CS5110: Complexity Theory

Shreyas Havaldar
CS18BTECH11042

September 30, 2020

1 Exercise Set 9

1.1 Given an NTM with multiple accepting states, to show that there is an equivalent NTM with a unique accepting state.

Let the original NTM be M. $M = (Q, \Sigma, \Gamma, \iota, A, R, \delta)$, where

- Q is a finite set of states
- Σ is a finite set of symbols, input alphabet not containing the blank symbol $\sqcup$
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\iota \in Q$ is the start state
- $A \subseteq Q$ is the set of accepting (final) states
- $R \subseteq Q$ is the set of reject states ($R \cap A = \phi$)
- $\delta : (Q \times \Gamma) \longrightarrow P(Q \times \Gamma \times \{L, S, R\})$ is a relation on states and symbols called the transition relation. L is the movement to the left, S is no movement, and R is the movement to the right. Note: Movement refers to the movement of the tape head. (At any point in a computation, the machine may proceed according to several possibilities along different branches.)

The computation of a non-deterministic turing machine is a tree whose branches correspond to different possibilities for the machine. If some branch of the computation leads to the accept state, the machine accepts its input.

We shall construct an equivalent NTM N. $N = (Q \cup T, \Sigma, \Gamma, \iota, T, R, \delta)$, where

- Q is a finite set of states, and we add our new states T and D to it. $Q \cup T \cup D = Q'$ Remember A is the set of accepting states from our original NTM M.
- Σ is a finite set of symbols, input alphabet not containing the blank symbol $\sqcup$
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$

- $\iota \in Q$ is the start state
- T is the single accepting state
- $R' = (R \subseteq Q) \cup D$ is the set of reject states ($R' \cap A = \phi$)
- $\delta' : \delta \cup (A \times \epsilon) \longrightarrow (T \times \Gamma \times S) \cup (T \times (\Gamma \backslash \sqcup)) \longrightarrow (D \times \Gamma \times S)$. Thus to the transition function in the original NTM M, we add a transition function such that whenever the NTM N is in a state $\in A$, without moving ahead on the input and irrespective of the symbol on the tape position the head is at, without replacing the symbol on the tape, the turing machine moves to state T, while the head stays on the same position the tape. Note: If there is a character to be read on the input tape even after some branch of the NTM N reaches T, that branch simply dies. That is, we define a transition from T to D, where D is a reject state with a self loop. Thus only if the input has a blank symbol, does the NTM stay in T. Therefore no false positives originate.

Thus by first copying the entire configuration of M into N and creating a new state T in N we have created a new NTM N. There is no outgoing transition from T to any other state. For every state a in the NTM such that a is an accepting state, we create an ϵ transition from a to T. Thus, $\forall V, V \in Q$, $\forall S, S \in A$ such that $\exists V - S$ path, $\exists A - T$ path in the tree of non-deterministic branches. Thus for when we reach any accepting state A in the NTM B, a copy of the NTM is created where we are in the state T. For N, all states s $\in$ A are no more accepting states. Thus we have created a single accepting state for NTM N, namely T, and N and M are equivalent as N would accept the input iff M accepts the input. (If and only if M is in state s $\in A$, N is in state T) It is easy to see that N cannot be in state T unless M is in state s $\in A$ at the end of the input, as there exists no transition in N, from $s' \longrightarrow T$, where $s' \notin A$.

Therefore N accepts iff for some branch, N is in state T and likewise M accepts iff for some branch M is in one of the sets in A. Both are equivalent. Thus we have constructed an NTM with an unique accepting state from a NTM with multiple accepting states.

1.2 To show that A_{NFA} is NL complete

$A_{NFA} = \{\langle N, w \rangle |$ N is an NFA that accepts string w$\}$.

