Student ID: 805167986

Collaborators: Sashwath Sundher, David Tran, Dylan Gunn

CS181 Winter 2021 – Problem Set 1

Due Monday, January 25, 11:59 pm

- Please write your student ID **and the names of anyone you collaborated with** in the spaces provided and attach this sheet to the front of your solutions. **Please do not include your name anywhere since the homework will be blind graded.**
- An extra credit of **5%** will be granted to solutions written using LaTeX. Here is one place where you can create LaTeX documents for free: `https://www.overleaf.com/`. The link also has tutorials to get you started. There are several other editors you can use. We have also posted a short LaTeX tutorial on CCLE under References.
- If you are writing solutions by hand, please write your answers in a neat and readable hand-writing.
- Always explain your answers. When a proof is requested, you should provide a rigorous proof.
- 20% of the points will be given if your answer is "I don't know". However, if instead of writing "I don't know" you write things that do not make any sense, no points will be given.
- The homework is expected to take anywhere between 10 to 16 hours. You are advised to start early.
- Submit your homework online on Gradescope. The Gradescope code is 5V7GW5.
- Homework points will be scaled according to the number of homework assignments. All assignments will be weighted equally. As per the syllabus, your homework assignments will together comprise 25% of your final grade.

Note: *All questions in the problem sets are challenging; you should not expect to know how to answer any question before trying to come up with innovative ideas and insights to tackle the question. If you want to do some practice problems before trying the questions on the problem set,* ***we suggest trying Exercise problems 1.4, 1.5, 1.9, 1.10, and 1.11 from the book. Do not turn in solutions to problems from the book.***

The machines that we called "Finite Automata" in class are also called "Deterministic Finite Automata (DFA)" and the machines we called "Magical Finite Automata" in class are also called "Non-Deterministic Finite Automata (NFA)".

Hint on all construction problems: If you want to prove that L is regular, it suffices to give an NFA for it. On the other hand, if you are told to assume that L' is regular, this means that there must exist a DFA recognizing L'.

1. **(15 points)** A *penta NFA* is a NFA that accepts a word w if there exist computation paths for w such that one-fifth or more of the ending states are accepting.

   (That is, let S_w be the set of states that the penta NFA could end up in after processing an input w. Let F be the set of accepting states of the penta NFA. Then, if $S_w \neq \emptyset$ and $\frac{|S_w \cap F|}{|S_w|} \geq \frac{1}{5}$, then the penta NFA should accept w. Otherwise, it should reject w.)

   Show that DFAs are equivalent to penta NFAs.

   In other words,

   (a) Show that for every DFA M there exists a penta NFA N that accepts the same language.

This means that for any DFA we can rename it to a *penta* NFA without changing the definition of the original DFA.
After renaming the original DFA M, the new NFA N will automatically accept M's language.
For every DFA

$$M = \{Q, \delta, \Sigma, q_0, F\}$$

there exists a *penta* NFA

$$N = \{Q, \delta', \Sigma, q_0, F\}$$

$$\delta'(q, a) = \{\delta(q, a) \mid \delta(q, \varepsilon) = \emptyset\}$$

□

   (b) Show that for every penta NFA N there exists a DFA M that accepts the same language.

For a *penta* NFA $N = \{Q, \delta, \Sigma, q_0, F\}$ there exists
DFA $M = \{Q', \delta', \Sigma', q_0', F'\}$ such that:
The states of M will be the power set of all states in N

$$Q' = \mathbb{P}(Q)$$

The transitions of M will be the union of the epsilon closure of all transitions in N at T which is an element of Q.

$$\delta' = \left(T \subseteq Q, a \in \Sigma', \bigcup_{q \in \varepsilon - closure(T)} \varepsilon - closure(\delta(q, a)) \right)$$

The alphabet of M is the same as the alphabet of N.

$$\Sigma' = \Sigma$$

The starting state of M will be the epsilon closure of the starting state q_0.

$$q_0' = \varepsilon - closure(\{q_0\})$$

The accepting states of M are the set S which is a subset of Q not including $\emptyset$ where the cardinality of the intersection of S and F divided by the cardinality of the set S is greater than or equal to $\frac{1}{5}$.

$$F' = \left\{ S \subseteq Q \mid S \neq \emptyset, \frac{|S \cap F|}{|S|} \geq \frac{1}{5} \right\}$$

□

Note that one direction will be harder than the other.

