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.”

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(s₀, a)?
What is N(s₂, b)?

<table>
<thead>
<tr>
<th>State</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₀</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s₁</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s₂</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: \( L(A) = \{ w \in I^* \mid N^*(s₀, w) \text{ is an accepting state and } I \text{ is the alphabet} \} \)

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

**Example 1.** Consider the finite-state automaton \( A \):

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
   iii. 011011
   iv. 00110

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 $\Sigma = \{a, b, c\}$.

$N^*( s_2, abaaacbcba) = \quad N^*( s_0, cbaabbba) =$
Kleene’s Theorem, Part 1

Given any language that is accepted by a finite state automaton, there is a regular expression that defines the same language.

Proof:

Suppose \( A \) is a finite-state automaton with a set \( I \) of input symbols, a set \( S \) of \( n \) states, and a next-state function \( N: S \times I \rightarrow S \). Let \( I^* \) denote the set of all strings over \( I \). Number the states \( s_1, s_2, s_3, \ldots, s_n \), using \( s_1 \), to denote the initial state, and for each integer \( k = 1, 2, 3, \ldots, n \), let \( L_{i,j}^k = \{ x \in I^* \text{ s.t. when the symbols of } x \text{ are input to } A \text{ in sequence, } A \text{ goes from state } s_i \text{ to } s_j \text{ without traveling through an intermediate state } s_h \text{ for which } h > k \} \).

If \( s_j \) is an accepting state and if \( k = n \) and \( i = 1 \),

Use mathematical induction to build up a set of regular expressions over \( I \). Let the property \( P(m) \) be the sentence, “For any pair of integers \( i \) and \( j \) with \( 1 \leq i, j \leq n \), there is a regular expression \( r_{i,j}^m \) that defines \( L_{i,j}^m \).”

Show that the property is true for \( m = 0 \):

Show that for all integers \( k \) with \( 0 \leq k \leq n \), if the property is true for \( m=k \) then it is true for \( m=k+1 \).
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 ( );
    }
}
```