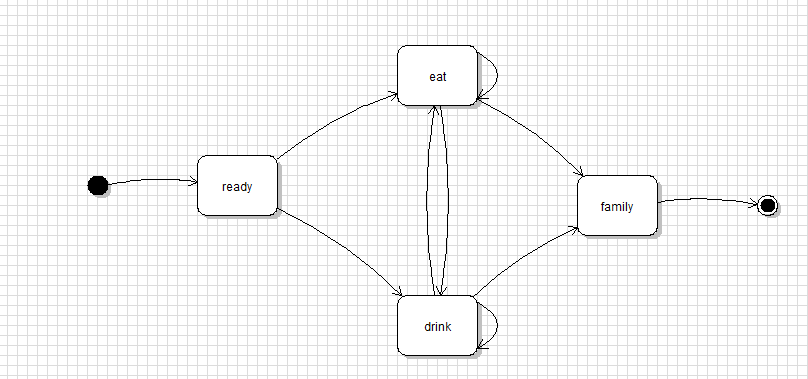
Part 2

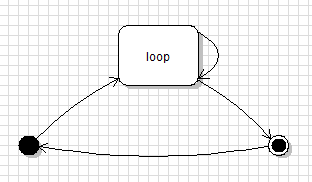
In Part 2, we draw five state diagrams in violet according to the five prolog databases. Each state diagram is in the corresponding folder in Part2File directory. Along with each diagram, there is a corresponding prolog database which is generated by the Main java classed we modified in part 1

My eating habit:



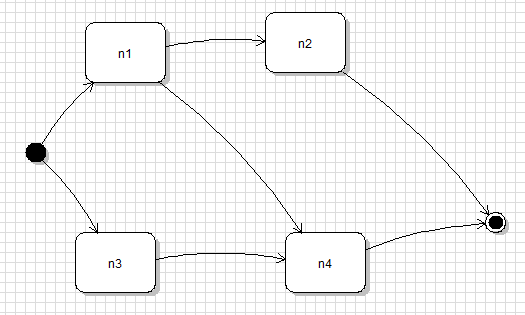
This diagram is straightforward. It is generated according to the prolog database online. It is stored in Part2Files/eatinghabit. The generated prolog database is equivalent to the prolog database.

Simple loop:



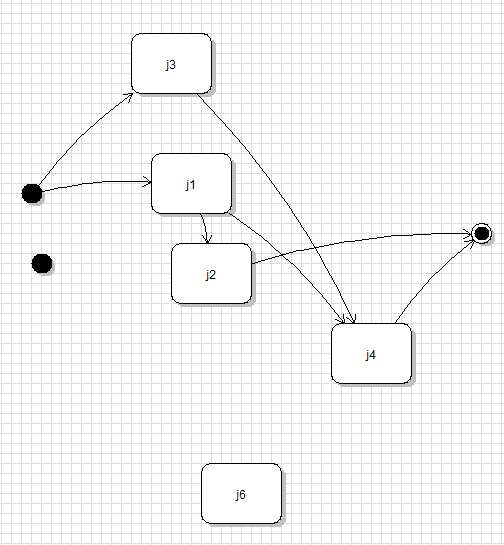
The simple loop is also straightforward. The diagram and the generated prolog database are in Part2Files/simpleloop

Fsm20:



The diagram and the generated prolog database is in Part2Files/fsm20. The generated prolog database is equivalent to the given one.

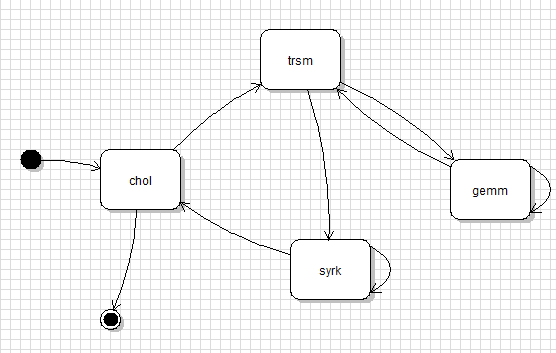
Fsm30:



There are three obvious errors. First, there are two start states and one of them is standalone. Second, node with id n6 is a standalone state. Third, the transition from n3 to n5 is not valid since there is no node with id n5.

In fact, we are technically not able to draw a diagram. The diagram above is the diagram without the last transition. However, violet does not support two start states and so the standalone start state above has type “CircularStateNode”. The Main java class generates the following line for this node: node(, x, state)

Therefore, the Main java class could not generated the correct prolog database.



In Part2Files/chol\_verify folder, Chol\_verify.pl is the expected prolog database and the diagram above is the correct state diagram. Therefore, the prolog database generated (chol\_vefity.pl) is equivalent to Chol\_verify.pl