16:198:538 Computational Complexity September 24, 2013

Lecture 3

Lecturer: Shubhangi Saraf *Scribe: Timothy Shields and Daniela Vianna*

1 Nondeterministic Turing Machines

Recall that in the previous lecture we introduced the Time Hierarchy theorem for (deterministic) Turing machines and proved a relaxed version of it. In this lecture we will begin by extending these concepts to their nondeterministic analogues.

Definition 1 *A nondeterministic Turing machine (NDTM) M is defined the same as a deterministic Turing machine (TM) except for the following differences:*

1. *M has two transition functions, δ_0 and δ_1*
2. *At each execution step, M must choose which transition function to apply*
3. *For an input x, $M(x) = 1$ if and only if there is some sequence of choices that results in an output of 1*

M is said to run in time $t(x)$ if for all sequences of choices M halts in $t(n)$ steps on all inputs of length n.

We have previously defined the complexity class NP to be the set of decision problems whose solutions can be verified by deterministic TMs in polynomial time. Curiously, this definition for a "nondeterministic" complexity class did not make use of anything nondeterministic. We will now see that NP has an equivalent definition in terms of NDTMs.

Definition 2 *The complexity class NTIME($T(n)$) is the set of languages L for which there is some NDTM and constant $c > 0$ such that M decides L in time $c \cdot T(n)$ on all inputs.*

Definition 3 *An alternative but equivalent definition for NP is*

$$NP = \bigcup_{k \in \mathbb{N}} NTIME(n^k)$$

We will now examine the nondeterministic analogue of the Time Hierarchy theorem for deterministic TMs.

2 Nondeterministic Time Hierarchy Theorem

Let $f, g : \mathbb{N} \to \mathbb{N}$ be time-constructible functions such that $f(n+1) = o(g(n))$. Then

$$\text{NTIME}(f(n)) \subset \text{NTIME}(g(n))$$

is a strict subset relation.

In the interest of facilitating a simple proof, consider the relaxed statement $NTIME(n) \subset NTIME(n^{10})$

Proof Since the non-strict subset relation is trivially true, the goal is to find a decision problem L such that $L \in NTIME(n^{10})$ but not $L \in NTIME(n)$. Let the sequence $\{M_i\}_{i \in \mathbb{N}}$ be an enumeration of all NDTMs such that each appears infinitely many times. Define $z : \mathbb{N} \to \mathbb{N}$ as

$$z(i) = \begin{cases} 1 & i = 1 \\ 2^{[z(i-1)]^2} & i > 1 \end{cases}$$

For each $i \in \mathbb{N}$, consider all strings of the form 1^n, where $z(i) < n \leq z(i+1)$. Define L as follows.

1. If $x \neq 1^n$ for any n, then $x \notin L$
2. If $x = 1^n$ for some n, then there is $i \in \mathbb{N}$ such that $z(i) < n \leq z(i+1)$
   (a) If $n < z(i+1)$, then $x \in L$ if and only if $M_i(1^{n+1})$ outputs 1
   (b) If $n = z(i+1)$, then $x \in L$ if and only if $M_i(1^{z(i)+1})$ halts in $(z(i)+1)^2$ steps and outputs 0

Observe that $L \in NTIME(n^{10})$ (with plenty of slack) because we can decide whether a string x of length n is in L as follows.

> Compute the i such that $z(i) < n \leq z(i+1)$
>
> If $n < z(i+1)$:
>
> > Use the universal NDTM to simulate $M_i(1^{n+1})$
>
> Else:
>
> > Use the universal TM to simulate $M_i(1^{z(i)+1})$ for all possible sequences of $(z(i)+1)^2$ choices at the execution steps of NDTM M_i

Now suppose that $L \in NTIME(n)$. Then L is decided by some NTDM M_i (with large enough i) such that on inputs of length n, M_i halts in at most n^2 steps. Now because M_i decides L, we have

$$M_i(1^k) = 1 \Leftrightarrow 1^k \in L, z(i) < k \leq z(i+1),$$

and because of 2b we have

$$1^k \in L \Leftrightarrow M_i(1^{k+1}) = 1, z(i) < k < z(i+1)$$

By chaining these equivalence relations together we conclude that $M_i(1^{z(i)+1}) \Leftrightarrow 1^{z(i+1)} \in L$. But this directly contradicts 2b. Therefore $L \notin NTIME(n)$ ■

Notice that the proof of the Nondeterministic Time Hierarchy theorem was significantly more involved than the proof for the deterministic Time Hierarchy theorem from before. This is because the simple diagonalization approach used before does not apply as directly in the case of NDTMs. The salient difficulty here was that a NDTM accepts if any sequences of choices results in 1, but an attempt at using the diagonalization argument would require a NDTM that accepts if all sequences of choices result in 1.

3 Ladner's Theorem

Switching topics, we now examine a theorem about the relationship between P and NP. The proof is not completely rigorous but provides an outline for reasoning. For the interested reader, Arora & Barak chapter 3 section 4 provides a much more rigorous proof.

Theorem 4 *(Ladner, 1975)*
If $P \neq NP$ then there exists $L \in NP \backslash P$ such that L is not NP-complete

Proof Before proceeding with the proof of the theorem, we need the following two lemmas concerning the language $SAT_H = \{\psi \circ 1^{H(|\psi|)} | \psi \in SAT\}$ where $H : \mathbb{N} \to \mathbb{N}$.

Lemma 5 *If H is polynomial-time-constructible, then $SAT_H \in NP$*

Proof Let x be an input to SAT_H. Binary search for n such that $\psi \circ 1^{H(n)}$ aligns with x, taking polynomial time. Extracting ψ, we now have a witness to a standard SAT instance, which can be verified in polynomial time since $SAT \in NP$. ■

Lemma 6 *If $P \neq NP$ and $H = n^{\omega(1)}$, then SAT_H is not NP-complete.*

Proof Suppose to the contrary that SAT_H is NP-complete. Then $SAT \leq_p SAT_H$. But since SAT_H is just SAT with an input padded with superpolynomially many 1s, this is effectively means there is a polynomial time reduction for reducing input ψ for SAT with $|\psi| = n$ to equivalent ψ' for SAT with $|\psi| = o(n)$. Repeating this reduction yields a polynomial time algorithm for SAT, a contradiction to $P \neq NP$ and SAT being NP-complete. ■

We now resume the proof of Ladner's theorem. Let us assume that $P \neq NP$ and that there is a polynomial-time-constructible function $t : \mathbb{N} \to \mathbb{N}$ such that $t = n^{\omega(1)}$ and SAT needs $t^{\omega(1)}$ time. Consider SAT_t. If $SAT_t \in P$, then SAT needs only time $t^{O(1)}$, because SAT_t is just SAT but with an input padded with 1's. But this is a contradiction to SAT needing $t^{\omega(1)}$ time, so $SAT_t \notin P$. By Lemmas 5 and 6, we have $SAT_t \in NP$ and SAT_t is not NP-complete.
■

4 Relativization

Can we use diagonalization to prove $P \neq NP$?

Before we answer the question of whether diagonalization can be used to answer the P vs NP problem, lets informally define diagonalization as any proof which does not use much more than the ability to simulate one Turing Machine (TM) using other. We also define a variant of Turing Machine called Oracle Turing Machine (OTM):

Definition 7 *(Oracle Turing Machine) An Oracle Turing Machine (OTM) is a TM that are given access to an "oracle" that can solve the decision problem for some language $O \subseteq \{0,1\}^*$. The OTM has an special tape called "oracle tape"; on it, the TM can write some query $q \in \{0,1\}^*$ and get an answer to the question "is $q \in O$?" in a single time step. It can also be repeated arbitrarily often.*

Definition 8 *Given an oracle O, we can define the complexity classes P^O and NP^O. P^O is a set of languages decided by a polynomial-time deterministic Turing Machine with oracle access O. NP^O is a set of languages decided by a polynomial-time non-deterministic Turing Machine with oracle access O.*

