ZNAČAJ ANALIZE JEZIKA

Dva računala (programabilna stroja) su ekvivalentna

- ako mogu riješiti isti skup (klasu) problema,
- bez obzira koliko vremena trebalo pojedinačnom računalu.

Rješenje problema je preslikavanje problema u rezultat.

Problem i rezultat **su opisani** ulaznim i izlaznim nizovima znakova - riječima

Stoga se rješenje problema može **modelirati**

- ispitivanjem da li riječ koja opisuje problem pripada skupu riječi - jeziku L (Language).

Postavlja se pitanje:

- kako prepoznati pripadnost riječi jeziku L

Koristimo strojeve:

- automate određenih sposobnosti (vrste) za konkretan skup (vrstu) jezika
- određene strukture za konkretan jezik.

PROBLEM IZRAČUNLJIVOSTI I KOMPLEKSNOSTI

Vrijeme postizanja rješenja (obrade):

- kako usporediti različite postupke postizanja istih rješenja?

Koristimo jedinicu vremena

- u smislu izvršenja jednog koraka Turingovog stroja (TM)
- pokušavamo izračunati odnos između broja koraka i duljine ulaznog niza znakova
- taj odnos označavamo s O(g(n)), n = duljina ulaznog niza.

Ako postoji rješenje

- pitanje je da li je ostvarivo u stvarnom vremenu
- raspoloživom tehnologijom

Prihvatljivost rješenja ovisi o prognozi vremena izvršenja:

ovisnost	formula	prognoza
beskonačno	$O(\propto)$	nema algoritamskog rješenja
eksponencijalna:	$O(k^n)$	loša
polinomska	$O(n^k)$	dobro
linearna	O(kn)	vrlo dobro
logaritamska	O(log(n))	izvrsno

Vrste strojeva za odlučivanje:

Pitanja glase:

- da li je niz član jezika L (TR, Turing Recognizable)
- da li niz ne pripada jeziku L (co-TR, Complementary Turing Recognizable)

Nekad u konačnom O() vremenu možemo odgovoriti samo na jedno ili ni na jedno pitanje

Formalno, pitamo da li niz $x \in L$ je TR i pritom nije co-TR.

TR	co-TR	odluka
0	0	ne postoji
0	1	ne postoji
1	0	ne postoji
1	1	postoji, eksponencijalno

Ukratko, problem kompleksnosti jednostavno rješavamo modelima na bazi Turingovog stroja.

Ponovimo:

Nadskup skupa S, simbolički 2^S, je skup svih pravih podskupova uključujući prazni skup {} i sam S. Nadskup ima 2^S članova jer svaki pravi podskup možemo prikazati kodnom riječi pripadnosti članova, npr. $S_1(\{a,b,c\}) = S_{001}(\{a,b,c\}) = \{a\}$. "001" je karakteristična sekvenca.

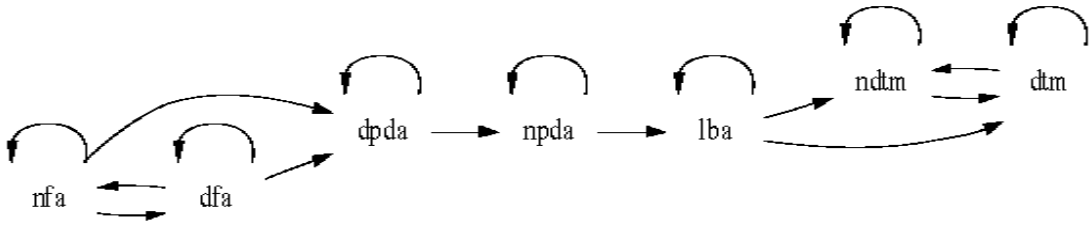
TAKSONOMIJA AUTOMATA

(nazivlje i sistematizacija)

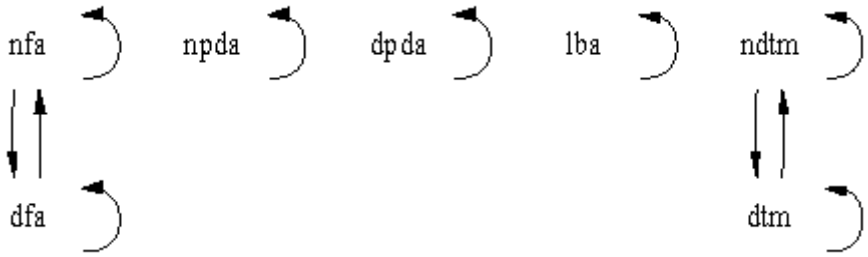
MT = {dfa; nfa; dpda; npda; lba; dtm; ndtm}

DFA (Deterministic Finite Automata)
NFA (Non Deterministic Finite Automata)
DPDA (Deterministic Push Down Automata)
NPDA (Non Deterministic Push Down Automata)
LBA (Linear Bounded Automata)
DTM (Deterministic Turing Machine)
NDTM (Non Deterministic Turing Machine)

Odnos "snage" raznih vrsta automata



Ekvivalentnost raznih vrsta automata



JEZICI I AUTOMATI

Jezik L je skup sekvenci izgrađenih od članova skupa simbola Σ.

