Yash Shah
405 565 567

Homework 2. Due October 24, 9:59PM.

CS181: Fall 2022

GUIDELINES:

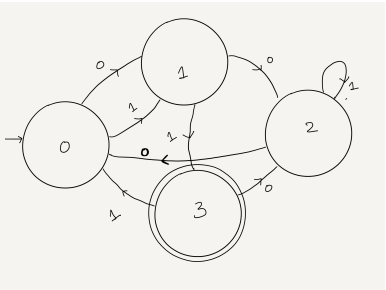
- Upload your assignments to Gradescope by 9:59 PM.
- Follow the instructions mentioned on the course webpage for uploading to Gradescope very carefully (including starting each problem on a new page and matching the pages with the assignments); this makes it easy and smooth for everyone. As the guidelines are simple enough, bad uploads will not be graded.
- You may use results proved in class without proofs as long as you state them clearly.
- Most importantly, make sure you adhere to the policies for academic honesty set out on the course webpage. The policies will be enforced strictly. Homework is a stepping stone for exams; keep in mind that reasonable partial credit will be awarded and trying the problems will help you a lot for the exams.
- All problem numbers correspond to our text 'Introduction to Theory of Computation' by Boaz Barak. So, exercise a.b refers to Chapter a, exercise b.

1. Exercise 5.4. You can assume that $m \leq 2^n$ for the problem. [1 point]

   [Hint: We already had an upper bound on the number of circuits of size s in class (this also applied to multi-output functions so you can use that as is). Next, what is the total number of functions $f : \{0,1\}^n \to \{0,1\}^m$? To get an upper bound use the reasoning we used in class: Think of the large truth table containing the outputs for every possible input string and the number of ways to fill the table. (How many options per cell and how many cells?)

   Now compare the number of circuits to the number of m-bit output functions as we did in class. You don't have to redo the arithmetic calculations we did in class - you can use those simplifications as is.]

2. Answer the following for the DFA shown in the figure [.75 point]:



(a) What is the set of accept states?

(b) What sequence of states does the machine go through on input 1010101?

(c) Write down an accepting input for the machine.

3. Draw a DFA that accepts strings which contain 011 as a substring. For instance, 001100, 00110, 00000110111 would be accepted whereas 010100 should not be. [1 point]

   [Hint: Think along the following lines: As you go over the input, you should initially skip all 1's. If you see a 0, then you may be onto something - it could be the first 0 of your sequence. If you see a 0 now, you can 'reset'. If you see a 1, you know you've seen a 0 and a 1. What should you look for next? You don't have to prove that you are correct - just the diagram with some explanations to make the grading easier would be enough.]

4. Use the construction we did in class to build a NFA for the Kleene star operation to build a NFA that accepts the strings $(\{01, 001\})^*$. [.75 point]

   [Hint: Start with a DFA that only accepts $\{01, 001\}$ and then use the construction in class. You don't have to prove your construction works - just a diagram is enough (with some verbal explanations for clarity).]

5. For a language L, define $DROPLAST(L)$ to be the language containing all strings that can be obtained by removing the last symbol of a string in L. Thus, $DROPLAST(L) = \{x : xb \in L, \text{ for some } b \in \{0, 1\}\}$. Show that if L is computable by a DFA, then DROPLAST(L) is computable by a NFA. [.5 points]

   [Hint: Think of adding appropriate ε transitions to accepting states. You don't have to prove your construction works - just a diagram is enough (with some verbal explanations for clarity).]

1. Exercise 5.4. You can assume that $m \leq 2^n$ for the problem. [1 point]

   [Hint: We already had an upper bound on the number of circuits of size s in class (this also applied to multi-output functions so you can use that as is). Next, what is the total number of functions $f : \{0,1\}^n \to \{0,1\}^m$? To get an upper bound use the reasoning we used in class: Think of the large truth table containing the outputs for every possible input string and the number of ways to fill the table. (How many options per cell and how many cells?)

   Now compare the number of circuits to the number of m-bit output functions as we did in class. You don't have to redo the arithmetic calculations we did in class - you can use those simplifications as is.]

As shown in lecture, for a particular circuit with size s, there exists an upper bound on the number of circuits. For a multi-output NAND, the upper bound for inputs n,m

$$\left|\text{size}_{n,m}(s)\right| \leq 2^{c \cdot s \cdot \log(s)}$$

For such a multi-input NAND, the total number of functions from $\{0,1\}^n \to \{0,1\}^m$ is $2^{2^n \cdot m}$

Using these properties, and the assumption that $m \leq 2^n$, we must prove that for a number $\delta > 0$, there exists at least one function $f : \{0,1\}^n \to \{0,1\}^m$ such that there are at least $\delta m 2^n$ NAND gates to compute.

if we let c be the constant such that $|\text{size}_{n,m}(s)| \leq 2^{cs\log(s)}$ and $\delta = \frac{1}{2c}$, then setting $s = \delta m \frac{2^n}{n}$ we see that

$$\begin{aligned}
\left|\text{size}_{n,m}\left(m\tfrac{2^n}{n}\right)\right| &\leq 2^{c\delta m \frac{2^n}{n}\log(s)} \\
&\leq 2^{c \cdot \frac{1}{2c} \cdot m \frac{2^n}{n}\log(s)} \\
&\leq 2^{\frac{m2^n}{2n}\log(s)} \\
&\leq 2^{m2^n \frac{\log(s)}{2n}}
\end{aligned}$$

