Small Group Activity - Encoding a FSM Diagram for the simulator and testing the simulator

Lynn Robert Carter, PhD

2019-02-10

Activity Kind

Small Group Activity

Purpose

The iterative cycle of design, code, and test is critical and, in many cases, each of these three steps themselves should be broken down into sub-steps that follow much the same pattern. Consider using a FSM machine as a design for a Java implementation. How do you ensure that the design is correct? In this assignment, we will use a FSM Simulator to check the design with actual input. If the simulator produces the expected output for each key test case, it is reasonable to conclude that the design is correct. When that happens, we will proceed to coding that design in Java.

Doing work and then verifying that is was done correctly before building upon it is a crucial concept. A formalism, such as Finite State Machines, make this possible. In engineering, we use such formalisms to check our work and to ensure that our results operate as expected **before** we release them for use. Learning such formalism in software development and using them is a key aspect of becoming a professional.

Pre-requisite

Students are expected to be current in their studies. If you are falling behind and have previous work that still needs to be done, it would be best to finish that work before beginning this work.

Students are expected to, at a minimum, watch the following videos, read all of this document, and do the practice worksheet, **before** coming to class. The two videos will require less than 20 minutes of your time to watch.

1. Watched - Video - An introduction to Finite State Machines (6:14)
2. Watched - Video - Finite State Machines and Specifications (12:48)

Tasking

During the activity, you will be asked to complete the tasks on the following worksheet. If you have questions, the time to get them resolved is **before** you do the work.

As you do the work, you are encouraged to take notes for your ENB. Please do not wait to capture and record these notes. Do the note taking as you are doing it so the information captured is much more likely to be accurate and useful. Delaying the note taking increases the probability that errors will creep into your notes, the quality of your notes will be reduced, and you will spend more time struggling to produce the notes that if you take them as you do the work.

Deliverable

You are responsible for producing the artifacts specified on the worksheet. As you create each of these deliverable, ensure they are saved and any notes about their creation or insights obtained from producing the deliverable is captured and recorded in your ENB.

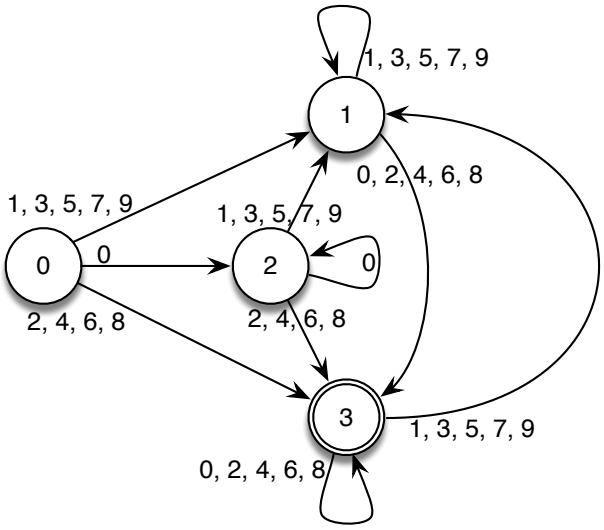
Submission

You are expected to submit your ENB with the worksheet deliverables and your notes from this activity by the specified deadline.

Finite State Machine Practice Worksheet

Practice using a recognizer application

From the previous work, you saw how an algorithm was used to convert this graphical representation



into the following textual representation.

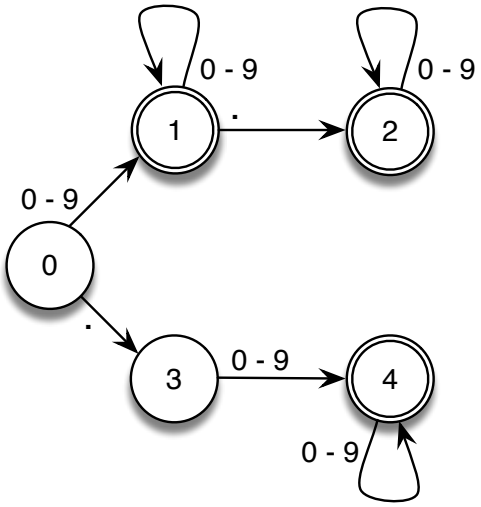
```
4
State 0
3
1 5 1 3 5 7 9
2 1 0
3 4 2 4 6 8

State 1
2
1 5 1 3 5 7 9
3 5 0 2 4 6 8

State 2
3
1 5 1 3 5 7 9
2 1 0
3 4 2 4 6 8

State F 3
2
1 5 1 3 5 7 9
3 5 0 2 4 6 8
```

1. The first task for this activity is for each student to produce a textual representation of the following FSM. This representation is the first of the deliverables for this worksheet.



2. Once you have produced the textual representation, obtain a copy of the FiniteStateMachine.zip file, place it into your Eclipse IDE's workbench folder, decompress it, and use it to create a new Java Project.

- Within Java, select File/New/Java Project…
- Uncheck the Use Default Location box that is just below and to the let of the Project name: label at the top of the pop-up wizard.
- Click on the Browse… button on the next line on the right-hand side.
- Using the pop-up navigation window, locate the folder "FiniteStateMachine" which was produced when you decompressed the zip archive, and click on it so that is highlighted. The "Finish" button at the bottom-right of the pop-up wizard should now be enabled.
- Click on the "Finish" button on the bottom-right of the pop-up wizard.
- You should now see the Java Project in the Package Explorer on the left side of the Eclipse window.
- Open up the project, the "src" folder and the "fsm" folder.
- Verify that the project works by running the following three test classes:
    1) TestCharCode.java.
    2) TestState.java
    3) TestStateTable.java
- The output of each test should indicate that all of the tests passed and none failed.
- Create a new text file within the project by selecting File/New/Untiled Text File.
- Place the textual representation of the Finite State Machine Recognizer into that test file.
- Save the file, specify the FiniteStateMachine folder as the parent folder, give it the name "DecimalFormatNumber", and save it.
- Run the "FiniteStateMachineSystem.java" class and you should see the following output:

```
Welcome to the Finite State Machine System

Please enter the name of the file containing the Finite State Machine description:
```

- Type the name "DecimalFormatNumber" following the colon and press return (enter). If you have properly entered the textual representation and stored it at the same level as the "src" folder, it should load the recognizer, display it to you, tell you that "No Errors were found!" and invite you to enter a string to see if it is recognized by the FMS using this definition.

- If there are no errors found during the reading in of the textual representation of the diagram, provide some input strings and verify that the recognizer properly produces a transition table and an appropriate result message. Copy the entire contents of the console and save it to a text file. This is the second deliverable for this worksheet.
- If you are successful with all of the above activities, look around to see if there are any students having problems. Don't be shy. Walk around and ask if you can help. Remember, you help by looking at what they have been doing and by asking questions about things that don't look right to you. Please **do not** just tell them what to do. If you are not exactly sure, suggest an experiment that could be used to address whatever the problem might be.
- After you help someone, take notes in your ENB about what the issue was, how you helped, and what the result of your help was.
- Once everyone appears to have their recognizer working, it is time to go to the next task.

3. Revisit the even number recognizer. Change it into an odd number recognizer. Convince yourself that each state in the recognizer is actually needed and be able to explain why. Once you have your new recognizer, go through the same process as above to produce a textual representation of it, produce a text file in Eclipse for it, save the file, run the application, specify the name of this new file, and verify the it works properly. Be sure you are producing notes and are recording them into your ENB. This and the previous activity will be peer-reviewed in the next class, so you better have evidence that you have done the work and that you followed the process.