CS 181: Homework 2

Einar Balan

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.]

   Answer

   We start by defining two functions (similar to what we did in class)

$$\text{All}_{n,m} = \{f : \{0,1\}^n \to \{0,1\}^m\}$$

$$\text{SIZE}_{n,m}(s) = \{\text{All functions with m bit outputs computable by a circuit of size } \leq s\}$$

   The goal is to show $|\text{All}_{n,m}| > |\text{SIZE}_{n,m}(\delta m \frac{2^n}{n})|$. If this is the case, then there must be some functions that require more than size $\delta m \frac{2^n}{n}$.

   We showed in class that $|\text{SIZE}_{n,m}(s)|$ is $\leq (2)2^{12(n+s)log_2(n+s)}$. To count the size of $|\text{All}_{n,m}|$, we consider the fact that there are 2^n rows and 2^m possible choices of output per row. This results in $(2^m)^{2^n}$ posisble functions. Now we must show that, for sufficiently large m and n (as stated by Professor on EdStem),

$$(2^m)^{2^n} > (2)2^{12(n+s)log_2(n+s)}$$

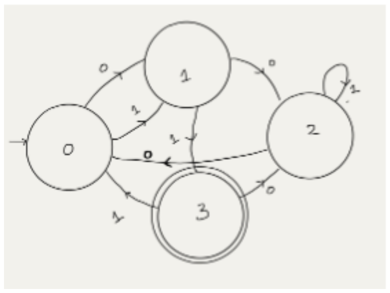
   Simplifying and substituting s we get

$$(2^m)^{2^n} > 2^{12(n^2+\delta m * 2^n)}$$

   Now we can simply pass in arbitrary values, $\delta = \frac{1}{120}, m = 7$, and $n = 7$. Computing this we get that $5.283 \times 10^{269} > 9.504 \times 10^{203}$, which is clearly true. This same thing holds as m and n increase. The goal is shown.

   □

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.

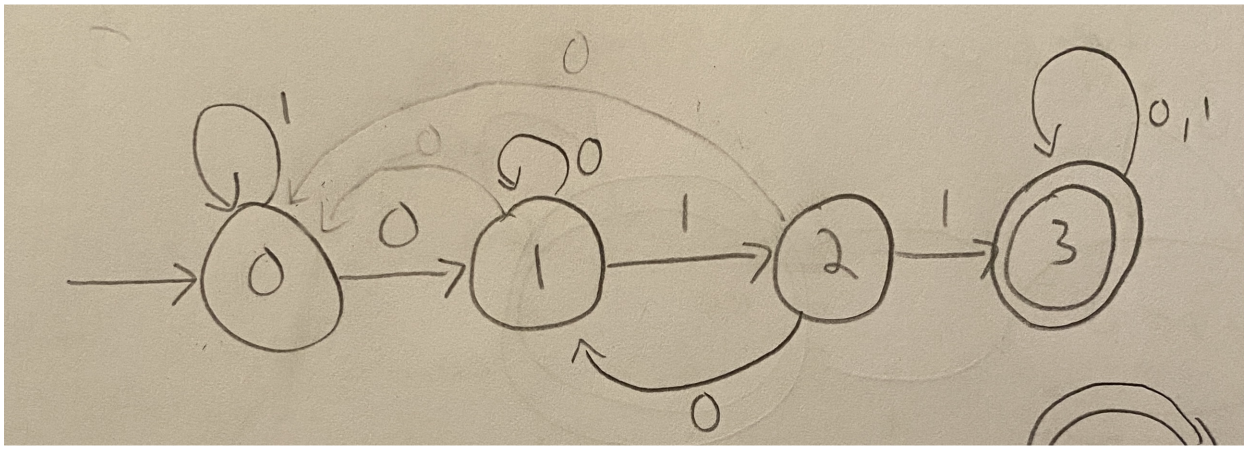
Answer

a) $S = \{3\}$
b) $0 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 2$
c) 01

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.]

