CSE 30151 Theory of Computing: Homework 4
GNFAs, Regex, Pumping Lemma

Version 2: Sept. 19, 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. 1.10 Regex to NFA
2. 1.20 Regex to strings
3. 1.21a DFA to regex
4. 1.28 regex to NFA
5. 1.29 a,c Pumping Lemma to show non-regularity.

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) 1.13 Language to DFA and regular expression.
2. (5 points) 1.19a Regular expression to NFA
3. (5 points) 1.21b DFA to regex. Show steps
4. (5 points) 1.29b Pumping Lemma
5. (5 points) 1.37 Show a language is regular. (Hint: remember that if for the binary number x, $mod(x, n) = k$, then the number $mod(2x + b, n)$, where b is another binary digit, is the same as $mod(2k + b, n)$)

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.

6. (5 points) For the following NFA ompute 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.

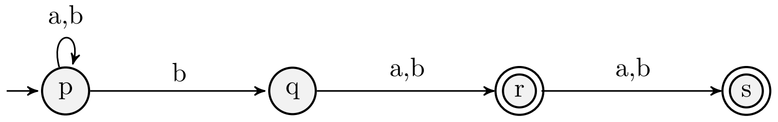


Figure 1: State diagram of NFA for question 6

Solution: (Satyaki Sikdar)

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.

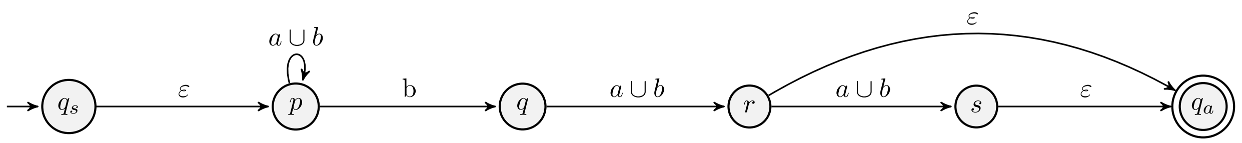


Figure 2: The GNFA G in step 1

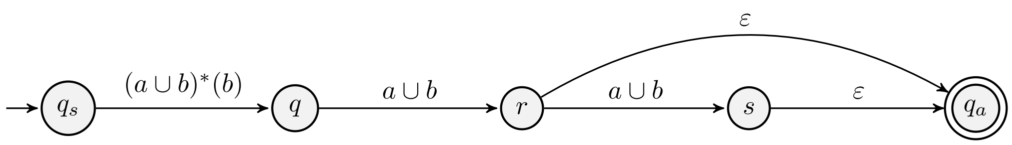


Figure 3: The GNFA G in step 2

7. (5 points) Do the same as above for the following NFA, i.e. compute the regex.

8. (5 points) 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

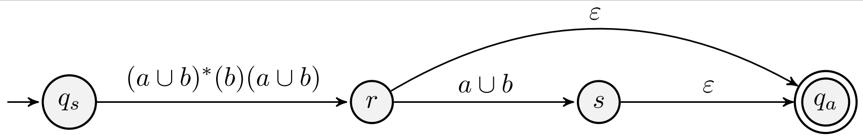


Figure 4: The GNFA G in step 3

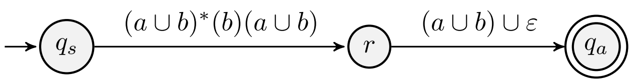


Figure 5: The GNFA G in step 4

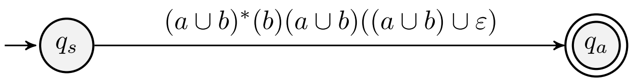


Figure 6: The final GNFA

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*

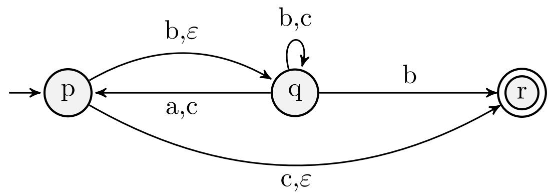
9. (5 points) Show that the language generated from $\Sigma = \{a, b\}$ where the only constraint is that the number of a's in any string is less than the number of b's, is not regular.

   Solution: (Ryan Mackey)

   I will prove by contradiction that language L, where $L = \{w \mid \textit{the number of } a's < \textit{ the number of } b's \textit{ in } w\}$, 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 $x = a^p b^{p+1}$, u and v will both lie in the first half of the word (the a region). Thus, with an $i = 3$, $x' = a^{(p+2*v)} b^{p+1}$. $v \geq 1$, so $p + 2 * v > p + 1$. This means that there are more a's than b's in x'. Thus, $x' \notin L$. This gives a contradiction and proves that language L is not regular. *QED*

10. (5 points) A *string homomorphism* h is a function that takes each character in its input string some alphabet Σ^*, replaces it by a string (typically from a different alphabet Σ'), and concatenates all resulting substrings together, i.e. $h : \Sigma^* \rightarrow (\Sigma')^*$. As an example if $h(0) = ab$ and $h(1) = \varepsilon$, then $h(00110) = ababab$

    Prove that if L is a regular language over Σ, then the language $L' = \{h(w) | w \text{ is in } L\}$ is a regular language over Σ'. Hint: show by induction over the construction of regular expressions.


p
q
r
b,ε
b,c
a,c
b
c,ε