Limbaje Formale și Compilatoare (LFC)
- Curs -

Ș.L.dr.ing. Octavian MACHIDON

octavian.machidon@unitbv.ro

Universitatea
Transilvania
din Brașov

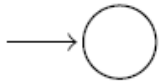
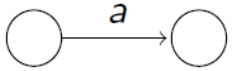
Astăzi

- Automate finite de stari
  - Definiție și reprezentări
  - Automate fiinite deterministe (AFD)
  - Automate finite nedeterministe (AFN)
  - Transformarea AFN în AFD

Automate finite

- Informal, un automat finit constă din:
  - O mulțime finită de stări
  - Tranzițiile între stări
  - O stare inițială
  - O mulțime de stări finale
- Există două tipuri de automate finite:
  - Automate finite deterministe (AFD): tranziția de la fiecare stare este determinată în mod unic de caracterul de intrare curent
  - Automate finite nedeterministe (AFN): pot fi mai multe posibile variante de tranziții, precum și tranziții „spontante” fără vreun caracter de intrare curent

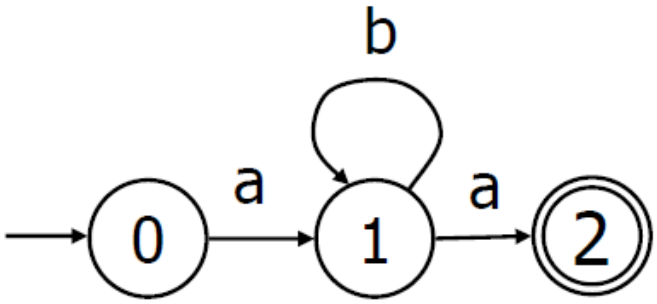
Automate finite deterministe

- Un automat finit determinist e un tuplu cu 5 elemente (cvintuplu) (Σ, S, s_0, δ, F)
  - Σ e un alfabet finit nevid de simboluri de intrare {a, 0, 1, ...}
  - S e o mulțime finită nevidă de stări
  - $s_0 \in S$ e starea inițială 
  - $\delta : S \times \Sigma \rightarrow S$ e funcția de tranziție (pentru fiecare stare și intrare, dă starea următoare) 

  - $F \subseteq S$ e mulțimea stărilor acceptoare (unde dorim să ajungem) 
- Automat finit determinist: o mulțime de *stări* (unele acceptoare), o *stare inițială*, și *tranziții* în funcție de *simbolurile* de intrare.

Exemplu de AFD

- Automat finit care acceptă cuvintele limbajului de forma **ab^na, $n \geq 0$**
- Descris de:
  - Graf

