**CS392F P1 Design Description and Instructions**

Jianyu Huang

Xiaohui Chen

**Running the examples**

./run.script.sh [the parts you want to run(optional)]

e.g

./run.script.sh ---- run all parts (Part 1,2&3)

./run.script.sh 3 1 ---- run part 3 and part 1

**Part 1**

The vm code could generate fsm.java with methods “goto[node name]()”. The only private variable of fsm class is currentState, which is of type State. The reason for this variable to be private is to obey the rules of encapsulation in object-oriented programming. The “goto” methods simply calls currentState.”gotoXXX” and returns a state. If the state is null, that means the transition between the two nodes is not possible. Otherwise a new state is returned and currentState is assigned to the new state. The transition status would be printed according to the project requirements. The getName() method would call the currentState.getName() and return the node name.

State.java is simply a java interface and all the methods there are public abstract methods

The nXXX.java are nodes with nodeid as class names. In the “goto” methods in each class, the tuples in transition tables are inspected. If a transition is possible, then the state in endsAt would be returned. Otherwise null would be returned.

**Part 2**

Part 2 has the same prolog database as that of part 1, but the fsm is more abstract. Here, fsm.java use enum to include all the possible states. Also, in each “gotoXXX” method, the vm evaluated the tuples in transition table. In this case a switch statement is used to test whether the transition from currentState to state XXX is possible. Therefore it would print out exactly the same result as that of part 1.

**Part 3**