CS 332: *Theory of Computation* — Prof. Mark Bun
Boston University — February 15, 2021

Homework 3 – Due Thursday, February 18, 2021 at 11:59 PM

Reminder Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to the course staff if asked. You must also identify your collaborators and write "Collaborators: none" if you worked by yourself. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.

Problems There are 5 required problems. Problem 2 will be autograded by AutomataTutor.

1. (**Regex to description**) Give plain English descriptions of the languages generated by each of the following regular expressions

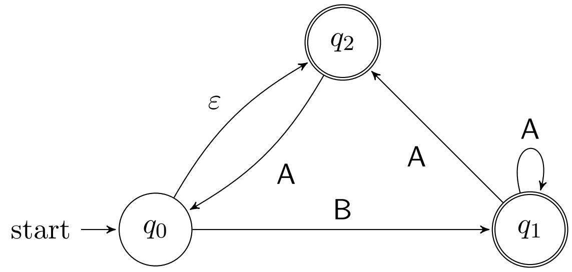
   (a) $(a \cup b)^* \cup c^*$
   (b) $1(000)^*1$
   (c) $a(ba)^*b$
   (d) $\emptyset^*$
   (e) $(\emptyset \cup \varepsilon)^*$

2. (**Regular expressions vs. finite automata**) Please log on to AutomataTutor to submit solutions for this question.

   (a) (**Description to regex**) Give regular expressions generating the following languages:
      i. $\{w \in \{0,1\}^* \mid w$ has exactly two 0's and at least one 1$\}$
      ii. $\{w \in \{0,1\}^* \mid w$ is not the string 01$\}$
      iii. $\{w \in \{0,1\}^* \mid$ the number of 1's in w is divisible by 3$\}$.

   (b) (**Regex to NFA**) Use the procedure described in class (also in Sipser, Lemma 1.55) to convert $(\mathsf{AT} \cup \mathsf{TA} \cup \mathsf{CG} \cup \mathsf{GC})^*$ to an equivalent NFA. Simplify your NFA.

   (c) (**NFA to regex**) Convert the following NFA to an equivalent regular expression.



3. (**Conversion procedures as algorithms**) Consider the following pseudocode describing an algorithm taking as input a regex and outputting the description of an equivalent NFA.

   Here, you can assume that the subroutines NFA.emptyLanguage(), NFA.emptyString(), and NFA.symbol(a) return NFAs recognizing the languages $\emptyset, \{\varepsilon\}, \{a\}$, respectively, as described in Sipser's proof of Lemma 1.55 or in Lecture 5, slide 24. Moreover, NFA.union(N_1, N_2) takes as input two NFAs and outputs the NFA recognizing $L(N_1) \cup L(N_2)$ described in Sipser's proof of Theorem 1.45, and similarly for NFA.concatenate and NFA.star.

RegexToNFA(R)
Input : Regular expression R
Output: Equivalent NFA N

```
if R = ∅ then
| return NFA.emptyLanguage();
else if R = ε then
| return NFA.emptyString();
else if R = a then
| return NFA.symbol(a);
else if R = R_1 ∪ R_2 then
| return NFA.union(RegexToNFA(R_1), RegexToNFA(R_2));
else if R = R_1 ∘ R_2 then
| return NFA.concatenate(RegexToNFA(R_1), RegexToNFA(R_2));
else if R = R_1^* then
| return NFA.star(RegexToNFA(R_1));
```

(a) If N_1 and N_2 are NFAs with s_1 and s_2 states, respectively, how many states does NFA.union(N_1, N_2) have? How about NFA.concatenate(N_1, N_2)? NFA.star(N_1)?

(b) The *size* of a regular expression R is the the number of appearances of $\emptyset, \varepsilon, \cup, \circ, ^*$ and alphabet symbols in R. If R is a regular expression of size 1, what is the maximum number of states in RegexToNFA(R)?

(c) For a natural number k, let $S(k)$ be the maximum number of states RegexToNFA(R) can have over all regexes R of size k. Prove by induction on k that $S(k) \leq 2k$.

Now consider the following pseudocode describing an algorithm taking as input an NFA and outputting an equivalent regex.

NFAtoRegex(N)
Input : NFA N
Output: Equivalent regular expression R

```
M_0 ← NFAtoGNFA(N);
k ← number of states of M_0;
for i ← 1 to k − 2 do
| Obtain M_i from M_{i−1} by ripping out state q_i and updating transitions appropriately;
end
return the regex labeling the transition from q_0 to q_accept in M_{k−2};
```

(d) Suppose the starting NFA N has exactly one symbol labeling each transition present in its state diagram. (This simplifying assumption makes the calculations cleaner, and in particular, independent of the alphabet size.)

Let $\ell(i)$ be the maximum possible size of a regular expression appearing on any transition in M_i. Prove by induction on i that $\ell(i) \leq 4^{i+1} - 3$.

(e) Show that if N is an NFA with s states, then NFAtoRegex(N) is a regular expression of size at most 4^{s+1}.

4. (**Distinguishing set method**)

(a) Let $REP_2 = \{ww \mid w \in \{0,1\}^2\}$. Show that $S = \{00, 01, 10, 11\}$ is pairwise distinguishable by REP_2. That is, for every pair $x, y \in S$, argue that there is a string z such that exactly one of xz and yz is in REP_2.

(b) What is the smallest number of states a DFA recognizing REP_2 can have? Explain your answer.

(c) For any $k \geq 1$, let $REP_k = \{ww \mid w \in \{0,1\}^k\}$. Show that every DFA recognizing REP_k requires at least 2^k states.

(d) Show that every NFA recognizing REP_k requires at least k states.

5. (**Non-regular languages**) Prove that the following languages are not regular. You may use the distinguishing set method and the closure of the class of regular languages under union, intersection, complement, and reverse.

(a) $L_1 = \{0^n 1^{2n} \mid n \geq 0\}$.

(b) $L_2 = \{w \in \{0,1\}^* \mid w \neq w^R\}$.

(c) $L_3 = \{www \mid w \in \{0,1\}^*\}$.

(d) $L_4 = \{x/y/z \mid x, y, z \in \{0,1\}^* \text{ are binary numbers such that } x + y = z\}$. The alphabet for this language is $\{0, 1, /\}$. For example, $10/10/100 \in L_3$ and $11/1/001 \notin L_3$.