Programming Assignment-5

Akanksha Bansal (akankshabansal90@gmail.com)
Gaurav Nanda (gaurav324@gmail.com)
Prateek Agarwal (prat0318@gmail.com)

For Part 1:

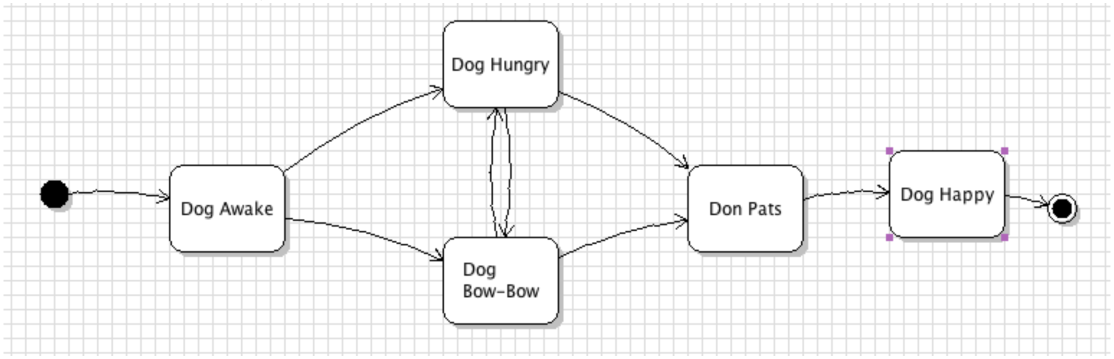
The First part of the assignment asks us to write parser for a violet state machine xml that will generate a prolog file. The State machine being drawn in violet is represented in form of an xml with the details of the nodes and the transitions. We will parse the xml file and extract the nodes and the transitions from it. Then write a .pl files out of the data that has been extracted to create tables of the State machine Meta model. With the nodes being defined in the first table and transitions being defined in the next.
The prolog file generated after reading the File should be representing the state machine present in the xml file.
Jar created for this execution is named: MDELite2.3.jar

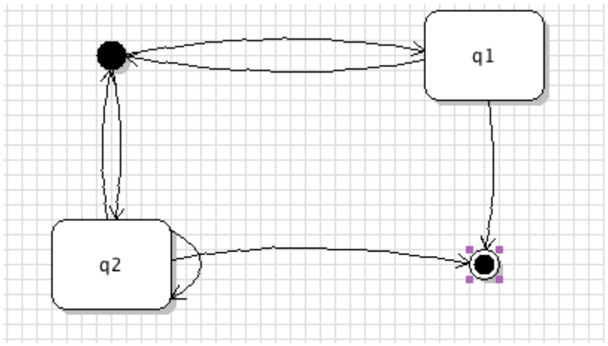
For Part 2:

This part uses the output of the previous part. From the code written for part 1 we get the prolog files needed for us to be able to perform a model to model transition on the data. This model transition helps in creating a prolog file for the model we have defined so far and then create a FSM with the details of the nodes and their corresponding transitions. Once the new Model prolog file has been generated we run constraints on it to make sure that the FSM being defined in the diagram represents a correct FSM. As part of this the xml corresponding to semiclique.state.violet fails.

In the final step we generate the java files from the generated FSM file.
We have then defined the app.java files for the various FSM's to show that the java files being generated work correctly.

New FSM's defined;





Running Instructions:
$ sh run.script
This script will compile the code for the above sections and then call functions to generate the prolog files from the xml's given in the argument. The output will indicate if the tests have run successfully or not.

This script will create java files (representing the FSM) and the .pl files for the given violet files present in the input directory. We then compile the code for each of the xml file individually and then run their corresponding test cases (present in outputs directory) to check whether the generated java files are correct.

You may observe following warning, which can be safely ignored.

```
- Compiling "eatingHabits.state.violet" to create the prolog file.
- Doing model to model transformation for eatingHabits.state.violet.

- Checking for constraints violations.
Warning: /Users/gnanda/fop/fop-shared/P5/outputs/merge.pl:6:
          Singleton variables: [Ready]
Warning: /Users/gnanda/fop/fop-shared/P5/outputs/merge.pl:7:
          Singleton variables: [Drink]
Warning: /Users/gnanda/fop/fop-shared/P5/outputs/merge.pl:8:
          Singleton variables: [Eat]
Warning: /Users/gnanda/fop/fop-shared/P5/outputs/merge.pl:9:
          Singleton variables: [Pig]
No invalid constraints found.
```