1 Automatic Doors Example

A finite state automaton (FSA) is the simplest model of computation. Many useful devices can be modeled using FSAs.

- Behavior of automatic doors:
  The door can be in one of two states: open or closed.
  If it is closed and a person is standing on the pad in front of the doorway, the door opens.
  If it is closed and a person is standing on the pad to the rear of the doorway, it remains closed.
  If it is closed and a person is standing on neither pad, it remains closed.
  If it is closed and people are standing on both pads, it remains closed.
  If it is open and a person is standing on the pad in front of the doorway, it remains open.
  If it is open and a person is standing on the pad to the rear of the doorway, it remains open.
  If it is open and a person is standing on neither pad, it becomes closed.
  If it is open and people are standing on both pads, it remains open.

- Behavior of automatic doors in tabular form:

<table>
<thead>
<tr>
<th></th>
<th>NEITHER</th>
<th>FRONT</th>
<th>REAR</th>
<th>BOTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSED</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Behavior of automatic doors in graph form:

States: open, closed
Input signals: front, rear, neither, both
2 String, Alphabet, Language

- An alphabet is any finite set. The members of the alphabet are the symbols of the alphabet. Examples:
  \[ \Sigma_1 = \{0, 1\} \]
  \[ \Sigma_2 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\} \]
  \[ \Gamma = \{0, 1, x, y, z\} \]

- A string over an alphabet is a finite sequence of symbols from that alphabet.
  If \( \Sigma_1 = \{0, 1\} \), then 011001 is a string over \( \Sigma_1 \).
  If \( \Sigma_2 = \{a, b, c, ..., z\} \), then abracadabra is a string over \( \Sigma_2 \).
  If \( w \) is a string over some alphabet, the length of \( w \), written \( |w| \), is the number of symbols in the string.
  \[ |011001| = 6 \]
  \[ |\text{abracadabra}| = 11 \]
  The string of length zero is called the empty string and is written \( \epsilon \).
  The reverse of string \( w \), written \( w^R \), is the string obtained by writing \( w \) in the opposite order.
  \[ 011001^R = 100110 \]
  String \( z \) is a substring of \( w \) if \( z \) appears consecutively within \( w \).
  11 is a substring of 011001.
  Given two strings \( x \) and \( y \), the concatenation of \( x \) and \( y \) is the string obtained by appending \( y \) to the end of \( x \).
  Concatenation of 011001 and abracadabra is 011001abracadabra.

- A language is a set of strings.

3 Cola Machine Example

- Consider a cola machine with the following characteristics:

  1. Drinks cost 25 cents.
  2. The only coins accepted by the machine are quarters (Q), dimes (D), and nickels (N).
  3. The machine accepts any combination of these coins, in any order, as long as it adds up to exactly 25 cents.
  4. The machine requires exact change.

- This cola machine is a finite state automaton. Before we put any money into it, it will be in a start state. Our job is to add appropriate coins that change the state of the machine until it is in a special final state. The machine will reach the final state (or accept state) when the total amount of money you insert into the machine adds up to exactly 25 cents. By convention, we use a double circle to mark a final state.
We can get to the final state in one step by inserting a quarter. But there are other ways of getting to the final state by inserting various combinations and permutations of nickels and dimes.

The following is a list of all the possible sequences of coins that the cola machine will accept, where acceptance means reaching the final state and allowing us to get a cola:
Q, DDN, DND, NDD, DNNN, NDNN, NNND, NNND, NNNNN

Let us make a transition from a mechanical machine to a linguistic one.
Think of the inputs to the machine not as coins but as symbols like Q, D, and N.
The set of valid symbols that the machine will accept is its alphabet.
The sequences of symbols that the machine will accept are strings.
The entire set of strings that the machine accepts or recognizes is its language.

Question:
1. What is the alphabet of the cola machine?
2. What are the strings that the cola machine accepts?
3. What is the language of the cola machine?
4 How to Read State Diagrams

The following figure depicts a finite state automaton called $M_1$:

- $M_1$ has three states, labeled $q_0$, $q_1$, $q_2$.
- The start state is $q_0$.
- The final state (or accept state) is the one with a double circle, $q_1$.
- The arrows going from one state to another are called transitions.

