Ethan Wong
305319001
10/19/21

# CS 181 HW2

pg 214 1. Prove there exists a number $\delta > 0$ such that for every $n, m$ there exists a function $f: \{0,1\}^n \to \{0,1\}^m$ that requires at least $\delta m \cdot \frac{2^n}{n}$ NAND gates to compute

$m \leq 2^n$

- Hint: How many functions from $\{0,1\}^n \to \{0,1\}^m$ exist?
  - This means there are $n$ inputs and $m$ outputs. Thus the possibilities for inputs is $2^n$, and there are $2^m$ possible outputs. Each input has $2^m$ possible output values, and a value must be picked for all $2^n$ inputs. Thus the number of functions is $2^{m2^n}$.

Now take Theorem 5.2 into account: The number of functions of size less than or equal to s is bounded by $|SIZE_{n,m}(s)| \leq 2^{Bs\log s}$ for a constant B.

Now suppose $s = \delta m \cdot \frac{2^n}{n}$ and $\delta = \min(1, \frac{1}{3}B)$
This would mean that $2^{m2^n} \leq 2^{Bs\log s}$
which simplifies to $m \cdot 2^n \leq Bs\log s$
Now substitute s into the rightside:

$$m \cdot 2^n \leq Bs\log s$$
$$m \cdot 2^n \leq B(\delta m \cdot \tfrac{2^n}{n}) \log(\delta m \cdot \tfrac{2^n}{n})$$
$$2^n \leq B(\delta \cdot \tfrac{2^n}{n}) \log(\delta m \cdot \tfrac{2^n}{n})$$
$$[\log(\delta m \cdot \tfrac{2^n}{n}) \to n \text{ as } n \to \infty]$$
$$2^n \leq B(\delta \cdot \tfrac{2^n}{n}) \cdot n$$
$$2^n \leq B\delta 2^n$$

The inequality seen here is not true for values from $0 \leq \delta < \frac{1}{B}$

There is a contradiction in these equalities, and this contradiction proves that there is a number $\delta > 0$ that satisfies the condition. Therefore, there does exist a function $f: \{0,1\}^n \to \{0,1\}^m$ that requires at least $\delta m \cdot \frac{2^n}{n}$ NAND gates to compute

Ethan Wong
10/19/21
305319001

CS 181 HW2

✓ 2. a) The set of accept states is {3}, as indicated by the double circle

b) Input: 1010101

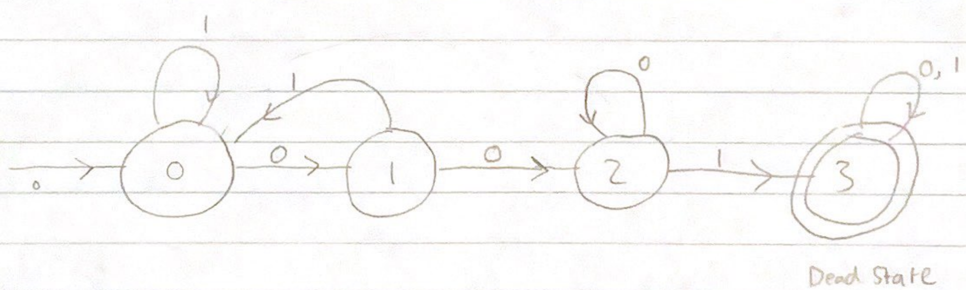
| start at | 0 |
|---|---|
| 1 | 1 |
| 0 | 2 |
| 1 | 2 |
| 0 | 0 |
| 1 | 1 |
| 0 | 2 |
| 1 | 2 |

The sequence of states is
(0, 1, 2, 2, 0, 1, 2, 2)
start state

c) An accepting input for the machine would simply be '11' since the sequence would end at state 3, the accept state.

