CS46 Homework 4

This homework is due at 11:50PM on Sunday, February 13. This is a **16 point** homework. This homework has 3 parts:

- part 1 is individual
- part 2 and part 3 will be completed with a partner

General instructions: It's ok to discuss approaches at a high level with other students, but for part 1 your work should be your own, and for parts 2 and 3 your discussions should be just with your partner. Your partnership's write-up and code is your own: do not share it, and do not read other teams' write-ups. If you use any out-of-class references (anything except class notes, the textbook, or asking Lila), then you **must** cite these in your post-homework survey. Please refer to the course webpage or directly ask any questions you have about this policy.

The main **learning goal** of this homework is to work with NFAs, regular expressions, and our tools for proving that languages are/are not regular. As always, we shall continue to monotonically improve our proof-writing, clarity, and organization skills.

1 Automata Tutor problems

These problems should be completed[1] on Automata Tutor. You are allowed three attempts at each problem. I recommend that you *first* try to solve the problems on paper, *then* use the site to debug your solutions.

1. Let $\Sigma = \{a, b\}$, let $L_1 = \{w \mid \text{the length of } w \text{ is even}\}$, and let $L_2 = \{w \mid w \text{ begins and ends with } a\}$.
   (a) Construct an NFA for the language $L = L_1 \cup L_2$.
   (b) Construct an NFA for $L_1 \circ L_2$.
2. Write a regular expression for the following language over the alphabet $\Sigma = \{a, b\}$:

   $$\{w \mid w \text{ contains exactly two } a\text{s or at least two } b\text{s}\}$$

   Ponder the fact that union is pretty easy with NFAs and with regular expressions.
3. Let $\Sigma = \{0, 1\}$ and $L = \{w \mid w \text{ contains the substring } 0ab0 \text{ or } 1ab1 \text{ where } a, b \in \Sigma\}$.
   (a) Construct a regular expression for L.
   (b) Construct a DFA that recognizes L.
   (It is *strongly* recommended that you plan on paper first!)
   For fun: Take a screenshot of your masterpiece, email it to Lila, treasure it forever.
   (c) Construct an NFA that recognizes L.
   Your NFA should have substantially fewer states than your DFA. (Phew!)

[1] If you want to use late days on this assignment, you will need to submit solutions to these problems via github. The automatatutor site has only one deadline.

4. Let $L = \{w \mid$ if w contains an a, then it contains at least three as in a row$\}$ over $\Sigma = \{a, b\}$. So for example, L contains *abaaa*, *bbaabaaabaa*, ε, and *bb*. L does *not* contain *baa*, *abaaba*, or *bbba*.

   (a) Construct a DFA that recognizes L.

   (b) Construct an NFA that recognizes L.

   (c) Construct a regular expression that recognizes L.

5. Let $\Sigma = \{a, b\}$ and let $L = \{w \mid w$ contains an even number of as and an odd number of bs$\}$.

   (a) Construct an NFA recognizing L^*.
   (Hint: You can use the construction of Theorem 1.49 to get a correct answer, but this NFA will be capable of being simplified. Try describing L^* in English first.)

   (b) Think for a little while about a regular expression for L. Ponder the fact that intersection was much easier in automata than in regular expressions.
   (If you choose to, you can try to submit this regular expression on Automata Tutor, but because of the number of fiddly little details and the lack of useful feedback on the site, it is **not required**.)
   You might try to approach this problem by starting with a DFA and converting it to a regular expression, or by coming up with a regular expression based on your intuition about what sort of strings are in this language.

2 Nondeterminism in practice

In this programming portion of the lab, you will write a Python implementation which takes as input an NFA and a string and runs the NFA on that string, outputting either accept or reject. Your code should go in the file `nfa-simulator.py`, which includes some starter code. Your code should accept two files as input. The **first file** is a description of the NFA.

Sample format, file `nfa1.txt`:

```
q0 0
q1 0
q2 1

q0 a q1
q1 a q1
q1 b q1
q1 b q2
```

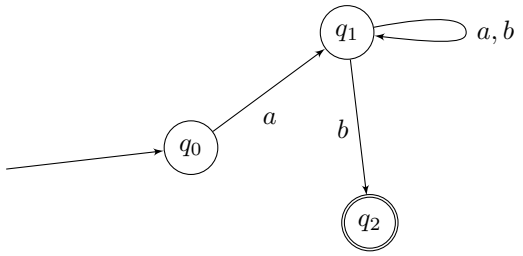

Figure 1: An NFA for the practice problem 3.1(b), $\{w \mid a$ begins with a and ends with $b\}$.