When this automaton receives an input string, it processes each symbol in that string from left to right and eventually produces an output. The output is either accept or reject. The processing begins in the machine’s start state, and upon reading each symbol it moves from one state to another along the transition that has that symbol as its label. When the machine reads the last symbol, the output is either accept if the machine is in a final (accept) state or reject if it is not.

When we feed the input string 1101 to machine $M_1$:

1. start in state $q_0$;
2. read 1, follow transition from $q_0$ to $q_1$;
3. read 1, follow transition from $q_1$ to $q_1$;
4. read 0, follow transition from $q_1$ to $q_2$;
5. read 1, follow transition from $q_2$ to $q_1$;
6. accept because $M_1$ is in an accept state $q_1$ at the end of the input.

Question:

Which of the following strings are accepted by the machine $M_1$?

a. 1
b. 101000
c. 0101010101

d. 0101000000

e. 0
f. 110000
g. 01
h. 11
i. 10
j. 100
k. 0100
l. 111010100000

4
5 Formal Definition of a Finite State Automaton

- Definition 1.1 A finite state automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where:

  1. \(Q\) is a finite set of states;
  2. \(\Sigma\) is a finite set called the alphabet;
  3. \(\delta : Q \times \Sigma \rightarrow Q\) is the transition function;
  4. \(q_0 \in Q\) is the start state;
  5. \(F \subseteq Q\) is the set of accept states.

- Question
  Can finite state machines have zero number of accept states?
  Must there be exactly one transition arrow exiting every state for each possible input symbol?

- We can describe \(M_1\) formally by writing \(M_1 = (Q, \Sigma, \delta, q_0, F)\), where

  1. \(Q = \{q_0, q_1, q_2\}\)
  2. \(\Sigma = \{0, 1\}\)

<table>
<thead>
<tr>
<th>(\delta)</th>
<th>0</th>
<th>1</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_2)</td>
<td>(q_1)</td>
</tr>
<tr>
<td>(q_2)</td>
<td>(q_1)</td>
<td>(q_1)</td>
</tr>
</tbody>
</table>

  3. \(\delta\) is defined as
  4. \(q_0\) is the start state
  5. \(F = \{q_1\}\)

- Question: Given the formal description of finite state automaton \(M_2\) below, draw a corresponding state diagram for \(M_2\).

\(M_2 = (Q, \Sigma, \delta, q_0, F)\), where

  1. \(Q = \{q_0, q_1\}\)
  2. \(\Sigma = \{0, 1\}\)
  3. \(\delta\) is defined as

<table>
<thead>
<tr>
<th>(\delta)</th>
<th>0</th>
<th>1</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_1)</td>
</tr>
</tbody>
</table>

  4. \(q_0\) is the start state
  5. \(F = \{q_1\}\)
• Question: Consider the state diagram for finite state automaton $M_3$, and give a formal description of $M_3$.

\[ M_3 = (Q, \Sigma, \delta, q_0, F), \] where

1. $Q =$
2. $\Sigma =$
3. $\delta$ is defined as

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. The start state is
5. $F =$

Does $M_3$ accept the empty string $\epsilon$?

• Question
Consider the state diagram for finite state automaton $M_4$, and give a formal description of $M_4$. 
6 The Regular Operations

6.1 Definition of regular operations

- Definition 1.10

Let $A$ and $B$ be languages. We define the regular operations union, intersection, concatenation and star as follows.

- Union: $A \cup B = \{x | x \in A \text{ or } x \in B\}$.
- Intersection: $A \cap B = \{x | x \in A \text{ and } x \in B\}$.
- Concatenation: $A \circ B = \{xy | x \in A \text{ and } y \in B\}$.
- Star: $A^* = \{x_1x_2x_3\ldots x_k | k \geq 0 \text{ and each } x_i \in A\}$.

- Examples

Let the alphabet $\Sigma$ be the standard 26 letters \{a, b, c, ..., z\}. If $A = \{\text{good, bad, boy}\}$ and $B = \{\text{boy, girl}\}$, then

$A \cup B = \{\text{good, bad, girl, boy}\}$.

$A \cap B = \{\text{boy}\}$.

$A \circ B = \{\text{goodboy, goodgirl, badboy, badgirl, boyboy, boygirl}\}$.

