بسم الله الرحمن الرحیم

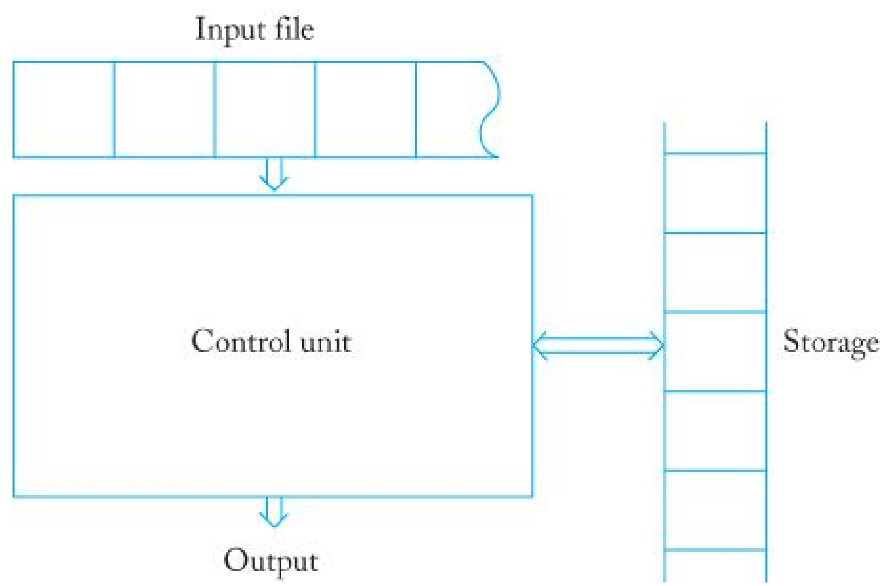
Formal Languages & Automata

Ali Shakiba
Vali-e-Asr University of Rafsanjan
<ali.shakiba@vru.ac.ir>

Chapter 2: Finite Automata

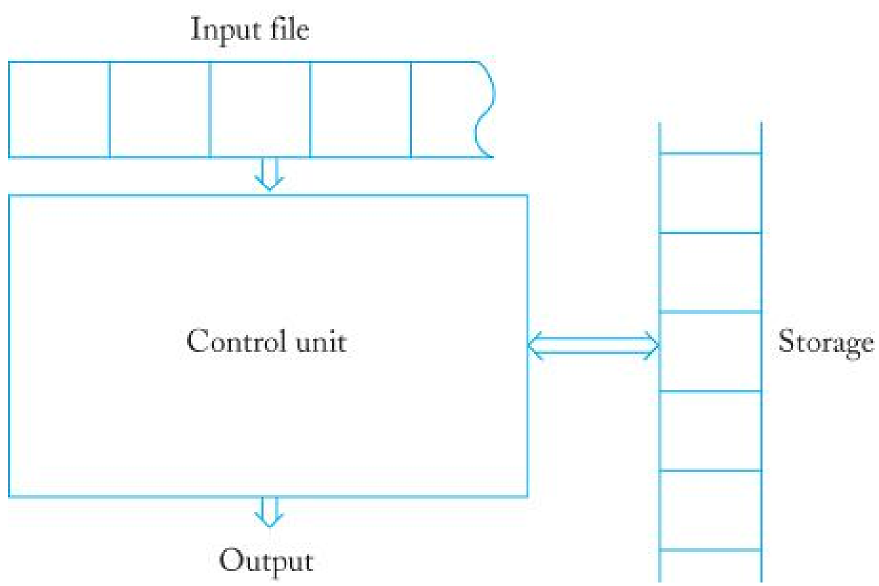
Automata

- An automaton is an abstract model of a digital computer.
  - Plural: automata



Transition Function

- The transition function determines the next state of the control unit based on:
  - the current state
  - the current input symbol
  - information currently in the temporary storage



Automaton Configuration

- A configuration refers to a particular state of:
  - the control unit
  - the input
  - the temporary storage
- The transition of the automaton from one configuration to the next is a move.

Acceptors and Transducers

- An automaton whose output is limited to "yes" or "no" for any given input is an acceptor.
  - It either accepts the input string or not.
- An automaton that can produce any string of symbols as output is a transducer.

Deterministic vs. Nondeterministic

- Deterministic automaton
  - Each move is uniquely determined by the current configuration.
- Nondeterministic automaton
  - At each point, the automaton can have several possible moves

The relationship between deterministic and nondeterministic automata will play a significant role in our study of formal languages and computation.