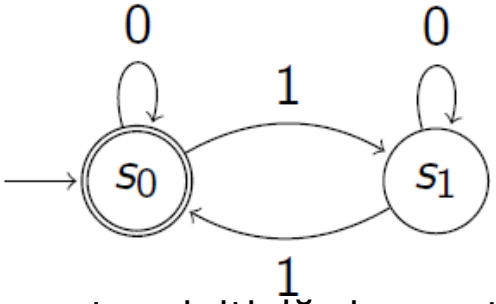


  - Tabel de tranziții

	a	b
0	1	Error
1	2	1
2	Error	Error

Alte exemple de AFD

- automat de paritate: acceptă șiruri de 0 și 1 cu număr par de 1

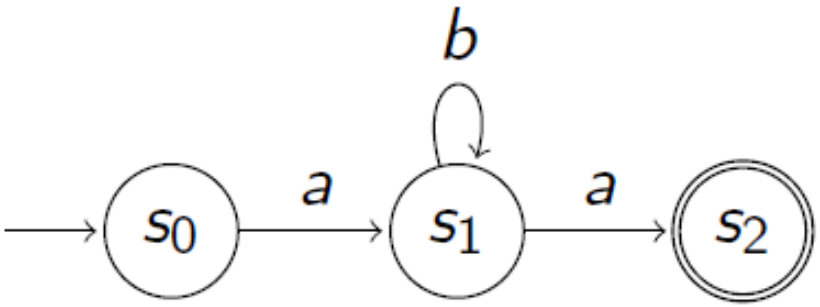


sau ca tabelă de tranziții

	0	1
s_0	s_0	s_1
s_1	s_1	s_0

  - s_0 e stare inițială și acceptoare în același timp

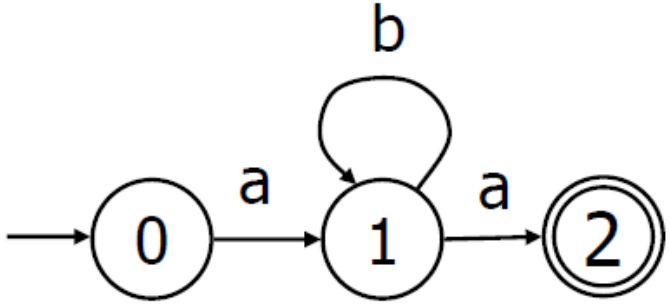
- automat care acceptă cuvinte cu oricâți de *b* (incl. 0) între doi a



„Simularea ” unui AFD

- Algoritm care determină dacă un AFD acceptă un cuvânt de intrare

```
trans_table[NSTATES][NCHARS]
accept_states[NSTATES]
state = INITIAL
while (state != Error) {
      c = input.read();
      if (c == EOF) break;
      state = trans_table[state][c];
}
return (state!=Error) && accept_states[state];
```

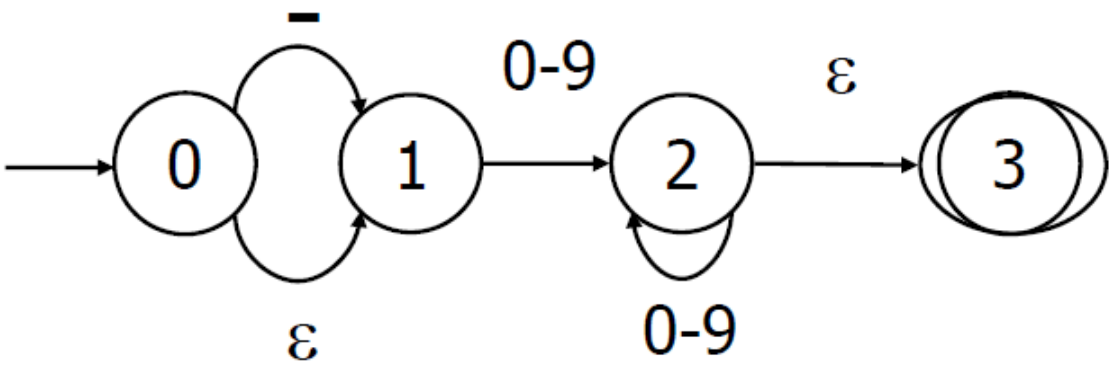


„Simularea ” unui AFN

- Problema: cum se execută un AFN?
- „cuvintele acceptate sunt acelea pentru care există o cale corespunzătoare de la starea de start la o stare acceptată”
- Soluție: se caută în paralel toate căile posibile în graf care corespund cuvântului
  - Se ține evidența sub-mulțimii de stări ale AFN-ului în care ar duce căutarea după parcurgerea prefixului cuvântului

Exemplu

- Șir de caractere de intrare: -23
- Posibile „parcurgeri” ale AFN-ului:
  - {0,1}
  - {1}
  - {0,1,2}
  - {0,1,2,3}

