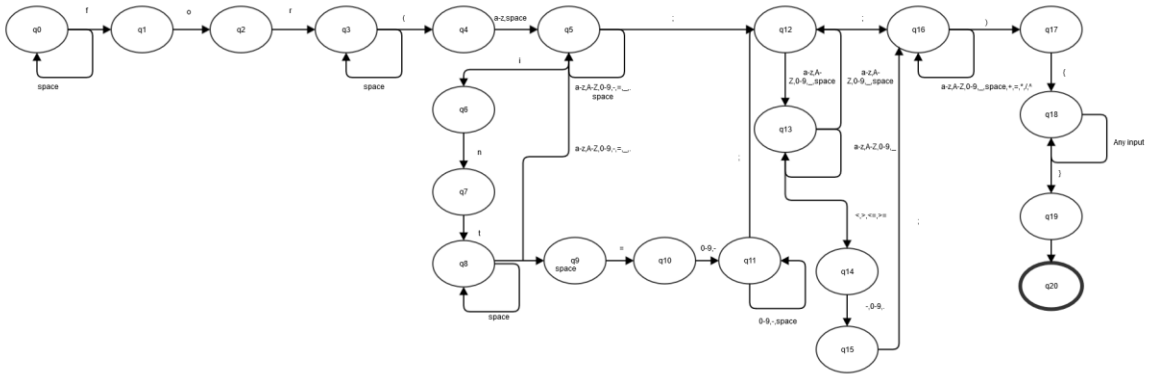
Richard Liao
Cmpt 440



Essay 1: The most difficult part of the problem was figuring out what kind of inputs could be accepted for the loop parameters. There are almost endless possibilities for parameter values that also count as valid input. In a java for loop, the initialization can contain a double, the termination can use any kind of method as long as it returns a true/false, and the increment is open ended as far as inputs go (it is also technically correct to have a mix of the 3 parameters). Without the use of tokens or regular expressions to give clearly defined inputs, it is almost impossible to fully encapsulate all correct and incorrect syntax variations for the for-loop. To compound the issues with defining the input, it should also be valid syntax to have as much or little white space in-between words. For instance, “int I” should be equally valid as “ int I” . To solve this I simply allowed for any input the work for the parameters (also lucid charts only allows for 60 objects to be placed for free accounts). While this method allows for incorrect inputs it also allows for all of the correct inputs.

Since the initialization parameter must be an int, a double, a float, or a previously defined identifier. This makes is considerably easier to define branches, but does not stop inputs such as “in the ”. While my DFA does not include the branches for float and double, the structure is there and the inclusion would be relatively simple. It is my opinion that the DFA, while capable of representing the full syntax of a for-loop, is not practical for such. The termination parameter in my DFA accounts for two possible entries, an outside variable/method or a comparison. Since ultimately, the language only cares about the result being either true or false, I left the DFA to only include those possibilities. The last parameter, increment, is left open ended since this can be several possibilities: i++,++i, i--, i+=2, i=x+3, 2, etc. Accounting for those numerous branches is difficult, so it was left open ended.

Essay 2: If I were to implement this into java I would use two objects the state object and a transition object. The state object will hold the name (i.e. q0, q1, etc.), a list of transition objects, and an accepted state flag. The transition object, on the other hand, will hold a target state, and accepted inputs. The reasoning behind putting the “next” state in the transition object is that it is difficult to pin down what the “next” state will be. In more complicated DFA’s, there may be several branching paths to the final state. By defining the pathways in the transition objects rather than the state objects, it is much less restrictive to define pathways between states. A transition object might direct a state to itself, but there also may be another transition object that defines the next path. Since the state object takes a list of transitions, there can be as few or many as needed. The final state can be any state since the definition is just a flag in the state object. The rules of the DFA can be further defined in the methods used in the implementation.