Automate si limbaje formale

Pentru a modela hardware-ul unui calculator (a simula functionarea lui) se introduce notiunea de **automat**. Automatul primeste niste date de intrare, este caracterizat printr-un set de decizii care conduc la datele de iesire. Poate dispune sau nu de un spatiu temporar de stocare.

Limbajul formal reprezinta o abstractizare a caracteristicilor generale ale limbajelor de programare. Un limbaj formal este dat de un set de simboluri si o multime de reguli care arata modul in care simbolurile se pot combina pentru a forma asa numitele **propozitii**. Limbajul formal este alcatuit din toate string-urile ce se pot forma pe baza regulilor date.

I. Automate finite

I.1. Automate finite deterministe

Definitia 1.1: $A = (Q , \sum , \delta , q_0 , F)$ se numeste **automat finit determinist**, unde:

Q este o multime finita denumita **multime de stari**
$\sum$ este multime de simbolurilor (**alfabetul** automatului)
δ: Q x $\sum \rightarrow$ Q **functie de tranzitie**
$q_0 \in$ Q este **starea initiala**
$F \subseteq$ Q este **multimea starilor finale**

Presupunem ca avem un sir de caractere ósô (de intrare). Un automat determinist functioneaza in felul urmator:

1. Automatul se pozitioneaza la inceputul sirului ósô (pe pozitia primului caracter) si intra in starea initiala q_0. Se trece la pasul 2.
2. Daca nu s-a ajuns la sfarsitul sirului s, atunci se trece la pasul 3, altfel se trece la pasul 4.
3. Automatul citeste caracterul curent ócô din string-ul ósô. Fie $q_j \in Q$ astfel incat $\delta(q_i,c) = q_j$ (q_i starea curenta). Se trece din starea curenta $q_i \in Q$ intr-o stare $q_j \in Q$, se avanseaza un caracter in sirul ósô si se reia pasul 2.
4. Daca starea curenta $q_i \in Q$ este stare finala ($q_i \in F$), se spune ca automatul A a acceptat sirul ósô, in caz contrar automatul nu recunoaste (nu accepta) sirul ósô.

Pentru automate se foloseste o reprezentare grafica printr-un graf orientat notat G_A = (V, E) denumit **graf de tranzitie**, in care nodurile (multimea V) este data de starile automatului (V = Q), iar arcele (multimea E) sunt date de tranzitiile automatului. Fiecare arc este etichetat cu simbolul prin care se trece din starea corespunzatoare nodului capat

initial al arcului in starea corespunzatoare capatului final al arcului. Cu alte cuvinte, tranzitiei $\delta(q_i,c) = q_j$ ii corespunde in graf de tranzitie arcul (q_i, q_j) etichetat cu simbolul ćcô

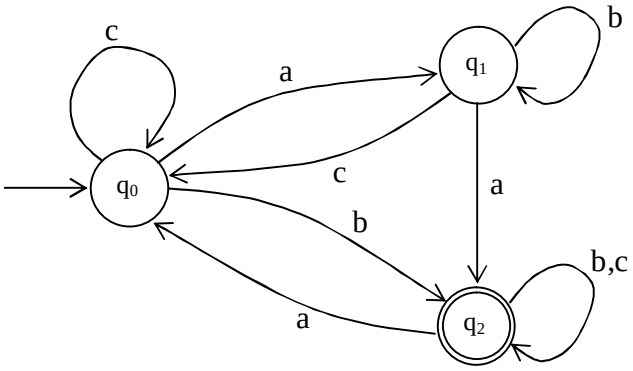
Consideram spre exemplificare automatul finit $A = (Q, \Sigma, \delta, q_0, F)$, unde:

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b, c\}$
$F = \{q_1\}$.

Functia de tranzitie este definita astfel:

$\delta(q_0, a) = q_1$
$\delta(q_0, b) = q_2$
$\delta(q_0, c) = q_0$
$\delta(q_1, a) = q_2$
$\delta(q_1, b) = q_1$
$\delta(q_1, c) = q_0$
$\delta(q_2, a) = q_0$
$\delta(q_2, b) = q_2$
$\delta(q_2, c) = q_2$

Graful de tranzitie atasat automatului A este:



Se observa ca starea initiala q_0 se recunoaste pe desen prin faptul ca nodul q_0 este marcat cu o sageata care intra in el. De asemenea, nodurile finale sunt marcate prin cerculete duble.

Definitia 1.2: Limbajul acceptat de automatul determinist $A = (Q, \Sigma, \delta, q_0, F)$ se noteaza cu L(A), unde:

$L(A) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$.

Cu $\sum^*$ am notat toate combinatiile posibile simboluri din $\sum$.

Un string din $w \in \sum^*$ se numeste **cuvant acceptat** de automatul A, daca $w \in L(A)$. In consecinta, un cuvant $w' \in \sum^*$ este neacceptat de catre A, daca:

$w' \in \sum^* - L(A) = \{u \in \sum^* \mid \delta^*(q_0, u) \notin F\} =: \overline{L(A)}$.

Definitia 1.3: δ^*: Q x $\sum^* \rightarrow$ Q se numeste **functie de tranzitie extinsa**. Valoarea lui $\delta^*(q, w)$ reprezinta starea in care se ajunge pornind din starea q dupa ce automatul citeste simbolurile lui w si face tranzitiile corespunzatoare.

Functia de tranzitie extinsa δ^* se defineste recursiv pornind de la definitia lui δ astfel:

$$\begin{cases} \delta^*(q, \lambda) = q \\ \delta^*(q, wa) = \delta(\delta^*(q, w), a) \end{cases}, \forall q \in Q, \forall w \in \Sigma^*, \forall a \in \Sigma.$$

In definitia de mai sus am folosit notatia λ pentru cuvantul vid (format din niciun caracter).

De exemplu, pentru automatul A din exemplul de mai sus avem:

$\delta^*(q_0, \text{cccca}) = q_1$.

