CSE 34151 Theory of Computing: Homework 3
Regexes and DFA/NFAs

Version 1: Feb. 6, 2018

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 is different, it will be listed as V3:x.yy/V2:x.zz. If either IE or V2 do 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 **legible PDF**. You can scan written solutions on the printer in the back of the classroom, or using a smartphone (with a scanner app like CamScanner). It is up to you to ensure that submissions are legible. REMEMBER THAT IF WE CAN"T READ IT OR SCAN IS CUT OFF, YOU DON"T GET A GRADE FOR IT.
- The problems marked as "TEAM" may be solved in a collaborative fashion with up to 2 other students. In such cases, your submission should have the word "TEAM" at the start of the problem, followed by the names of your collaborators (must be other students in this class). When such problems are graded, the first submission encountered by the grader for the team will be used for a common grade for all identified team members.
- Please give every PDF file a unique filename.
  - If you're making a complete submission (all problems), name your PDF file `netid-hw#.pdf`, where `netid` is replaced with your `NetID` and # is the homework number.
  - If you're submitting some problems now and other problems later, name your file `netid-hw-1-2-3.pdf`, where `1-2-3` is replaced with just the problems you are submitting now. This may be useful for team submissions.
  - 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".
- You may also find the website http://madebyevan.com/fsm/ a useful tool for drawing state diagrams via drop and drag. It will output both .png image files and latex in the tikz format.
- Submit your PDF file in Sakai to the appropriate directory. 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. 1.10 Regex to NFA
2. 1.20 Regex to strings
3. 1.21a DFA to regex
4. 1.28 regex to NFA

5. 1.29a,c Pumping Lemma
6. 1.40a CLosure

Book Exercises

These problems are found in the text book and are to be answered and submitted by each student. **If they are not marked as "TEAM," you are to solve them individually.** If they are marked as "TEAM" you may submit the same answer as the others in your team. In any case, 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 not part of your TEAM.

1. (5pt) 1.12 Language to DFA and regular expression
   Solution: We are given the language:
   D = {w | w contains an even number of a's and an odd number of b's and does not contain the string ab}. We are asked to give a DFA with 5 states that recognizes D and a regular expression that generates D. Right away we can note that we will never be able to have an 'a' followed by a 'b' because the substring 'ab' will never be accepted. Using the formal definition of a DFA:

   $$(Q,\ \Sigma,\ \delta,\ q_0,\ F)$$

   We can make the following DFA: $Q = \{q_1, q_2, q_3, q_4, q_5\}$ $\Sigma = \{a, b\}$ $q_0 = q_1$ $F = \{q_4\}$ $\delta =$

state	a	b
q_1	q_5	q_2
q_2	q_3	q_1
q_3	q_4	q_5
q_4	q_3	q_5
q_5	q_5	q_5

   Based on this DFA, we can make the regex: b(bb)*(aa)*
2. (TEAM-5pt) Book 1.19a. Regex to NFA
3. (5pt) Book 1-19b. Regex to NFA
4. (TEAM-5pt) V3:1-41;V2:1-41;IE:1.31 - Show Closure. Hint Remember that showing a language is regular is equivalent to showing how to construct a DFA/NFA for any language constructed from operator applied to a regular language.
   Solution:
   Assume $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ accepts A and $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ accepts B. We want to build M that alternates between M_1 and M_2. To do this define $Q = Q_1 x Q_2 x \{1, 2\}$ where the third component says are we looking at A or B. So states are now triples.
   Now $\delta((q_1, q_2, 1), a) = (\delta(q_1, a), q_2, 2)$
   and $\delta((q_1, q_2, 2), a) = (q_1, \delta(q_2, a), 1)$
   $s = (s_1, s_2, 1)$
   $F = \{(q_1, q_2, 1)\}$ where q_1 is in F_1 and q_2 is in F_2
5. (10pt - Part a only is TEAM) V3:1.46a,c; V2:1-46a,c;IE:11.51a,c. Use Pumping Lemma
   Solution:
   Part ba. Try string 0^p10^p. x must be 0^a for some a between 0 and $p-1$, and y is 0^b for some b from 1 to $p-b$. Thus xy^iz for $i=0$ is $0^(p-b)1o^p$ where $b \geq 1$, which is not in L. Also, you could use $i=2$ which yields $0^(p+b)10^p = 0^p0^b10^p$, which again fails (0^b1 is not a valid 1^m
   Part c. If a language is regular then so is its complement, and vice versa. So let's look at $\{w|$ w is a palindrone $\}$. Try the string 0^p10^p. It does not pump.

