FINITE STATE MACHINES
(AUTOMATA)
Think about the On/Off button
Switch Example 1

The corresponding Automaton

Input:

Push Push Push Push Push
Vending Machine Example 2

Vending machine dispenses Cola for $0.45
Vending Machine Example 2

Input: D Q Q Q D

45 is final state
Input Accepted
The machine accepts a string if the process ends in a final state.
The machine accepts a string if the process starts in the start state ($q_0$) and ends in a final state ($F$).
An alphabet $S$ is a finite set of symbols (in Ex3, $S = \{0,1\}$)

A string over $S$ is a finite sequence of elements of $S$ (e.g. 0111)

For a string $s$, $|s|$ is the length of $s$

The unique string of length 0 will be denoted by $\epsilon$ and will be called the empty string

The reversal of a string $u$ is denoted by $u^R$. Example: $(\text{banana})^R = \text{ananab}$
The *concatenation* of two strings is the string resulting from putting them together from left to right. Given strings $u$ and $v$, denote the concatenation by $u \cdot v$, or just $uv$.

**Example:**

\[ \text{jap . an} = \text{japan}, \ \text{QQ . DD} = \text{QQDD} \]

**Q1:** What’s the Java equivalent of concatenation?

**Q2:** Find a formula for $|u \cdot v|$?

<table>
<thead>
<tr>
<th>The + operator on strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
</tr>
</tbody>
</table>
If S is an alphabet, 
\( S^* \) denotes the set of all strings over S.

A **language** over S is a subset of \( S^* \)

i.e. a set of strings *each* consisting of sequences of symbols in S.
Example 1: in our vending machine we have

\[
\begin{align*}
\Sigma &= \{ \text{D}, \text{Q} \} \\
\Sigma^* &= \{ \lambda, \\
&\quad \text{D, Q,} \\
&\quad \text{DD, DQ, QD, QQ,} \\
&\quad \text{DDD, DDQ, DQD, DQQ, QDD, QDQ, QQD, QQQ,} \\
&\quad \text{DDDD, DDDQ, \ldots } \}
\end{align*}
\]

\[L = \{ u \in \Sigma^* \mid u \text{ successfully vends} \}\]

Example 2: in our switch example we have

\[
\begin{align*}
\Sigma &= \{ \text{Push}\} \\
\Sigma^* &= \{ \lambda, \\
&\quad \text{Push,} \\
&\quad \text{Push Push,} \\
&\quad \text{Push Push Push,} \\
&\quad \text{Push Push Push Push, \ldots } \}
\end{align*}
\]

\[L = \{ \text{Push}^n \mid n \text{ is odd} \}\]
A finite automaton is a 5-tuple $M = (Q, S, \delta, q_0, F)$

- $Q$ is the set of states
- $S$ is the alphabet
- $\delta$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of final states

$L(M) =$ the language of machine $M$

$= $ set of all strings machine $M$ accepts
Definitions

State Diagram and Table

Definitions:

- \( Q = \{ q_0, q_1, q_2 \} \)
- \( \Sigma = \{ a, b \} \)
- \( F = \{ q_2 \} \)

Transition Table:

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( q_0 )</td>
<td>( q_1 )</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( q_0 )</td>
<td>( q_2 )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( q_2 )</td>
<td>( q_2 )</td>
</tr>
</tbody>
</table>
FINITE STATE MACHINES (AUTOMATA)

- Deterministic Finite Automata (DFA)
- Non-Deterministic Finite Automata with empty move (?-NFA)
- Non-Deterministic Finite Automata (NFA)
Deterministic & Nondeterministic

Deterministic

One choice

Non-Deterministic

Multi choice → Backtrack
Deterministic & Nondeterministic

Deterministic

From ONE state machine can go to another ONE state on one input

One choice

Non-Deterministic

From ONE state machine can go to MANY states on one input

Multi choice
Deterministic Computation

accept or reject

Non-Deterministic Computation

accept

reject
DETERMINISTIC FINITE AUTOMATA (DFA)
A DFA is a 5-tuple $M = (Q, S, \delta, q_0, F)$

- $Q$ is the set of states
- $S$ is the alphabet
- $\delta : Q \times S \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

$L(M) =$ the language of machine $M$
$= $ set of all strings machine $M$ accepts
Deterministic Finite Automata (DFA)

Example 1

Input Rejected

is not final state

Input Rejected
Q: What kinds of bit-strings are accepted?

A: Bit-strings that represent binary even numbers.
Deterministic Finite Automata (DFA)

Example 2

<table>
<thead>
<tr>
<th>Input</th>
<th>State Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td>reject</td>
</tr>
<tr>
<td>11</td>
<td>accept</td>
</tr>
<tr>
<td>010100100100100</td>
<td>accept</td>
</tr>
<tr>
<td>0100000100100</td>
<td>accept</td>
</tr>
<tr>
<td>λ</td>
<td>reject</td>
</tr>
</tbody>
</table>
Exercise

$L(M) = \{0, 1\}^*$
$L(M) = \{ w \mid w \text{ has an even number of 1s} \}$
Build an automaton that accepts all and only those strings that contain 001
Strings over \{a, b\} containing even number of a’s and odd number of b’s.

\[ q_0 = \text{Even}(a).\text{Even}(b) \]
\[ q_1 = \text{Even}(a).\text{Odd}(b) \]
\[ q_2 = \text{Odd}(a).\text{Even}(b) \]
\[ q_3 = \text{Odd}(a).\text{Odd}(b) \]
Strings over \{a,b,c\} that has the form \((ab)^*c\)

- \(q_0 = \text{End}(b)\)
- \(q_1 = \text{End}(a)\)
- \(q_2 = \text{End}(c)\)
- \(q_3 = \text{Error}\)
Exercise

Design with a friend a machine that tells us when a base-10 number is divisible by 3.

What should your alphabet be?

How can you tell when a number is divisible by 3?
Answer
Find the formal description of this automaton.
Answer

\[ Q = \{ 0 \text{ mod } 3, 1 \text{ mod } 3, 2 \text{ mod } 3 \} \quad (\text{rename: } \{ q_0, q_1, q_2 \}) \]
\[ \Sigma = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \} \]
\[ q_0 = 0 \text{ mod } 3 \]
\[ F = \{ 0 \text{ mod } 3 \} \]

\[ d : Q \times \Sigma \rightarrow Q \]
\[ d(q_0, 2) = q_2, \quad d(q_0, 9) = q_0, \quad d(q_1, 2) = q_0, \]
\[ d(q_1, 7) = q_2, \quad d(q_2, 3) = q_2, \quad d(q_2, 5) = q_1. \]

Question: \[ d(q_i, j) = ? \]

\[ d(q_i, j) = q_{(i+j) \text{ mod } 3} \]