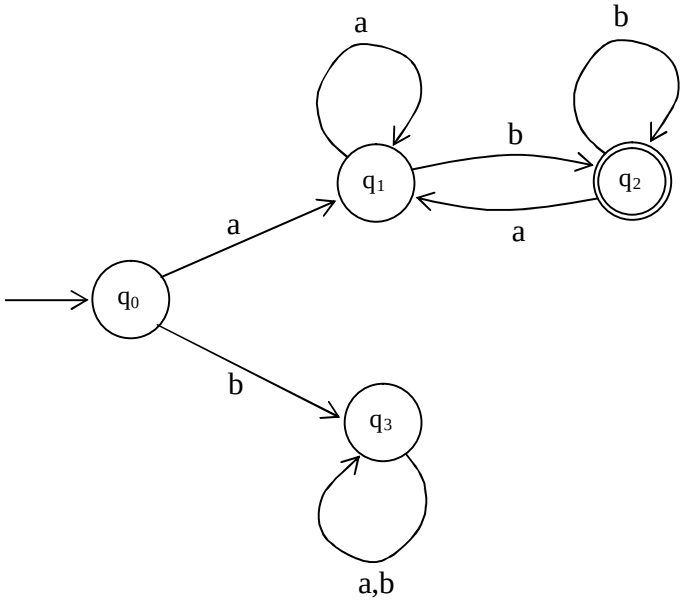
Teorema 1.1: Oricarui cuvant $w \in \sum^*$ cu proprietatea ca $\delta^*(q_i, w) = q_j$ ii corespunde in graful de tranzitie G_A un drum de la q_i la q_j etichetat cu caracterele cuvantului w. Reciproc, oricarui drum de la nodul q_i la nodul q_j din graful G_A ii corespunde un cuvant $w \in \sum^*$ cu proprietatea ca $\delta^*(q_i, w) = q_j$.

Teorema este extrem de intuitiva, de aceea nu o vom demonstra formal.

Definitia 1.4: Se numeste **limbaj** o multime de cuvinte (finita sau infinita) notata cu L, cu proprietatea ca $L \subseteq \sum^*$.

Definitia 1.5: Se numeste **limbaj regulat** un limbaj L cu proprietatea ca exista un automat finit determinist A cu proprietatea ca L = L(A).

De exemplu, pentru limbajul $L = \{ awb \mid w \in \{a, b\}^* \}$ putem construi automatul finit determinist A dat prin graful de tranzitie de mai jos.



Se observa ca L = L(A). Drept consecinta, L este un limbaj regulat.

Teme

I. Dupa cum am vazut, constructia unui automat finit determinist pentru a demonstra ca un limbaj este regulat nu este simpla. De aceea, pentru a capata experienta, lasam ca exercitiu constructia automatelor finite deterministe pentru a justifica faptul ca limbajele urmatoare sunt regulate:

1. $L_1 = \{ wb \mid w \in \{a, b\}^* \}$
2. $L_2 = \{ aw \mid w \in \{a, b\}^* \}$
3. $L_3 = \{ awa \mid w \in \{a, b\}^* \}$
4. $L_4 = \{ awbb \mid w \in \{a, b\}^* \}$.

II. O alta tema este aceea de a realiza un program care simuleaza functionarea unui automat finit determinist. Programul citeste dintr-un fisier text urmatoarele date:

1. n = numarul de stari ale automatului (vom avea evident starile $q_0, q_1,$ é ., q_{n-1})
2. m = numarul de simboluri ale automatului
3. $c_0, c_1,$ é ., c_{m-1} alfabetul automatului (m caractere distincte doua cate doua)
4. O matrice T de dimensiuni n x m corespunzatoare tranzitiilor recunoscute de automat. Astfel, pentru oricare $i \in \{0, 1,$ é ., n-1$\}$ si oricare $j \in \{0,1,$é .,m-1$\}$ avem ca $t_{ij} \in \{0, 1,$ é ., n-1$\}$ si t_{ij} reprezinta indicele starii in care se trece din starea q_i cu simbolul c_j. Cu alte cuvinte avem:

$$\delta(q_i, c_j) = q_{t_{ij}}, \forall i \in \{0,1,....,n-1\} \; si \; \forall j \in \{0,1,....,m-1\}$$

5. p = numarul de stari finale. Evident $0 \leq p \leq n$.
6. f_0, f_1, é ., f_{p-1} indicii starilor finale. Evident ca avem f_0, f_1, é ., $f_{p-1} \in$ {0,1,é .,n-1} si componentele vectorului f sunt distincte doua cate doua.

Programul dupa ce citeste din fisierul text datele de mai sus, citeste de la tastatura un string. Daca automatul recunoaste string-ul, afiseaza pe ecran mesajul ñacceptatò, in caz contrar afiseaza mesajul ñneacceptatò.

De exemplu, pentru automatul din prima figura fisierul text de intrare va contine:

```
3       // n = numarul de stari
3       // m = numarul de simboluri
a b c   // simbolurile automatului
1 2 0   // matricea T de tranzitii
2 1 0
0 2 2
1       // p = numarul de stari finale
2       // indicele singurei stari finale
```

I.2. Automate finite nedeterministe

Daca in cazul automatului finit determinist din orice stare cu fiecare simbol (caracter) trecem intr-o alta stare, automatul nedeterminist da posibilitatea ca dintr-o stare cu un simbol (caracter) sa nu putem trece in nicio stare sau dintr-o stare cu un caracter la un moment dat sa alegem dintr-o multime in ce stare sa trecem. Cu alte cuvinte, dintr-o stare cu un caracter exista o multime finita de tranzitii posibile, care poate fi chiar nula.

Definitia 2.1: A = (Q , $\sum$, δ , q_0 , F) se numeste **automat finit nedeterminist**, unde:

Q este o multime finita denumita **multime de stari**

$\sum$ este multime de simbolurilor (**alfabetul** automatului)

δ: Q x ($\sum \cup \{\lambda\}$) $\rightarrow$ P_Q defineste **setul de tranzitii**, unde P_Q este multimea partilor multimii Q.