Non-book Problems

The following problems are not found in the text book. **If they are not marked as "TEAM," you are to solve them individually.** If they are marked as "TEAM" you may submit the same answer as the others in your team. 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.

7. (TEAM - 5pt) 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 $\{w | w \text{ is in } L_1 \text{ but not in } L_2\}$. Show by construction of an NFA that L_3 is regular. Describe informally how your construction works.

   Solution: We know that L_1 is regular so there exists a machine $M_1 = (Q_1, \Sigma_1, \delta_1, q_{0_1}, F_1)$, and that L_2 is regular so there exists a machine $M_2 = (Q_2, \Sigma_2, \delta_2, q_{0_2}, F_2)$. We define M_3 to be the product of M_1 and M_2 where the final states of M_3 are those states in M_3, (q_i, q_j) with $q_i \in Q_1, q_j \in Q_2$ s.t. $(q_i, q_j) \in F_3$ if $q_i \in F_1$ and $q_j \notin F_2$. What we mean by the product of two machines is that we take $Q_1 \times Q_2$ to be the states of the new machine. The start state is the tuple which has both the start state of M_1 and the start state of M_2. The alphabet is simply the union of the two alphabets of the original machine, and the delta function does element-wise transition, similar to how the NFA to DFA construction works.

   Informally, this works by simultaneously checking both machines since we have their product. When we are done, we simply check if we are in a final state of M_1 paired with not a final state of M_2. If both were accepting, we reject. If neither were accepting, we still reject. And finally, if it was only accepted by M_2 we also reject. We get precisely the strings we desire.

Remark	Points
No proof by construction	-1

8. (5pt) Show using the Pumping Lemma that the language $L = \{(ab)^n(cd)^n \mid n \geq 0\}$ is not regular.

   Solution: (Ryan Mackey)

   I will prove by contradiction that language L is not regular. Assume that language L is regular. Thus, L can be represented by DFA M where $M = (Q, \Sigma, s, \delta, F)$. The amount of states in M is finite, so assume that based on the pigeonhole principle and $|Q|$ that the pumping length is p. Now consider a word $x \in L$ of length greater than p. x can be written in the form $x = uvw$ where $|uv| \leq p$ and $|v| \geq 1$. Then for $\forall i \in Z, uv^i \in L$, according to the definition of regular languages. But if $x = (ab)^p(cd)^p$, u and v will both lie in the first half of the word (the ab region). Thus, with an $i = 2$, $x' = (ab)^{(p+v)}(cd)^p$. $x' \notin L$. This gives a contradiction and proves that language L is not regular. *QED*

9. (Team - 5 pt) For the following NFA compute the regular expression accepted by converting it to a GNFA and then using the CONVERT algorithm of from Lemma 1.60 on it. Show work. Feel free to use variables like R_i to stand for intermediate expressions on an edge (with R_i written out below), but show entire final answer. You don't need to simplify.

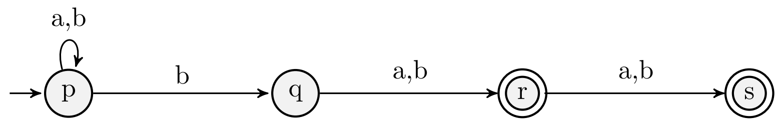

Figure 1: State diagram of NFA for question 6

   Solution:
   Any DFA/NFA can be converted to a GNFA by adding a start state q_s and a new accepting state q_a and by making appropriate ε transitions such that q_a is the only accepting state. Then, the procedure to convert a k-state GNFA (G) to regular expression is given below.
   CONVERT(G):

   1. Let k be the number of states of G

   2. If $k = 2$, then G must consist of only a start state and an accepting state, with a single arrow connecting the states labeled with regular expression R. Return R.

