2.1 Finite Automata: Examples and Definitions

- A finite automaton is a simple type of computer.
  - Its output is limited to “yes” or “no”.
  - It has very primitive memory capabilities.

- For this chapter, consider that:
  - The input comes in the form of a string of individual input symbols.
  - The computer gives an answer for the current string.
Finite Automata: Examples and Definitions (cont’d.)

- A *Finite Automaton* (FA) or *finite state machine* is always in one of a finite number of *states*.
- At each step it makes a move that depends only on the state it’s currently in and the input symbol it gets.
- The move is to enter a particular state.
- States are either *accepting* or *nonaccepting*:
  - Entering an accepting state means answering “yes”.
  - Entering a nonaccepting state means “no”.
- An FA has an *initial state*, which is an accepting state if and only if the language the FA accepts includes $\Lambda$.

 finite Automata: Examples and Definitions (cont’d.)

- An FA can be described by the set of states, the input alphabet, the initial state, the set of accepting states, and a *transition function*.
- This can be described by a transition diagram or a transition table.
Finite Automata: Examples and Definitions (cont’d.)

The program accepts a string if the process ends in a double circle.
Finite Automata: Examples and Definitions (cont'd.)

- This FA accepts the language of strings that end in $aa$.

- This FA accepts strings ending with $b$ and not containing $aa$. 
Finite Automata: Examples and Definitions (cont’d.)

- This FA accepts strings that contain $abbaab$.

- What do we do when a prefix of $abbaab$ has been read but the next symbol doesn’t match?
  - Go back to the state representing the longest prefix of $abbaab$ at the end of the new current string.

Finite Automata: Examples and Definitions (cont’d.)

- An FA that accepts binary representation of integers divisible by 3
  - States 0, 1, and 2 represent the current “remainder”
  - The initial state is non-accepting: at least one bit is required
  - Leading zeros are prohibited
Finite Automata: Examples and Definitions (cont’d.)

- FAs are ideally suited for lexical analysis, the first stage in compiling a computer program.

- A lexical analyzer takes a string of characters and provides a string of “tokens”.
  - Tokens: reserved words, punctuation symbols, identifiers, operators, numeric literals, ...... separated by spaces.
  - Accepting states represent scanned tokens; each accepting state represents a category of token.

Finite Automata: Examples and Definitions (cont’d.)

- Tokens: identifiers, semicolons, =, aa, numerical literals (digits + decimal point)
  - D: any digit
  - L: a lowercase letter other than a
  - M: D or L
  - N: D or L or a
  - Δ: a space
  - All transitions not shown explicitly go to an error state and stay there
Finite Automata: Examples and Definitions (cont’d.)

- Definition: A finite automaton is a 5-tuple $M = (Q, \Sigma, q_0, A, \delta)$, where:
  - $Q$ is a finite set of states
  - $\Sigma$ is a finite input alphabet
  - $q_0 \in Q$ is the initial state
  - $A \subseteq Q$ is the set of accepting states
  - $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
- From state $q$ the machine will move to state $\delta(q, \sigma)$ if it receives input symbol $\sigma$.

Build an automaton that accepts all and only those strings that contain 001.
The notation $\delta^*(q, x)$ describes the state the FA is in after starting in state $q$ and receiving input string $x$. The extended transition function $\delta^*: Q \times \Sigma^* \rightarrow Q$ is defined recursively as follows:

1. For every $q \in Q$, $\delta^*(q, \Lambda) = q$
2. For every $q \in Q$, every $y \in \Sigma^*$, and every $\sigma \in \Sigma$, $\delta^*(q, y\sigma) = \delta(\delta^*(q, y), \sigma)$

Evaluate $\delta^*(q_0, baa)$ in the following FA:

$\delta^*(q_0, baa) = \delta(\delta^*(q_0, ba), a) = \delta(\delta(\delta^*(q_0, b), a), a)$
$= \delta(\delta(\delta(q_1, \Lambda), b), a), a)$
$= \delta(\delta(q_0, b), a), a) = \delta(\delta(q_0, a), a)$
$= \delta(q_1, a) = q_1$
Finite Automata: Examples and Definitions (cont’d.)

- Definition: Let $M=(Q, \Sigma, q_0, A, \delta)$ be a FA, and let $x \in \Sigma^*$. Then $x$ is accepted by $M$ if $\delta^*(q_0, x) \in A$ and rejected otherwise.

- The language accepted by $M$ is $L(M) = \{x \in \Sigma^* \mid x$ is accepted by $M\}$

---

Finite Automata: Examples and Definitions (cont’d.)

- $L(M) = \{\varepsilon \mid \varepsilon$ is a string of 0s and 1s$\}$

- $L(M) = \{\varepsilon\}$
Design an FA to accept the language
\[ L = \{ w | w \text{ has both an even number of 0's and an even number of 1's} \} \]

\[ M = (Q, \Sigma, q_0, A, \delta) \]
- \( Q = \{q_0, q_1, q_2, q_3\} \)
- \( \Sigma = \{0, 1\} \)
- \( A = \{q_0\} \)

Transition Table:

<table>
<thead>
<tr>
<th>State</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( q_2 )</td>
<td>( q_1 )</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( q_2 )</td>
<td>( q_0 )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( q_0 )</td>
<td>( q_3 )</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( q_1 )</td>
<td>( q_2 )</td>
</tr>
</tbody>
</table>

DFS Graph: