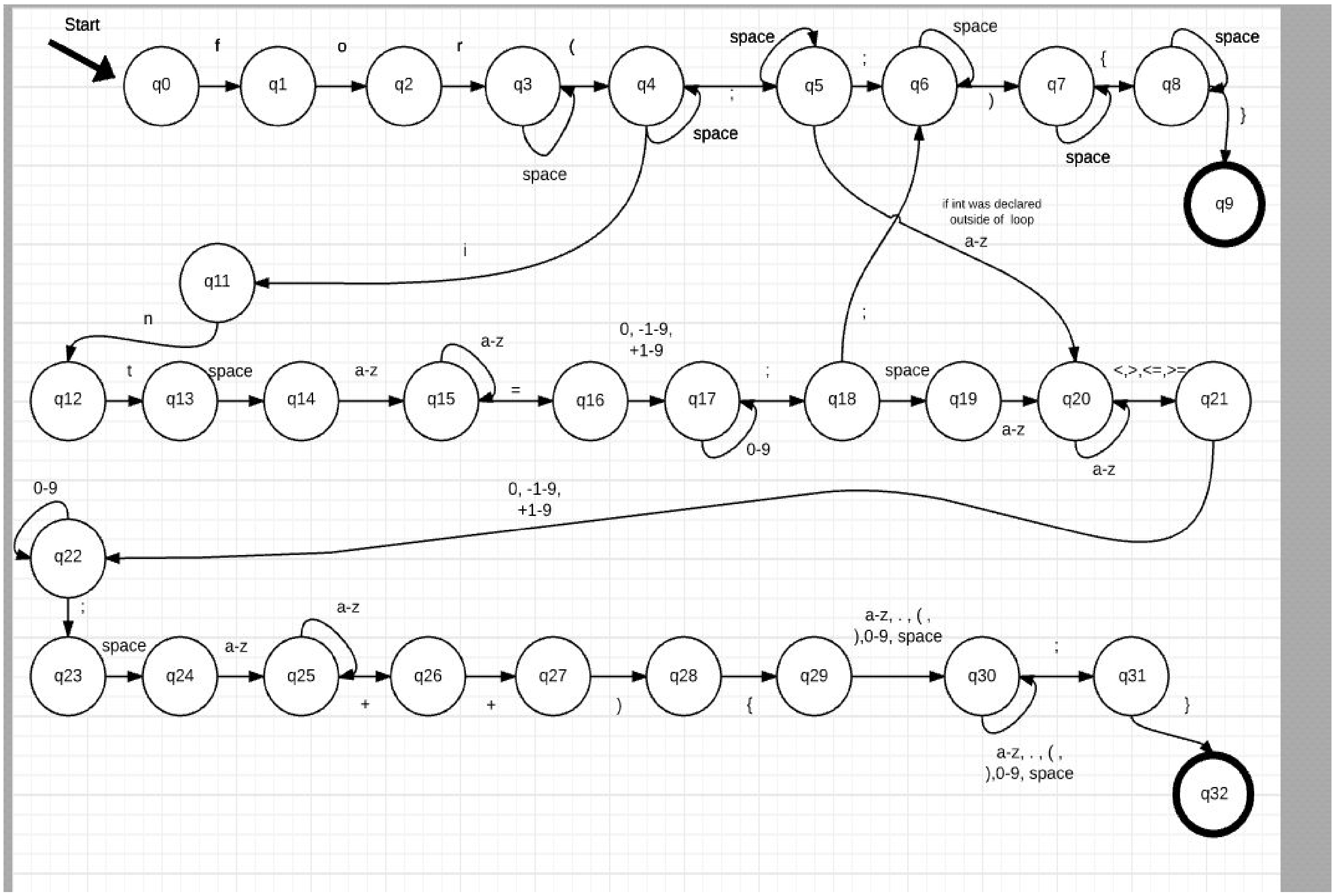
Formal Languages
Prof Rivas
Christopher Piccirillo
Feb 1, 2016
Lab 1

1)

DFA:

There are definitely multiple ways to do this as there are multiple ways to declare for loops. E.g. are we allowing for a variable to be declared outside of the loop? I accounted for that in my DFA, but there are other variations of the for loop as well. Another example is eliminating the upper bound check to decrease run time. I did NOT account for this in my DFA because it was not necessary to complete my diagram. I created my DFA like this because it allows for SOME variation in the assigned for loops but also covers the two assigned loops exactly.



2)

Creating this DFA definitely proved to be a challenge. It took me multiple attempts to get it into a final state that I was pleased with. I ran through this assignment at first in the literal sense. What I mean by that is that I wrote a DFA to work for these two assigned loops exactly how they were typed out. This was obviously a mistake and about halfway through I decided to completely restart to account for other variations, i.e different lengths of numbers and variable declarations. Another thing I had to take into account was how efficient I wanted to make my DFA. I could have easily created two separate paths for these two loops but I did my best to intertwine their pathways to give the user many choices down the line, no matter how far they progressed into their for loops.

Another challenge was to try to keep this neat. A goal of mine was to try an avoid any line overlaps and keep this chart readable and easy to follow. For the most part I would say I achieved my goal with only one overlap.

Out of this whole lab I would say it definitely helped me learn about the for loop syntax in java. I was not aware of the many different ways you could declare a for loop and have it come out as valid syntax. I had no idea that you could declare a for loop as follows and have java accept it.

```
for(;;i++){}
```

Obviously this won't do very much, but if Java accepted it then a complete DFA for all types of for loops would definitely have to consider this. The main takeaway from this is that I really know a lot less about what is valid syntax in java let alone all programming languages. I should really spend more time playing around with languages I like to get to know all the intricacies they possess.

3)

Implementing this is java would definitely be something completely new to me in the language. I have never had to tackle something like this before. If I had to validate a certain string to exact syntax patterns then I believe that could be done very linearly. The first thing that comes to mine would be inserting each character one by one into an array. So the array would contain the exact pattern of the string. I would then parse through the given string with a counter for each letter. As the counter increase I would compare the current character and the character in the array that matches the counter number. This is very strict and does not allow for any variance in the for loop so I do not think this would be the best way to implement this.

If I was to attempt to implement this to account for multiple different types of for loops with different declarations and format and the such I would have to completely change my take on this. I would attempt to define some pattern matching methods to check the given string for validity. For example it would be similar to the DFA. It would check the string for a for at first followed by some variation of an counter with () { }. This would be much more difficult to implement but would be a better option in my opinion as what use is a method that only accepts a certain string.