Deterministic Acceptors

- A deterministic finite acceptor (DFA) is the quintuple
$$M = (Q, \Sigma, \delta, q_0, F)$$
  where:
  - Q is the finite set of internal states
  - Σ is the input alphabet, a finite set of symbols
  - $\delta: Q \times \Sigma \rightarrow Q$ is a total transition function
  - $q_0 \in Q$ is the initial state
  - $F \subseteq Q$ is a set of final states

Total function: A function that is defined for all inputs of the right type (*i.e.*, for all inputs from a given domain).

DFA Operation

- At the initial time:
  - In internal state q_0
  - Input mechanism on the leftmost input symbol

- During each move:
  - Consume one input symbol by advancing the input one symbol to the right.

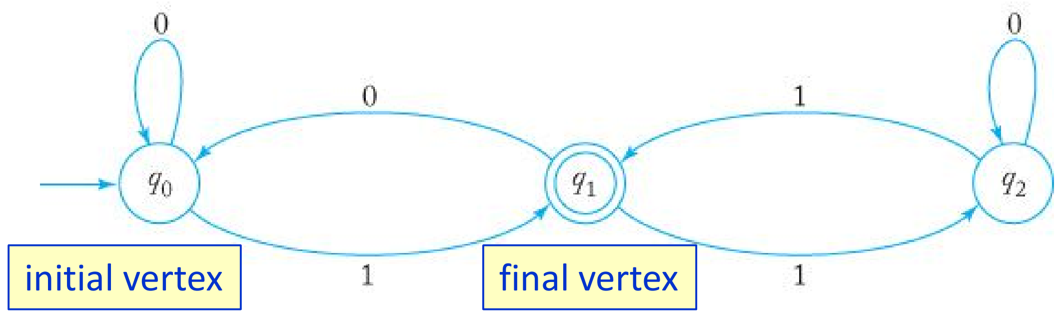
DFA Operation, *cont'd*

- The transition from one internal state to another is governed by the transition function

- Example:
  - If the automaton is at the initial state q_0, and
  - If the input symbol is a, and
  - If $\delta(q_0, a) = q_1$, then the DFA will go to state q_1.

- We can visualize and represent a finite automaton with a transition graph.

Transition Graph Example



$$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

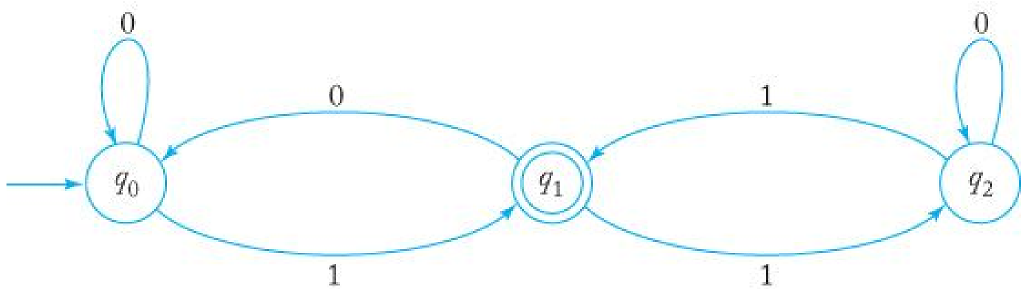
where δ is given by

$\delta(q_0, 0) = q_0$	$\delta(q_0, 1) = q_1$
$\delta(q_1, 0) = q_0$	$\delta(q_1, 1) = q_2$
$\delta(q_2, 0) = q_2$	$\delta(q_2, 1) = q_1$

DFA Operation, *cont'd*

- The automaton accepts its input string if:
  - The automaton reaches the end of the input string.
  - And it's in one of its final states.
- Otherwise, the automaton rejects the string.

Transition Graph Example, *cont'd*



Will this automaton accept or reject?

- 01
- 00
- 101
- 0111
- 11001
- 100
- 1100

JFLAP demo

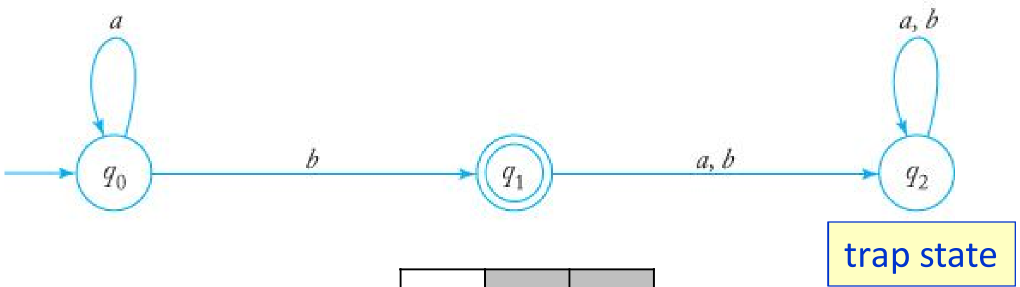
Extended Transition Function

$$\delta^*: Q \times \Sigma^* \to Q$$

- Transition on a string rather than a single symbol.
  - Do a regular transition on each symbol of the string.

- Give the state after reading the entire string.

State Transition Matrix



	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

A Practical Application

- You are given the text of the novel *War and Peace* as a plain text file.

- Write a program to search the text for these names:
  - Boris Drubetskoy
  - Joseph Bazdeev
  - Makar Alexeevich

- For each name found, print the line number and the position within the line.
  - Line and position numbers start with 1

Mehr, 13th, 1395

A Practical Application, *cont'd*

- A name can span two lines.
  - First name at the end of one line.
  - Last name at the beginning of the next line.

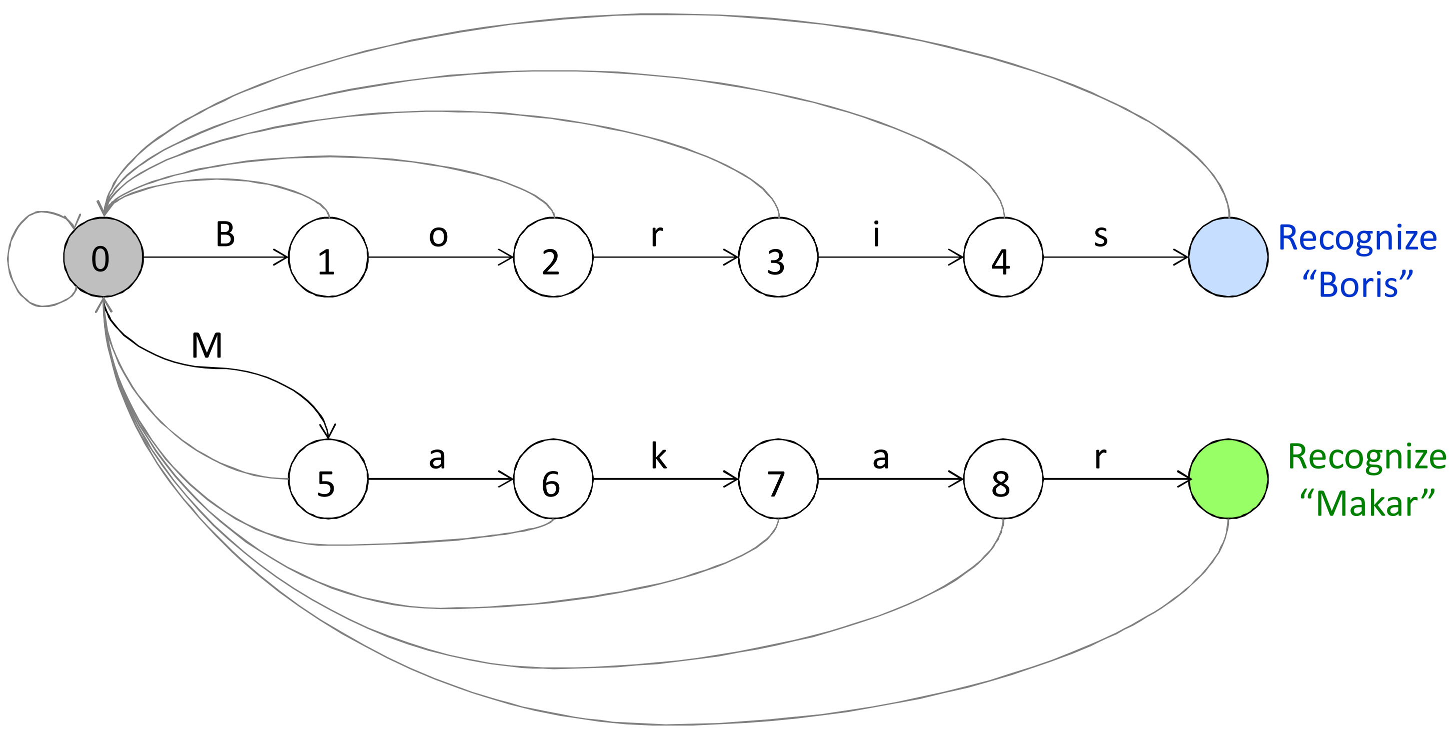
- How efficiently can this program run?
  - Run and time the program 10 times and time each run.
  - Print the minimum, maximum, and median times (in milliseconds).
  - To compare among different machines, also calculate the "performance number":