   3. If $k > 2$, we select any state q_{rip} different from q_s and q_a and let G' be the GNFA $(Q', \Sigma, q_s, \delta', q_a)$, where

$$Q' = Q - \{q_{rip}\},$$

and for any $q_i \in Q' - \{q_a\}$ and any $q_j \in Q' - \{q_s\}$, let

$$\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4),$$

for $R_1 = \delta(q_i, q_{rip}), R_2 = \delta(q_{rip}, q_{rip}), R_3 = \delta(q_{rip}, q_j), R_4 = \delta(q_i, q_j)$.

4. Compute CONVERT(G') and return the value.

Step 1: Starting from the given NFA $N : (Q, \Sigma, q_0, \delta, F)$, where $Q = \{p, q, r, s\}$, $\Sigma = \{a, b\}$, $q_0 = p$, δ can be inferred from the state diagram in figure 1, $F = \{r, s\}$, we make a six-state GNFA $G : (Q', \Sigma, q_0, \delta', F')$ by adding a start (q_s) and an accepting state (q_a). Therefore, $Q' = \{q_s, p, q, r, s, q_a\}$, $q_0 = q_s$, δ' can be inferred from the figure 2, and $F' = \{q_a\}$. We also convert the transitions to formal regular expressions.

Step 2: Since the G has more than 2 states, we need to remove a state q_{rip} which is different from q_s and q_a. Let $q_{rip} = p$. Thus, as per the CONVERT algorithm, we have $q_i = q_s$ and $q_j = q$. $R_1 = \delta(q_s, p) = \varepsilon, R_2 = \delta(p, p) = a \cup b, R_3 = \delta(p, q) = b$ and $R_4 = \delta(q_s, q) = \emptyset$.
Thus, $\delta(q_s, q) = (\varepsilon)(a \cup b)^*(b) \cup (\emptyset) = (a \cup b)^*(b)$. The state diagram of the reduced GNFA is given in figure 3.

Step 3: Since the G has more than 2 states, we need to remove a state q_{rip} which is different from q_s and q_a. Let $q_{rip} = q$. Thus, as per the CONVERT algorithm, we have $q_i = q_s$ and $q_j = r$. $R_1 = \delta(q_s, q) = (a \cup b)^*(b), R_2 = \delta(q, q) = \varepsilon, R_3 = \delta(q, r) = a \cup b$ and $R_4 = \delta(q_s, r) = \emptyset$.
Thus, $\delta(q_s, r) = ((a \cup b)^*(b))(\varepsilon)^*(a \cup b) \cup (\emptyset) = ((a \cup b)^*(b))(a \cup b)$. The state diagram of the reduced GNFA is given in figure 4.

Step 4: Since the G has more than 2 states, we need to remove a state q_{rip} which is different from q_s and q_a. Let $q_{rip} = s$. Thus, as per the CONVERT algorithm, we have $q_i = r$ and $q_j = q_a$. $R_1 = \delta(r, s) = (a \cup b), R_2 = \delta(s, s) = \varepsilon, R_3 = \delta(s, q_a) = \varepsilon$ and $R_4 = \delta(r, q_a) = \varepsilon$.
Thus, $\delta(r, q_a) = (a \cup b)(\varepsilon)^*(\varepsilon) \cup (\varepsilon) = (a \cup b) \cup \varepsilon$. The state diagram of the reduced GNFA is given in figure 5.