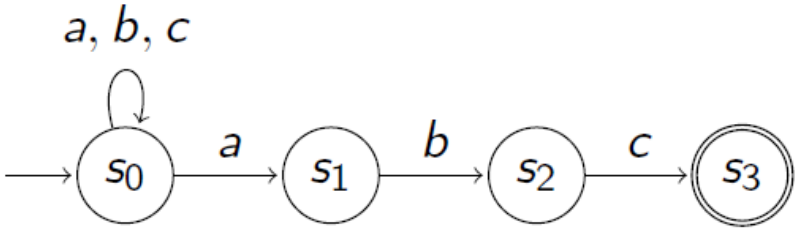


Limbajul acceptat de un automat

- Notăm $\varepsilon \in \Sigma^*$ cuvântul *vid* (fără niciun simbol).
- Definim inductiv o funcție de tranziție δ^* cu intrări *cuvinte*:
  - în ce stare ajunge automatul pentru un cuvânt dat la intrare?
- Pentru orice stare $s \in S$:
  - $\delta^*(s, \varepsilon) = s$; cuvânt vid: nu face nimic
  - $\delta^*(s, a_1 a_2 \ldots a_n) = \delta^*(\delta(s, a_1), a_2 \ldots a_n)$ pentru $n > 0$
- Altfel spus, $\delta^*(s_0, a_1 a_2 \ldots a_n) = (s_1, a_2 \ldots a_n)$ cu $s_1 = \delta(s_0, a_1)$
  - obținem starea s_1 după intrarea a_1, și aplicăm δ^* pe șirul rămas
- Automatul *acceptă* cuvântul $w \in \Sigma^*$ dacă și numai dacă $\delta^*(s_0, w) \in F$
  - (cuvântul *duce* automatul într-o stare *acceptoare*)

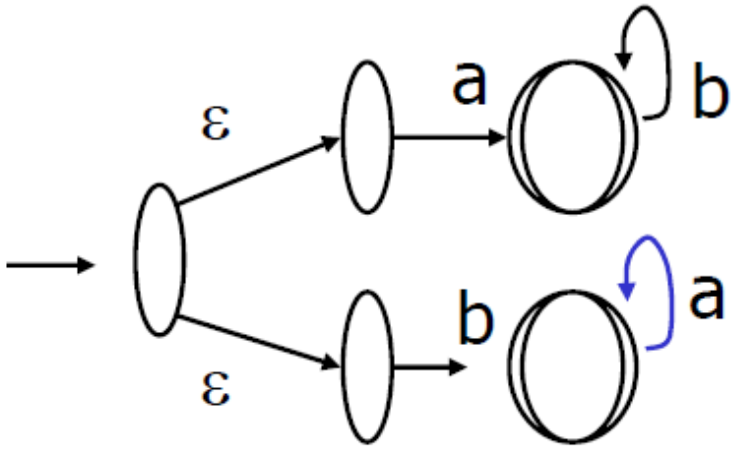
Automat finit nedeterminist

- Exemplu: toate șirurile de *a, b, c* care se *termină* în *abc*
- Din s_0, primind *a*, automatul poate
  - încerca să vadă dacă vor mai fi exact 3 simboluri (trece în s_1)
  - rămâne în s_0 (prespunând că urmează mai mult de 3)
- ***Automatul poate urma una din mai multe căi***
- Un AFN acceptă un cuvânt dacă *există* o alegere de stări succesive ducând într-o stare acceptoare.
- Funcția de tranziție e acum: $\delta: S \times \Sigma \rightarrow \mathcal{P}(S)$
  - dă o *mulțime de stări în care poate trece automatul*
- Avantaje:
  - uneori se scrie mai ușor (permite să „ghicim” tranziția bună)
  - când *specificăm* un sistem, permite o alegere la implementare



Automat finit nedeterminist

- Un AFN este deci un automat unde:
  - Putem aveam mai multe ε-tranziții (tranziții care nu presupun caractere de intrare)
- Pot fi mai tranziții multiple de la aceeași stare cu același caracter