$$(\text{run time in ms}) \times (\text{processor speed in GHz})$$

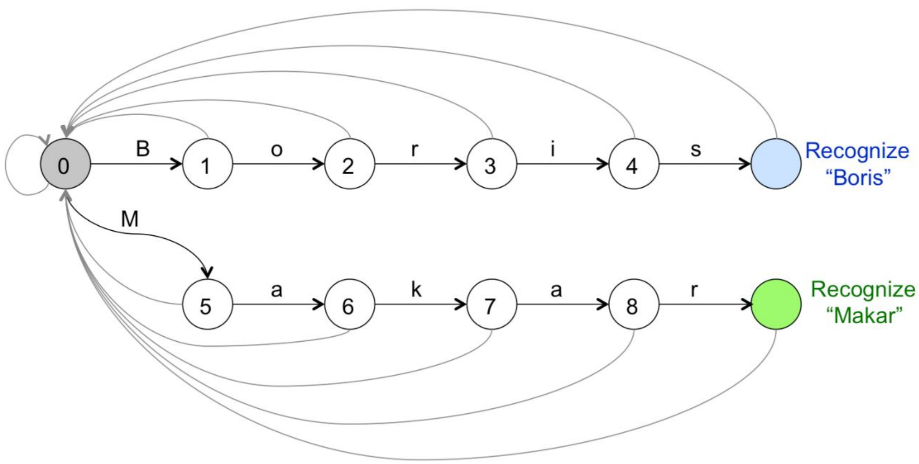
(lower performance numbers are better)

A Practical Application, *cont'd*

- State transition diagram to recognize "Boris" and "Makar":



A Practical Application, *cont'd*



The state transition matrix

		B	M	a	i	k	o	r	s
0	0	1	5	0	0	0	0	0	0
1	0	0	0	0	0	0	2	0	0
2	0	0	0	0	0	0	0	3	0
3	0	0	0	0	4	0	0	0	0
4	0	0	0	0	0	0	0	0	
5	0	0	0	6	0	0	0	0	0
6	0	0	0	0	0	7	0	0	0
7	0	0	0	8	0	0	0	0	0
8	0	0	0	0	0	0	0		0

A Practical Application, *cont'd*

```
private static final int MATRIX[][] = {

    // Starting state 0

    /*    other,A,B,D,J,M,a,b,c,d,e,h,i,k,l,o,p,r,s,t,u,v,x,y,z,sp,\n */
    /*  0 */ {0,0,1,0,16,29,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

    // Boris Drubetskoy

    /*    other,A,B,D,J,M,a,b,c,d,e,h,i,k,l,o,p,r,s,t,u,v,x,y,z,sp,\n */
    /*  1 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0},
    /*  2 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0},
    /*  3 */ {0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    /*  4 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0},
    /*  5 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6},
    /*  6 */ {0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    /*  7 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,0,0,0,0,0},
    /*  8 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,0,0,0,0,0,0},
    /*  9 */ {0,0,0,0,0,0,0,10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    /* 10 */ {0,0,0,0,0,0,0,0,0,0,11,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    /* 11 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,12,0,0,0,0,0,0,0},
    /* 12 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,0,0,0,0,0,0,0,0},
    /* 13 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,14,0,0,0,0,0,0,0,0,0,0,0,0,0},
    /* 14 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,15,0,0,0,0,0,0,0,0,0,0,0},
    /* 15 */ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,BD,0,0,0},


   …
};
```