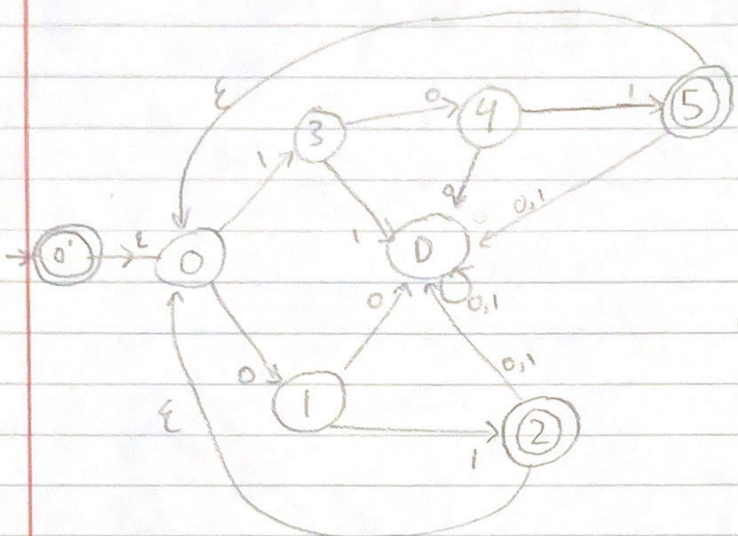
✓ 3.

001




This DFA is based on the hint. Starting at state 0, you want to keep looking through the bits. Ignore all the 1's, but when there's a 0 we take note and move to state 1. If we encounter a 1, we reset to where we started. If we see a 0, we take note that there have now been two consecutive 0's. and move to state 2. Next, if we see a 1 then we know we have found our '001' and can advance to the accept state (state 3) that acts as a dead state upon arrival. If we see a 1 then we can self-loop, as there are technically still 2 consecutive 0's.

4.




Note: 'D' is a dead state

The logic for this NFA is as such:

- If the string is '01' it follows the bottom path and reaches the accept state (2). If there is more to the string it follows the ε back to the start and continues from there. Otherwise it is just accepted.
- If the string is '101', it basically does the same thing except it follows the top path.

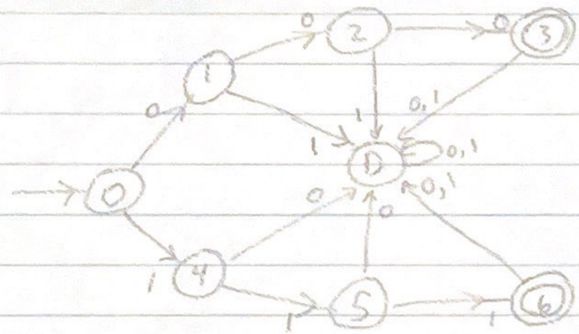
The ε edges lead from every accept state back to the start state so that repetition of strings can be checked for. If at any point there is a bit that doesn't match either '01' or '101', it is sent to the dead state and the string is never accepted.

Note: There is a dummy starting state with an ε edge so that the empty string is still accepted.

5. Language $L$, DROPFIRST($L$) is langoge containing all strings that can be obtained by removing first symbol of a string in $L$

DROPFIRST($L$) = $\{x : bx \in L$, for some $b \in \{0,1\}\}$

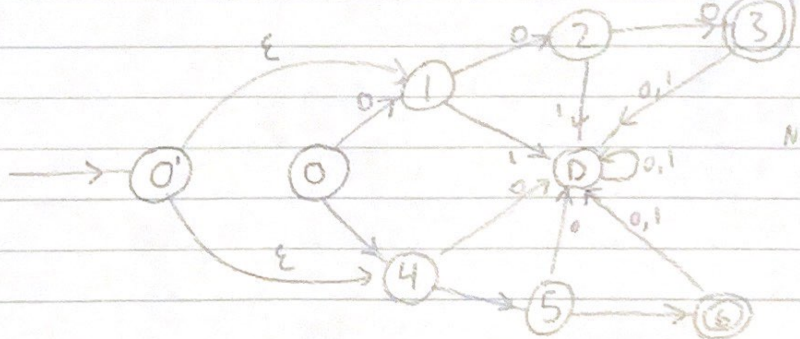
Say we have a language $L = \{111, 000\}$. A DFA for $L$ would be:



Note: D is a dead State

We can create an NFA for DROPFIRST($L$) like this:

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



Note: D is dead state

Basically, once you have a DFA for a language $L$ you can create an NFA by adding a dummy state that has $\varepsilon$ edges to the original start state's next states. This basically lets the DROPFIRST($L$) NFA skip over the first symbol in the string - which is what we wanted.

Note: I'm assuming the empty string won't be a part of $L$, because then there won't be any symbol to drop.