Using the fact that $m \leq 2^n$, $m \cdot 2^n \leq 2^{2n}$. Therefore,

$$\delta m \tfrac{2^n}{n} = s < 2^{2n} \Rightarrow s < 2^{2n}$$

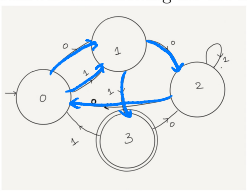
$$\log(s) < 2n \Rightarrow \frac{\log(s)}{2n} < 1$$

Henceforth, taking the upper bounds of these respective equalities we get

$$|\text{size}_{n,m}(s)| \leq 2^{m2^n \frac{\log(s)}{2n}} \Rightarrow |\text{size}_{n,m}(s)| < 2^{m2^n(1)} = |\text{ALL}_{n,m}|$$

since $|\text{size}_{n,m}(s)| < 2^{m2^n}$ which is what we found to be the total number of functions mapping n bits $\to m$ bits, there must be at least one function that requires more than $8m2^n$ NAND gates to compute.

2. Answer the following for the DFA shown in the figure [.75 point]:



(a) What is the set of accept states?
(b) What sequence of states does the machine go through on input 1010101?
(c) Write down an accepting input for the machine.

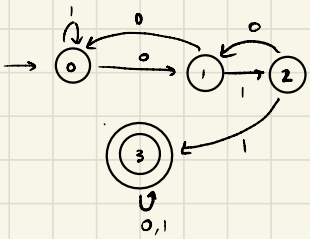
a) {3}

b) 0 → 1 → 2 → 2 → 0 → 1 → 1 → 3
1 0 1 0 1 0 1

c) 00011

3. Draw a DFA that accepts strings which contain 011 as a substring. For instance, 001100, 00110, 00000110111 would be accepted whereas 010100 should not be. [1 point]

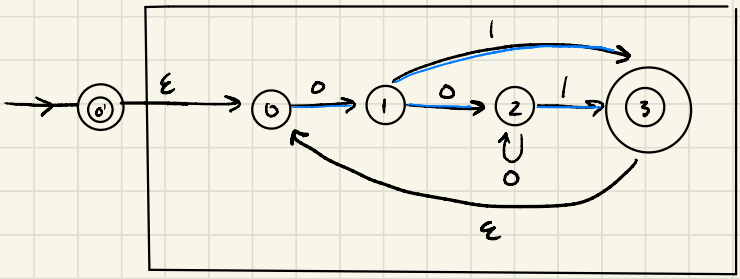
   [Hint: Think along the following lines: As you go over the input, you should initially skip all 1's. If you see a 0, then you may be onto something - it could be the first 0 of your sequence. If you see a 0 now, you can 'reset'. If you see a 1, you know you've seen a 0 and a 1. What should you look for next? You don't have to prove that you are correct - just the diagram with some explanations to make the grading easier would be enough.]



In this solution, the shortest substring contains the path $0 \to 1 \to 2 \to 3$. Thus any strings that contain 011 is supported. In order to reach the accepted path, that sequence of 011 is required and any number of 0s and 1s can be appended from the third state to the accepted state.

4. Use the construction we did in class to build a NFA for the Kleene star operation to build a NFA that accepts the strings $(\{01, 001\})^*$. [.75 point]

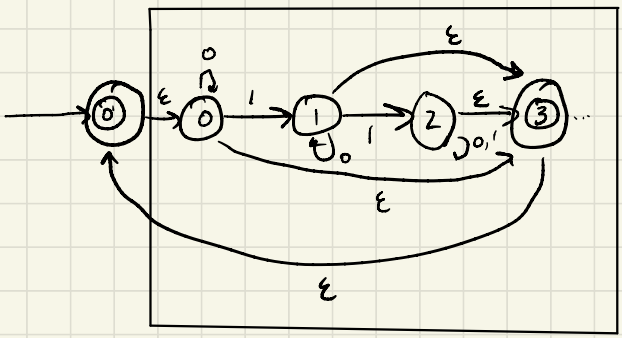
   [Hint: Start with a DFA that only accepts $\{01, 001\}$ and then use the construction in class. You don't have to prove your construction works - just a diagram is enough (with some verbal explanations for clarity).]



This construction has a dummy node connected by a ε to account for the empty string. If the string isn't empty, then the strings {01, 001} are in the DFA as exemplified by the blue color.

5. For a language L, define $DROPLAST(L)$ to be the language containing all strings that can be obtained by removing the last symbol of a string in L. Thus, $DROPLAST(L) = \{x : xb \in L, \text{ for some } b \in \{0,1\}\}$. Show that if L is computable by a DFA, then DROPLAST(L) is computable by a NFA. [.5 points]

   [Hint: Think of adding appropriate ε transitions to accepting states. You don't have to prove your construction works - just a diagram is enough (with some verbal explanations for clarity).]



For all bits in the DFA, each incoming edge to the accepted state must have an ε transition.

This diagram has all incoming edges into the accepted state as ε so that when it hits the second to last bit, it goes to the accepted state. If the accepted state itself hits, it goes to the dummy node to indicate an empty string.

Therefore, if L is computed by the DFA, DROPLAST(L) is computable by NFA