Step 5: Since the G has more than 2 states, we need to remove a state q_{rip} which is different from q_s and q_a. Let $q_{rip} = r$. Thus, as per the CONVERT algorithm, we have $q_i = q_s$ and $q_j = q_a$. $R_1 = \delta(q_s, r) = (a \cup b)^*(b)(a \cup b), R_2 = \delta(r, r) = \varepsilon, R_3 = \delta(r, q_a) = (a \cup b) \cup \varepsilon$ and $R_4 = \delta(q_s, q_a) = \emptyset$.
Thus, $\delta(q_s, q_a) = (a \cup b)^*(b)(a \cup b)(\varepsilon)^*((a \cup b) \cup \varepsilon) \cup (\emptyset) = (a \cup b)^*(b)(a \cup b)((a \cup b) \cup \varepsilon)$. The state diagram of the reduced GNFA is given in figure 6.

Step 6: Since the G has 2 states, the CONVERT algorithm terminates and returns $\delta(q_s, q_a) = (a \cup b)^*(b)(a \cup b)((a \cup b) \cup \varepsilon)$ as the required regular expression.

Therefore, the required regular expression is $(a \cup b)^*(b)(a \cup b)((a \cup b) \cup \varepsilon)$, representing the strings over the alphabet $\{a, b\}$ having a b in either the second last or the third last position.

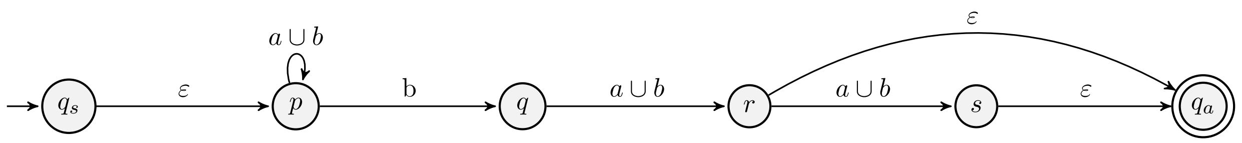

Figure 2: The GNFA G in step 1

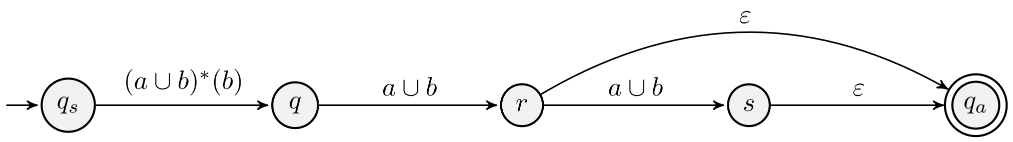

Figure 3: The GNFA G in step 2

-

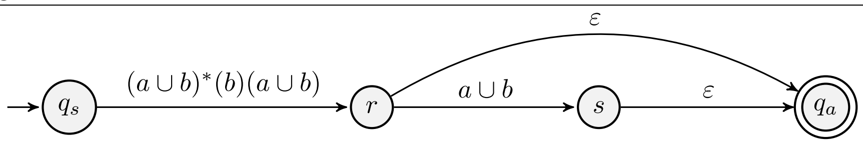

Figure 4: The GNFA G in step 3

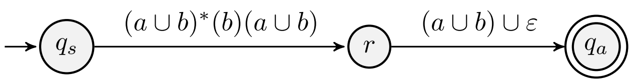

Figure 5: The GNFA G in step 4

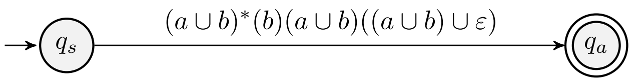

Figure 6: The final GNFA

10. (5pt) Do the same as above for the NFA in figure 7, i.e. compute the regex.
Solution:
Any DFA/NFA can be converted to a GNFA by adding a start state q_s and a new accepting state q_a and by making appropriate ε transitions such that q_a is the only accepting state. Then, the procedure to convert a k-state GNFA (G) to regular expression is given below.
CONVERT(G):

1. Let k be the number of states of G

2. If $k = 2$, then G must consist of only a start state and an accepting state, with a single arrow connecting the states labeled with regular expression R. Return R.

