10/21/21
Haley Kim

CS181
HW 2

1. $m \leq 2^n$

Prove that $\exists$ a number $\delta > 0$ s.t. $\forall n, m$, $\exists$ a function $f: \{0,1\}^n \to \{0,1\}^m$ that requires at least $\delta m \cdot 2^n / n$ NAND gates to compute.
Hint: how many functions from $\{0,1\}^n \to \{0,1\}^m$? $(2^m)^{2^n} = 2^{m \cdot 2^n}$

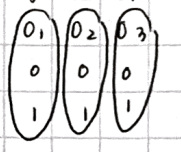
Thm: every $f: \{0,1\}^n \to \{0,1\}^m$ can be computed by a boolean circuit of size $O(n \cdot m \cdot 2^n)$
Remark: If we start w/ size s boolean circuit, then we get a size $\leq 3s$ NAND circuit.

For $\{0,1\}^n$, $\exists$ 2^n different truth tables.
For $\{0,1\}^m$, $\exists$ 2^m different truth tables.
Thus, $\exists$ $2^{m \cdot 2^n}$ different functions $f: \{0,1\}^n \to \{0,1\}^m$.

* Counting Truth Tables, $n = 2$, $m = 2$

	i_1	i_0	o_1 o_0	
R = 2^n	0	0	s_1	4 different outputs per input
	0	1	s_2	2^m different outputs per input
	1	0	⋮	total # functions = $(2^m)^{2^n} = 2^{m \cdot 2^n}$
	1	1		

* single-output counting



$\Rightarrow 2^R$ ← rows
↑ # outputs P per row

* multi-output counting

o_{11} o_{10} o_{21} o_{20} ⋯ o_{R0} $\quad m \cdot R = m \cdot 2^n$
↑ row ↑ col

We W.T.S. that $ALL_n \notin SIZE\left(\frac{\delta m 2^n}{n}\right)$
As shown above, $|ALL_n| = 2^{m \cdot 2^n}$
$|SIZE(s)| \equiv$ how many NCPs there are with size $\leq s$.
$|\#$ programs of size $s| \leq 2^{20 s \log(s)}$ (as shown in lecture)
$|\#$ programs of size $s| \leq |\#$ prog.s of size $1| + |\#$ prg.s of size $2| + \cdots + |\#$ prg.s of size $s|$

$$\leq \sum_{\ell=1}^{s} 2^{20\ell \cdot \log(\ell)}$$

$$\leq 2^{40 s \log s} \quad \text{(as shown in lecture)}$$

Thus, $|size_n(s)| \leq 2^{40 s \cdot \log(s)}$
If $ALL_n \subseteq size_n(s)$, then $|ALL_n| \leq |SIZE_n(s)|$ must hold.
Thus, $2^{m \cdot 2^n} \leq 2^{40 s \log(s)}$ must hold for $ALL_n \subseteq SIZE_n(s)$ to be true.

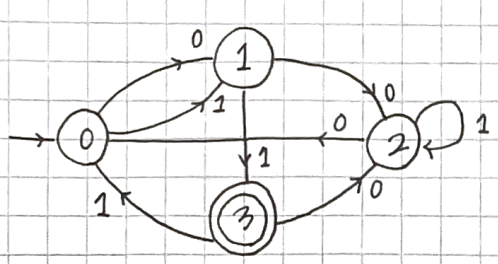
$$m \cdot 2^n \leq 40 \cdot s \cdot \log(s), \quad m \cdot 2^n \leq 40\left(\frac{\delta m \cdot 2^n}{n}\right) \log\left(\frac{\delta m \cdot 2^n}{n}\right)$$

$$2^n \leq 40 \delta \frac{2^n}{n} \log\left(\delta m \frac{2^n}{n}\right), \quad \log\left(\delta m \frac{2^n}{n}\right) \Rightarrow n \quad \text{(as shown in lecture, also as } n \to \infty)$$

$$2^n \leq 40 \delta \frac{2^n}{n} \cdot n, \quad 2^n \leq 40 \delta 2^n$$

This final inequality does not hold when $0 \leq \delta < \frac{1}{40}$.
Thus, this assumption does not hold, and $ALL_n \notin SIZE\left(\delta m \frac{2^n}{n}\right)$.
Since $ALL \notin SIZE\left(\delta m \frac{2^n}{n}\right)$, $\exists$ a function $\{0,1\}^m \to \{0,1\}^n$ that requires at least $\frac{\delta m 2^n}{n}$ gates.

