Event Driven Computing - Assignment 2

August 18, 2007

1 Objective

This assignment is the first of three in which you will develop an editing and simulation environment for finite state automata (FSA). For this assignment you will develop the basic infrastructure for reading in a representation of a FSA, and some simulation facilities to be used in the subsequent stages.

Although each assignment is of equal value. the quality of your submission for this part might affect your marks in the subsequent practicals. We urge you to do a good job!

2 Description

The goal of this assignment is to create a Java class to store deterministic and non-deterministic FSAs, and allow the behaviour of the Fsa to be simulated. In addition, you will write a class that can read/write an Fsa from/to a text file. The external behaviour is presecribed by interfaces we provide.

3 Fsa file format

The file describing an FSA is a series of records, structured as lines of text. The are four different type of lines to specify states, transitions, initial-state(s), and comments.

3.1 State record

state stateName xpos ypos

Where: **state** is the character string "state". *stateName* is the name of the state, a string of characters beginning with a letter, followed by zero or more letter, digit, or underscore characters. (This is essentially the definition of "idfentifier" in Java.) *xpos* and *ypos* are integer values that gives the x- and y-position of the state on the computer's display screen. Although you will not need these values for this stage of the project, you must store and reproduce them correctly.

3.2 Transition record

transition fromState event output toState

Where: **transition** stands for the character string "transition"; *fromState* is the name of the state from which this transition begins; *event* is the name of the event that causes this transition to occur. An event name is character string consisting *only* of letters. *output* is the output to be generated when the transition occurs. An output string is a character string consisting *only* of letters. If no output is required from a given transition, the output string has the special value "-"; and *toState* is the name of the state where the transition terminates.

An *epsilon*-transition is specified by the special event-name "?". An *epsilon*-transition *must* have an output action of "-".

3.3 Initial record

initial stateName

Where: *stateName* specifies an initial state of the FSA. *stateName* must comply with the rules decsribed earlier for state-names.

3.4 Comment record

A line that begins with the character "#" is a comment, and is ignored by the system.

3.5 Order of records

Records can appear in any order in the file, with one restriction: a state name *must* have been defined via a **state** record before its name can be used in a **transition** or **initial** record.

4 What we provide

We provide the following java files for you:

Fsa.java contains an interface specification for a class that stores a representation of an FSA.

State.java contains an interface specification for a class that stores a representation of a State.

Transition.java contains an interface specification for a class that stores a representation of a Transition.

FsaSim.java is an interface specification that allows a sequence of events to be fed to an FSA.

FsaIo.java contains an interface specification for a class that can read or write an FSA from a textfile.

FsaFormatException.java Specifies an exception that is thrown when an error is detected while reading an FSA specification.

5 Version control system

You must use the SVN software version control system to manage your work. The submission key for this exercise is: **edc-07-s2-assign2**.

If you are unfamiliar with SVN, a document describing it is available on the web at: **http://www.cs.adelaide.edu.au/docs/svn-instr.pdf**

The following Unix commands will create a repository for this assignment, and check out a working directory, ready for work.

Please note the following:

- Perform these steps ONCE in the order written.
- Replace aaaaaa, where it appears in the commands, with YOUR student id.
- Some commands are long — they must be typed on one line.

Use the Unix *cd* command to change to the place where you want your exercise to be stored, then type these commands:

```
svn mkdir -m "initial creation" edc-07-s2-assign2
https://repos.cs.adelaide.edu.au/svn/aaaaaa/edc-07-s2-assign2/

svn checkout https://repos.cs.adelaide.edu.au/svn/aaaaaaa/edc-07-s2-assign2/
edc-07-s2-assign2/
```

6 What you must do

You must write two java classes: *FsaImpl.java* that implements the *Fsa* interface, and the *FsaSim* interface, and contains a default constructor: `public FsaImpl()`.

FsaReaderWriter.java that implements the *FsaIo* inteface, and contains a default constructor `FsaReaderWriter()`.

7 Assessment

We will test the behaviour of your classes using an automated test script, that will be made available a few days before the deadline. You will need to thoroughly test your code *before* the deadline.

8 Submitting your work

The deadline for this assignment is 5:00pm Wednesday 5th September. Late handins will be penalised 25% per day or part thereof (including weekends!).

David Knight
Travis Olds