3. If $k > 2$, we select any state q_{rip} different from q_s and q_a and let G' be the GNFA $(Q', \Sigma, q_s, \delta', q_a)$, where

$$Q' = Q - \{q_{rip}\},$$

and for any $q_i \in Q' - \{q_a\}$ and any $q_j \in Q' - \{q_s\}$, let

$$\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4),$$

for $R_1 = \delta(q_i, q_{rip}), R_2 = \delta(q_{rip}, q_{rip}), R_3 = \delta(q_{rip}, q_j), R_4 = \delta(q_i, q_j)$.

4. Compute CONVERT(G') and return the value.

Step 1: Starting from the given NFA $N : (Q, \Sigma, q_0, \delta, F)$, where $Q = \{p, q, r\}$, $\Sigma = \{a, b\}$, $q_0 = p$, δ can be inferred from the state diagram in figure 7, $F = \{r\}$, we make a six-state GNFA $G : (Q', \Sigma, q_0, \delta', F')$ by adding a start (q_s) and an accepting state (q_a). Therefore, $Q' = \{q_s, p, q, r, q_a\}$, $q_0 = q_s$, δ' can be inferred from the figure 8, and $F' = \{q_a\}$. We also convert the transitions to formal regular expressions.

Step 2: Since the G has more than 2 states, we need to remove a state q_{rip} which is different from q_s and q_a. Let $q_{rip} = r$. Thus, as per the CONVERT algorithm, we have $q_i = \{q, p\}$ and $q_j = q_a$.
For $q_i = q$, $R_1 = \delta(q, r) = b, R_2 = \delta(r, r) = \emptyset, R_3 = \delta(r, q_a) = \varepsilon$ and $R_4 = \delta(q, q_a) = \emptyset$.
Thus, $\delta(q, q_a) = (b)(\emptyset)^*(\varepsilon) \cup (\emptyset) = b$.
For $q_i = p$, $R_1 = \delta(p, r) = (c \cup \varepsilon), R_2 = \delta(r, r) = \emptyset, R_3 = \delta(r, q_a) = \varepsilon$ and $R_4 = \delta(p, q_a) = \emptyset$.
Thus, $\delta(p, q_a) = (c \cup \varepsilon)(\emptyset)^*(\varepsilon) \cup (\emptyset) = c \cup \varepsilon$. The state diagram of the reduced GNFA is given in figure 9.

Step 3: Since the G has more than 2 states, we need to remove a state q_{rip} which is different from q_s and q_a. Let $q_{rip} = q$. Thus, as per the CONVERT algorithm, we have $q_i = p$ and $q_j = \{p, q_a\}$.

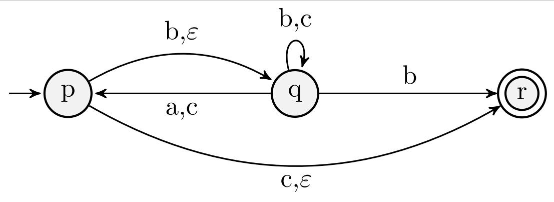

Figure 7: NFA for problem 7

For $q_j = p$, $R_1 = \delta(p, q) = (b \cup \varepsilon)$, $R_2 = \delta(q, q) = (b \cup c)$, $R_3 = \delta(q, p) = (a \cup c)$ and $R_4 = \delta(p, p) = \emptyset$. Thus, $\delta(p, p) = (b \cup \varepsilon)(b \cup c)^*(a \cup c) \cup (\emptyset) = (b \cup \varepsilon)(b \cup c)^*(a \cup c)$.

For $q_j = q_a$, $R_1 = \delta(p, q) = (b \cup \varepsilon)$, $R_2 = \delta(q, q) = (b \cup c)$, $R_3 = \delta(q, q_a) = b$, $R_4 = \delta(p, q_a) = (c \cup \varepsilon)$. Thus, $\delta(p, q_a) = (b \cup \varepsilon)(b \cup c)^*(b) \cup (c \cup \varepsilon)$. The state diagram of the reduced GNFA is given in figure 10.