$q_0 \in$ Q este **starea initiala**

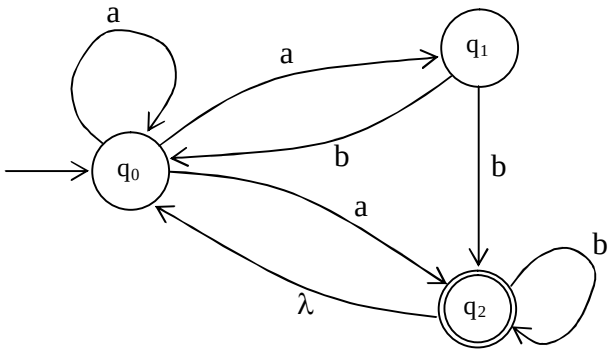
F $\subseteq$ Q este **multimea starilor finale**

Sa observam ca δ poate fi aplicat unei stari din Q impreuna cu un simbol din $\sum$ sau impreuna cu λ, situatie in care putem opta intre mai multe stari din Q ca destinatie

finala a tranzitiei. Daca δ se aplica unei stari impreuna cu λ spunem ca avem o λ-**tranzitie**, adica tranzitia se face fara a consuma vreun simbol din sirul de intrare ósô

Ca si in cazul automatelor finite deterministe, automatele finite nedeterministe pot fi reprezentate intuitiv printr-un graf denumit tot graf de tranzitie.

In figura de mai jos este prezentat graful de tranzitie al unui automat nedeterminist:



Sa vedem care sunt tranzitiile pentru automatul al carui graf de tranzitii este prezentat mai sus:

$\delta(q_0, a) = \{ q_0, q_1, q_2 \}$
$\delta(q_0, b) = \varnothing$
$\delta(q_0, \lambda) = \varnothing$
$\delta(q_1, a) = \varnothing$
$\delta(q_1, b) = \{ q_0, q_2 \}$
$\delta(q_1, \lambda) = \varnothing$
$\delta(q_2, a) = \varnothing$
$\delta(q_2, b) = \{ q_2 \}$
$\delta(q_2, \lambda) = \{ q_0 \}$

Pentru un automat finit nedeterminist putem defini de asemenea functia de tranzitie extinsa δ^*.

Definitia 2.2: $\delta^*: Q \times \Sigma^* \rightarrow P_Q$ se numeste **functie de tranzitie extinsa** a unui automat finit nedetermist. Valoarea lui $\delta^*(q, w)$ reprezinta multimea starilor in care se poate ajunge pornind din q dupa ce automatul citeste simbolurile lui w si face tranzitiile corespunzatoare.

Functia de tranzitie extinsa δ^* se poate defini si folosindu-ne de notiunea de graf de tranzitie G_A atasat automatului A astfel:

Definitia 2.2': δ^*: Q x $\sum^*$ $\rightarrow$ P_Q se numeste **functie de tranzitie extinsa** daca pentru oricare q, q' $\in$ Q si oricare w $\in \sum^*$ avem q' $\in \delta^*$(q, w) daca si numai daca exista un drum in G_A de la q la q' etichetat cu simbolurile lui w.

Definitia 2.3: Limbajul L(A) acceptat de un automat finit nedeterminist A se defineste in felul urmator:

$$L(A) = \{ w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \varnothing \}.$$

Se observa ca pentru a simula functionarea unui automat finit determinist scriem un algoritm de tip Greedy. In schimb, pentru a simula functionarea unui automat finit nedeterminist suntem nevoiti sa recurgem la un algoritm de tip backtracking, care este evident ineficient d.p.d.v. al complexitatii.

Tema

Sa se scrie un program care simuleaza functionarea unui automat finit nedeterminist. Programul citeste dintr-un fisier text urmatoarele date:

1. n = numarul de stari ale automatului (vom avea evident starile q_0, q_1, é ., q_{n-1})
2. m = numarul de simboluri ale automatului
3. c_0, c_1, é ., c_{m-1} alfabetul automatului (m caractere distincte doua cate doua)
4. un simbol pentru λ, evident diferit de c_0, c_1, é ., c_{m-1}
5. t = numarul de tranzitii posibile ale automatului
6. Cele t tranzitii ale automatului, date fiecare sub forma unor triplete (k, c, h), prin aceasta intelegand ca $q_h \in \delta(q_k, c)$, evident k, h $\in$ {0, 1, é ., n-1}
7. p = numarul de stari finale. Evident $1 \leq p \leq n$.
8. f_0, f_1, é ., f_{p-1} indicii starilor finale. Evident ca avem f_0, f_1, é ., $f_{p-1} \in$ {0,1,é .,n-1} si componentele vectorului f sunt distincte doua cate doua.

Programul dupa ce citeste din fisierul text datele de mai sus, citeste de la tastatura un string. Daca automatul recunoaste string-ul, afiseaza pe ecran mesajul ñacceptatò, in caz contrar afiseaza mesajul ñneacceptatò.

De exemplu, pentru automatul nedeterminist prezentat in figura din acest capitol fisierul text va contine:

```
3       // n = numarul de stari
2       // m = numarul de simboluri
a b     // simbolurile automatului
L       // simbolul folosit pentru λ
```

```
7       // t = numarul de tranzitii
0 a 0   // cele 7 tranzitii
0 a 1
0 a 2
1 b 0
1 b 2
2 L 0
2 b 2
1       // p = numarul de stari finale
2       // indicele singurei stari finale
```

II.3. Echivalenta intre automatele finite deterministe si cele nedeterministe

Definitia 3.1: Doua automate finite A_1 si A_2 se numesc **echivalente** daca ambele accepta aceleasi limbaje, adica:

$$L(A_1) = L(A_2).$$

Este evident ca pentru orice automat finit determinist A exista un automat finit nedeterminist A' echivalent cu el. Pentru aceasta nu avem decat sa descriem automatul nedeterminist A' corespunzator grafului de tranzitii G_A atasat automatului A.