$A^* = \{\varepsilon, \text{good, bad, boy, goodgood, goodbad, badgood, badbad, goodgoodboy, goodbadbadgoodboy, goodbadboy, goodgoodgood, goodgoodboy, badbadbadboyboy, badbadbadbadgoodgood, ...}\}$

Note that the empty string $\varepsilon$ is always a member of $A^*$, no matter what $A$ is.
6.2 Properties of regular operations

- Theorem (Sipser’s Theorem 1.12)
  The class of regular languages is closed under the union operation.
  That is, if $A$ and $B$ are regular languages, then $A \cup B$ is also a regular language.

- Theorem
  The class of regular languages is closed under the intersection operation.
  That is, if $A$ and $B$ are regular languages, then $A \cap B$ is also a regular language.

- Theorem (Sipser’s Theorem 1.13)
  The class of regular languages is closed under the concatenation operation.
  That is, if $A$ and $B$ are regular languages, then $A \circ B$ is also a regular language.

- Theorem
  The class of regular languages is closed under the star operation.
  That is, if $A$ is a regular language, then $A^*$ is also a regular language.

6.3 Proof of Theorem 1.12

- Remember that if $A$ is a regular language, then there is a finite state automaton, call it $M_A$, such that $A = L(M_A)$. That is, there must be a finite state automaton that accepts $A$.

So, if $A$ and $B$ are regular languages, to show that $A \cup B$ is a regular language we have to show that there is a finite state automaton, $M_{A \cup B}$, that accepts $A \cup B$.

In particular, suppose $M_A$ is the automaton that recognizes $A$ and $M_B$ is the automaton that recognizes $B$. We want to show how to automatically build a finite state automaton, $M_{A \cup B}$, from $M_A$ and $M_B$ such that $A \cup B$ is accepted by $M_{A \cup B}$.

This is a proof by construction. We construct a machine $M'$ that simulates $M_A$ and $M_B$ for any input. In a sense, $M'$ would take the input and run $M_A$ and $M_B$ on it in parallel. It would accept the input if and only if either $M_A$ or $M_B$ accepts the input.
What does it mean for a finite state machine to simulate two other machines in parallel?
At each step, $M'$ would look at an input symbol from the string and enter a new state that is a function of what $M_A$ does on the string and what $M_B$ does on the string. That is, $M'$ needs to remember a pair of states.

This suggests that we should construct $M'$’s states from the Cartesian product of $M_A$’s states and $M_B$’s states. The transitions of $M'$ go from pair to pair, in a sense updating the current state for both $M_A$ and $M_B$.

Here is how to construct such a machine.
Let $M_A$ recognize $A$, where $M_A = (Q_A, \Sigma, \delta_A, q_1, F_A)$, and let $M_B$ recognize $B$, where $M_B = (Q_B, \Sigma, \delta_B, q_2, F_B)$.

Construct $M'$ to recognize $A \cup B$, where $M' = (Q', \Sigma, \delta', q_0, F')$.

1. $Q' = \{(q_i, q_j) | q_i \in Q_A \text{ and } q_j \in Q_B\}$
   This set is the Cartesian product of $Q_A$ and $Q_B$ $(Q_A \times Q_B)$. It is the set of all pairs of states, the first from $Q_A$ and the second from $Q_B$.

2. $\Sigma$, the alphabet, is the same as in $M_A$ and $M_B$, namely, $\{0, 1\}$.

3. $\delta'$, the transition function, is defined as follows.
   For each $(q_i, q_j) \in Q'$ and each $a \in \Sigma$,
   \[
   \delta'(q_i, q_j, a) = (\delta_A(q_i, a), \delta_B(q_j, a))
   \]
   That is, $\delta'$ gets a state of $M'$ (which is a pair of states from $M_A$ and $M_B$), together with an input symbol, and returns $M'$’s next state, which is the pair of states returned by $\delta_A$ on $a$ and $\delta_B$ on $a$.

4. The start state, $q_0$, is the pair $(q_1, q_2)$.

5. $F' = \{(q_i, q_j) | q_i \in F_A \text{ or } q_j \in F_B\}$.
   That is, $F'$ is the set of pairs in which either member is a final state of $M_A$ or $M_B$.

