Gramatici regulate. Automate finite

10.03.2020

Outline

Introducere in automate.

- Letia & Chifu 2.2: 2.2.1
- capitolul 1.1: Finite automata, FORMAL DEFINITION OF A FINITE AUTOMATON , EXAMPLES OF FINITE AUTOMATA(FA), FORMAL DEFINITION OF COMPUTATION "Introduction to the Theory of computation" 3rd edition, Michael Sipser
- Introduction to Automata Theory, Languages, and Computation sections 2.1, 2.2, Ullman

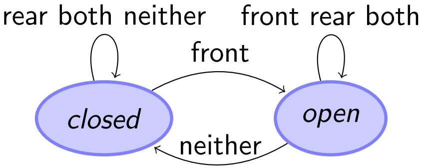
Automate finite

- modele pentru calculatoare cu extrem de putina memorie
- colectie finita de stari cu reguli de tranzitie care determina trecerea dintr-o stare in alta
- aplicatii initiale - sequential switching circuits = starea = valoarea bitilor interni
- azi: mai multe aplicatii software pot fi modelate ca FA

Reprezentarea FA - State diagram:

- noduri
- arce - indica tranzitia starilor
- etichete (labels) pe arce care definesc ce cauzeaza tranzitia

Exemplu: Controller pentru usi automate



	neither	front	rear	both
closed	closed	open	closed	closed
open	closed	open	open	open

start:closed

- secventa: front, rear, neither, front, both, neither, rear, neither
- stari: closed, open, opeb, closed,open, open, closed, closed, closed

computer cu doar un bit de memorie (vezi Sipser)

FA - Good models for computers with an extremely limited amount of memory

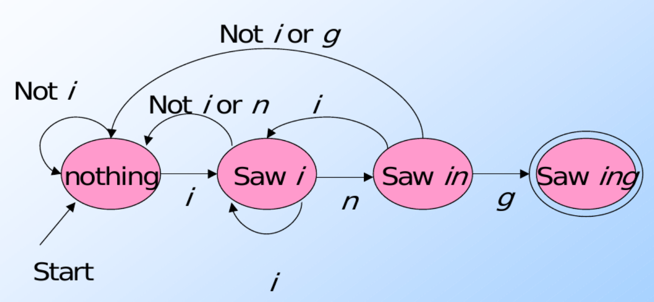
Exemplu: Recunoasterea cuvintelor care se termina in “.ing”

```
2: /* i seen /*
 c = getNextInput();
 if (c=='n') goto 3;
 else if (c=='i') goto 2;
 else goto 1;
3:  /* "in" seen */
...
```

ingest, reading

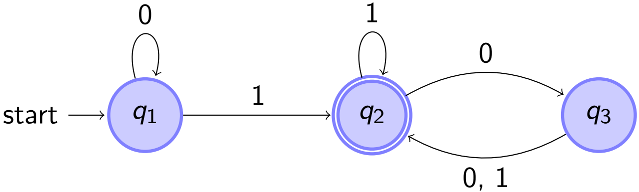
Automat $\rightarrow$ Cod

1. citeste urmatorul input
2. decide starea urmatore
3. sari la inceputul codului pentru acea stare



de fapt: expresii regulate .*ing

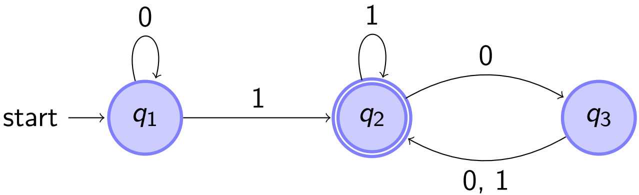
Exemplu: Automat - Diagrama de stare



- 3 stari; start state, accept state
- transitions

Automatul primeste un input string si produce *accept* sau *reject*.
fie 1101:

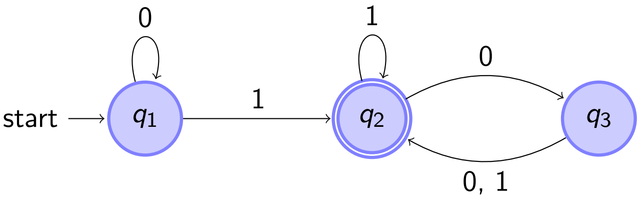
1. Start in q_1
2. Citeste 1 si urmeaza tranzitia q_1 to q_2
3. Citeste 1 si urmeaza tranzitia q_2 to q_2
4. Citeste 0 si urmeaza tranzitia q_2 to q_3
5. Citeste 1 si urmeaza tranzitia q_3 to q_2
6. *accept* deoarece se afla in starea accept q_2 la sfarsitul input-ului



- Accepta 1, 01, 11, 0101010101?
- Dar 100, 0100, 110000, 0101000000?
- dar 0, 10, 101000?

care sunt toate stringurile pe care automatul le accepta?
Setul tuturor sirurilor recunoscute de an automat A: $L(A)$

L(A) $=$?



- Accepta 1, 01, 11, 0101010101? DA
- Dar 100, 0100, 110000, 0101000000? Da
- dar 0, 10, 101000? le respinge

care sunt toate stringurile pe care automatul le accepta?
Setul tuturor sirurilor recunoscute de an automat A: $L(A)$

L(A) $= \{w | w$ contine cel putin un 1 si se termina cu un numar par de 0-uri dupa ultimul 1$\}$

Table of Contents

Automat finit determinist - *formal* definition

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

- un alfabet de intrare Σ- set de simboluri
- un set finit de stari Q
- o functie de tranzitie δ
- o stare de start q_0
- un set de stari finale $F \subseteq Q$ (final state, accepting states)

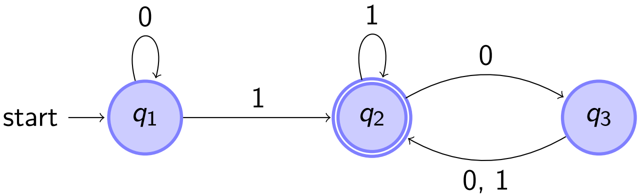
Functia de tranzitie $\delta : Q \times \Sigma \rightarrow Q$: $\delta(q,\ a)$ starea in care automatul DFA trece cand este in starea q si primeste ca input a.

Setul tuturor sirurilor recunoscute de un automat A este limbajul $Ł(A) = \{w | A\ accepta\ w\}$

Descrierea formala a automatului exemplu

$D_1 = (\{0,1\}, \{q_1, q_2, q_3\}, \delta, q_1, \{q_2\})$

	δ	0	1
$\rightarrow$	q_1	q_1	q_2
$*$	q_2	q_3	q_2
	q_3	q_2	q_2



Acceptare 011 $?\exists\delta(q_1, 0)$:

$$\delta(q_1, 0) = q_1; \delta(q_1, 1) = q_2; \delta(q_2, 1) = q_2 \in F$$

s-a gasit secventa de stari: q_1, q_1, q_2

Definitie formala a calculului

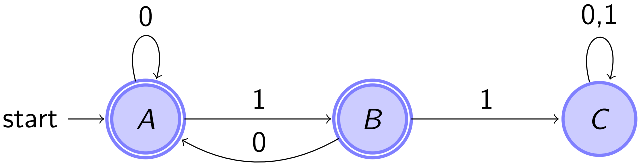
Fie $A = (\Sigma, Q, \delta, q_0, F)$ si $w = w_1 w_2 ... w_n$, $w_i \in \Sigma$. Automatul recunoaste w daca exista o secventa $r_0, r_1, ..., r_n \in Q$:

Definitie formala a calculului

Fie $A = (\Sigma, Q, \delta, q_0, F)$ si $w = w_1 w_2 ... w_n$, $w_i \in \Sigma$. Automatul recunoaste w daca exista o secventa $r_0, r_1, ..., r_n \in Q$:

- $r_0 = q_0$
- $\delta(r_i, w_{i+1}) = r_{i+1}$ pt $i = 0, ..., n-1$
- $r_n \in F$

Exemplu:



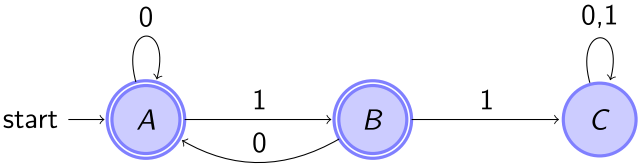
Extended $\hat{\delta}$

$$\hat{\delta}(A, 011) = \delta(\delta(\delta(A, 0), 1), 1) =$$

$$\delta(\delta(A, 1), 1) = \delta(B, 1) = C$$

Ce accepta?

Exemplu:



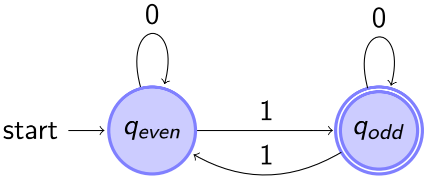
Extended $\hat{\delta}$

$$\hat{\delta}(A, 011) = \delta(\delta(\delta(A, 0), 1), 1) =$$

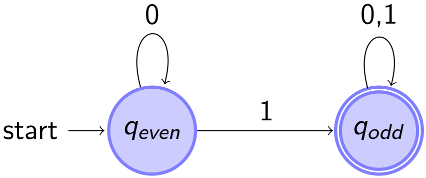
$$\delta(\delta(A, 1), 1) = \delta(B, 1) = C$$

Ce accepta? Accepta toate stringurile care nu includ doua simboluri consecutive 1

Exemple

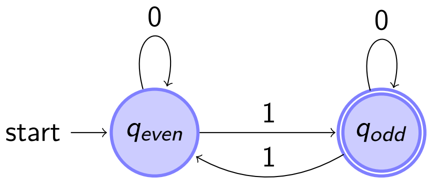


Ce accepta?

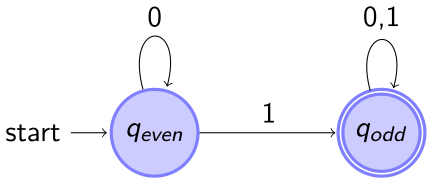


Ce accepta?

Exemple



Ce accepta? Accepta toate stringurile care includ numar par de 1



Ce accepta? Accepta toate stringurile care contin cel putin un 1

Table of Contents

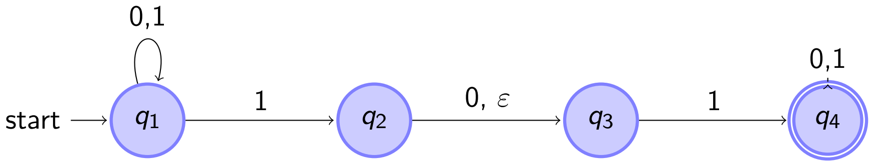
Automat finit nedeterminist - *formal* definition

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

- un alfabet de intrare Σ- set de simboluri
- un set finit de stari Q
- o functie de tranzitie δ
- o stare de start q_0
- un set de stari finale $F \subseteq Q$ (final state, accepting states)

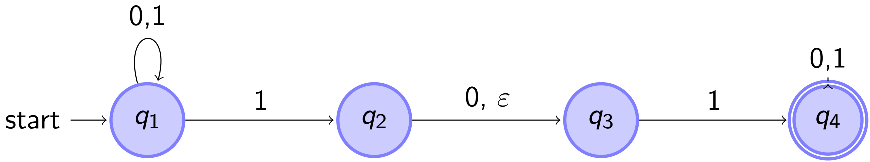
Functia de tranzitie $\delta : Q \times \Sigma_\varepsilon \to P(Q)$: $\delta(q,\ a)$ starea/starile in care automatul NFA poate trece cand este in starea q si primeste ca input a.

Exemplu - automat nedeterminist



Input: 010110

Exemplu - automat nedeterminist



Input: 010110 Calcul nedeterminist: accept/reject

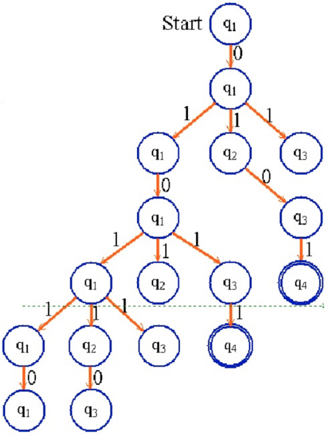


Table of Contents

Outline

Automat finit - definitie formala ca sistem de rescriere

Automat finit (finite automaton, finite state acceptor):

$$A = (T, Q, R, q_0, F)$$

- Q set nevid - setul starilor interne
- $(T \cup Q, R)$ sistem de rescriere; $T \cap Q = \emptyset$
- $q_0 \in Q$ - starea initiala
- $F \subseteq Q$ - stari finale
- fiecare element din R are forma $qt \to q'$, $q, q' \in Q, t \in T$

Definitie formala *Automat finit*

- automatul A accepta/recunoaste setul de stringuri

$$L(A) = \{\tau \in T^* | q_0\tau \Rightarrow^* q, q \in F\}$$

- Doua automate A si A' sunt echivalente daca si numai daca $L(A) = L(A')$

Interpretare

- masina care citeste la intrare un input string; citeste simbol cu simbol si isi schimba starea interna
- automatul se afla in starea q cand sirul curent din derivare este $q\tau$
- automatul face o tranzitie din q in q' daca $\tau = t\chi$ si ?? $\in R$
  $q\tau = qt\chi \Rightarrow ???$
- fiecare tranzitie sterge un simbol din stringul de intrare

Definitie formala *Automat finit*

- automatul A accepta/recunoaste setul de stringuri

$$L(A) = \{\tau \in T^* | q_0\tau \Rightarrow^* q, q \in F\}$$

- Doua automate A si A' sunt echivalente daca si numai daca $L(A) = L(A')$

Interpretare

- masina care citeste la intrare un input string; citeste simbol cu simbol si isi schimba starea interna
- automatul se afla in starea q cand sirul curent din derivare este $q\tau$
- automatul face o tranzitie din q in q' daca $\tau = t\chi$ si $qt \rightarrow q' \in R$ $q\tau = qt\chi \Rightarrow q'\chi$
- fiecare tranzitie sterge un simbol din stringul de intrare

Exemplu automat vazut ca sistem de rescriere

$$A = (T = \{0, 1\}, Q = \{q_0, q_1\}, R, q_0, F = \{q_1\})$$

$$\begin{aligned} R = \{ & q_0 1 \to q_1 \\ & q_0 0 \to q_0 \\ & q_1 1 \to q_0 \\ & q_1 0 \to q_1 \\ \} & \end{aligned}$$

Intrebare: 1001 apartine limbajului automatului? Dar 10?
?Exista derivarea

$$q_0 1001 \Rightarrow^* q_1$$

Table of Contents

Teorema

- Pentru fiecare gramatica regulata G exista un automat finit A a.i. $L(A) = L(G)$.

Reamintire: gramatica regulata (din ierarhia lui Chomsky)
$G = (T, N, Z, P)$

- fiecare productie are forma

$$X \rightarrow t, \quad X \in N, \ t \in T \cup \{\varepsilon\}$$

sau

$$X \rightarrow tY, \ X, Y \in N, t \in T$$

Construirea AF pentru gramatica **regulata** G

- **Algoritm** Construirea automatului $A = (T, N \cup \{f\}, R, Z, F)$, $f \notin N$ pentru gramatica $G = (T, N, Z, P)$.
  1. daca $X \to t \in P$, $X \in N, t \in T$, atunci $Xt \to f \in R$
  2. daca $X \to tY \in P$, $X, Y \in N, t \in T$, atunci $Xt \to Y \in R$
  3. $F = \{f\} \cup \{X | X \to \varepsilon \in P\}$

Exemplu: Gramatica pentru constante reale - gramatica regulata

Fie gramatica G_3

- $T = \{n, ., +, -, E\}$
- $N = \{C, F, I, X, S, U\}$
- $P = \{$

$$
\begin{aligned}
&C \to n, C \to nF, C \to .I,\\
&F \to .I, F \to ES,\\
&I \to n, I \to nX,\\
&X \to ES,\\
&S \to n, S \to +U, S \to -U,\\
&U \to n\}
\end{aligned}
$$

Exemple de derivare:

- $C \Rightarrow n$
- $C \Rightarrow .I \Rightarrow .n$
- $C \Rightarrow nF \Rightarrow n.I \Rightarrow n.nX \Rightarrow n.nES \Rightarrow n.nE + U \Rightarrow n.nE + n$