Vom arata in continuare ca si reciproca este adevarata. Pentru un automat finit nedeterminist dat A vom arata cum se construieste un automat finit determinist A' echivalent cu A, adica L(A) = L(A').

Fie un automat finit nedeterminist:

$$(1)\ A = (Q , \sum , \delta , q_0 , F).$$

Construim automatul finit determinist:

$$(2)\ A' = (Q' , \sum , \delta' , q'_0 , F'), \text{ unde:}$$

$Q' = P_Q$, adica A' are ca stari multimea partilor lui Q. Sunt in total 2^n stari in Q', unde n este numarul de stari al lui A, adica n = |Q|.

$q'_0 = \{q_0\}$.

F' contine ca stari din Q' multimile de stari din Q in care exista cel putin un nod din F, adica $F' = \{ q' \in Q' \mid \exists\ q \in F \text{ a.i. } q \in q' \}$. Daca automatul A accepta cuvinte vide obtinute prin λ-tranzitii pornind din q_0 pana intr-o stare finala, atunci in F' se introduce si $\{q_0\} = q'_0$.

Tranzitiile automatului A' se definesc astfel:

$$\forall q' \in Q' \setminus \{\phi\}, \forall c \in \Sigma : \delta'(q', c) = \bigcup_{q \in q'} \delta^*(q, c)$$

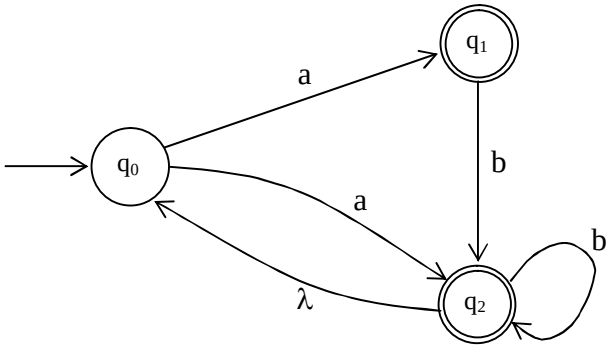
si $\delta'(\phi, c) = \phi, \forall c \in \Sigma$.

Sa observam ca daca automatul A nu are λ-tranzitii, atunci:

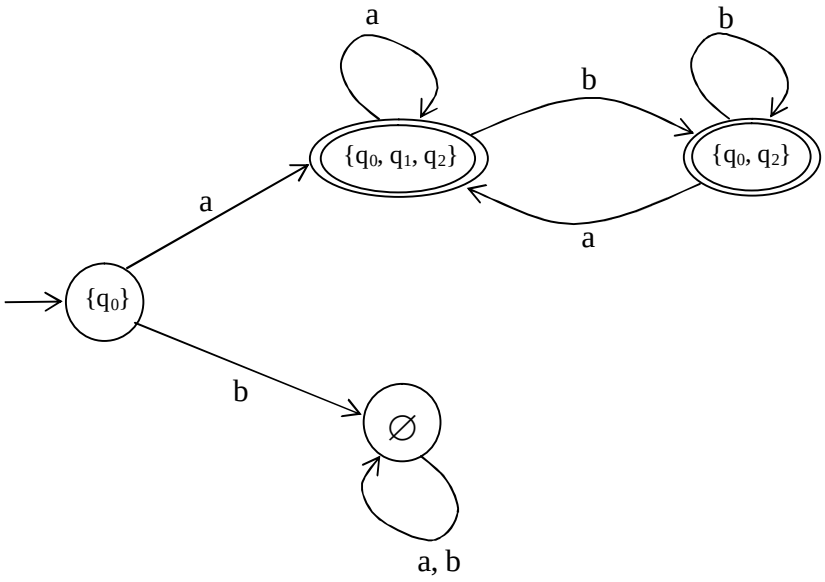
$$\forall q' \in Q' \setminus \{\phi\}, \forall c \in \Sigma : \delta'(q', c) = \bigcup_{q \in q'} \delta(q, c).$$

Pentru a ilustra modul de constructie al automatului finit determinist A' pornind de la un automat finit nedeterminist dat A consideram un exemplu sugestiv.

Fie A = (Q , ∑ , δ , q_0 , F) un automat nedeterminist al carui graf de tranzitii este:



Automatul determinist echivalent A' este urmatorul:



Dupa cum se poate observa in figura de mai sus, pentru automatul A' nu am reprezentat decat 4 stari, celelalte 4 nefiind accesibile din starea initiala $\{q_0\}$. Cu alte

cuvinte, pornind din nodul $\{q_0\}$ nu se poate ajunge in nodurile $\{q_1\}$, $\{q_2\}$, $\{q_0, q_1\}$ si $\{q_1, q_2\}$. In consecinta, automatul finit determinist A' este echivalent cu automatul finit determinist A'' obtinut din A' prin eliminarea starilor $\{q_1\}$, $\{q_2\}$, $\{q_0, q_1\}$ si $\{q_1, q_2\}$ si, evident, a tranzitiilor legate de aceste trei stari.

Urmatoarea teorema justifica echivalenta automatelor A si A'.

Teorema 3.1: Pentru orice automatul finit nedeterminist A definit in (1), automatul finit determinist A' definit in (2) este echivalent cu A, adica avem L(A) = L(A').

Demonstratie:

I. Pentru a arata ca $L(A) \subseteq L(A')$ folosim inductia matematica.

Fie propozitia P_k: *"daca pentru un cuvant $w \in \Sigma^*$ de lungime cel mult k exista un drum etichetat w in G_A de la q_0 la q_i, atunci exista un drum etichetat w si in $G_{A'}$ de la q'_0 la un nod q'_i, unde $q_i \in q'_i$"* (k este numar natural).

Evident, P_0 este adevarata.

Sa aratam acum ca $P_k \rightarrow P_{k+1}$.

Presupunem P_k adevarata.