Automate finite

- AF:
  - Stări și tranzițiile între stări
  - Stare inițială și mulțime de stări finale
- AFD: Automat Finit Determinist
  - Fiecare tranziție consumă un caracter de intrare
  - Fiecare tranziție este determinată în mod unic de caracterul de intrare
- AFN: Automat finit nedeterminist
  - Permite ε-tranziții, care nu consumă caractere de intare
  - Pot exista mai multe tranziții din aceași stare prin același caracter de intrare
- Problemă de implementare:
  - AFN este „ambiguu”, deci Compilatoarele folosesc AFD-uri
  - AFN-urile sunt mai simplu de specificat/proiectat
  - **Este nevoie de conversia de la AFN la AFD**

Conversie AFN→ AFD

Orice AFN poate fi transformat, prin aplicarea unui algoritm simplu, într-un AFD echivalent, adică un AFD care recunoaşte acelaşi limbaj ca AFN-ul iniţial.

Exemplul 1:

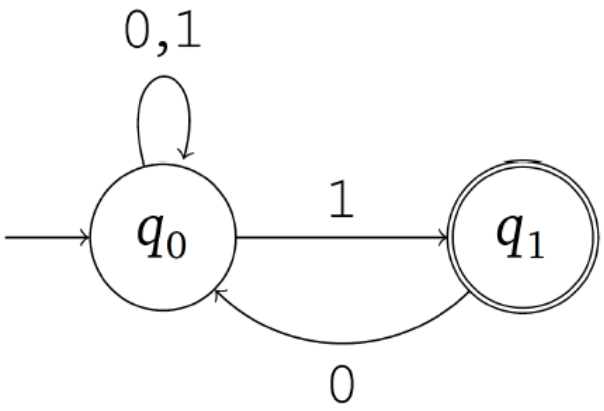


Figura 3-6: Un AFN care acceptă limbajul şirurilor ce se termină cu simbolul 1.

În Figura 3-6 de mai sus este prezentat un AFN care acceptă limbajul şirurilor de 0 şi 1 care se termină în 1. Obiectivul este să se obţină AFD-ul echivalent. Pentru acest lucru, primul pas îl reprezintă reprezentarea AFN-ului sub forma tabelului de tranziţii.

Conversie AFN→ AFD

Tabelul 3-2: . Funcţia de tranziţie al AFN-ului din Figura 3-6.

δ	0	1
q_0	q_0	$\{q_0, q_1\}$
q_1	q_0	-

Fiind vorba despre un AFN, se observă faptul că există în tabel mulţimi de stări în care se ajunge cu acelaşi simbol de intrare din aceeaşi stare. Pasul următor al algoritmului implică adăugarea la acest tabel de noi linii care reprezintă tranziţiile pentru aceste mulţimi de stări.

În acest exemplu, se va adăuga o nouă stare la tabel, $\{q_0, q_1\}$, din care se ajunge cu 0 în q_0 iar cu 1 în aceaşi stare $\{q_0, q_1\}$. Algoritmul continuă până când au fost adăugate toate mulţimile de stări apărute pe parcurs. Noul tabel astfel obţinut este tabelul de tranziţii al AFD-ului rezultat.

Tabelul 3-3: Funcţia de tranziţie al AFD-ului echivalent AFN-ului din Figura 3-6.

δ'	0	1
q_0	q_0	$\{q_0, q_1\}$
q_1	q_0	-
$\{q_0, q_1\}$	q_0	$\{q_0, q_1\}$

Conversie AFN→ AFD

Pe baza noului tabel de tranziţii poate fi reprezentat AFD-ul şi sub forma de graf, cum este afişat în Figura 3-7. în noul AFD echivalent, starea iniţială rămâne cea din AFN, iar stările finale sunt atât cele din AFN, cât şi orice stare nou apărută (o mulţime de stări din AFN) care are în componenţă măcar o stare finală din AFN.

