TOC PROBLEM SET-2

SIDDHANT CHAUDHARY
BMC201953

Problem 1. Prove that if a DFA A with n states accepts a word of length $m \geq n$, then its language will be infinite. Borrowing the idea from the previous proof, show that if $w \in L(A)$ where A is a DFA with n states and $|w| \geq n$, then w can be partitioned as $w = xyz$ with $y \neq \epsilon$ such that $xy^kz \in L(A)$ for all $k \in \mathbb{N}$. This is called the *pumping lemma* (Infact, we can partition w in such a way that $|xy| \leq n$)

Solution: Let $A = (Q, \Sigma, \delta, F, Q_0)$ be a DFA with $|Q| = n$ such that there is some word w with $|w| \geq n$ that is accepted by A. Suppose the word w is

$$w = a_1a_2...a_{|w|}$$

where $a_i \in \Sigma$ for each $1 \leq i \leq |w|$ and suppose the accepting run is

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} ... \xrightarrow{a_{|w|}} q_{|w|}$$

(where the q_is are not necessarily distinct). By the pigeon-hole principle, there are indices $0 \leq i < j \leq |w|$ such that $q_i = q_j$ (this is because $|w| \geq n$, and in this run we have $|w| + 1 > n$ states). Then, let the word w be written as

$$w = xyz$$

where x is the substring of w occuring in the run from q_0 to q_i, y is the substring of w occuring in the run from q_i to q_j (and $y \neq \epsilon$ because $i < j$), and similarly z is the substring occuring in the run from q_j to $q_{|w|}$. Then, it is easy to see that if $k \in \mathbb{N}$, xy^kz will always be accepted, because the run will end at state $q_{|w|}$ (which is a final state). This also shows that the language of A will be infinite, hence completing the proof.

Problem 2. Are DFAs with only one final state as *powerful* [1] as those with two final states? Prove or give a counterexample. What can you say about the *power* of the DFAs with k final states and those with $k+1$ final states? Justify. Does the same hold for an NFA?

Solution: Let $\Sigma = \{1\}$, i.e there is only one letter in the alphabet. Let $k \in \mathbb{N}$, and define

$$C_k := \{L \subset \Sigma^* | L \text{ is accepted by some DFA with } k \text{ final states}\}$$
$$C_{k+1} := \{L \subset \Sigma^* | L \text{ is accepted by some DFA with } k+1 \text{ final states}\}$$

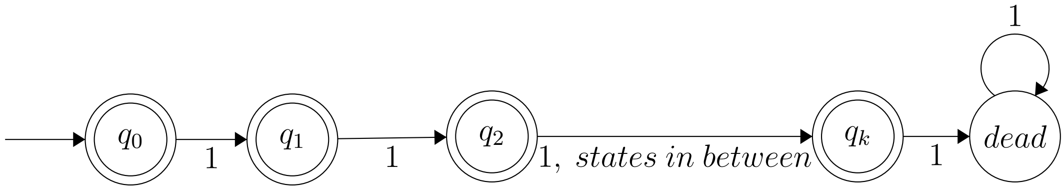
We will show that C_k is a *proper subset* of C_{k+1}, and this will show that DFAs with $k+1$ final states are more powerful than those with k final states. First, suppose $L \in C_k$, that is L is a language accepted by some DFA with k final states. To this DFA, add a garbage state, make it final, and let all transitions coming out

Date: August 2020.

[1] An automaton is said to be *more powerful* than another if the language accepted by the second is a *proper* subset of the language accepted by the first.

of this state be self loops. Then, we have found a DFA with $k+1$ final states that accepts L, so that $C_k \subset C_{k+1}$.