Fie un cuvant wa ($w \in \Sigma^*$, $a \in \Sigma$) de lungime cel mult k+1 pentru care exista un drum etichetat wa in G_A de la q_0 la q_i. Acest lucru inseamna ca exista un drum etichetat w in G_A de la q_0 la o stare $q_j \in Q$ si $q_i \in \delta^*(q_j, a)$ (3).

Cum cuvantul w are lungime cel mult k, rezulta din P_k faptul ca exista un drum D' etichetat w in $G_{A'}$ de la $q'_0 = \{q_0\}$ la un nod q'_j, unde $q_j \in q'_j$ (4).

Avem: $\delta'(q'_j, a) \overset{def}{=} \bigcup_{q \in q'_j} \delta^*(q, a) \overset{(4)}{\supseteq} \delta^*(q_j, a)$. De aici si din (3) rezulta ca $q_i \in \delta'(q'_j, a)$. Notam $q'_i = \delta'(q'_j, a)$. Rezulta de aici ca exista un arc in $G_{A'}$ etichetat cu a de la q'_j la starea q'_i, unde $q_i \in q'_i$ (5).

Din (4) si (5) rezulta ca exista un drum (obtinut prin adaugarea arcului (q'_j, q'_i) la sfarsitul drumului D') etichetat wa in $G_{A'}$ de la q'_0 la q'_i, unde $q_i \in q'_i$ si astfel propozitia P_{k+1} este demonstrata.

Evident, din faptul ca P_k este adevarata pentru orice numar natural k se obtine imediat si faptul ca $L(A) \subseteq L(A')$.

II. Sa demonstram acum includerea inversa, adica $L(A') \subseteq L(A)$ folosind tot inductia matematica.

Fie propozitia R_k: *"daca pentru un cuvant $w \in \Sigma^*$ de lungime cel mult k exista un drum etichetat w in $G_{A'} \setminus \varnothing$ (graful de tranzitie fara nodul $\varnothing$) de la q'_0 la q'_i, atunci exista un drum etichetat w si in G_A de la q_0 la un nod q_i, unde $q_i \in q'_i$"* (k este numar natural).

Evident, R_0 este adevarata.

Sa aratam acum ca $R_k \rightarrow R_{k+1}$.

Presupunem R_k adevarata.

Fie un cuvant wa ($w \in \Sigma^*$, $a \in \Sigma$) de lungime cel mult k+1 pentru care exista un drum etichetat wa in $G_{A'}\setminus\varnothing$ de la q'_0 la q'_i. Acest lucru inseamna ca exista un drum etichetat w in $G_{A'}\setminus\varnothing$ de la q'_0 la o stare $q'_j \in Q'$ si $\delta'(q'_j, a) = q'_i$ (3').

Cum cuvantul w are lungime cel mult k, rezulta din R_k faptul ca exista un drum D_1 etichetat w in G_A de la q_0 la q_j, unde $q_j \in q'_j$ (4').

Avem: $\delta'(q'_j, a) \overset{def}{=} \bigcup_{q \in q'_j} \delta^*(q, a) \overset{(4')}{\supseteq} \delta^*(q_j, a)$. De aici si din (3') rezulta ca $\delta^*(q_j, a) \subseteq q'_i$, adica exista un drum D_2 in $G_{A'}$ etichetat cu *a* de la q_j la o stare $q_i \in q'_i$ (5').

Din (4') si (5') obtinem drumul $D = D_1D_2$ etichetat wa in G_A de la q_0 la q_i (care trece prin q_j) si astfel propozitia R_{k+1} este demonstrata.

Evident, din faptul ca R_k este adevarata pentru orice numar natural k se obtine imediat si faptul ca $L(A') \subseteq L(A)$.

Din $L(A') \subseteq L(A)$ si $L(A) \subseteq L(A')$ rezulta ca $L(A) = L(A')$ (q.e.d.).

□

Tema:

1. Determinati un automat finit determinist echivalent cu automatul finit nedeterminist pentru care s-a prezentat graful de tranzitie in subcapitolul I.2.
2. Scrieti un program care citeste dintr-un fisier text un automat finit nedeterminist si scrie intr-un alt fisier text automatul finit determinist echivalent. Memorarea intr-un fisier a unui automat finit determinst consideram ca se face asa cum este descris la tema B de la sfarsitul subcapitolul I.1, iar memorarea unui automat finit nedeterminist intr-un fisier este descrisa la tema de la sfarsitul capitolul I.2.

I.4. Reducerea numarului de stari pentru un automat finit

Fiecare automat finit accepta un singur limbaj, dar evident reciproca nu este adevarata. Atunci prezinta interes pentru un automat finit dat A sa determinam un automat finit determinist A' echivalent cu A astfel incat A' sa aiba cel mai mic numar de stari.

In acest capitol vom prezenta o metoda de gasire a unui automat finit determinist A' cu numar minim de stari echivalent cu un automat finit determinist dat A.

Definitia 4.1: Doua stari p si q ale unui a.f.d. se numesc **asemenea**, daca pentru orice cuvant $w \in \Sigma^*$ avem simultan:

$$\begin{cases} \delta^*(p,w) \in F \Rightarrow \delta^*(q,w) \in F \\ \delta^*(p,w) \notin F \Rightarrow \delta^*(q,w) \notin F \end{cases}.$$

Se observa ca relatia ñasemeneaò este reflexiva (daca p si q sunt asemenea, atunci q si p sunt asemenea) si tranzitiva (daca p si q sunt asemenea si q si r sunt asemenea, atunci p si r sunt asemenea).