2.

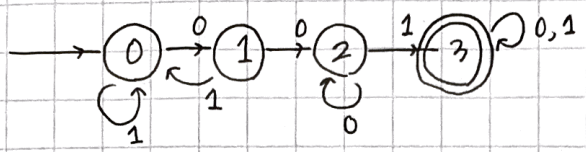


(a) $S = \{3\}$

(b) sequence of states (1 0 1 0 1 0 1) = 0, 1, 2, 2, 0, 1, 2, 2

(c) accepting input = 1 1

3. DFA that accepts strings containing "001."



Explanation: If we see any 1's in the beginning, we remain in starting state.

If we see a 0, then we progress to the next state. We have found 0 out of 001.

Here, if we prematurely find a 1, becoming 01, we go back to starting state to "reset."

If we find another 0, we now have 00, and we move to the next state.

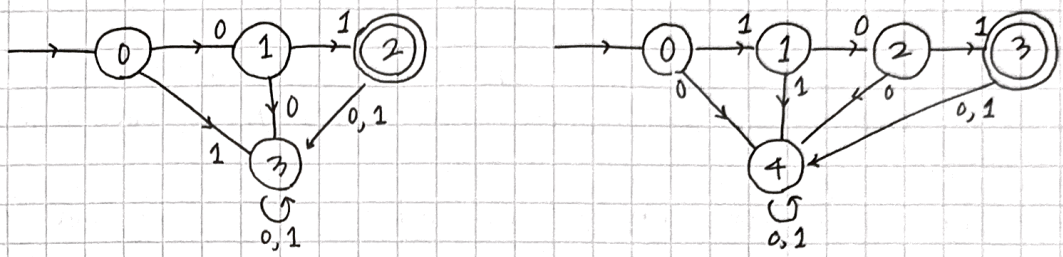
Here if we find another 0, making it 000, nothing changes, and we still only need a 1. We remain put.

If we find a 1, we have found 001. We move to accepting state. Anything from here is irrelevant.

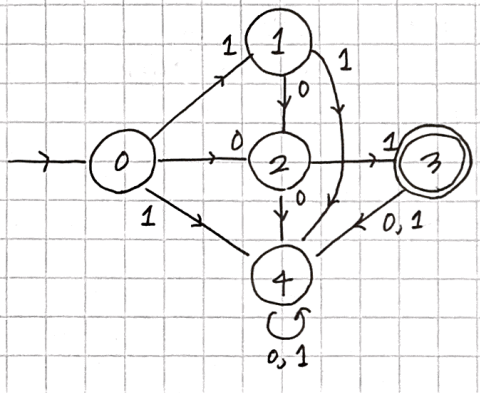
4. NFA that accepts $(\{01, 101\})^*$

DFA that only accepts $\{01, 101\}$?

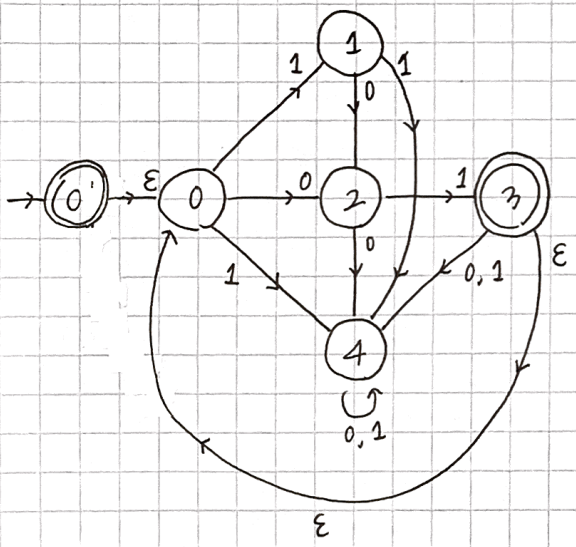
DFA that only accepts $\{01\}$ · + DFA that only accepts $\{101\}$



