generat şi pornind din acele neterminale (A, B sau C). Dar dacă existau doar ele şi nu exista şi S, acest lucru nu ne ajuta la nimic, cuvântul nu ar fi fost generat de gramatică.

Obs: Unde apare S în tabel înseamnă că gramatica generează şi cuvintele corespunzătoare pentru acele perechi (i, j).

Pentru ij = 12 avem cuvântul "ba": $\underline{S} \Rightarrow \underline{B}A \Rightarrow b\underline{A} \Rightarrow ba$.

Pentru ij = 25 avem "aabca", pentru ij = 34 avem "abca", pentru ij = 43 avem "bca".

~ Seminar 6 ~

(Leme de pompare pentru limbajele regulate)

Lema de pompare pentru limbajele regulate:

Fie L un limbaj regulat. Atunci $\exists n \in \mathrm{N}$ (număr natural) astfel încât pentru orice cuvânt $\alpha \in L$, cu $|\alpha| \geq n$, putem scrie $\alpha = u \cdot v \cdot w$ cu proprietățile:

i) $|u \cdot v| \leq n$

ii) $|v| \geq 1$

iii) $u \cdot v^i \cdot w \in L, \forall i \geq 0$

Vrem să demonstrăm că următoarele limbaje nu sunt regulate.
Schema demonstrației este să presupunem prin absurd că limbajul este regulat și atunci putem aplica lema anterioară. Alegem un cuvânt α din limbajul dat care să respecte ipoteza lemei, adică să aibă lungimea cel puțin n, iar descompunerea sa $\alpha = u \cdot v \cdot w$ să respecte condițiile i) și ii). Apoi alegem convenabil un număr natural i pentru care să obținem o contradicție a condiției iii), adică să rezulte că cuvântul $u \cdot v^i \cdot w \notin L$, și deci presupunerea făcută este falsă.

L1 = {a^k^2, k ≥ 1} = {a^1, a^4, a^9, a^16, a^25, a^36, a^49, a^64, a^81, a^100, ...}

Vrem să demonstrăm că L1 nu este limbaj regulat. Observăm că L1 conține cuvinte formate doar din litera ”a” și având lungimi egale cu pătratele perfecte strict pozitive. Presupunem prin absurd că L1 este limbaj legulat. Atunci $\exists n \in \mathrm{N}$ și putem aplica lema de pompare. Alegem cuvântul $\alpha = a^{n^2}$, cu $|\alpha| = n^2 \geq n$ *(deci lungimea cuvântului respectă ipoteza lemei)*.
Conform lemei, cuvântul poate fi scris sub forma $\alpha = u \cdot v \cdot w$, iar $1 \leq |v| \leq n$ *(cele două inegalități rezultă din condițiile ii) și i) și din faptul că nu se spune că ”u” nu ar putea fi cuvântul vid, deci de lungime 0).*
De asemenea, condiția iii) spune că $u \cdot v^i \cdot w \in L, \forall i \geq 0$. Alegem $i = 2$. Atunci avem cuvântul $u \cdot v^2 \cdot w$ de lungime $|u \cdot v^2 \cdot w| = |u \cdot v \cdot w| + |v| = |\alpha| + |v| = n^2 + |v|$.
La inegalitatea de mai sus adunăm peste tot n^2 și obținem: $n^2 + 1 \leq n^2 + |v| \leq n^2 + n$, adică $n^2 + 1 \leq |u \cdot v^2 \cdot w| \leq n^2 + n$.
Dar $n^2 < n^2 + 1$ și $n^2 + n < (n+1)^2$. Rezultă că $n^2 < |u \cdot v^2 \cdot w| < (n+1)^2$, adică $u \cdot v^2 \cdot w \notin L1$ (pentru că lungimea cuvântului este inclusă strict între două pătrate perfecte alăturate, deci $|u \cdot v^2 \cdot w|$ nu poate fi pătrat perfect).

L2 = {$a^k b^k$, k ≥ 1} = {ab, a^2b^2, a^3b^3, a^4b^4, a^5b^5, ...}

Vrem să demonstrăm că L2 nu este limbaj regulat. Observăm că L2 conține cuvinte formate din k litere de ”a” urmate tot de k litere de ”b”, deci având lungimi numere pare strict pozitive.

Presupunem prin absurd că L2 este limbaj legulat. Atunci $\exists n \in \mathrm{N}$ și putem aplica lema de pompare. Alegem cuvântul $\alpha = a^n b^n$, cu $|\alpha| = 2n \geq n$ *(deci lungimea cuvântului respectă ipoteza lemei)*.

Conform lemei, cuvântul poate fi scris sub forma $\alpha = u \cdot v \cdot w$.

Din condiția i), avem $|u \cdot v| \leq n$. Rezultă că cuvântul $u \cdot v$ conține doar litere de ”a” (pentru că $u \cdot v$ este un prefix al primelor n caractere din α).

Atunci fie $v = a^p$. Din condițiile i) și ii) avem $1 \leq |v| \leq n$. Deci $1 \leq |a^p| \leq n$, adică $1 \leq p \leq n$.

De asemenea, condiția iii) spune că $u \cdot v^i \cdot w \in L, \forall i \geq 0$. Alegem $i = 0$. Atunci avem cuvântul $u \cdot v^0 \cdot w = u \cdot w = a^{n-p} b^n \notin L2$ (pentru că p>0, deci numărul de ”a”-uri este strict mai mic decât numărul de ”b-uri”).