In vederea reducerii numarului de stari ne intereseaza sa gasim starile care nu sunt asemenea. Pentru gasirea perechilor de stari (p, q) care nu sunt asemenea putem folosi urmatorul algoritm:

Procedura GasireStariNeasemenea

1. Se elimina toate nodurile inaccesibile (daca exista) din q_0, folosind un algoritm de parcurgere a grafului de tranzitii G_A.
2. Se construieste matricea M initializata cu 0, de dimensiuni n x n, unde n este numarul de stari ramase dupa executarea primului pas.
3. Pentru fiecare i, j $\in$ {0, 1, 2, é ., n-1} daca $q_i \in F$ si $q_j \in Q \setminus F$, atunci in matricea M pe pozitia (i, j) se pune valoarea 1 si pe pozitia (j, i) se pune valoarea 1, adica $m_{ij} = 1$ si $m_{ji} = 1$.
4. Pentru fiecare i, j $\in$ {0, 1, 2, é ., n-1} si fiecare c $\in \Sigma$ daca $\delta(q_i, c) = q_k$ si $\delta(q_j, c) = q_h$ si $m_{kh} = 1$, atunci $m_{ij} = 1$ si $m_{ji} = 1$.
5. Daca la pasul 4 s-a facut vreo modificare in matricea M, atunci se reia pasul 4.

Procedura de mai sus gaseste toate perechile de stari care nu sunt asemenea. Astfel, o pereche de stari (q_i, q_j) nu este asemenea daca si numai daca $m_{ij} = 1$.

Evident, o pereche de stari (q_i, q_j) este asemenea daca si numai daca $m_{ij} = 0$.

Pentru a gasi clasele de echivalenta pentru relatia ñasemeneaò putem construi graful neorientat G = (V = {0, 1, é ., n-1}, U), unde multimea de muchii U este formata cu perechile de noduri neordonate [i, j] (i, j $\in$ V, i $\neq$ j) cu proprietatea ca $m_{ij} = 0$. In graful G se identifica toate componentele conexe, care vor fi clasele de echivalenta de stari asemenea:

Procedura GasireClaseStariAsemenea

1. Se apeleaza procedura procedura GasireStariNeasemenea;
2. Folosind matricea M generata in primul pas se construieste graful G = (V, U);
3. Folosind un algoritm de parcurgere a grafului G se identifica toate componentele conexe K_0, K_1, é ., K_{t-1} ale grafului G.

Teorema 4.1: Procedura GasireClaseStariAsemenea determina clasele de echivalenta ale relatiei ñasemeneaò.

Demonstratie:

Intai sa demonstram corectitudinea pasului 1, adica faptul procedura ñGasireStariNeasemeneaò gaseste toate perechile de stari ale lui care nu sunt asemenea.

Este clar ca executia procedurii ñGasireStariNeasemeneaò se incheie in numar finit de pasi, deoarece la fiecare iteratie a pasului 4 in matricea M cel putin doua valori 0 se schimba in 1, iar la inceput in M avem n^2 valori 0 (numar finit).

Sa aratam acum faptul ca la iesire din procedura avem $m_{ij} = 1$ daca si numai daca q_i si q_j nu sunt asemenea.

Pentru implicatia directa ($m_{ij} = 1 \Rightarrow q_i$ si q_j nu sunt asemenea) avem una dintre urmatoarele doua situatii, in functie de momentul in care m_{ij} devine 1:

Situatia 1: m_{ij} a devenit 1 in pasul 3. Inseamna ca una si numai una dintre starile q_i sau q_j este finala. Consideram cuvantul vid $w = \lambda \in \Sigma^*$. Evident, $\delta^*(q_i, w) = q_i$ si $\delta^*(q_j, w) = q_j$, ceea ce inseamna ca una si numai una dintre starile $\delta^*(q_i, w)$ si $\delta^*(q_j, w)$ este stare finala, adica starile q_i si q_j nu sunt asemenea.

Situatia 2: m_{ij} a devenit 1 intr-una dintre iteratiile pasului 4. Inseamna ca exista $k, h \in \{0, 1, 2, é ., n-1\}$ si exista $c \in \Sigma$ a.i. $\delta(q_i, c) = q_k$ si $\delta(q_j, c) = q_h$ si $m_{kh} = 1$. Se obtine ca starile q_k si q_h nu sunt asemenea, adica exista un cuvant $w \in \Sigma^*$ astfel incat $\delta^*(q_k, w) = q_u$, $\delta^*(q_h, w) = q_v$, iar q_u si q_v sunt doua stari dintre care una si numai una este finala (1).

Consideram cuvantul $cw \in \Sigma^*$. Avem:

(2) $\delta^*(q_i, cw) = \delta^*(\delta(q_i, c), w) = \delta^*(q_k, w) = q_u$ (din (1) si def. lui δ^*)

si

(3) $\delta^*(q_j, cw) = \delta^*(\delta(q_j, c), w) = \delta^*(q_h, w) = q_v$.

Din (1), (2) si (3) rezulta ca starile q_i si q_j nu sunt asemenea (q.e.d.).

SŁ justificam implicatia inversa, adica daca starile q_i si q_j nu sunt asemenea, atunci $m_{ij} = 1$.

Din faptul ca starile q_i si q_j nu sunt asemenea rezulta ca exista un cuvant $w \in \Sigma^*$ si starile q_k si q_h astfel incat:

$\delta^*(q_i, w) = q_k$, $\delta^*(q_j, w) = q_h$ (4)

si

una si numai una dintre starile q_k si q_h sunt finale, ceea ce inseamna dintr-una dintre iteratiile pasului 3 al algoritmului ca $m_{kh} = 1$ (5).

Daca $w = \lambda$, atunci $q_i = q_k$ si $q_j = q_h$, adica $m_{ij} = m_{kh} = 1$.

Daca $w \neq \lambda$, $w = w'c$, $c \in \Sigma$ si exista starile $q_{k'}$ si $q_{h'}$ a.i.:

$\delta^*(q_i, w'c) = \delta(\delta^*(q_i, w'), c) = \delta(q_{k'}, c) = q_k$, unde $q_{k'} = \delta^*(q_i, w')$
si
$\delta^*(q_j, w'c) = \delta(\delta^*(q_j, w'), c) = \delta(q_{h'}, c) = q_h$, unde $q_{h'} = \delta^*(q_j, w')$.

