CSE 30151 Theory of Computing: Homework 3
NFAs and Regex

Version 2: Sept. 8, 2017

Instructions

- Unless otherwise specified, all problems from "the book" are from Version 3. When a problem in the International Edition is different from Version 3, the problem will be listed as V3:x.yy/IE:x.zz, where x.zz is the equivalent number. When Version 2 has a different number, it will be listed as V3:x.yy/V2:x.zz. If either IE or V2 does not have a matching number, the problem text will be duplicated.
- You can prepare your solutions however you like (handwriting, LaTeX, etc.), but you must submit them in PDF. You can scan written solutions in the library or using a smartphone (with a scanner app like CamScanner). It is up to you to ensure that submissions are legible.
- Please give every PDF file a unique filename.
  - If you're making a complete submission (all problems), name your PDF file `netid-hw2.pdf`, where `netid` is replaced with your `NetID`.
  - If you're submitting some problems now and other problems later, name your file `netid-hw2-123.pdf`, where `123` is replaced with just the problems you are submitting now.
  - If you use the same filename twice, only the most recent version will be graded.
  - The time of submission is the time the most recent file was uploaded.
- If you use LaTeX and want to draw something like a state diagram, consider using the `tikz` package. A reference document is on the website under "Assignments".
- Submit your PDF file in Sakai. Don't forget to click the Submit (or Resubmit) button!

Practice Problems

These problems are from the book, and most have solutions listed for them. They are listed here for you to practice on as needed and any answers you generate **should not** be submitted. You are free to discuss these with others, but you are not allowed to post solutions to any public forum.

1. 1.7a,f NFA state diagrams with fixed number of states
2. 1.11 NFAs need only 1 accept state
3. 1.16a NFA to DFA
4. 1.10b,c Regex to NFA
5. 1.20 Regex to strings
6. 1.21a DFA to regex
7. 1.28 regex to NFA

Book Exercises

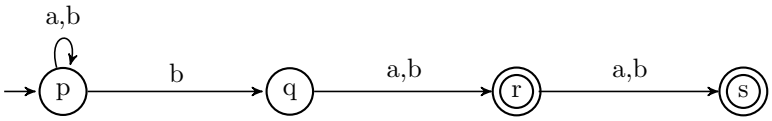
These problems are found in the text book and are to be answered and submitted by each student. You are to solve them yourself. Use of solution manuals from any source or shared solutions is a violation of the ND Honor Code. You are also not allowed to show your solutions to another student.

1. (5 points) 1.17 Build an NFA and convert to DFA
2. (5 points) 1.12 Language to DFA and regular expression
3. (5 points) 1,24a,b,f,g,h Finite State Transducers (FSTs): DFAs with outputs
4. (5 points) 1.25 Formal definition of FSTs
5. (5 points) 1.26 Define sample FSTs

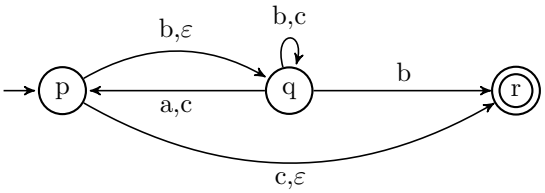
Non-book Problems

The following problems are not found in the text book. You are to solve them yourself. Use of any resource you used other than the text book or class notes must be cited. You are also not allowed to show your solutions to another student.

5. Consider the two DFAs M_1 and M_2 in Figs. 1.6 and 1.8 respectively. Build a DFA that takes the union of the two languages.
6. (5 points) Convert the following NFA called M to a DFA called M'. Identify all formal components of the DFA. For δ', either a state diagram or transition table is sufficient.



7. (5 points) Do the same on the following graph to convert to a DFA. Note that there are εs in this. Use the approach of Theorem 1.39 and explicitly show the corresponding E(q) as the set of states reachable by 0 or more ε edges from state q.



8. (5 points) Assume regular languages L_1 and L_2 are accepted by DFAs M_1 and M_2, where $M_1 = (S, \Sigma, \delta_1, s_1, F_1)$, $S = \{s_1, ..., s_n\}$, and $M_2 = (R, \Sigma, \delta_2, r_1, F_2)$., $R = \{r_1, ..., r_m\}$. Define $L_3 = L_1 - L_2$ as $\{w | w$ is in L_1 but not in $L_2\}$. Show by construction of an NFA that L_3 is regular. Describe informally how your construction works.
9. (5 points) For the DFA M_4 in Fig. 1.12 in the book, create a FA M_{4R} that accepts the reverse of any string accepted by this machine. Show that state sequences you would go through for the string "aaabba" for M_4 and the string "abbaaa" for M_{4R}.
10. (5 points) Now consider a generalization of the above question. Given any NFA N that accepts some language L, describe the process you would use to get a DFA M that accepts any string that is either in L or L_R, where L_R is the language of any string that is the reverse of L, i.e. if "abc" is in L, then "cba" is in L_R. Make sure you consider the case where N has multiple accept states.