

CS 311 Language Translation and Automata
Project #1
(Due: 2 PM, Monday, 4/23/2012)

Once a finite state automaton (FSA) is designed, its transition diagram can be translated in a straightforward manner into program code. However, this translation process is considerably tedious if the FSA is large and troublesome if the design is modified. The reason is that the transition information and mechanism are combined in the translation.

To do it differently, we can design a general data structure such as a table or a list to hold the transition information, and to implement in program code only a general transition mechanism. Using this approach, the resulted program is not only smaller, but it can also be modified easily to simulate a different FSA by just changing the transition information and the general data structure holds. We shall refer to such a program as a *universal finite state automaton*.

Your job for this assignment is to implement such a universal finite state automaton. To convince you that it is very easy to do so, I have suggested an algorithm below; however, you are free to modify it. Note that your program has to be designed to handle the FSA where the input transition function is partial, which means there is a default dead-end, or trap state.

```

1  state ← initial_state; exit ← false;
2  while not exit do
3    begin
4      symbol ← NextSymbol();
5      if symbol is in alphabet then begin
6        state ← NextState(state, symbol);
7        if state is dead_end then begin exit ← true; Reject(); end
8      end
9      else begin
10       exit ← true;
11       if symbol is not the endmarker then Reject()
12       else if state is final then Accept()
13       else Reject();
14     end //if
15   end //while

```

You must represent each FSA in the input file with the following format, and put all input machines in one file:

(1) The number of states, say N.

For ease of implementation, number states from 0 to N-1, with 0 representing the initial state, and N the dead-end state.

(2) The set of final states.

You need a boolean array FINAL [0..N-1].

(3) The alphabet.

Symbols in the alphabet should be numbered internally so that the value returned by the function NextSymbol is an integer.

(4) A sequence of transitions of the form (p,a,q).

The triple (p,a,q) means that in state p, looking at input symbol a, the FSA will change its state to q. For this project, store this information in a table, say next_state, so that the value returned by the function NextState is next_state [state, symbol].

Test your program with the following 5 finite state automata (test strings are also given):

(1) A FSA which recognizes the set of all strings over {a, b} such that each a in the string is immediately preceded and immediately followed by a b. Strings: a, b, bab, abb, bbaba, babaab, bbbababab, bbbbbb

(2) A FSA which recognizes email addresses. A valid email address is defined as follows: [userName@serverName.domainName](#), where the user name consists of at least one symbol with any combination of letters, digits, hyphens, underscores, or periods. The server name is any string of at least one symbol with any combination of letters, digits, hyphens, and underscores. The domain name must have 2 to 4 letters. Strings: a.b.c@d.e.fg, jsmith, jsmith@olympus, jsmith@olympus.gov, _jsmith-example.olympus@states.us, jsmith.edu, jsmith@a.b.c, john@mail.office

(3) A FSA for a vending machine that provides only candy bars, each costs 25 cents. The initial state indicates that nothing has happened yet. If you insert a nickel, the machine will change to a state indicating that the total amount received so far is 5 cents. If you then insert another nickel, it will change to a state indicating that the total amount received is now 10 cents. If you then press a button to select a candy bar, the machine will ignore this input, since the proper amount has not yet been paid. When the total amount reaches at least 25 cents, however, the machine will now respond if you select a candy bar. It will deliver the candy bar and return to its starting state, which is also an accepting state. $\Sigma = \{n, d, q, s\}$, where n stands for nickel, d for dime, q for quarter, and s for the select button. Strings: s, sds, nnns, ddds, qqqs, sdsnsns, sdqsd, ndnqdqndqn

(4) A FSA which recognizes all identifiers that begin with a letter (both upper and lower) or underscore, followed by any number of letters, digits, or underscores. Strings: ϵ , a, _, java, 1st_Assignment, _finite_automaton, X3Y7, X*Y

(5) A FSA which recognizes the set of all signed or unsigned decimal numbers without superfluous leading or trailing zeros. For instance, 0.0, -0.5, +120.01, and 123000.0 are in the language, but 0.00, 00.5, and 0123.4 are not. Each decimal number has the form of A.B, where A and B are strings of digits and they cannot be empty at the same time. Strings: +1.23, -.123, 123., -0.0, 01234.5, -789, ., 56.30, +120.0001, 123000.0

Sample input of a FSA:

```
2
1
0 1
(0 0 0)
(0 1 1)
(1 0 1)
(1 1 0)
1000
10001
.....
```

Corresponding output:

Finite State Automaton #1.

(1) number of states: 2

(2) final states: 1

(3) alphabet: 0, 1

(4) transitions:

```
0 0 0
0 1 1
1 0 1
1 1 0
```

(5) strings:

```
1000      Accept
10001     Reject
.....
```

Submit both your program and the input file to the instructor on Bb, and turn in the printout of (1) program code with proper comments including your name and clear instructions to compile, link, and run your program, (2) input file containing the description of the above five machines and corresponding test strings, and (3) output of executions.

Grading:

90% – correctness including turn-in materials and output format

10% – program structure and readability

Your project must be your own work. Copying programs or parts of programs will receive a zero grade and will be reported to the CS department and the University. No late project will be accepted. If you cannot complete the project by the due date, then submit whatever you have completed. Partial credit will be given for reasonable partial solutions. Note that your project will not be graded if I cannot run your program based on your instructions or if the turn-in material is not complete. The highest grade you can get is 70% of the earned score if your project requires being graded second time.