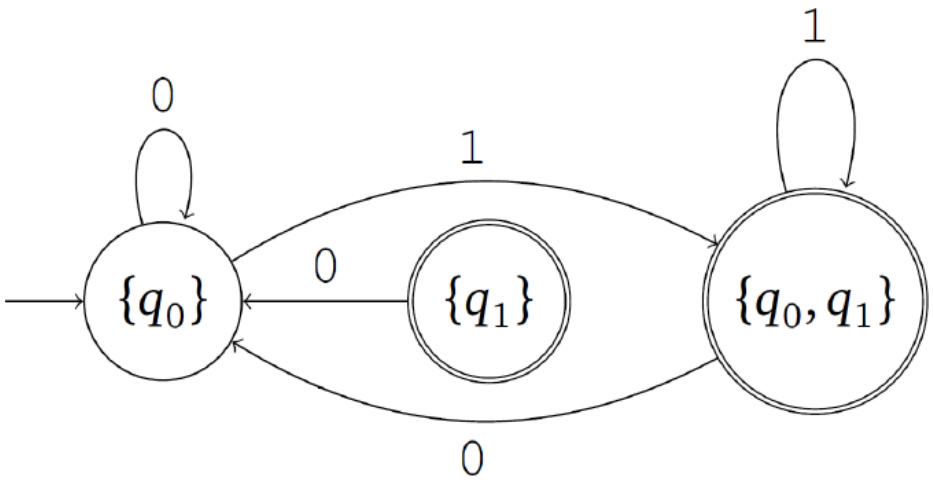


Figura 3-7: AFD-ul echivalent AFN-ului din Figura 3-6.

Conversie AFN→ AFD

La o analiză a graf-ului AFD-ului echivalent rezultat, se observă faptul că starea q_1, deşi stare finală este inaccesibilă din moment ce nu există nicio tranziţie dintr-o altă stare către q_1. Astfel, AFD-ul poate fi minimizat prin reducerea stărilor inaccesibile. În acest caz se obţine automatul din Figura 3-8.

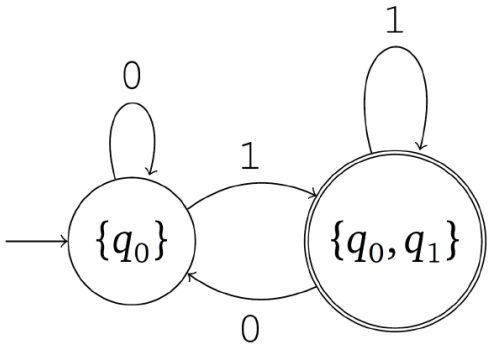


Figura 3-8: AFD-ul minimizat.

Acesta a fost un exemplu minimalist. În practică pot exista şi situaţii în care un AFN ce se doreşte convertit în AFD prezintă şi ε–tranziţii. În acel caz, pentru stabilirea tabelului de tranziţie trebuie ţinut cont şi de acest fapt, întrucât automatul poate ajunge într-o stare şi fără a consuma un caracter de intrare. Astfel, se introduce mulţimea de stări $E(R)$, care se numeşte extensia mulţimii de stări R şi conţine toate stările în care se poate ajunge din stările lui R prin oricâte ε-tranziţii (inclusiv prin niciuna).

Conversie AFN→ AFD

Fie AFN-ul din Figura 3-9. Pentru acest automat, extensiile stărilor sunt prezentate mai jos:

$$E(\{q_0\}) = \{q_0\}$$
$$E(\{q_1\}) = \{q_0, q_1\}$$
$$E(\{q_2\}) = \{q_0, q_1, q_2\}$$
$$E(\{q_2, q_3\}) = \{q_0, q_1, q_2, q_3\}$$