Step 4: Since the G has more than 2 states, we need to remove a state q_{rip} which is different from q_s and q_a. Let $q_{rip} = p$. Thus, as per the CONVERT algorithm, we have $q_i = q_s$ and $q_j = q_a$. Therefore, $R_1 = \delta(q_s, p) = \varepsilon$, $R_2 = \delta(p, p) = (b \cup \varepsilon)(b \cup c)^*(a \cup c)$, $R_3 = \delta(p, q_a) = (b \cup \varepsilon)(b \cup c)^*(b) \cup (c \cup \varepsilon)$, $R_4 = \delta(q_s, q_a) = \emptyset$. Thus, $\delta(q_s, q_a) = (\varepsilon)[(b \cup \varepsilon)(b \cup c)^*(a \cup c)]^*(b \cup \varepsilon)(b \cup c)^*(b) \cup (c \cup \varepsilon) = [(b \cup \varepsilon)(b \cup c)^*(a \cup c)]^*(b \cup \varepsilon)(b \cup c)^*(b) \cup (c \cup \varepsilon)$. The state diagram of the reduced GNFA is given in figure 11.

Therefore, the required regular expression is $[(b \cup \varepsilon)(b \cup c)^*(a \cup c)]^*(b \cup \varepsilon)(b \cup c)^*(b) \cup (c \cup \varepsilon)$

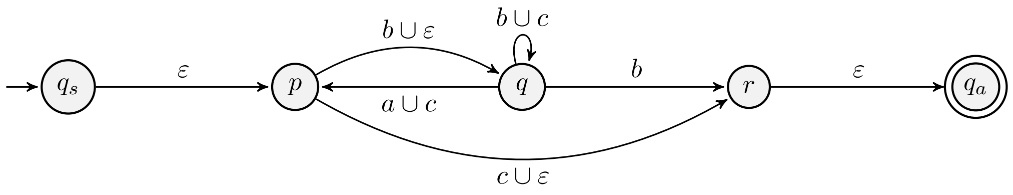

Figure 8: The GNFA G in step 1

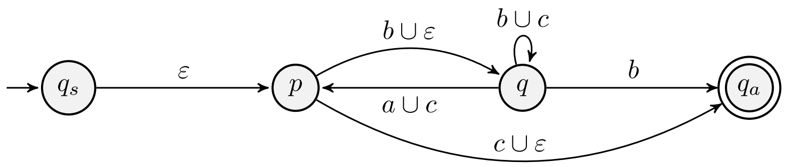

Figure 9: The GNFA G in step 2

Reason	Points
Correct small rip	1
Two medium large rips	$2 \times 2 = 4$
Transitions not regex	-0.5
Not following CONVERT	-5
Not adding the dummy start and end states	-0.5

Table 1: Rubric for problem 7 - Note: the errors in the GNFA propagate through the steps. They will be penalized accordingly.

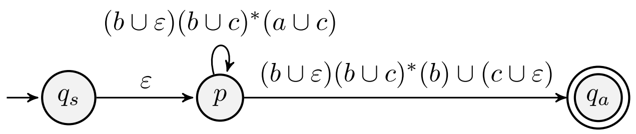
$(b \cup \varepsilon)(b \cup c)^*(a \cup c)$
q_s
ε
p
$(b \cup \varepsilon)(b \cup c)^*(b) \cup (c \cup \varepsilon)$
q_a

Figure 10: The GNFA G in step 3

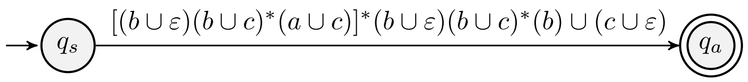
q_s
$[(b \cup \varepsilon)(b \cup c)^*(a \cup c)]^*(b \cup \varepsilon)(b \cup c)^*(b) \cup (c \cup \varepsilon)$
q_a

Figure 11: The GNFA G in step 4