So we need to show two things. $A_{NFA} \in NL$ and A_{NFA} is NL-hard. For the former we just need to that the space occupied by a non-deterministic turing machine for deciding A_{NFA} is $O(\log n)$. Let the turing machine deciding A_{NFA} be TM. TM has a work tape. We use this work tape to save only the current state of the NFA N running on input w. We can non-deterministically guess the next state on the input w and proceed saving only the state information in our work tape and we would need atmost $\log n$ bits to store a value from 1 to n using binary notation, where n is the number of states in the NFA N. Thus as the input must atleast list out all the states of the NFA N in the input, thus the input size say m, is $\Omega(n)$. Our TM will return true if for any branch of non-determinism, our state on the work tape belongs to set of accepting states in the NFA N when we reach the end of the input w. Therefore our work tape requires atmost $\log n$ space and therefore our non-deterministic TM

occupies only $\log n$ space, which is necessarily $\log m$. Thus we have shown A_{NFA} can be decided by a non-deterministic TM in $\log m$ space. Thus, $A_{NFA} \in NL$.

Now for proving A_{NFA} is NL-hard. We know that PATH is NL-Complete where PATH = $\{\langle G, s, t\rangle |$ G is a directed graph and has an s-t directed path$\}$. Which implies, PATH is NL-Hard, therefore $\forall L \in NL, L \leq_L PATH$. If $PATH \leq_L A_{NFA}$, we can conclude that A_{NFA} is NL-Hard as we can reduce every language L $\in NL$ to PATH first and then can use another reduction to reduce PATH $\in$ NL to A_{NFA} sequentially in total $O\log$ space. Thus, we just need to show a reduction from PATH to A_{NFA}. To show $PATH \leq_L A_{NFA}$:

- Let the source vertex s in G be the start state S of the NFA N.
- Let the target vertex t in G be the accepting state T of the NFA N. Note: Our accepting state T has a self loop.
- For every edge (u,u') in G, there exists a transition from state U to state U' on reading 1 from the input tape in the NFA.

Let the input w be 1^{n-1}, that is the input tape to N would consist of n-1 ones, where n is the number of vertices in G.

Thus, this construction ensures that if and only if there is a path from s-t, we have a sequence of transitions in the NFA that causes an accepting configuration T at some point of reading the input tape, and due to self loop the NTM TM stays in T till the end of the input. This reduction can be implemented in logarithmic space as we only need to represent the current state in the NTM TM which can be done in $O(\log n)$ space in binary notation where n is the number of vertices in G and equivalently the number of states in N. The reduction will not take more than $\log m$ space as the construction of the NFA N, would just require us move one vertex of the graph G at a time in each branch, and we would only store information corresponding to the state in N representing the vertex. Note: Our output tape would require polynomial space in input to save the configuration of the entire NFA N, but only the space taken by the work space is considered for the reduction and thus we are using $O(\log m)$ space. The reduction is also deterministic as we are proceeding in a sequential manner, in the order of vertex identities from 1 to n. There is no non-determinism involved at any point of the reduction.

An shortest s-t path cannot be of more than n-1 length. Thus if there is a shortest path s-t, it must be of length less than n, and the NTM TM would necessarily reach the state T in less than n transitions. Thus we defined w of length n-1. On reaching T, NTM TM stays there so at the end of the input, that branch of N stays in T, because we are considering all shortest s-t paths. thus if the length of an s-t path is smaller than n-1, we still accept.

Thus, $N = (Q, \Sigma, \Gamma, \iota, A, R, \delta)$, where

- Q is a finite set of states corresponding to vertices in G
- $\Sigma = \{1\}$, input alphabet

- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $S \in Q$ is the start state
- $T \in Q$ is the accepting state
- $R \subseteq Q$ is the set of reject states ($R \cap A = \phi$) which here is all states other than T. ($Q \backslash T$)
- $\delta : (Q \times \Gamma) \longrightarrow P(Q \times \Gamma \times \{L, S, R\})$ is a relation on states and symbols called the transition relation. L is the movement to the left, S is no movement, and R is the movement to the right. (At any point in a computation, the machine may proceed according to several possibilities along different branches.)

Thus if any branch of our NTM TM is in state T at the end of the input, we can say that there exists a path from s-t in the graph G. If there is no s-t path, there are no possible sequence of transitions that on reading w, can take the NFA N from state S to state T. Thus we have successfully reduced PATH to A_{NFA} in log space. Thus, A_{NFA} is NL-Hard.

As $A_{NFA} \in NL$ and A_{NFA} is NL-Hard, A_{NFA} is NL-Complete.

Hence Proved!