   Answer

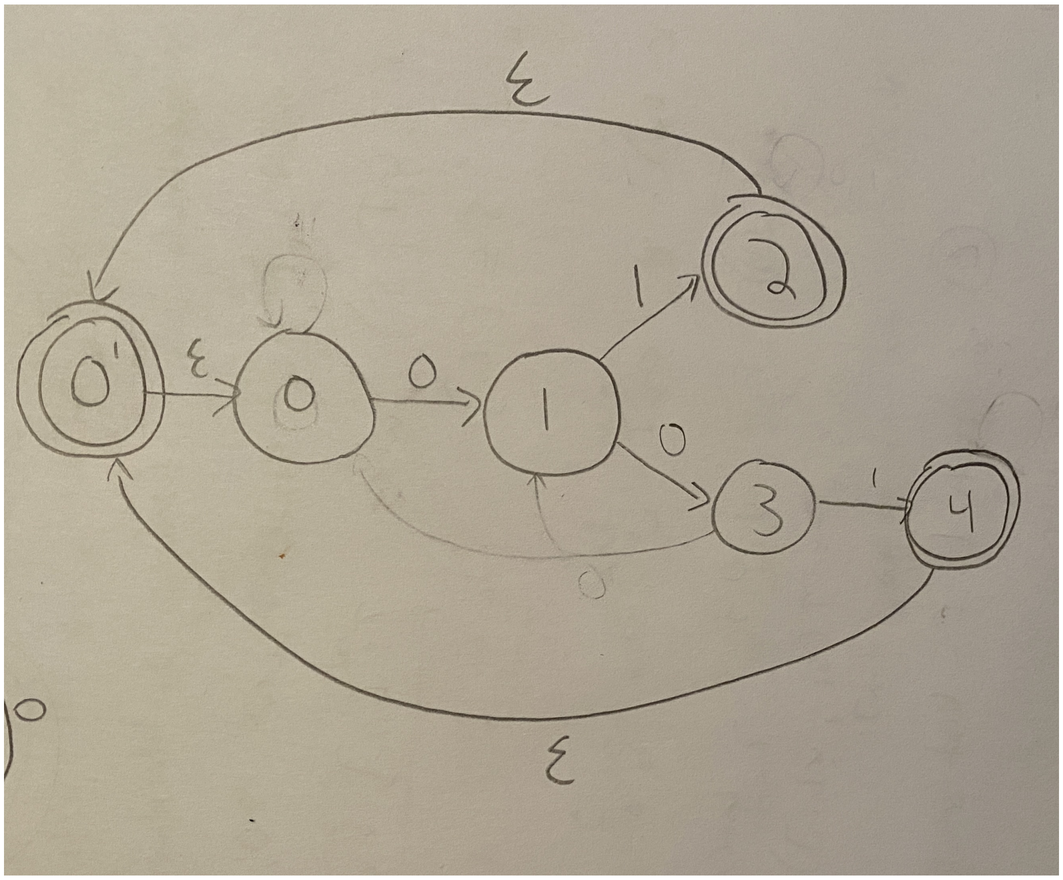


   Following the hint, the DFA shown will only progress towards the accepting state if the sequence 011 is followed. If it is not, then the state is reset back to the state where a 0 has been seen. However, if 011 is encountered then the DFA enters a persistent accepting state. That is, it is not possible to leave the accepting state once it has been entered. This is consistent with the fact that if 011 is encountered within an input then the input string will always contain 011 (it won't magically disappear somewhere).

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).]

   Answer



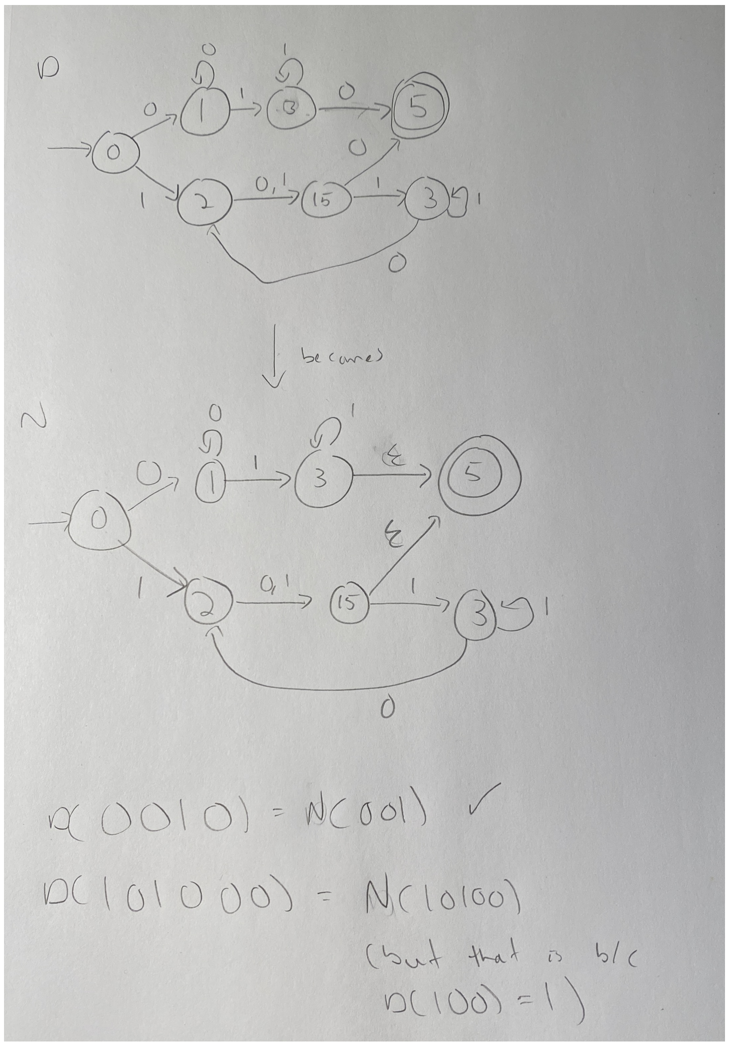
   We use the same idea from class, and in the previous problem except we have a branching path now to allow for two different accepted strings. Dead state when the string does not follow the pattern of one of the strings within the set.

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).]

   Answer

   Considering we have a DFA computing L, we simply need to add ε transitions from all states feeding into accepting states and remove any 0,1 transitions into the accepting states. The diagram below shows an example of this.


D
becomes
N
D(0010) = N(001) ✓
D(101000) = N(10100)
(but that is b/c D(100) = 1)