CSCI 162 Lab 9

Purpose: To create a FSA in Prolog to Predicate.

Objectives: After this lab you should be able to

- Create a Prolog Finite Automaton (FA).

Background:

A particular transition function is given by the transitions among the states, and the set of final states. For example,

```
transition(s,a,p).
transition(s,b,s).
transition(p,a,s).
transition(p,b,p).

final(s).
```

specifies that the FA, starting at state s, will move to state p if it sees an `a', and remain in state s if it sees a `b', etc.; it also states that the only final state in the FSM is s. You are to write the structure accept so that whatever FA is encoded using the transitions and final states that you have programmed, and assuming the start state is called s, prolog will return ``Yes" if the string is accepted by the FA and "No" otherwise. For example, if the transitions and final states are programmed as given above, the prolog query

```
accept([a,a,b,b]).
```

will provoke the response "Yes", but if you query

```
accept([a,a,b,b,a]).
```

the prolog interpreter will say "No" -- in this case, only the strings with an even number of a's will be given a ``Yes" response. To do this, you are required to write a prolog structure called computation, so that

```
computation(X,L,Y)
```

is satisfied if and only if there is a sequence of transitions that take the FSM from state X to state Y consuming the entire list L of symbols. Suppose you have a correct structure for computation: how do you write accept so that it is satisfied if and only if the FSM, starting at a state called s, reads the input, making transitions, and ends up at a final state? Write accept, and write computation so that it is general enough to perform the computation on any FSM given in the form indicated. Here are some more to test on.

```
%Tests to see if `ab' is a substring.
transition(s,a,p).
transition(s,b,s).
transition(p,a,p).
transition(p,b,q).
transition(q,a,q).
transition(q,b,q).

final(q).


%Tests to see if the number of a's is not a
multiple of 3.
transition(s,b,s).
transition(a1,b,a1).
transition(a2,b,a2).
transition(a1,a,a2).
transition(a2,a,s).
transition(s,a,a1).

final(a1).
final(a2).
```

Instructions:

For this lab, you will write prolog programs that have two parts: the transition function and final states for a particular Finite Automaton (FA); and structures called `computation' and `accept'.

You are to write hand in via Moodle the following files. They will be text files in compilable prolog format, with your name and the file name in comments (%) at the top of the file. Their names will be fa1, fa2, and fa3.

1. A file called fa1 that includes definitions (structures) for accept and computation, and for the transpositions and final states for a FA that accepts all strings over {0,1} that start and end in different symbols.
2. A file called fa2 that includes definitions (structures) for accept and computation, and for the transpositions and final states for a FA that accepts all strings over {0,1} that contain 111 as a substring.
3. A file called fa3 that includes definitions (structures) for accept and computation, and for the transpositions and final states for a FA that accepts all strings over {a,b} that are in the language described by the regular expression a*(ba + ab)*b*.