CS 2200 Spring 2019 HW 04

Dr. George Markowsky
Computer Science Department
Missouri Science & Tech

Due Tuesday, March 19, 2019, 11:59PM

Submit your HW as a SINGLE PDF file that includes your answers, your programs, and their output. Also include any data files if you used them. Please make sure that your PDF is well-organized so we can quickly find all the pieces of each problem that belong together.

PROBLEMS

1. (25 points) Write a Python NDFSA+ϵ simulator that meets the following specifications. The machine definition and tape will be stored in a text file that has the extension ".ndfsa". Note that we are not going to write a Python NDFSA simulator because it is very similar to this simulator. Also, if you have a file with no ϵ moves the NDFSA+ϵ simulator can run it without any issues.

   Each line of the text file shall begin with one of the following capital letters: A, B, D, O, S, T. We will explain the meaning of these letters below. The letters appear at the very beginning of the line and there is a single blank space after the letter for readability. Each line must begin with a letter. If the file does not meet the specifications, it is rejected. If a line does not have a letter on some line, the file is rejected.

   The first line must be an A line which contains just a single string consisting of all the characters in the alphabet. There is only one A line per file. To simplify coding, we make the restriction that the alphabet can never contain the character @ since we want to use it to represent ϵ. Note that you can enter ϵ into a Graphviz file and Graphviz will incorporate the ϵ in arrow labels but not in state labels. Depending on what text editor you are using, entering ϵ might be tricky, which is why I am letting you use @ instead of ϵ if you wish.

   The A line is followed by a series of S lines. S stands for state. An S line states that the line gives information about states such as their names and whether they are final or not. An S line has the following structure S S1,Bool1,S2,Bool2,...,Sk,Boolk where Si is the name of a state and Booli is 1 if the state is final and 0 is the state is not final.

Next comes a B line which contains the beginning (start) state There is only one such line and it looks like B State.

Next come the D lines where D stands for delta. The structure of a D line is as follows: D S_1,$C*_1$,N_1,S_2,$C*_2$,N_2...,S_k,$C*_k$,N_k where S_i is the current state, $C*_i$ is the character being read or @ if we are denoting an ϵ move, and N_i is the new state that the machine enters. Each line contains a whole number of triples although the number of complete triples per line is not specified. The lines could lack any entries, but they still should have the blank following the letter. Note, the .ndfsa file does not do anything special for non-determinism – it just lists all the possible moves just as for a FSA. It is up to your simulator to group the moves in a useful manner.

After the last D line, come a series of alternating T and O lines. T stands for tape and contains a character string representing a initial tape. Every T line is followed by an O line. The O line is blank if the preceding T line has not been processed by the machine. The line contains the word "accepted" if the machine would accept the string on the T line and contains the word "rejected" if the machine would reject the string on the T line.

After the program processes the file it should process any T lines for which there are no answers on the corresponding O lines. Once the simulator finishes, it should reconstruct the file that it read, but add any new computations to the updated file.

Include your program with comments in the PDF file that you will submit. Be sure that the comments include the information that I gave you about the format of the file. You will use this simulator to test all of the solutions to the remaining problems. For each problem you will include the .ndfsa file that contains the solution.

2. (20 points) Write a program that can generate a Graphviz file from either a .fsa or .ndfsa file. The straightforward way of doing this can generate multiple arrows between the same two nodes. Your generator should combine the arrows into a single arrow and merge the labels into one. For example, let's say that a machine transistions from state 1 to state 2 when it encounters either an a or a b. The Graphviz file should have just a single arrow from state 1 to state 2 with the label 'a,b'. Of course, if there are additional arrows from state 1 to state 2, the labels on those arrows should be added to the 'a,b' label. To test your program, produce the Graphviz files for all the FSA's that you produced for HW 02 and also for the NDFSA+ϵ that you will produce for the next problem.

3. (30 points) Write a Python program that takes regular expression written in our notation, and produces an NDFSA+ϵ that has exactly the same language as the regular expression. The NDFSA+ϵ should be specified by a .ndfsa file. To test your program, produce the NDFSA+ϵ to recognize the language generated by the regular expression (a+b)*abc(b+c)*. Produce at least 10 strings generated by the

regular expression of length 15 and run them through your NDFSA+ϵ simulator to prove that they are all accepted. Be sure to include a Graphviz diagram in your PDF of the NDFSA+ϵ that you produced.

4. (25 points) For each of the following, find a FSA automaton that recognizes the language or prove that there is no FSA that recognizes the language. If it can be solved by a FSA give the .fsa file, at least 10 sample cases, and a Graphviz diagram. If there is no FA that can recognize the language, prove that fact.

   (a) L_1 = { all binary strings divisible by 2 }

   (b) L_2 = { all binary strings divisible by 7 }

   (c) L_3 = { all unary strings that represent prime numbers}

   (d) L_4 = { all unary strings that represent composite numbers }

   (e) L_5 = { w $\in$ {a,b,c}* such that w is a palindrome}