The following Claim illustrates the definition of Oracle Turing Machines:

Claim 9 *$\overline{SAT} \in P^{SAT}$ where $\overline{SAT}$ denotes the language of unsatisfiable formulae*

Proof Given oracle access to SAT, a deterministic polynomial-time OTM can query its oracle whether a formula $\varphi \in SAT$ and then give the opposite answer. ■

We learnt that diagonalization is a technique that relies on the fact that Turing Machines can be represented by strings and their ability to simulate any other TM. As a normal TM, an Oracle Turing Machine can also be represented as strings and we can affirm that exist an Universal Turing Machine M (that simulates other TMs) with oracle access to O.

Observation 10 *Proof only using diagonalization relativize in the sense that they hold also for TMs with oracle access to O, for every oracle $O \subseteq \{0,1\}^*$ (**"Diagonalization proof relativize"**). For example, The proof of the time-hierarchy theorem relativizes.*

The following theorem shows the limitation of diagonalization in proving the P vs NP question.

Theorem 11 *(Backer, Gill, Solovay '75)*
There exist oracles A, B such that $P^A = NP^A$ and $P^B \neq NP^B$

Proof

1. **There exist an oracle A such that $P^A = NP^A$**

   Let A be an EXP-complete language.

   Recall that
   $$EXP = \bigcup_{c \geq 0} DTIME(2^{n^c})$$

   L is EXP-complete if $L \in EXP$ and $\forall L' \in EXP, L' \leq_p L$.

   An EXP-COM set is defined as $\{< M, x, 1^n > \mid M \text{ outputs 1 on input } x \text{ in at most } 2^n \text{ steps}\}$.

   $$EXP \subseteq P^{EXP-COM} \subseteq NP^{EXP-COM} \subseteq EXP \Longrightarrow P^{EXP-COM} = NP^{EXP-COM}$$

2. **Find B such that $P^B \neq NP^B$**

   For any language B, let
   $$\bigcup\nolimits_B = \{1^n \mid \exists \text{ some } x \in B, x \in \{0,1\}^n\}$$

   Claim 12

   $$\forall B, \bigcup\nolimits_B \in NP^B$$

   Now, we want to find B such that $\bigcup_B \notin P^B$.

   Let M_i be the oracle TM given by the binary representation of i. Enumerate all TMs with oracle access to B (each TM appears infinitely often) and force them to halt in time $\frac{2^n}{10}$ on inputs of length n. We will construct B in stages, where stage i guarantees that M_i does not decide $\bigcup_B$ in time $\frac{2^n}{10}$. Initially, B is the empty language and in each stage we will add strings to B.

   Stage i: Choose n large enough, larger than the length of all strings in B that have been decided, and run M_i on (1^n) for $\frac{2^n}{10}$ steps. Want to choose B such that $M_i(1^n)$ messes up:

   Run $M_i(1^n)$ for $\frac{2^n}{10}$ steps
   - all queries decided in the past (earlier stage), answer consistently with the earlier decision
   - all other queries, answer NO/ZERO

   Now, if $M_i(1^n)$ outputs 1, choose B such that are no strings of length n. If $M_i(1^n)$ outputs 0, let $x \in \{0,1\}^n$ not queried by M_i and add it to B.

   In both cases, ${M_i}^B$ could not decide $\bigcup_B$.

■

5 Space Complexity

Definition 13 *($SPACE(s(n))$ Let $S : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$. We say that L is in $SPACE(s(n)$ if there is a constant c and a Turing Machine M deciding L such that at most $c \cdot s(n)$ locations on M's work tapes (separated from input tape) ever visited by M's head on every input of length n.*

Definition 14 *($NSPACE(s(n))$ Similarly, we say that $L \in NSPACE(s(n))$ if there is a NDTM M deciding L that never uses more than $c \cdot s(n)$ nonblank tape locations on length n inputs, regardless of its nondeterministic choices.*

$$DTIME(S(n)) \subseteq SPACE(S(n)) \subseteq NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$$