If $M_A$ and $M_B$ are finite state automata, then so is $M'$ since it is built from the Cartesian product of the states in $M_A$ and $M_B$, which must be finite.
• Example

Let $M_A$ be defined as follows:

1. $Q = \{q_0, q_1, q_2\}$
2. $\Sigma = \{0, 1\}$
3. $\delta$ is defined as

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q0</td>
<td>q1</td>
</tr>
<tr>
<td>q1</td>
<td>q2</td>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
<td>q1</td>
<td>q1</td>
</tr>
</tbody>
</table>

4. The start state is $q_0$
5. $F = \{q_1\}$

And let $M_B$ be defined as follows:

1. $Q = \{q_3, q_4\}$
2. $\Sigma = \{0, 1\}$
3. $\delta$ is defined as

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>q3</td>
<td>q3</td>
<td>q4</td>
</tr>
<tr>
<td>q4</td>
<td>q3</td>
<td>q4</td>
</tr>
</tbody>
</table>

4. The start state is $q_3$
5. $F = \{q_3\}$
Construct $M'$ that accepts $L(M_A) \cup L(M_B)$.

1. $Q = \{ <q_0, q_3>, <q_0, q_4>, <q_1, q_3>, <q_1, q_4>, <q_2, q_3>, <q_2, q_4> \}$

2. $\Sigma = \{0, 1\}$

3. $\delta$ is

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;q_0, q_3&gt;$</td>
<td>$&lt;q_0, q_3&gt;$</td>
<td>$&lt;q_1, q_4&gt;$</td>
</tr>
<tr>
<td>$&lt;q_0, q_4&gt;$</td>
<td>$&lt;q_0, q_3&gt;$</td>
<td>$&lt;q_1, q_4&gt;$</td>
</tr>
<tr>
<td>$&lt;q_1, q_3&gt;$</td>
<td>$&lt;q_2, q_3&gt;$</td>
<td>$&lt;q_1, q_4&gt;$</td>
</tr>
<tr>
<td>$&lt;q_1, q_4&gt;$</td>
<td>$&lt;q_2, q_3&gt;$</td>
<td>$&lt;q_1, q_4&gt;$</td>
</tr>
<tr>
<td>$&lt;q_2, q_3&gt;$</td>
<td>$&lt;q_1, q_3&gt;$</td>
<td>$&lt;q_1, q_4&gt;$</td>
</tr>
<tr>
<td>$&lt;q_2, q_4&gt;$</td>
<td>$&lt;q_1, q_3&gt;$</td>
<td>$&lt;q_1, q_4&gt;$</td>
</tr>
</tbody>
</table>

4. The start state is $<q_0, q_3>$

5. $F = \{ <q_0, q_3>, <q_1, q_3>, <q_2, q_3>, <q_1, q_4> \}$

- Question
  
  Give a set of instructions for processing a string accepted by $M_A$ in $M'$ and a set of instructions for processing a string accepted by $M_B$ in $M'$. (For examples, see Partee page 454 or page 457, or see Sipser page 34).
6.4 Proof of closure under intersection

The class of regular languages is closed under the intersection operation. That is, if \( A \) and \( B \) are regular languages, then \( A \cap B \) is also a regular language.

- Let \( M_A \) recognize \( A \), where \( M_A = (Q_A, \Sigma, \delta_A, q_1, F_A) \), and let \( M_B \) recognize \( B \), where \( M_B = (Q_B, \Sigma, \delta_B, q_2, F_B) \).

- We construct \( M' = (Q', \Sigma, \delta', q_0, F') \) that accepts only strings in the intersection of \( A \) and \( B \) according to the following rules:

1. \( Q' = Q_A \times Q_B = \{(q_i, q_j) | q_i \in Q_A \text{ and } q_j \in Q_B \} \)
2. \( \Sigma = \{0, 1\} \)
3. \( \delta'((q_i, q_j), a) = (\delta_A(q_i, a), \delta_B(q_j, a)) \)
4. \( q_0 \) is the pair \( (q_1, q_2) \)
5. \( F' = \{(q_i, q_j) | q_i \in F_A \text{ and } q_j \in F_B \} \)

That is, \( F' \) is the set of pairs in which both members are final states of \( M_A \) and \( M_B \), respectively.

Question: Draw \( M' \).