Za neki Σ jezika može biti beskonačno, jer karakteristična sekvenca (kodna riječ pripadnosti sekvenci jeziku) može biti beskonačna

Operacije među jezicima

Unija $L_1 \cup L_2 = \{x \mid x \in L_1 \vee x \in L_2\}$

Presjek $L_1 \cap L_2 = \{x \mid x \in L_1 \wedge x \in L_2\}$

Razlika $L_1 \setminus L_2 = \{x \mid x \in L_1 \wedge x \notin L_2\}$

Simetrična razlika $(A \setminus B) \cup (B \setminus A)$

Pripajanje $L_1 \circ L_2 = \{xy \mid x \in L_1 \wedge y \in L_2\}$

Eksponenciranje $L^n = \{x_1 \dots x_n \mid x_i \in L\}$

Iteracija (Kleene star) $L^* = \{x \mid \exists k \in N : x \in L^k\}$

Negacija $\overline{L} = \{x \mid x \in \Sigma^* \setminus L\}.$

DETERMINIRANI KONAČNI AUTOMATI (DFA)

zadan je petorkom

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

- Q, a *finite nonempty* set of states;
- Σ, a *finite nonempty* alphabet;
- $\delta : Q \times \Sigma \to Q$, a *total* transition function;
- $q_0 \in Q$, an initial state; and
- $F \subseteq Q$, a *finite, possibly empty* set of final states.

Primjer tablice prijelaza i izlaza (F skup akceptorskih stanja)

	input	
state	0	1
I	I	F
*F	I	F

DFA odgovaraju regularni jezici

$$\mathcal{L}(d) = \{x \mid \exists q \in F : (q_0, x) \vdash^* (q, \varepsilon)\}$$

MANE DFA:

1. mana DFA: ne postoji DFA za:

$$L = \{(^n)^n \mid n \geq 0\}$$

L nije regularan, treba nam PDA!

2. mana DFA: prekompliciran, koristimo NFA:

RE = regular expression:

RE $r_1 r_2$	denotes	$\mathcal{L}(r_1) \circ \mathcal{L}(r_2)$	encoded in grail as	`r1 r2`
RE $r_1 + r_2$	denotes	$\mathcal{L}(r_1) \cup \mathcal{L}(r_2)$	encoded in grail as	`r1+r2`
RE r^*	denotes	$(\mathcal{L}(r))^*$	encoded in grail as	`r*`
RE (r)	denotes	$\mathcal{L}(r)$	encoded in grail as	`(r)`

Za jezik:

$$L_k = \{x0z \mid x \in \{0,1\}^* \; and \; z \in \{0,1\}^k\}$$

sekvence izražene RE su:

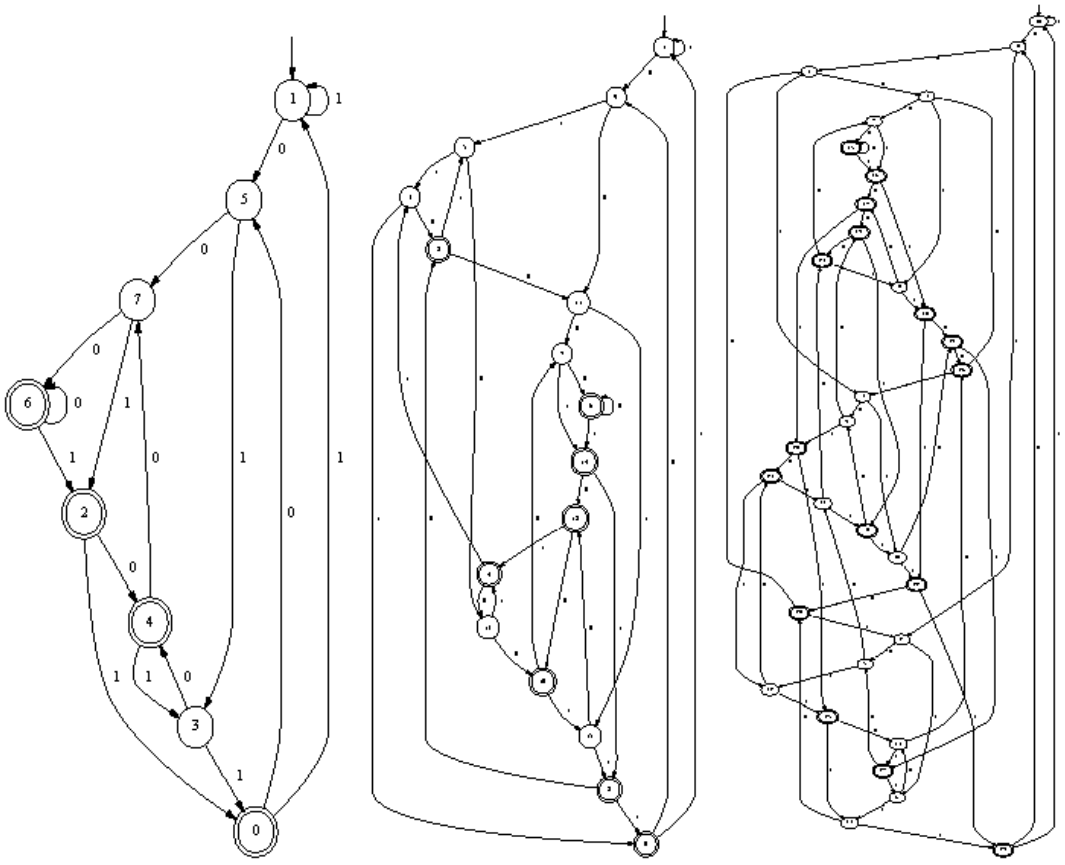
$k = 0$: `(0+1)*0`

$k = 1$: `(0+1)*0(0+1)`

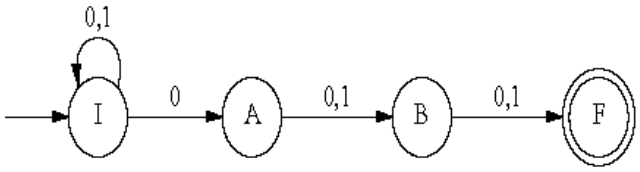
$k = 2$: `(0+1)*0(0+1)(0+1)`

$k = 3$: `(0+1)*0(0+1)(0+1)(0+1)`

Automati koji ih prepoznaju za k=2,3,4 su:



NFA za gornji slučaj i k=2 je:



NEDETERMINIRANI KONAČNI AUTOMATI (DFA)

zadan je petorkom

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

gdje je

Q, a *finite nonempty* set of states;

Σ, a *finite nonempty* alphabet;

$\delta : Q \times \Sigma_\varepsilon \to 2^Q$, a *total* transition function. (Note, however, that the range includes ϕ; thus, if for some state q, $\delta(q, x) = \phi$, the move from state q upon input x is undefined.)

$q_0 \in Q$, an initial state; and

$F \subseteq Q$, a *finite, possibly empty* set of final states.

Primjer:

Q ={I,A,B,F}
Σ ={0,1}
q_0 =I
F =F
$\delta : \{I, A, B, F\} \times \{0, 1, \varepsilon\} \to 2^{\{I,A,B,F\}}$

Input / States	0	1	e
I	{I,A}	{ I }	{ }
A	{ B }	{ B }	{ }
B	{ F }	{ F }	{ }
F	{ }	{ }	{ }

PRETVORBA NFA to DFA:

NFA

States	Input 0	1	e
I	{I,A}	{ I }	{ }
A	{ B }	{ B }	{ }
B	{ F }	{ F }	{ }
F	{ }	{ }	{ I }

DFA

States	Input 0	1
{I}	{I,A}	{ I }
{I,A}	{I,A,B}	{I,B}
{I,B}	{I,A,F}	{I,F}
{I,A,B}	{I,A,B,F}	{I,B,F}
{I,A,B,F}	{I,A,B,F}	{I,B,F}
{I,B,F}	{I,A,F}	{I,F}
{I,A,F}	{I,A,B}	{I,B}
{I,F}	{I,A}	{I}

TAKSONOMIJA JEZIKA I STROJEVA:

Machines	Languages	Nature of grammar
DFA/NFA	Regular	Left-linear productions only OR Right-linear productions only
DPDA	Deterministic CFL	Each left-hand side has one non-terminal (the productions are deterministic)
NPDA (or ``PDA'')	CFL	Each left-hand side has one non-terminal
LBA	Context Sensitive Languages	Left-hand sides may have length > 1 but |LHS| <= |RHS| - ignoring epsilon productions
DTM/NDTM	Recursively Enumerable	General grammars (|LHS| >= |RHS| allowed)