COMS 3261 Spring 2019: HW 1

This homework is due by 11:59 pm on Wednesday, Feb. 13, 2019.

*Problems marked with a * are potentially harder in the sense that they are less analogous to proofs we have done in class. It is recommended that you start thinking of these early, and come to office hours for guidance or collaborate with others if you are stuck for too long.

Problem 1 [20 pts] Consider the DFA defined by the following transition table, where q_0 is the start state, and accepting states are marked with an *. Describe the language it accepts. You do not need to prove your answer, but you should informally explain the reasoning you took to arrive at your answer.

	0	1
q_0	q_3	q_1
q_1	q_1	q_2
$*q_2$	q_1	q_2
q_3	q_3	q_3

Problem 2 [20 pts] Design a DFA that accepts the language of numbers written in base 10 such that the sum of the digits is divisible by 4. Draw either a transition diagram or a table to specify your DFA. Make sure you indicate the start state and the accepting state(s). Also provide a human-interpretable meaning for each of your states. Prove that your DFA accepts all strings representing numbers base 10 whose digits sum to a value that is divisible by 4, and nothing else. It is recommended that you use an induction on the length of input strings, and your inductive hypothesis should reflect the intended meaning of your states.

Problem 3 [20 pts] Consider the following inductive "proof". Explain why it is incorrect. Then discuss whether you think the claim is true or false - this can be informal and heuristic.

Claim Any language L over the binary alphabet containing only strings of length exactly n (where n is a positive integer) corresponds to the accepted language of some DFA with only $n + 2$ total states.

Proof We will prove the claim by induction on n.

Base Case We start with $n = 1$. In this case, L must either contain a single string of length 1, e.g. $L = \{0\}$, or L contains both strings of length 1, i.e. $L = \{0, 1\}$.

If $L = \{0\}$, we can design a 3-state DFA A such that $L(A) = L$ as follows: the start state will be q_{start}, and we will have additional states q_{accept} and q_{reject}. The only accepting state is q_{accept}. The transition function δ is defined by the following table:

	0	1
q_0	q_{accept}	q_{reject}
q_{accept}	q_{reject}	q_{reject}
q_{reject}	q_{reject}	q_{reject}

Clearly the case of $L = \{1\}$ is analogous.
If $L = \{0, 1\}$, we can simply change the transition function to the following:

	0	1
q_0	q_{accept}	q_{accept}
q_{accept}	q_{reject}	q_{reject}
q_{reject}	q_{reject}	q_{reject}

Inductive Step We can assume that the claim is true for $n-1$ (where $n \geq 2$), and we wish to prove it now for n. We stregnthen our inductive hypothesis to further assume that we can design a $n+1$ state DFA for any language containing only strings of length $n-1$ such that it has a single accepting state, and also a state q_{reject} that is not accepting and such that all transitions from q_{reject} go back to q_{reject}.

Now consider a language L containing only strings of length precisely n. Each string $x \in L$ can thus be written as $x = ya$, where y is a binary string of length $n-1$, and $a \in \{0, 1\}$. Let A be a finite automata with $n+1$ states that accepts exactly the length $n-1$ strings y such that $x = ya \in L$ for some $a \in \{0, 1\}$. Let q denote the single accepting state in A. We'll make a new automata by taking A, declaring q to no longer being an accepting state, and adding a new state q_{accept} that is now our only accepting state. For each $a \in \{0, 1\}$, we will modify the transition function so that $\delta(q, a) = q_{accept}$ if $x = ya$ for some $x \in L$, and $\delta(q, a) = q_{reject}$ otherwise. All transitions out of q_{accept} go to q_{reject}. Now the resulting $n+2$ state automata will accept all of the strings in L, and only the strings in L.

Problem 4 [20 pts] Design an NFA that accepts the language of all binary strings that contain *either* 000 *or* 111 as a substring (this is not an exclusive or, so containing both should still be accepted). Prove that your NFA accepts precisely this language.

Problem 5* [20 pts] Consider the example in lecture 3 of a DFA that accepted the language of all binary numbers that are divisible by 3. It had 3 states. Do you think it is possible to design a DFA with only two states that accepts precisely this language? If so, demonstrate it with a two-state DFA and a proof that the accepted language is precisely binary strings representing numbers divisible by 3. Otherwise, prove that such a two-state DFA is impossible.