A Practical Application, *cont'd*

```
private int index(char ch)
{
    switch (ch) {
        case 'A'  : return 1;
        case 'B'  : return 2;
        case 'D'  : return 3;
        case 'J'  : return 4;
        case 'M'  : return 5;
        case 'a'  : return 6;
        case 'b'  : return 7;
        case 'c'  : return 8;
        case 'd'  : return 9;
        case 'e'  : return 10;
        case 'h'  : return 11;
        case 'i'  : return 12;
        case 'k'  : return 13;
        case 'l'  : return 14;
        case 'o'  : return 15;
        case 'p'  : return 16;
        case 'r'  : return 17;
        case 's'  : return 18;
        case 't'  : return 19;
        case 'u'  : return 20;
        case 'v'  : return 21;
        case 'x'  : return 22;
        case 'y'  : return 23;
        case 'z'  : return 24;
        case ' '  : return 25;
        case '\n' : return 26;
        default   : return 0;
    }
}
```

Languages and DFAs

- Recall that an acceptor is an automaton that either accepts or rejects input strings.

- The set of all strings that the DFA
$$M = (Q, \Sigma, \delta, q_0, F)$$
accepts constitutes the language
$$L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$$

- The DFA represents the language's rules.

DFA and Associated Transition Graph

- If we have a DFA

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

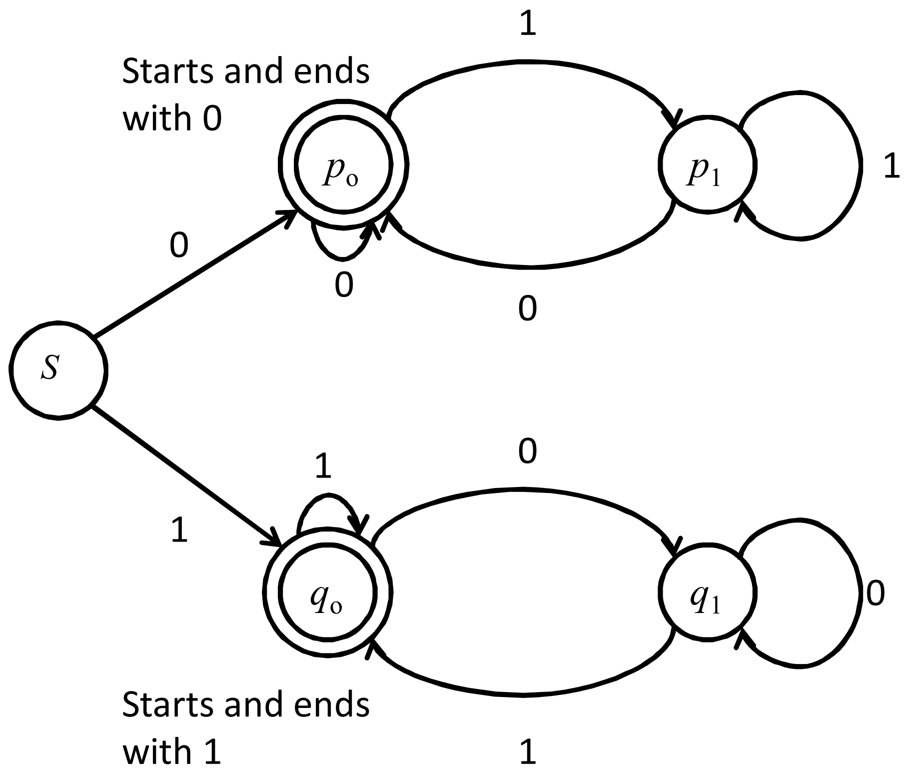
and its associated transition graph G_M, can we treat them both equally?

- Theorem 2.1 of the textbook basically says yes.
  - For every q_i, q_j in Q, and w in Σ^+:
  - $\delta^*(q_i, w) = q_j$ if and only if there is a path labeled w in G_M from q_i to q_j.
  - Proof by induction on the length of w.

DFA Example #1

- Create a DFA that accepts all strings on {0, 1} that begin and end with the same symbol.
- How can an automaton remember what was the beginning symbol of a string?
- Have a different set of states depending on the first symbol!

DFA Example #1, *cont'd*



DFA Example #2

- Create a DFA that accepts all strings on {0, 1} that do <u>not</u> contain the substring 001.

- The basic idea is that if the automaton ever reads 001, it should be in a non-final state.
  - Actually, that state should be a trap.

- How can the automaton remember the previous two symbols whenever it reads a 1?

DFA Example #2, *cont'd*

- Again, we must accomplish this with states:
  - A state for having read a 0.
  - A state for having read 00.
  - A state for having read 001.

- We can label the states accordingly.