Cum $m_{kh} = 1$, rezulta din pasul 4 al algoritmului ca $m_{k'h'} = 1$. In plus, avem:

$\delta^*(q_i, w') = q_{k'}$, $\delta^*(q_j, w') = q_{h'}$,

adica am obtinut o situatie similara cu cea de la punctele (4) si (5) numai ca pentru un cuvant w' mai scurt cu un simbol decat w. Cum cuvantul w este finit ca lungime, rezulta ca dupa |w| reduceri succesive ale cuvantului w cu cate un simbol obtinem $m_{ij} = 1$ (q.e.d.).

Sa justificam acum faptul ca procedura GasireClaseStariAsemenea determina clasele de echivalenta ale relatiei ñasemeneaò.

Este evident faptul ca starile $K_0 \cup K_1 \cup$ é .$\cup K_{t-1}$ reprezinta o partitionare a multimii V, adica $K_0 \cup K_1 \cup$ é .$\cup K_{t-1} = V$ si $K_u \cap K_v = \phi$, deoarece K_0, K_1, é ., K_{t-1} sunt componente conexe in G.

Sa aratam ca pentru $\forall$ u $\in \{0, 1,$ é ., t-1$\}$ si $\forall$ i, j $\in K_u$ rezulta ca starile q_i si q_j sunt asemenea.

Din cauza ca multimea K_u este componenta conexa, rezulta ca in graful G exista un drum D = (i = d_1, d_2, é ., d_s = j) de la i si j. Pentru fiecare arc (d_k, d_{k+1}) din D avem $m_{d_k d_{k+1}} = 0$. Acest lucru inseamna ca starile q_{d_k} si $q_{d_{k+1}}$ sunt asemenea, pentru orice k $\in \{1, 2,$ é ., s$\}$. In consecinta starile $q_i = q_{d_1}$ si $q_j = q_{d_s}$ sunt si ele asemenea (folosim proprietatea de tranzitivitate a relatiei).

Deci, procedura ñGasireClaseStariAsemeneaò gaseste clasele de echivalenta ale relatiei ñasemeneaò.

□

Pentru reducerea numarului de stari ale a.f.d.-ului A apelam urmatoarea procedura, care construieste a.f.d.-ul redus A' = (Q', $\sum$, δ', q'_0, F') echivalent cu A, adica L(A) = L(A') si A' are numar minim de stari:

Procedura ReducereStariAutomat

1. Se apeleaza procedura GasireClaseStariAsemenea;
2. Componentele conexe K_0, K_1, é ., K_{t-1} determinate la pasul 1 reprezinta starile automatului redus A', adica Q' = { K_0, K_1, é ., K_{t-1} } ;
3. $q'_0 = K_i$ astfel incat $0 \in K_i$;
4. F' = { $K_i \in Q' \mid \exists\ q_j \in F$ a.i. $j \in K_i$ } ;
5. Pentru fiecare tranzitie a automatului A: $\delta(q_i, c) = q_j$ ($q_i, q_j \in Q$ si $c \in \Sigma$) se identifica K_u si K_v (u, v $\in$ {0, 1, é ., t-1}) astfel incat $i \in K_u$ si $j \in K_v$ si in automatul A' se defineste tranzitia $\delta'(K_u, c) = K_v$.

Teorema 4.2: Automatul finit determinist A' gasit de procedura ReducereStariAutomat este echivalent cu automatul A si are numar minim de stari.

Demonstratie:

I. Sa demonstram intai ca $L(A) \subseteq L(A')$.

Fie $w \in L(A)$. Cuvantul w corespunde unui drum D etichetat cu w in graful G_A de la q_0 la $q_f \in F$.

Pentru fiecare arc $(q_i, q_j) \in D$ etichetat cu simbolul c identificam K_u si K_v (u, v $\in$ {0, 1, é ., t-1}) astfel incat $i \in K_u$ si $j \in K_v$. Avem din constructia lui δ' ca $\delta'(K_u, c) = K_v$, ceea ce corespunde in graful de tranzitii $G_{A'}$ unui arc (K_u, K_v) etichetat cu c. Se obtine astfel un drum D' etichetat w de la q'_0 la K_p, unde $f \in K_p$. Evident, K_p este stare finala in A', deoarece q_f este stare finala in A (v. constructia lui F').

II. Sa aratam acum ca $L(A') \subseteq L(A)$.

Fie $w \in L(A')$. Cuvantul w corespunde unui drum D' etichetat cu w in graful $G_{A'}$ de la q'_0 la $K_p \in F'$.

Pentru fiecare arc $(K_u, K_v) \in D'$ etichetat cu simbolul c avem ca $\delta'(K_u,c) = K_v$. Din constructia lui δ' rezulta ca exista $i \in K_u$ si $j \in K_v$ astfel incat $\delta(q_i, c) = q_j$, ceea ce corespunde in graful de tranzitii G_A unui arc (q_i,q_j) etichetat cu c. Se obtine astfel un drum D etichetat w de la q_0 la $q_f \in F$, unde $f \in K_p$.

Din $L(A') \subseteq L(A)$ si $L(A) \subseteq L(A')$ rezulta ca $L(A) = L(A')$. Deci, A si A' sunt echivalente.

Sa justificam in final faptul ca A' este automatul finit determinist echivalent cu A care are cele mai putine stari. Demonstratia o vom face prin reducere la absurd.

Presupunem ca exista un automat $A_1 = (Q_1=\{p_0,p_1, é .,p_{z-1}\}, \sum, \delta_1, p_0, F_1)$ echivalent cu A si care are mai putine stari ca A' (z < t).

Din cauza ca A' nu are stari inaccesibile din q'_0 rezulta ca exista t cuvinte distincte doua cate doua w_0, w_1, é ., w_{t-1} astfel incat:

(1) $\delta'^{*}(q'_0, w_i) = K_i, \forall\ i \in \{0, 1, é\ .., t\text{-}1\}$.

Din cauza ca z < t, din (1) si din faptul ca A' si A_1 sunt echivalente rezulta ca:

(2) $\exists$ u, v $\in \{0, 1, é\ .., t\text{-}1\}$ a.i. $\delta_1^{*}(p_0, w_u) = \delta_1^{*}(p_0, w_v)$.

