# CS 181 - Homework 10

Kuruvilla Cherian, Jahan
UID: 104436427
Section 1B

December 6, 2017

## Problem 3.20

The idea here is that we can create a *DFA* that functions exactly like a read-only section of a single tape TM, which in turn signifies its ability to only recognize regular languages. The key is to recognize the string input read so far by the TM. We essentially assign the DFA a set of finite control functions (of which there are a finite number) that map the set of the TM's states (read so far) and the *first* state to the set of all states read so far and either *accept* or *reject*. What this means is the following:

1. The function $F_s$ on first is the state the TM enters when it is about to move off the right of the string $s$ for the first time.
2. If the TM accepts or rejects before ever moving off the right end of $s$ then $F_s(first)$ is accept or reject accordingly.
3. To model a transition from the string read so far to the next symbol in the alphabet we just utilize the TM's transition function as normal noting that on an $\varepsilon$ transition on *first* we remain on the start state for the TM, and on any other state, we just remain at the given state.
4. If for any two strings, the finite control function is the same, then we know the TM either accepts or rejects both strings. Thus the accept and reject states for the DFA are the equivalent to the states at which the TM accepts or rejects before moving off the right end.

## Problem 4.12

We can construct a DFA $A$ to recognize the language of odd number of 1's. By closure we can perform intersection on the languages recognized by $A$ and $M$, $L(A) \cap L(M)$ which itself will be a DFA, call it $B$. The TM accepts if $L(B) = \emptyset$ else it rejects. To test this, we run the TM on $B$, trying to see if a path exists from start to an accept state. We do this by marking states visited on $B$ on the TM, noticing that if no accept state is marked, then we have shown that the language is the empty set thereby accepting, else rejecting.

## Problem 4.14

We know that the intersection of a regular language and a CFG produces a CFG. Thus let us create the CFG $N = 1^* \cap L(G)$. We then just use run a decider TM on this CFG to see if it accepts any string at all. This can be done by trying out every candidate parse tree in $N$ of depth $\leq |V|$, where $V$ is the set of all non-terminals in the grammar. If a valid parse tree is found then we accept, else we reject.

## Problem 4.17

$L(A) = L(B)$ for DFA's $A$ and $B$ respectively, if and only if both DFA's accept the same strings up to length $ab$ where $a$ is the length of $A$ and $b$ is the length $B$. If the two languages are the same then we are done, but if they are not the same then we need to show that the languages differ on some string of length at most $ab$. Let us consider a string $s$ that is the shortest string on which the languages differ. If the length of this string is less than $ab$ then we are done. If not then let us consider the set of states on both $A$ and $B$ that accept the string $s$. Because of the length of both these DFA's we can confidently say that there exist only $ab$ pairwise distinct states for the acceptance of these states. We notice that because of this we can state that pairs of states must be identical because of the limit of how many pairs we can have. By this logic we can remove a certain section of the string $s$ so as to make it shorter than it was before, thereby implying that the string can be of no length greater than $ab$.

## Problem 4.27

We can construct a *PDA* $N$ to recognize the language of more 1's than 0's by maintaining some kind of counting state on the stack. Thereby construct a new PDA $P = L(M) \cap L(N)$, and test if the CFG recognized by this PDA recognizes any string. Essentially we run a decider TM to see if the *L(P)* is empty or not (by the same algorithm stated in the problem above). If we find that the language is empty then we reject else accept.