Each nonempty line has either 2 or 3 strings separated by spaces.

- A line with two strings represents a state.

  - the first string is the state name
  - the second string is 0 or 1 (1 means that this state is a final state)
  - the first state in the file is the start state (in the example, `q0`)

- A line with three strings represents a transition. "`q0 a q1`" is interpreted as "if the machine is in state `q0` and reads an `a`, then transition to state `q1`". An epsilon transition will be written with capital `E`, so `q0 E q1` means that the machine can move from state `q0` to `q1` without consuming a letter of the input string.

- You may assume that all states are listed before all transitions.

The **second file** is just a list of strings that are inputs to the machine.
Sample format, file `strings1.txt`:

```
a
ab
aabb
baa
bbb
aaabbb
bbabababababaaabab
```

Your code should test each input string against the machine and print out whether the string was accepted or not.

```
python3 nfa-simulator.py nfa1.txt strings1.txt
a: False
ab: True
aabb: True
baa: False
bbb: False
aaabbb: True
bbabababababaaabab: False
```

You should **write your own tests** and **check that your program works on many different NFAs and strings**, beyond the one example included in the repository.

The key to this project is (1) understanding how an NFA works, and (2) understanding how a python dictionary or other hash map, list, or set data structure works. You are welcome to come up with your own implementation details and algorithm. You are encouraged to make a top-down design with your partner, both as good coding practice and as a way to handle the challenge of implementing a deterministic python program which is simulating nondeterminism.

3 Written problems

These problems should be typeset in LaTeXand submitted using **github**.

1. **Understanding regular expressions.**

   For each of the following regular expressions over $\Sigma = \{a, b\}$, explain in English what language they describe. (Show your thought process.)

   (a) $b^*a(b^*a^*)^*$

   (b) $(a^*\emptyset b \cup ab \cup b^*\emptyset^*a)(b \cup \emptyset)$

   (c) $\epsilon \cup a(ba)^* \cup b(ab)^*$

2. **Regular expression identities.**

   Let R and S be regular expressions. Prove or disprove the following "identities". To prove the identity, you must argue that a string in the language defined on the left-hand side is in the language defined on the right-hand side, and vice versa. To disprove the identity, you must give a small counterexample string with real examples of regular expressions for R and S.

   (a) $(R^*)^* = R^*$

   (b) $(R \cup S)^* = R^* \cup S^*$

   (c) $(R^*S^*)^* = (R \cup S)^*$

3. Consider the following languages. For each, is the language regular? Support your claim with a proof.

   (a) $L_1 = \{a^k u a^k \mid k \geq 1 \text{ and } u \in \Sigma^*\}$ where $\Sigma = \{a, b\}$.

   (b) $L_2 = \{a^k b u a^k \mid k \geq 1 \text{ and } u \in \Sigma^*\}$ where $\Sigma = \{a, b\}$.

   (c) $L_3 = \{a^n b^m a^m b^n \mid m, n \geq 0\}$ where $\Sigma = \{a, b\}$.

   (d) $L_4 = \{a^{m-n} \mid \frac{m}{n} = 5\}$ where $\Sigma = \{a, b\}$.

   (e) $L_5 = \{w \mid w \text{ is not a palindrome}\}$ where $\Sigma = \{a, b\}$.

   (f) $L_6 = \{w \mid w = x_1 \# x_2 \# \cdots \# x_k \text{ for } k \geq 0, \text{ each } x_i \in L(a^*), \text{ and } x_i \neq x_j \text{ for } i \neq j\}$, where $\Sigma = \{a, \#\}$.

4. **(extra credit) Yes, but does the alphabet *really* matter?**

   We saw in lab that the language $L = \{w \mid w \text{ is a palindrome}\}$ is non-regular for some alphabet, but is regular for $\Sigma = \{a\}$. For this problem, let's set $\Sigma = \{a\}$.

   (a) Argue why there must exist at least one language $L \subseteq \Sigma^*$ which is not regular.

   (b) Give an example of a non-regular language $L \subseteq \Sigma^*$.

5. **(self-referentially cool, *definitely* extra credit) Are "regular expressions" regular?**

   We have a set of rules that describe how to build regular expressions, given an alphabet Σ and some additional symbols, '$\emptyset$', '$\cup$', '$\circ$', '*', '`(`', and '`)`'. Consider the language L over the alphabet given by Σ with these added symbols, defined as:

   $$L = \{w \mid w \text{ is a regular expression over } \Sigma\}$$

Prove that L is not regular.

(Thinking dizziness warning: be careful not to think yourself in circles with this one! A cool corollary of this claim is that there is no regular expression that matches all regular expressions.)