Lecture 18: Finite State Automata

The wicked witch has hung her magic mirror and finds that, in spite of the owner’s manual, the mirror can speak only 2 phrases:

“You’re the fairest in the land.”
“Snow White is looking better than you.”

Peaceful  Agitated

Suppose the mirror acquires another phrase:
“Snow White is history.”

Finite State Automaton:
1. a set of states, S
2. a set of input symbols, I
3. a next-state function N (tells us where to go)
   \[ N : S \times I \rightarrow S \]
4. one state called the initial state \( s_0 \)
5. one or more accepting states

If the FSA transition diagram involves no decision making, it is a deterministic FSA.
(All transitions are “determined”—you have no choice.)
You display a (deterministic) FSA either
1. with a transition diagram or
2. with a (annotated) Next-State Table.

What is $N(s_0, a)$?
What is $N(s_2, b)$?

<table>
<thead>
<tr>
<th>State</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Definition:** The set of all strings accepted by a deterministic FSA is called a **regular language**.

Formally: $L(A) = \{ w \in I^* \mid N^* (s_0, w) \text{ is an accepting state and } I \text{ is the alphabet} \}$

$(N^* \text{ is the eventual state function, which is defined below.)}$

**Example 1.** Consider the finite-state automaton $A$:

![Finite-state automaton diagram]

a) To what states does $A$ go if symbols of the following strings are input to $A$ in sequence, starting from the initial state?

i. 11
ii. 0101

b) Which of the strings in part (a) send $A$ to an accepting state?

c) What is the language accepted by $A$?

d) Is there a regular expression that defines the same language?
**Example 2.** Draw a transition diagram for a FSA that accepts the language that consists of all strings that contain only a’s and b’s and end in b.

What are the input symbols (alphabet), I?

What is the annotated next-state table for this FSA?

**Eventual state function** \( N^* : S \times I^* \rightarrow S \)

Tells where the string (in \( I^* \)) will eventually leave us.

For example 2: \( N^* (s_0, aba) = \quad N^* (s_1, abaabb) = \)

**Example 3.** Draw a transition diagram for a FSA that accepts the language that consists of strings containing exactly 4 b’s and \( \Sigma = \{ a, b, c \} \).

\[
N^* (s_2, abaacbcba) =
\]

\[
N^* (s_0, cbaabbbba) =
\]
Example 4. Draw a transition diagram for a FSA that accepts the language that consists of strings containing only a's and b's and consists of a number of a's followed by an equal number of b's.

Can we use FSAs to represent what's going on in a program?

```java
public static void main()
{
    int i;
    int numE = 0;
    int numO = 0;
    Scanner scan = new Scanner (System.in);
    i = scan.nextInt ( );
    while ( i != 999)
    {
        if (i % 2 == 0)
            numE++;
        else
            numO++;
        i = scan.nextInt ( );
    }
}
```
Q: What is an NFA?
It is a finite state machine where several choices may exist for the next state at any point.

   a. The next state function takes a state and an input symbol or the empty string and produces the set of possible next states.
   b. For a string to be accepted by the NFA, only one of the possible paths has to lead to an accept state.

Every NFA is equivalent to a DFA

Q: What does equivalent mean?

Every NFA is can be transformed to a DFA

But the complexity of the resulting FSA may increase dramatically!

To convert a NFA to a DFA, use subsets of the original set of states: these will be the new states in the DFA.

NFA:

Start with the start state, s

<table>
<thead>
<tr>
<th>$\mathcal{P}(S)$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{s}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{p}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{q}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{s,p}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{s,q}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{p,q}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{s,p,q}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Any new "subset" state that contains an accepting state (of the original NFA) is accepting.
A. Find an equivalent DFA

B. Find an equivalent DFA
C. Find DFA, A, that is equivalent to this NFA:

What is $L(A)$, i.e. $\, ?$

Is there a simpler DFSA that accepts this language?

What is it?
There is a method that is guaranteed to find the simplest DFA equivalent to any given DFA.

Identify k-equivalence classes. Recreate the transitions on the k-equivalence classes

Let $A$ be a finite-state automaton with next-state function $N$. Given any states $s$ and $t$ in $A$,

<table>
<thead>
<tr>
<th>1) $s$ is 0-equivalent to $t$</th>
<th>$\iff$</th>
<th>either $s$ and $t$ are both accepting states or they are both nonaccepting states.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2) for every integer $k \geq 1$, $s$ is $k$-equivalent to $t$</td>
<td>$\iff$</td>
<td>$s$ and $t$ are $(k-1)$-equivalent and for any input symbol $m$, $N(s,m)$ and $N(t,m)$ are also $(k-1)$-equivalent.</td>
</tr>
</tbody>
</table>

Find the 0-equivalence classes, the 1-equivalence classes, and the 2-equivalence classes for the DFA

Redraw DFA using 2-equivalence classes. Map transitions onto the equivalence classes.