FA pentru G_3

Gramatica regulata

- $T = \{n, ., +, -, E\}$
- $N = \{C, F, I, X, S, U\}$
- $P = \{$

$C \to n, C \to nF, C \to .I,$

$F \to .I, F \to ES,$

$I \to n, I \to nX,$

$X \to ES,$

$S \to n, S \to +U, S \to -U,$

$U \to n\}$

Automat finit

- $T = \{n, ., +, -, E\}$
- $Q = \{C, F, I, X, S, U, q\}$
- $P = \{$

$Cn \to q, Cn \to F, C. \to I,$

$F. \to I, FE \to S,$

$In \to q, In \to X,$

$XE \to S,$

$Sn \to q, S+ \to U, S- \to U,$

$Un \to q\}$

- $q_0 = C$
- $F = \{q\}$

Derivare $n.n$

Gramatica

$$C \Rightarrow nF \Rightarrow n.I \Rightarrow n.n$$

Automat

$$Cn.n \Rightarrow F.n \Rightarrow In \Rightarrow q$$

Pentru orice $Z\tau\chi \Rightarrow^* X\chi \Rightarrow^* q$, $\tau, \chi \in T^*$, $X \in N$, $\tau\chi \in L(A)$, $q \in F$,
starea X specifica nonterminalul din G care ar fi trebuit utilizat pentru derivarea lui χ

Derivare $n.n$

Gramatica

$C \quad \Rightarrow nF \quad \Rightarrow n.l \quad \Rightarrow n.n$

Automat

$Cn.n \quad \Rightarrow F.n \quad \Rightarrow ln \quad \Rightarrow q$

Pentru orice $Z\tau\chi \Rightarrow^* X\chi \Rightarrow^* q$, $\tau, \chi \in T^*$, $X \in N$, $\tau\chi \in L(A)$, $q \in F$,
starea X specifica nonterminalul din G care ar fi trebuit utilizat pentru derivarea lui χ

Derivare $n.n$

Gramatica

$$C \Rightarrow nF \Rightarrow n.I \Rightarrow n.n$$

Automat

$$Cn.n \Rightarrow F.n \Rightarrow In \Rightarrow q$$

Pentru orice $Z\tau\chi \Rightarrow^* X\chi \Rightarrow^* q$, $\tau, \chi \in T^*$, $X \in N$, $\tau\chi \in L(A)$, $q \in F$,
starea X specifica nonterminalul din G care ar fi trebuit utilizat pentru derivarea lui χ

Proprietati ale automatului

Pentru orice $Z\tau\chi \Rightarrow^* X\chi \Rightarrow^* q$, $\tau, \chi \in T^*$, $X \in N$, $\tau\chi \in L(A)$, $q \in F$,
starea X specifica nonterminalul din G care ar fi trebuit utilizat pentru derivarea lui χ

Demonstratie prin inductie

- daca $\tau\chi \in L(G)$. afirmatia este adevarata pentru Z stare initiala
- proprietatea ramana adevarata pana la starea finala q, care nu genereaza alte simboluri

Fiecare propozitie din $L(G)$ apartine lui $L(A)$ si invers

Evitarea backtrackingului

- in exemplu: Automatul generat este nedeterminist: stare I cu inputul n sunt mai multe tranzitii posibile
- la implementare: backtracking necesar in cazul unei decizii incorecte
- motive pentru evitarea backtrackingului:
  - timpul necesar analizarii unui string cu backtracking poate creste exponential cu lungimea stringului (spre deosebire de cazul DFA - unde e liniar cu lungimea stringului)
  - daca automatul nu accepta stringul, stringul va fi recunoscut drept incorect. Pinpointingul (tratarea erorilor) devine dificial cu backtracking
  - deoarece in compilator, tranzitiilor de stare le sunt asociate anumite actiuni, la revenire ar trebui anularea acelor actiuni

Table of Contents

Automat finit determinist

Un automat este determinist daca fiecare derivare poate fi continuata prin cel mult o mutare.
$\rightarrow$ Determinist daca Partile stanga ale tuturor productiilor sunt distincte

Backtrackingul poate fi intotdeauna evitat cand se recunosc stringuri pentru limbaje regulate

Automat finit determinist (deterministic finite automaton

Pentru orice gramatica regulata G, exista un automat finit determinist A (DFA) a.i. $L(A) = L(G)$

Algoritm construire DFA

Idee: construim un automat pentru gramatica $G = (T, N, Z, P)$ a.i. in timpul acceptarii unei propozitii din $L(G)$, starea la fiecare pas sa mentioneze elementul N utilizat pentru a deriva restul stringului.

Daca $X \rightarrow tU$ si $X \rightarrow tV \in P$,
atunci cand t este urmatorul simbol, restul stringului poate fi derivat atat din U cat si din V
dar pentru a avea DFA, R trebuie sa contina o singura productie $Xt \rightarrow q'$
deci starea q' trebuie sa contina un set de nonterminale - acelea care puteau fi utilizate pentru derivarea restului sirului

Algoritm construire DFA pt G=(T,N,Z,P)

$$A = (T, Q, R, q_0, F), ostare\ reprezinta\ N_q \subseteq N \cup \{f\}, f \notin N$$

1. initial $Q = \{q_0\}$ si $R = \emptyset, N_{q_0} = \{Z\}$
2. pentru $q \in Q$ netratat se efectueaza pasii 3-5 pentru fiecare $t \in T$
3. fie $next(q, t) = \{U | \exists X \in N_q \text{ a.i. } X \to tU \in P\}$
4. daca exista un $X \in N_q$ a.i. $X \to t \in P$, atunci adauga f la $next(q, t)$ daca nu era deja adaugat;
   daca exista $X \in N_q$ a.i. $X \to \varepsilon \in P$ atunci adauga f la N_q
5. daca $next(q, t) \neq \emptyset$, atunci fie q' starea ce reprezinta $N_{q'} = next(q, t)$. Adauga q' la Q si $qt \to q'$ in R
6. daca toate starile din Q au fost considerate, atunci $F = \{q | f \in N_q\}$ si terminat; altfel continua cu pasul 2

Exemplu DFA pentru gramatica regulata

Fie gramatica G_3

- $T = \{n, ., +, -, E\}$
- $N = \{C, F, I, X, S, U\}$
- $P = \{$

$$
\begin{aligned}
&C \to n, C \to nF, C \to .I,\\
&F \to .I, F \to ES,\\
&I \to n, I \to nX,\\
&X \to ES,\\
&S \to n, S \to +U, S \to -U,\\
&U \to n\}
\end{aligned}
$$

	n	.	$+$	$-$	E	N
q_0						$\{C\}$
...						

$next(q_0, n) = \{F, f\}$...

DFA

	n	.	$+$	$-$	E	N
q_0	q_1	q_2				$\{C\}$
q_1		q_2			q_3	$\{f, F\}$
q_2	q_4					$\{I\}$
q_3	q_5		q_6	q_6		$\{S\}$
q_4					q_3	$\{f, X\}$
q_5						$\{f\}$
q_6	q_5					$\{U\}$

- $T = \{n, ., +, -, E\}$, $F = \{q_1, q_4, q_5\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$
- $P =$

$$q_0 n \rightarrow q_1, q_0. \rightarrow q_2,$$
$$q_1. \rightarrow q_2, q_1 E \rightarrow q_3,$$
$$q_2 n \rightarrow q_4$$
$$q_3 n \rightarrow q_5, q_3+ \rightarrow q_6, q_3- \rightarrow q_6,$$
$$q_4 E \rightarrow q_3$$
$$q_6 n \rightarrow q_5\}$$

DFA

	n	.	$+$	$-$	E	N
q_0	q_1	q_2				$\{C\}$
q_1		q_2			q_3	$\{f, F\}$
q_2	q_4					$\{I\}$
q_3	q_5		q_6	q_6		$\{S\}$
q_4					q_3	$\{f, X\}$
q_5						$\{f\}$
q_6	q_5					$\{U\}$

- $T = \{n, ., +, -, E\}$, $F = \{q_1, q_4, q_5\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$
- $P =$

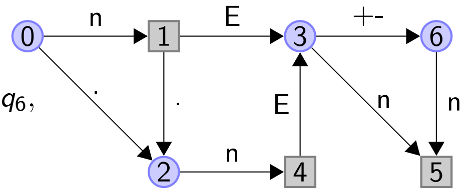
$$\begin{aligned} &q_0 n \to q_1, q_0 . \to q_2, \\ &q_1 . \to q_2, q_1 E \to q_3, \\ &q_2 n \to q_4 \\ &q_3 n \to q_5, q_3 + \to q_6, q_3 - \to q_6, \\ &q_4 E \to q_3 \\ &q_6 n \to q_5 \} \end{aligned}$$

Diagrama de stare

Fie $T = \{n, ., +, -, E\}$, $F = \{q_1, q_4, q_5\}$
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$P = \{q_0 n \rightarrow q_1, q_0. \rightarrow q_2,$
$q_1. \rightarrow q_2, q_1 E \rightarrow q_3,$
$q_2 n \rightarrow q_4$
$q_3 n \rightarrow q_5, q_3 + \rightarrow q_6, q_3 - \rightarrow q_6,$
$q_4 E \rightarrow q_3$
$q_6 n \rightarrow q_5\}$