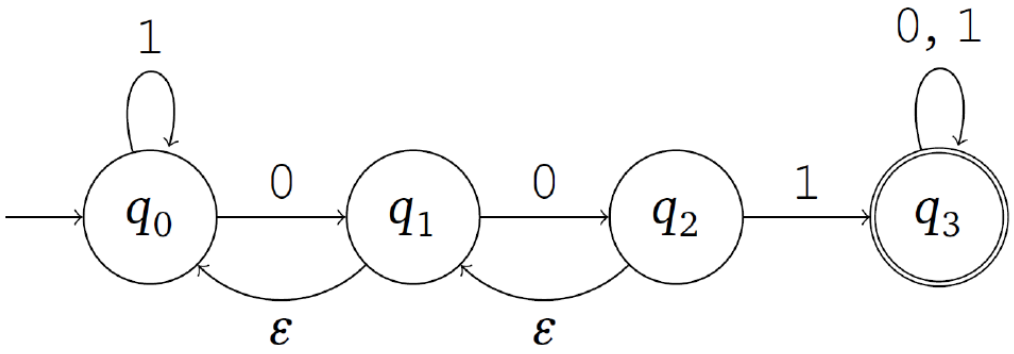
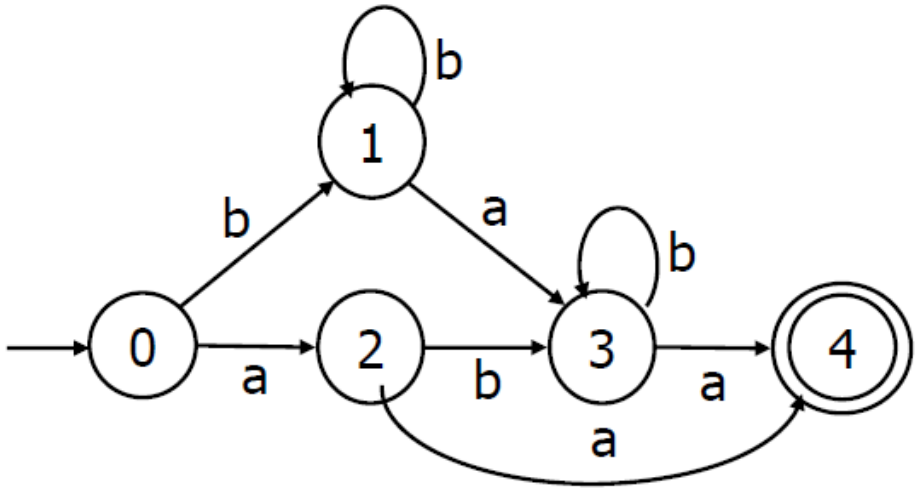


Figura 3-9: AFN ce recunoaşte limbajul şirurilor care conţin subşirul 001.

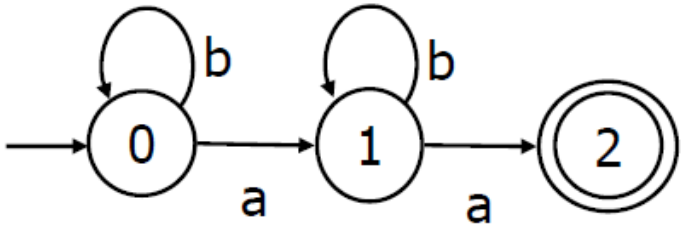
- Exemplul de mai sus va fi dezvoltat și explicat pe larg la seminar

