HW2

Due: September 28, 2021

Reminder: Your name should not appear anywhere on your handin; each individual page of the homework should include your Banner ID only. For your digital submission, each page should include work for only one problem (i.e., make a new page/new pages for each problem).

While collaboration is encouraged in this class, please remember not to take away notes from any labs or collaboration sessions, and to list your collaborators at the beginning of your homework submission.

Please monitor Piazza, as we will post clarifications of questions there. Hand in your solutions by 10:20am to Gradescope. See the course syllabus for the late policy.

Problem 1

(a) Convert the following two NFAs to equivalent DFAs.

**Hint:** Use the construction given in Theorem 1.39 from the textbook.

![Diagram 1](image1)

(i)

![Diagram 2](image2)

(ii)

(b) Give a regular expression generating the language. \{ w | w contains at least 3 1s \} over \( \Sigma = \{0, 1\} \).
Solution:

(a)  (i) The first step is to determine the states of the new DFA, which we’ll call $D$. Because the transition function of an NFA formally maps to the power set of its states, these will be the states of $D$ (so we have only one transition for a letter to a combination of states representing the multiple parallel possibilities in the NFA). So $Q = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$. 

Next, we determine $D$’s start and accept states. The start state will simply be $\{1\}$ since there are no $\epsilon$ transitions leaving it. The accept states will be any states in $Q$ that include $\{1\}$, thus $\{\{1\}, \{1, 2\}\}$. 

Finally, we determine the transition function, keeping in mind that the DFA must have exactly one arrow exiting each state per symbol in the alphabet, even if the state that arrow goes to is $\emptyset$. All letters loop at $\emptyset$, $a$ takes state 1 to states 1 and 2, so $(\{1\}, a) \rightarrow \{1, 2\}$, and we can continue in this fashion mapping transition functions in the NFA to transition functions in the DFA where instead of having multiple arrows for a letter, we have a single arrow to a state representing a set of states.

This gives us the following DFA:

![DFA Diagram]

(ii) Same general process as above, but this time we also have to deal with the epsilon transition. All this means is the start state for our DFA will be $E(1) = \{1, 2\}$, since we could jump immediately to 2 upon arriving at 1. This applies to all our other transitions—we include in any states that can be reached by epsilon transitions as the result of a transition via symbol, i.e. after identifying a transition that goes to some subset of $Q$, $R \in \mathcal{P}(Q)$, we actually use $E(R)$ as the resultant state in the DFA. For example, $\{2\}$ goes to $E(\{1\}) = \{1, 2\}$, since 2 is reachable via $\epsilon$-transition.
Now we prune any states that are not reachable (usually because they have no incoming arrows or the states pointing to them don’t have incoming arrows). We are left with the following DFA:

\[(b) \ (1 \cup 0)^*1(1 \cup 0)^*1(1 \cup 0)^*1(1 \cup 0)^*\]
Problem 2

Conver the following regular expressions to NFAs using the procedure given in Theorem 1.54. In all parts, $\Sigma = \{a, b\}$.

(a) $a(abb)^* \cup b$

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

A possible solution: These are just example solutions, pared down from an NFA with more states and more epsilon transitions that would follow directly from the constructions in the textbook.

Problem 3

Let $\Sigma = \{0, 1\}$ and let:

$D = \{w \mid w$ contains an equal number of occurrences of the substrings 01 and 10$\}$

Thus $101 \in D$ because 101 contains a single 01 and a single 10, but $1010 \notin D$ because 1010 contains 2 10s and 1 01. Show that $D$ is a regular language.
There are a couple ways you could show that $D$ is a regular language, including constructing an NFA or DFA that recognizes $D$ or providing a regular expression that describes it. Here, I'll do the latter, but it helps to think about or develop both approaches as you're working through the problem. Let's start by noticing that $D$ contains $\{\epsilon, 0^*, 1^*\}$, as these are all the possible strings of $\Sigma = \{0, 1\}$ that have no substrings that are either 01 or 10. If, after some long initial stretch of 0s or 1s, we see the other symbol, we know that the string at that point cannot be accepted yet because we've seen only one of 01 or 10. That means that we will eventually have to see the starting symbol again eventually to close out the pair of 01 with 10 or 10 with 01. We can sort of think of every time we change symbols from 0 to 1 or 1 to 0 within the string, a switch is being flipped between balance and imbalance. So 00000 is balanced, 000001 is imbalanced (1 occurrence of 01 and 0 occurrences of 10), 0000011111 is imbalanced (still 1-0), 00000111110 is balanced once again (equal occurrences of 01 and 10), 0000011111101 imbalanced and so on. Inductively, we can conclude that so long as the string is "sandwiched" by the same symbol, it will be in the language, i.e. have the same number of occurrences of 01 and 10.

This is starting to look somewhat like the recipe for a regular expression! We can either start and end with 0 or start and end with 1 (which sounds like a union of two possibilities), and we can repeat any symbol as many times as we want in a row (which sounds like a star operation), and flip between symbols as often as we like so long as we end in the start symbol, again a star operation, and everything is glued together with concatenation. So can write a regular expression like this:

$$D = 0^*(011^*00^*)^* \cup 1^*(100^*11^*)^*$$

This says that strings in $D$ that start with some number of 0s can be followed up with any number of 010 sandwiches (of whatever thickness... 00110 and 0111110 and 0000010000 are all what I'm calling 010 sandwiches), and the same goes for strings that start with 1. Both of these expressions in the union can be $\epsilon$.

Because we have written a regular expression that describes this language $D$, we can conclude that $D$ is a regular language.