Diagrama de stare



cale care incepe in q_0 si se termina intr-o stare finala $\in L(A)$

n	.n	n.n
nEn	nE+n	nE-n
.nEn	.nE+n	.nE-n
n.nEn	n.nE+n	n.nE-n

Comparatie DFA FA pentru gramatica regulata

- $T = \{n, ., +, -, E\}$, $F = \{q_1, q_4, q_5\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$
- $P = \{$

$q_0 n \to q_1, q_0. \to q_2,$

$q_1. \to q_2, q_1 E \to q_3,$

$q_2 n \to q_4$

$q_3 n \to q_5, q_3 + \to q_6, q_3 - \to q_6,$

$q_4 E \to q_3$

$q_6 n \to q_5\}$

q	N_q	q	N_q
0	C	1	f, F
2	I	3	S
4	f, X	5	f

- $T = \{n, ., +, -, E\}, F = \{q\}$
- $Q = \{C, F, I, X, S, U, q\}$
- $P = \{$

$Cn \to q, Cn \to F, C. \to I,$

$F. \to I, FE \to S,$

$In \to q, In \to X,$

$XE \to S,$

$Sn \to q, S+ \to U, S- \to U,$

$Un \to q\}$

Table of Contents

Diagrama de stare

Fie $A = (T, Q, R, q_0, F)$ un automat finit,

- $D = \{(q, q')|\exists t, qt \rightarrow q' \in R\}$,
- $f : (q, q') \rightarrow \{t|qt \rightarrow q' \in R)$ o mapare de la D la $P(T)$

Graful directionat (Q, D) cu etichetele muchiilor $f((q, q'))$ este diagrama de stare a automatului A

Table of Contents

Teorema

Pentru fiecare automat finit A exista o gramatica regulata G a.i. $L(A) = L(G)$

Din automatul $A = (T, Q, R, q_0, F)$ construim gramatica $G = (T, Q, q_0, P)$:

$$P = \{q \to tq' | qt \to q' \in R\} \cup \{q \to \varepsilon | q \in F\}$$

Gramatici pentru automat

$$F = \{q_1, q_4, q_5\} \quad P = \{ \begin{array}{l} q_0 n \to q_1, q_0. \to q_2, \\ q_1. \to q_2, q_1 E \to q_3, \\ q_2 n \to q_4 \\ q_3 n \to q_5, q_3+ \to q_6, q_3- \to q_6, \\ q_4 E \to q_3 \\ q_6 n \to q_5 \} \end{array}$$

Productii gramatica

$$\begin{array}{l} q_0 \to nq_1|.q_2, \\ q_1 \to .q_2|Eq_3|\varepsilon, \\ q_2 \to nq_4 \\ q_3 \to nq_5| + q_6| - q_6, \\ q_4 \to Eq_3|\varepsilon \\ q_5 \to \varepsilon \\ q_6 \to nq_5 \} \end{array}$$

Productii gramatica fara productii ε

$$\begin{array}{l} q_0 \to n|nq_1|.q_2, \\ q_1 \to .q_2|Eq_3, \\ q_2 \to n|nq_4 \\ q_3 \to n| + q_6| - q_6, \\ q_4 \to Eq_3 \\ q_6 \to n\} \end{array}$$

Gramatici pentru automat

$F = \{q_1, q_4, q_5\}$ $P = \{$

$$q_0 n \to q_1, q_0. \to q_2,$$
$$q_1. \to q_2, q_1 E \to q_3,$$
$$q_2 n \to q_4$$
$$q_3 n \to q_5, q_3+ \to q_6, q_3- \to q_6,$$
$$q_4 E \to q_3$$
$$q_6 n \to q_5\}$$

Productii gramatica

$$q_0 \to nq_1|.q_2,$$
$$q_1 \to .q_2|Eq_3|\varepsilon,$$
$$q_2 \to nq_4$$
$$q_3 \to nq_5| + q_6| - q_6,$$
$$q_4 \to Eq_3|\varepsilon$$
$$q_5 \to \varepsilon$$
$$q_6 \to nq_5\}$$