To show that the inclusion is strict, consider the following example. We design a DFA that accepts precisely those words that have length atmost k, i.e the words can have length only in $\{0, 1, ..., k\}$. Suppose q_0 is an initial state. Clearly, q_0 is a final state. Let $(q_0, 1, q)$ be a transition. Then, $q \neq q_0$, otherwise all words will be accepted, so put $q = q_1$. Also, observe that q_1 is a final state, because the word 1 must be accepted ($k \geq 1$). Continuing this process k times, we see that the only DFA that accepts this language is the following DFA:



(Between q_2 and q_k there are more final states). Clearly, we require atleast $k+1$ final states. Hence, the inclusion is proper. Also, note that this example works over *any* alphabet, since we are only interested in the *length* and not the letters of a word.

I think the same does not hold true in an NFA , strictly because of the fact that ϵ-transitions between final states can exist. I will try to come up with a proof of this.

Problem 3. Prove that swapping the final and non-final states in a DFA A gives us a DFA that recognizes the complement of $L(A)$.

A finite state automaton is said to be an *incomplete* DFA if for each state q there is *atmost* one transition on each $a \in \Sigma$. Prove (or disprove) that the above holds for an incomplete DFA.

Solution: Let A be a DFA, and let B be the DFA obtained by swapping the final and non-final states of A. We will show that $L(B) = L(A)^c$, where the complement is taken inside the set Σ^*. Here is the important observation. If w is any word in Σ^* and if

$$q_0 \xrightarrow{w} q_f$$

is the run of w in A, then this is also the run of w in B, and vice-versa. This is true because the initial state and the transitions are the same in A and B.

Now, suppose $w \in L(B)$, and let the the run be

$$q_0 \xrightarrow{w} q_f$$

where q_f is a final state in B. By what we showed above, this is the run of w in A is well, but in A, q_f is *not* a final state in A, and hence $w \notin L(A)$, implying $w \in L(A)^c$. This shows $L(B) \subset L(A)$. To show the reverse inclusion, suppose $w \in L(A)^c$, which means $w \notin L(A)$, and hence q_f is *not* a final state in A. The run of w in B will be the same as that in A, but in B, q_f will be a final state, and hence $w \in L(B)$. This shows $L(A)^c \subset L(B)$. So, $L(B) = L(A)^c$, completing the proof.

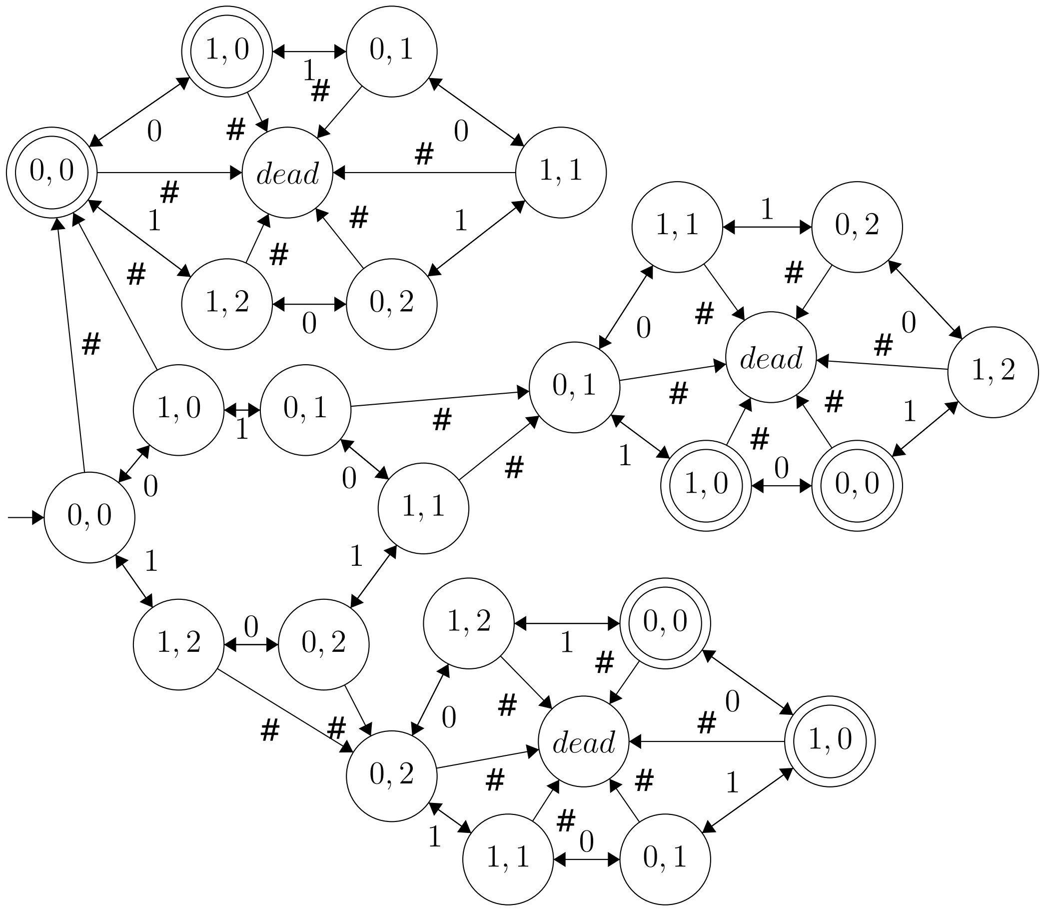
The same proof works for *incomplete* DFAs as well, because of the same observation: the run in both the incomplete DFAs will be the same, but the last state in one DFA would be final, while in the other DFA it won't be final. (This is wrong, try to figure out why.)