For each part, provide a **complete and rigorous** construction of the penta NFA or DFA as appropriate and provide an explanation as to why your construction works.

2. **(15 points).** Let L be any language and let L_R be the set of reverse strings, i.e.

$$L_R = \{x \mid \exists y \in L \text{ such that } |y| = |x| \text{ and } x_1 x_2 \dots x_n = y_n y_{n-1} \dots y_1\}.$$

As an example, if $L = \{happy, computation, automata, finite\}$, then
$L_R = \{yppah, noitatupmoc, atamotua, etinif\}$.
Show that, if L is regular, so is L_R.

Given L is a regular language, then there is a DFA $M = \{Q, \delta, \Sigma, q_0, F\}$ which solves L. We now are going to define an NFA $N = \{Q', \delta', \Sigma', q_0', F'\}$ which solves the reverse of L, L_R. q_0' is a state which epsilon transitions to all states in F so our NFA has one starting state.

$$q_0' = q_{start}$$

$$\delta_{start}(q_{start}, \varepsilon) = F$$

The states we have for N are the same as the states in M plus an additional state q_0'.

$$Q' = Q \cup \{q_0'\}$$

A transition function $\delta' = Q' \times (\Sigma' \cup \{\varepsilon\}) \to \mathbb{P}(Q')$ gives us the inverse of the transition function $\delta = Q \times \Sigma \to Q$, but what we want is an NFA N because we could have many edges going to single states in our DFA M but now we have many states exiting from a single state all on the same in N because it is the reverse. Therefore, in N we can be in many states at once when we solve L_R, which is unlike what we could be doing when we were solving L. In addition to this we must union with the new ε transitions out of the new starting state q_{start} and this set of transitions is denoted as δ_{start}.

$$\delta'(q' \in Q', \sigma \in \Sigma) = \{q \in Q \mid \delta(q, \sigma) = q'\} \cup \delta_{start}$$

Our alphabet is the same in N as it was in M.

$$\Sigma' = \Sigma$$

Our accepting state for N is just the starting state of M.

$$F' = \{q_0\}$$

Because we have defined an NFA N which solves L_R we can say L_R is a regular language. □

---

3. **(20 points)** Let L be any language and let L_{alt} be the set of strings in L with every other character removed, i.e.

$$L_{alt} = \{x \mid \exists y \in L \text{ such that } x_1x_2x_3\ldots = y_1y_3y_5\ldots\}.$$

As an example, if $L = \{happy, computation, automata, finite\}$, then
$L_{alt} = \{hpy, cmuain, atmt, fnt\}$.
Show that if L is a regular language then L_{alt} is regular.

Given a DFA $M = \{Q, \delta, \Sigma, q_0, F\}$ which solves the language L.
We want to make an NFA $N = \{Q', \delta', \Sigma', q_0', F'\}$ which solves the language L_{alt}.

The set of states in Q' for N is the set of states created from the Cartesian product between Q and itself.

$$Q' = Q \times \mathbb{P}(Q)$$

This makes every state in N a tuple which I will denote as

$$(q, Q_{alt}) \text{ such that } q \in Q \text{ and } Q_{alt} \subseteq Q$$

where we accept any input to go to any other state q is connected to and that creates the set of states Q_{alt}.
Because we are representing each state in our NFA as a tuple of states the initial state can be the tuple consisting of q_0 and the set $\{q_0\}$.

$$q_0' = (q_0, \{q_0\})$$

Our new transition function δ' will be transitions from the states in Q_{alt}. We then have 1 transition in N be the equivalent of 2 transitions in M.

$$\delta'((q, Q_{alt}) \in Q', \sigma \in \Sigma') = \bigcup_{q_{alt} \in Q_{alt}} ((p, \{\delta(p, \sigma_{alt})\}) \mid \sigma_{alt} \in \Sigma, p = \delta(q_{alt}, \sigma))$$

The alphabet for N is the same as the alphabet of M.

$$\Sigma' = \Sigma$$

The accepting states for N, F', will be the the set of states where $q \in F$ or at least one state in Q_{alt} is an accepting state. Because we have the first transitioned to state as well as the set of second transitioned to states, we can be certain that we will not miss an accepting state.

$$F' = \{(q, Q_{alt}) \in Q' \mid (q \in F) \vee (Q_{alt} \cap F \neq \emptyset)\}$$