= DFA that only accepts $\{01, 101\}$



NFA that accepts $(\{01, 101\})^*$

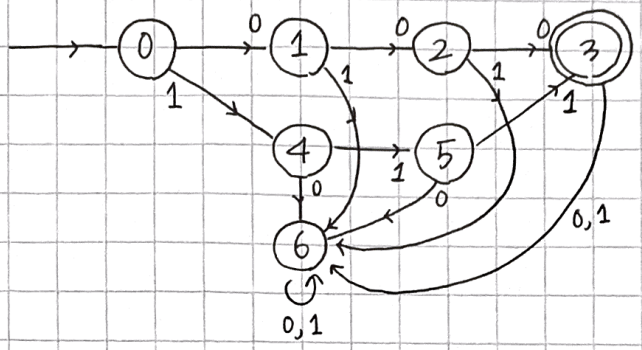


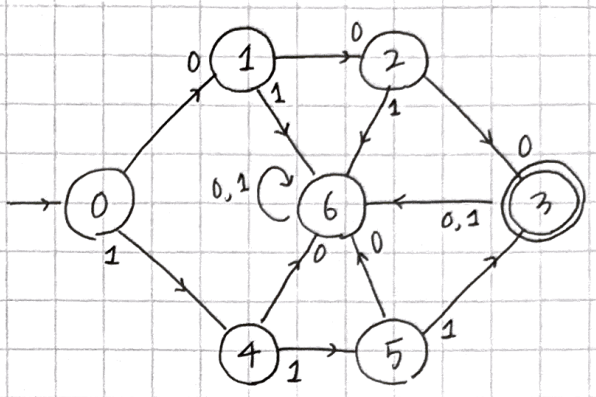
Explanation: First, find DFAs that only accept $\{01\}$ and only accepts $\{101\}$, separately.
Combine those two DFAs to find one that accepts $\{01, 101\}$.
Add a dummy starting state and "ε" edges to make it accept L^* instead of L (L being $\{01, 101\}$).

5. language L, $DROPFIRST(L) = \{x : bx \in L, \text{ for some } b \in \{0,1\}\}$
Show that if L is computable by DFA, $DROPFIRST(L)$ is computable by NFA.

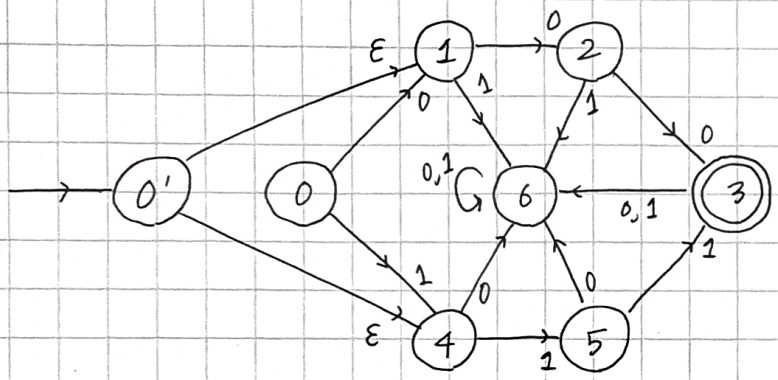
Let $L = \{000, 111\}$.

DFA for L:





$DROPFIRST(L) = \{00, 11\}$



Explanation: in general, adding a dummy start state, and adding ε edges from the dummy start state to all of the original start state's next states will allow the NFA to bypass the first symbol of a string in L, effectively computing $DROPFIRST(L)$.