Productii gramatica fara productii ε

$$q_0 \to n|nq_1|.q_2,$$
$$q_1 \to .q_2|Eq_3,$$
$$q_2 \to n|nq_4$$
$$q_3 \to n| + q_6| - q_6,$$
$$q_4 \to Eq_3$$
$$q_6 \to n\}$$

- Pentru orice gramatica regulata G, exista un automat finit A a.i. $L(A) = L(G)$
- Pentru fiecare automat finit A exista o gramatica regulata G a.i. $L(A) = L(G)$

Gramaticile regulate si automatele finite sunt echivalente

Table of Contents

DFA vs NFA

Ambele automate, deterministe si nedeterministe, sunt capabile sa recunoasca toate limbajele regulate:

$$L(NFA) = L(DFA)$$

Diferenta principala: spatiu vs timp:

- DFA sunt mai rapide decat NFA
- DFA sunt exponential mai mari decat NFA

FA sunt folosite ca mdoele pentru:

- software for designing digital circuits
- lexical analyzer of a compiler
- software for verifying finite state systems, such as communication protocols: exemplul cu planeta

Rezumat

Introducere in automate

- Definitie formala - automat determinist
- Definitie formala - automat nedeterminist

Gramatici regulate si automate finite

- Definitie automat ca sistem de rescriere + Limbaj automat
- Pentru orice gramatica regulata exista un automat finit cu acelasi limbaj
- Pentru orice gramatica regulata exista un automat finit determinist cu acelasi imbaj
- Diagrama de stare - definire formala
- Pentru fiecare automat finit exista o gramatica regulata cu acelasi limbaj
- Automat finit determinist - automat finit nedeterminist - limbaje regulate

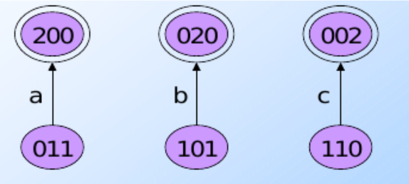
Extended Example - ullman slides - Optional - un exemplu interesant modelat cu automate

- On a distant planet, there are three species, a, b, and c. Any two different species can mate. If they do:
  1. The participants die.
  2. Two children of the third species are born.
- The planet fails if at some point all individuals are of the same species. Then, no more breeding can take place.
- State = sequence of three integers the numbers of individuals of species a, b, and c.

- In a given state, must the planet eventually fail?
- In a given state, is it possible for the planet to fail, if the wrong breeding choices are made?

$a - event$ - individuals of species b and c breed and are replaced by two a's

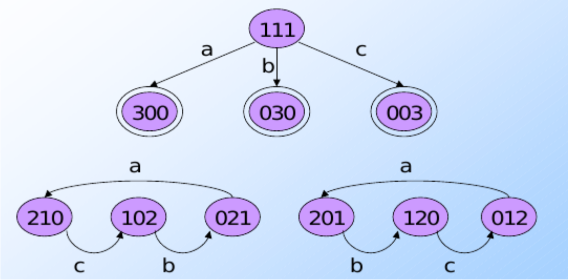
idee: state = #a#b#c



Toate starile sunt **must-fail**

- In a given state, must the planet eventually fail?
- In a given state, is it possible for the planet to fail, if the wrong breeding choices are made?

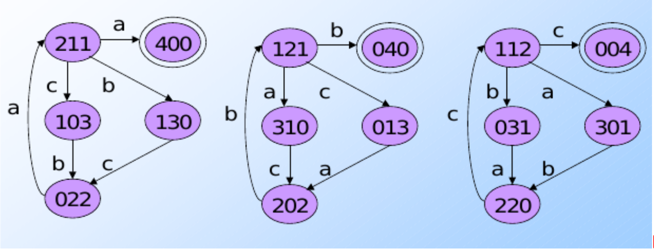
$a - event$ - individuals of species b and c breed and are replaced by two a's
idee: state = #a#b#c



4 stari sunt **must-fail**, celelalte sunt **can't fail**

- In a given state, must the planet eventually fail?
- In a given state, is it possible for the planet to fail, if the wrong breeding choices are made?

$a -$ *event* - individuals of species b and c breed and are replaced by two a's
idee: state $= \#a\#b\#c$



must fail

dar, si **might fail**

- In a given state, must the planet eventually fail?
- In a given state, is it possible for the planet to fail, if the wrong breeding choices are made?

a — *event* - individuals of species *b* and *c* breed and are replaced by two *a*'s
idee: state = #a#b#c