Cum starile din K_u si K_v nu sunt asemenea (din constructia lor), rezulta ca exista un cuvant $\pi \in \Sigma^{*}$ astfel incat $\delta'^{*}(q'_0,w_u\pi) \overset{(1)}{=} \delta'^{*}(K_u,\pi) \in F'$ si $\delta'^{*}(q'_0,w_v\pi) \overset{(1)}{=} \delta'^{*}(K_v,\pi) \notin F'$ sau avem $\delta'^{*}(q'_0,w_u\pi) \overset{(1)}{=} \delta'^{*}(K_u,\pi) \notin F'$ si $\delta'^{*}(q'_0,w_v\pi) \overset{(1)}{=} \delta'^{*}(K_v,\pi) \in F'$, ceea ce inseamna ca unul si numai unul dintre cuvintele este $w_u\pi$ si $w_v\pi$ este acceptat de A' si de A (3).

Din (2) obtinem ca:

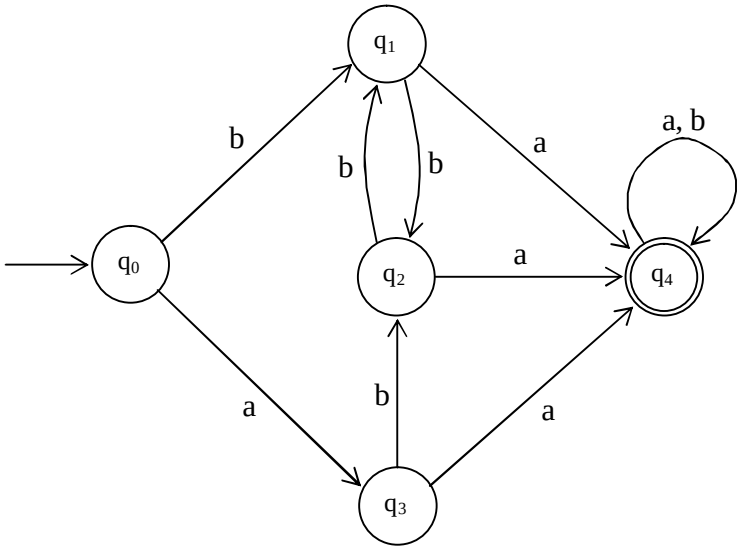
(4) $\delta_1^{*}(p_0,w_u\pi) = \delta_1^{*}(\delta_1^{*}(p_0,w_u),\pi) \overset{(2)}{=} \delta_1^{*}(\delta_1^{*}(p_0,w_v),\pi) = \delta_1^{*}(p_0,w_v\pi) =: p_y$.

Din (4) rezulta ca $w_u\pi$ si $w_v\pi$ sunt ambele cuvinte acceptate de A_1, daca p_y este stare finala in A_1 (si de catre A) sau $w_u\pi$ si $w_v\pi$ sunt ambele cuvinte neacceptate de A_1 (si de catre A), daca p_y nu este stare finala in A_1, ceea ce contrazice afirmatia (3).

In consecinta, A' este automatul finit determinist echivalent cu A cu numar minim de stari.

□

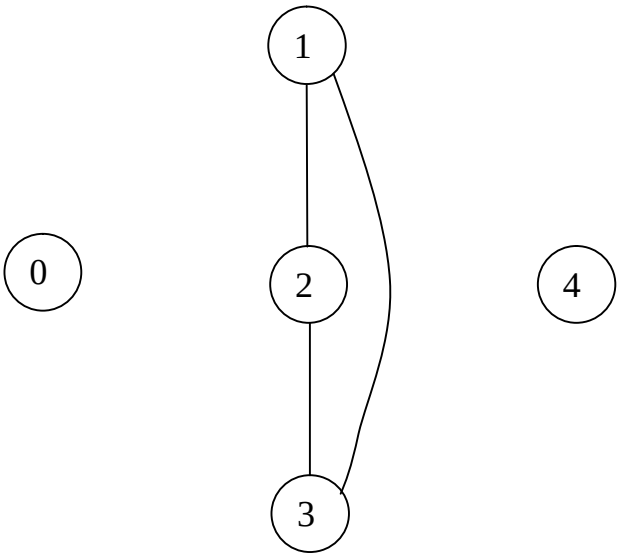
In finalul capitolului ne propunem sa aplicam algoritmul de reducere a numarului de stari pentru urmatorul automat finit determinist:



Dupa apelarea procedurii ñGasireStariNeasemeneaò se obtine matricea:

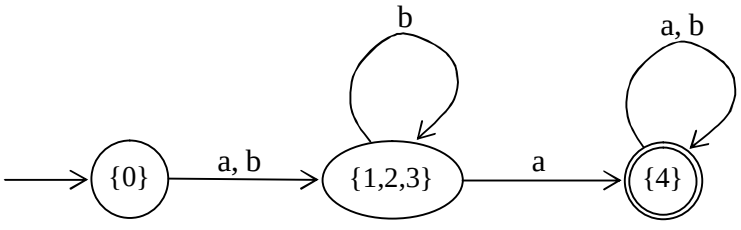
$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Graful G construit pornind de la matricea M, in pasul 2 de la procedura GasireClaseStariAsemenea este:



Graful G are t = 3 componente conexe: $K_0 = \{0\}$, $K_1 = \{1, 2, 3\}$ si $K_2 = \{4\}$.

Dupa ce se executa pasii 2, 3, 4 si 5 ai procedurii ñReducereStariAutomatò obtinem automatul redus A':



Teme:

1. Aplicati procedura ñReducereStariAutomatò pentru un alt automat finit determinist.

2. Scrieti un program care citeste dintr-un fisier text un automat finit determinist si scrie intr-un alt fisier text automatul finit determinist redus. Memorarea intr-un fisier text a unui automat finit determinst consideram ca se face asa cum este descris la tema B de la sfarsitul subcapitolul I.1.
3. Evaluati complexitatea algoritmului de reducere a unui automat finit determinist.