Problem 4. Is the language

$$L = \{x\#y|\ x, y \in \{0,1\}^*,\ (x)_2 + (y)_2 \text{ is divisible by } 3\}$$

recognizable? If yes, give a DFA/NFA, otherwise justify.

Solution: Yes, this language is recognizable, and we give a DFA for it. We assume that strings of the form x# and #y are acceptible if $3|(x)_2$ and $3|(y)_2$. In PSET-1, we designed a DFA to check whether a binary number string is divisible by 3. Here, the basic idea is as follows: we keep scanning until we hit a # (if we don't, we don't accept the string), and while scanning, we keep track of the first number modulo 3 (which is $(x)_2$). If a # is scanned, we start over again, and we keep track of the second number modulo 3 (which is $(y)_2$), starting from the residue of $(x)_2$ modulo 3. The final states are those ones which have a residue of 0 modulo 3. All dead states are not final, and have self-loops corresponding to every symbol in the alphabet.



Note: If strings of the form x# and #y are not accepted, we can fix the above automaton pretty easily.

Problem 5. The classical Sudoku is a 9×9 grid that has nine 3×3 sub-grids. The goal of the game is to fill the digits from 0 to 9 such that each of the row, column and the nine 3×3 sub-grids have all the digits from 0 to 9. A filled Sudoku that satisfies these constraints is said to be *correctly filled*.

Let S be a completely filled Sudoku (whether correctly filled or not), and $\texttt{flatten}(S)$ be the flattened version of the Sudoku obtained by concatenating all the nine rows of S one after another (preserving the order) to form a string over $\Sigma = \{1, 2, ..., 9\}$.

Is it possible to construct a DFA A which only accepts the words w over $\Sigma = \{1, 2, ..., 9\}$ that are flattening of some correctly filled 9×9 Sudokum i.e

$$L(A) = \{w \in \Sigma^* | \exists S \text{ such that } \texttt{flatten}(S) = w\}$$

Solution(Incomplete): For this problem, I have the following conjecture:

Regular languages are closed under intersection

If this conjecture is false, then this strategy won't work. If this is true, then we only need to design three FAs to: (1) check that no two numbers in a row are the same, (2) check that no two numbers in a column are the same and (3) check that no two numbers in the same 3×3 block are the same. And then, combine these three to a single one, which will be the required FA (we know that ϵ-NFAs are equivalent to DFAs, so it is enough to make an ϵ-NFA). I think designing DFAs for these three tasks individually is definitely possible. (Update: this is a finite language, and hence must be regular.)