Overall, if we input a string s_{alt} into our new NFA we will start at the state represented by $(q_0, \{q_0\})$. Then we essentially will transition twice for every input in the string s_{alt}, which will allow us to mimic the behavior of inputting s into the original DFA M which solved L. This way of constructing the NFA will allow us to notice an accepting state even if s_{alt} is an odd or even length because we will accept a state $a = (q, Q_{alt})$ if either $q \in F$ or if $\exists q_{alt} \in Q_{alt}$ where $q_{alt} \in F$
Because we have defined an NFA N which solves L_{alt} we can say L_{alt} is a regular language. $\square$

---

4. **(30 points)** Let L be any language and let $L_{\frac{1}{2}\_}$ be the set of all the first halves of strings in L, i.e.

$$L_{\frac{1}{2}\_} = \{x \mid \exists y \in \Sigma^* \text{ such that } |x| = |y| \text{ and } xy \in L\}.$$

As an example, if $L = \{happy, computation, automata, finite\}$, then $L_{\frac{1}{2}\_} = \{auto, fin\}$.
Show that if L is a regular language then $L_{\frac{1}{2}\_}$ is regular.
Hint: Think about the way we implemented two machines in "parallel" by using the Cartesian product. That idea may be useful for this problem.

If L is a regular language, then there is a DFA $M = \{Q, \delta, \Sigma, q_0, F\}$ which solves the language L.
We want to make an NFA $N = \{Q', \delta', \Sigma', q_0', F'\}$ which solves the language $L_{\frac{1}{2}\_}$.
This is the new state which will have epsilon transitions to the original DFA and all $|Q|$ NFAs.
To solve this we will essentially run the supposed half string through the original DFA while also running it through $|Q|$ NFA's which are duplicates of the original DFA but each transition is replaced with accepting the whole language Σ.
Our set of states will consist of a tuple where

(a) The first element of the tuple will be the state where we would be if we just inputted the string into the original DFA.

(b) The second element will be the label number of the NFA. This number will tell us which starting state we started at for each NFA.

(c) The third element will be the state which the NFA is currently at.

$$Q' = (Q \times \{1, 2, \ldots, |Q|\} \times Q) \cup \{q_{\frac{1}{2}\_}\}$$

Our new starting state will be

$$q_0' = q_{\frac{1}{2}\_}$$

where

$$\delta_{start}(q_{\frac{1}{2}\_}, \varepsilon) = \{(q_0, a, q_a) \mid a \; \forall \; a \in \{1, 2, \ldots, |Q|\}\}$$

This means that after the ε transition out of the starting state we will have $|Q|$ computation paths each starting at every state while also moving though the original DFA.
This delta function transitions the first element through the original DFA, leaves the second element as it is throughout computation so we can remember the starting state of each of the $|Q|$ NFAs, and it transitions on the entire alphabet to all states reachable in each of the $|Q|$ NFAs.
Then we union this with the ε-transitions from $q_{\frac{1}{2}\_}$.

$$\delta'(q = (x, y, z) \in Q', \sigma \in \Sigma') = \left\{ \bigcup_{\sigma_{nfa} \in \Sigma'} (\delta(x, \sigma), y, \delta(z, \sigma_{nfa})) \right\} \cup \delta_{start}$$

The alphabet for N is the same as the alphabet of M.

$$\Sigma' = \Sigma$$

The accepting states will be the states where f is in the set of final states from the DFA and the state which we are at in the DFA q_a matches the starting state of one or more of the $|Q|$ NFAs.

$$F' = \{q = (q_a, a, f) \mid f \in F\}$$

Overall, what we are doing on a higher level is taking a string $s_{\frac{1}{2}\_} \in L_{\frac{1}{2}\_}$ and we are putting this string $s_{\frac{1}{2}\_}$ into the original DFA. While we are also putting it into $|Q|$ NFAs which all are the same as the original DFA but have all their transition function replaced with accepting the entire alphabet Σ. This is represented by having each state represented by a tuple (x, y, z) where x is the state we are at in the original DFA, y is the label of the NFA, and z is the state in the NFA we are at. To accept we will run the string $s_{\frac{1}{2}\_}$ into the DFA then where it stops we will see if an accepting state is exactly $|s_{\frac{1}{2}\_}|$ states away.
Because we have defined an NFA N which solves $L_{\frac{1}{2}\_}$ we can say $L_{\frac{1}{2}\_}$ is a regular language. $\square$
