Formal Languages: Lab 1
Hiroki Hayashi
02/01/2016
Github repository: https://github.com/hiroki94/Formal-Languages-2016-Spring

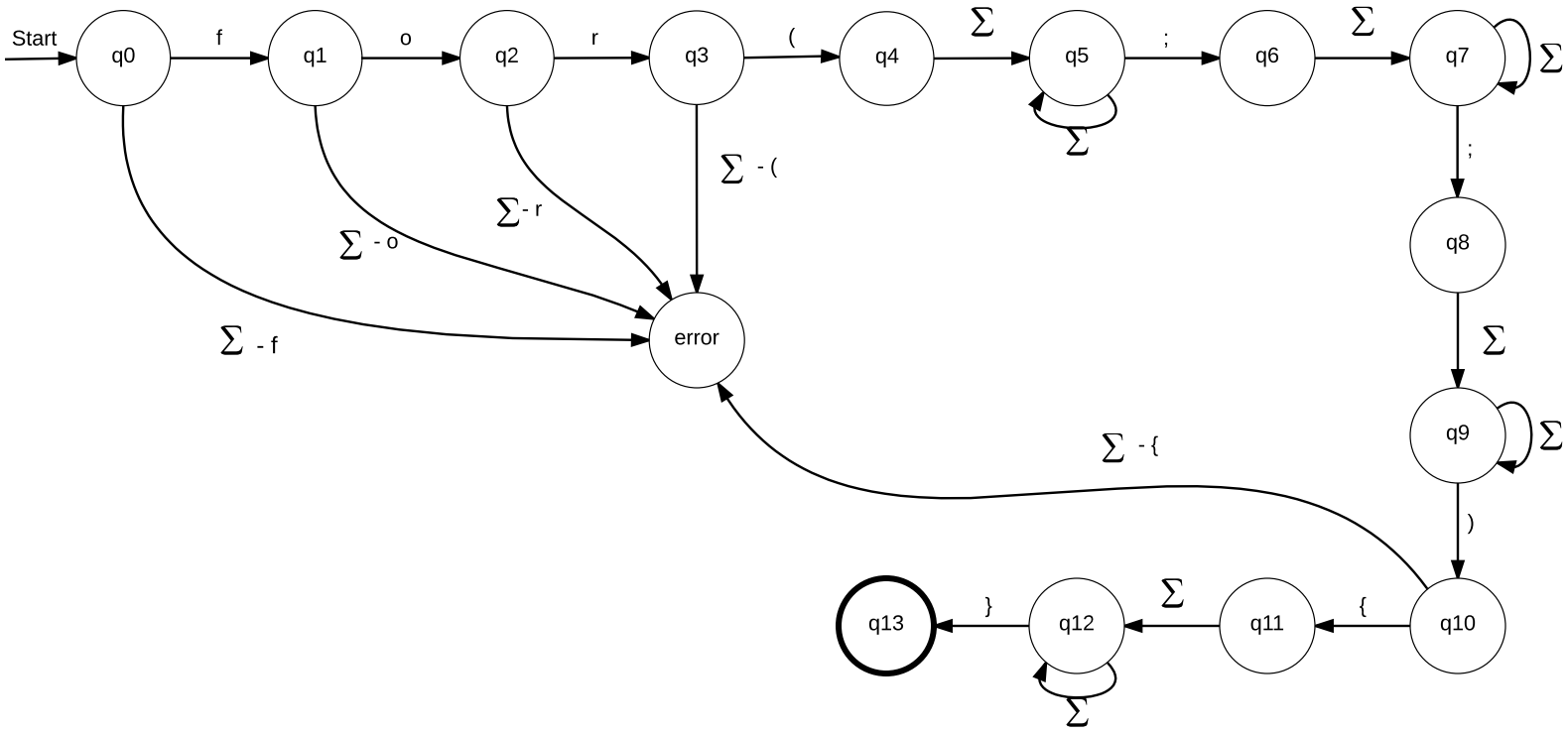
Essay 1:

I was faced with a few problems when I started working on the DFA for the for loop that was provided. One was that in this specific for loop, "i" was initially set to 1, the loop was to go until 10, and the i was to be incremented every loop. However, at first I was confused with how to write that in the DFA, and if other variables would also be valid. Next, I was faced with a looping problem: how I am supposed to loop the print statement in an FDA? I ultimately solved these two problems by using the sigma sign, so that any character in the alphabet can be the variable, and it can be looped as many times as it desired. However, I ended up not being able to create an error state for those that I assigned the sigma sign, due to the fact that if every possible character were abled to be valid, then no character could be sent to the error state. This is a problem that I could not figure out, although this did fix the problem, perhaps temporarily. This DFA also fits the second for loop, which contained nothing; just the skeleton of a for loop. The way I chose was maybe an easy way out, and problems may occur with this DFA, or it may prove to be a little too vague. I learned from this lab that the easiest solutions can be the answer at that moment, but not always in the long run. I ended up creating a second DFA that took into account every single character in the first for loop. This proved to have even more problems, as it became too complex to loop and to fit other variations of the for loop. In the end, I went back to the original simple version.

Formal Languages: Lab 1
Hiroki Hayashi
02/01/2016

Essay 2:

I have always though that this whole implementation looks quite like the linked lists that some languages have. Java is one of them, and I think that the for loop syntax could be validated using a linked list. Similar to the DFA, we could assign each character in the string to a node in a linked list. Then, we "link" these nodes so that it goes in the same order of the string. Then, we can go through every possible node, and check if every character at that position is valid for a for loop. Although this may sound tedious, this is something that is very close to the DFA structure, and it would be easy to visualize. Also, due to each node having a next (and a previous, depending on the linked list), we can easily take a node out or add one in a position we desire. This is why an array, although provides similar functions, would not be able to be implemented as a DFA.


Start
q0
f
q1
o
q2
r
q3
(
q4
Σ
q5
;
q6
Σ
q7
Σ
;
q8
Σ
q9
Σ
)
q10
{
q11
Σ
q12
}
q13
Σ
Σ
Σ - (
Σ - r
Σ - o
Σ - f
error
Σ - {