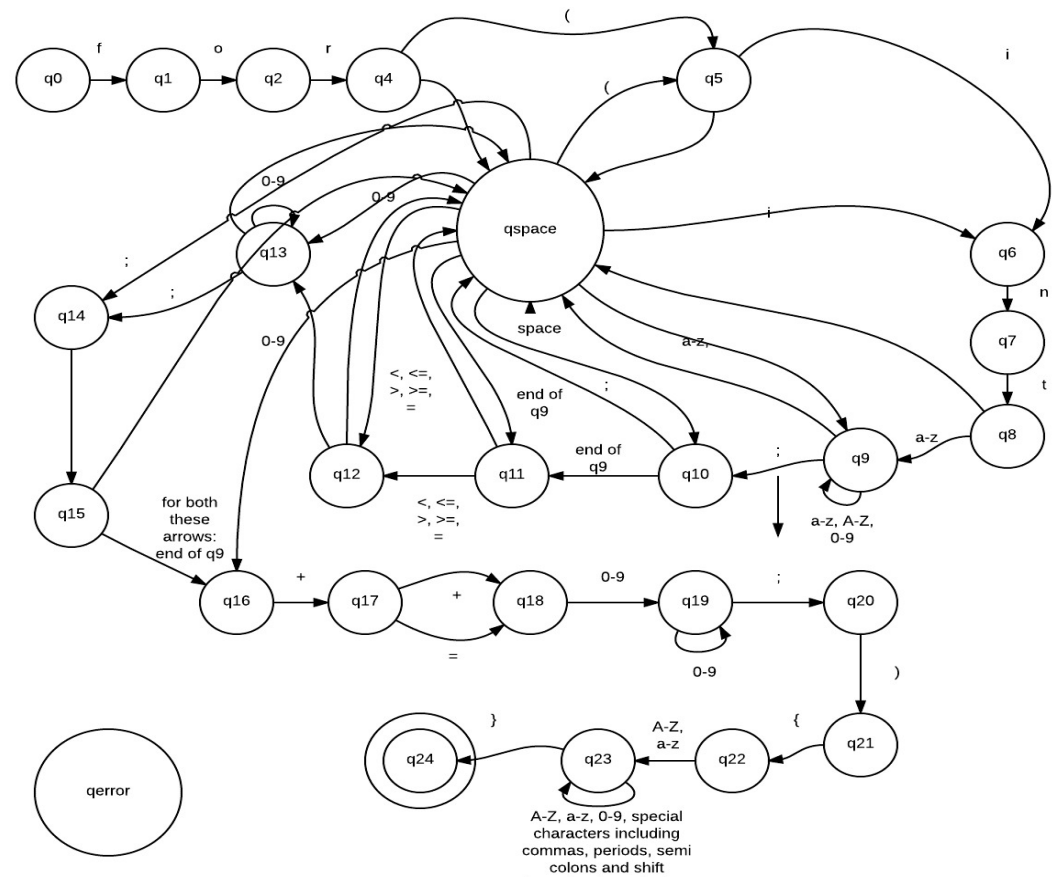
Connor Loughlin
Formal Computability
Professor Rivas
February 1, 2016

Lab 1

1.



Literally every state on this diagram shold have an extra arrow pointing to this error state, this error state should have no outgoing arrows. For purposes of making my diagram more easy to read I have not added any of these arrows.

In order to account for the second specified variant of the for loop on this lab, I should have at allow for spacing at the beginning of the increment section of the loop, this way a semi colon can be added without needing any characters in that section. I need to get better at making these diagrams.

2. While figuring out how to create a DFA that accepts or rejects for loop syntax in java I encountered many problems. First and foremost, I find that actually creating the DFA so that it is nice and readable is very difficult to do. Second, I had a bit of a difficult time figuring out just how to incorporate the use of white space into my DFA. At first, I kept creating new states that accepted a space, and then transitioned back to themselves if another space was used. However, I then decided to make a "qspace" state. This state is entered if a space is entered anywhere in the for loop that white space is allowed. From this state, an infinite number of spaces can be used, or depending on where you are in the for loop, you can enter a character, semi colon, parentheses, and a few other symbols to continue onward. I also added a qerror state that receives an arrow from every other state. The arrows are not included in my diagram, mainly because I want you to be able to (sort of) read it. However, upon finishing my DFA, I realize that because of this qspace state, I also allow for loop syntax that doesn't exist in java to be accepted. This is because I can enter in certain characters form the qspace state that will take into a later section of the syntax without actually having done any of the necessary syntax in between. However, I do still allow for both the required syntax to be accepted. I learned quite a bit from doing this problem. Mainly, I learned that I hate using lucidchart, and that DFA's can be tricky and require huge diagrams to map out some seemingly simple syntax. One more thing, please read my message on the diagram. It explains why my diagram appears to not allow for spacing in the increment section of the for loop, as well as why spacing is not allowed between the) and { characters (it was mainly because I've been up all night and am way too tired to include it. Rest assured, I understand the concept).

3. Given that I took this class last year, I do know a little bit about how to do this. In java, in order to implement a DFA, I would first create a class that represents a DFA. This class could contain a constructor that contains everything that a 5-tuple would have. This includes the set of all states, the alphabet, the transitions, the start state, and the set of end states. Additionally, there would need to be a class that represents what a state is. A state would be a simple integer. The main difficulty with implementing the DFA in java would be implementing the transitions. I would do this by using a hashTable of integers. For example if I had an entry in my hashtable that was <1, 2> that would mean that given a certain character, state 1 transitions into state 2. There would need to be helper methods, such as a "getNextState" method, which would return the integer of the next state depending on what character of the string is being processed. Though I don't' know if any of this would work until I actually write the code for it, I can see one flaw with the solution I have described. I need to find some way to include the current character in the string in the key of the transition hashtable. The key must be the current character and state, and the value must be the next state. Then, the next character must be grabbed. If at any point, an error state is reached, the string is invalid. If the accept state is reached, the string is valid.