Minimizarea stărilor AFD-ului

- Exemplu:
  - AFD1



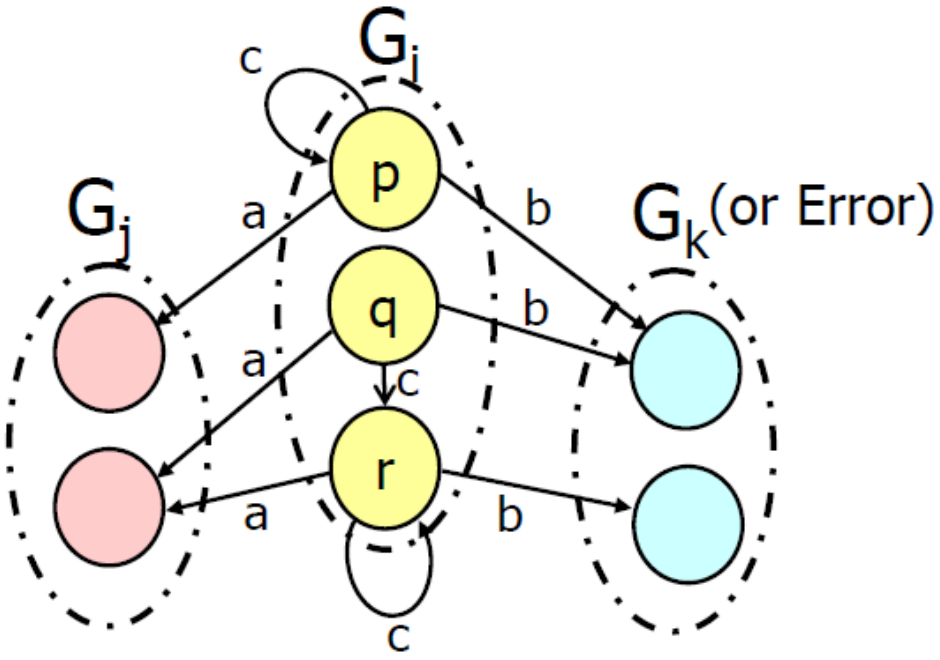
  - AFD2



  - Ambele acceptă $b^n a b^n a$, $n \geq 0$

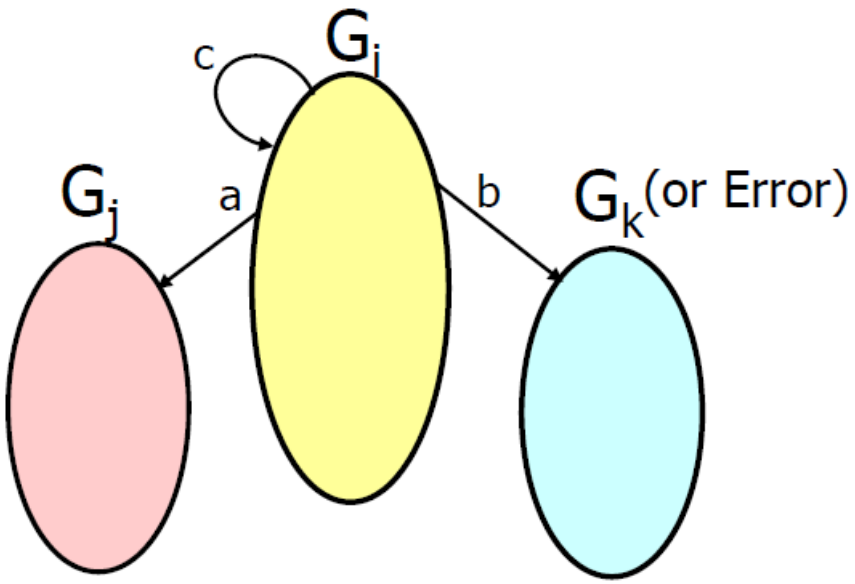
Minimizarea AFD – echivalența

- Toate stările din mulțimea G_i sunt echivalente dacă pentru oricare două stări p și q din G_i , și pentru orice simbol σ, tranziția (p,σ) și tranziția (q,σ) sunt ambele fie eronate, fie conduc la stări din același mulțime G_j (sau chiar



Minimizarea AFD – echivalența

- Toate stările din mulțimea G_i sunt echivalente dacă pentru oricare două stări p și q din G_i , și pentru orice simbol σ, tranziția (p,σ) și tranziția (q,σ) sunt ambele fie eronate, fie conduc la stări din același mulțime G_j (sau chiar G_i).



Cursul viitor:

- Expresii regulate
  - Definiție
  - Proprietățile expresiilor regulate. Exemple de expresii regulate
  - Conversia unei expresii regulate într-un AFD
  - Conversia unui AFD într-o expresie regulată

Întrebări?