CS 6810: Theory of Computing **Fall 2021**

Lecture 4: Sep 7, 2021

Lecturer: Eshan Chattopadhyay *Scribe: Adarsh Srinivasan*

1 Introduction

In this lecture, we prove Ladner's theorem using the technique of diagonalisation. Previously, we had defined the complexity classes **P** and **NP**. The class **P** is a subset of **NP**. It is widely believed, but not been proved that $\mathbf{P} \neq \mathbf{NP}$. We have also defined the class of **NP**-complete problems which are at least as hard as any other problem in **NP**. Ladner's theorem proves the existance of **NP**-intermediate problems. That is, assuming that $\mathbf{P} \neq \mathbf{NP}$, there exist problems that are not **NP**-complete but also not in **P**.

2 Ladner's theorem

Theorem 2.1 (Ladner's theorem)**.** *Assuming that* $\mathbf{P} \neq \mathbf{NP}$*, there exists a language* $L \in \mathbf{NP} \setminus \mathbf{P}$ *that is not* **NP***-complete.*

We prove this theorem by explicitly constructing a language that is not **NP**-complete but not in **P**, assuming $\mathbf{P} \neq \mathbf{NP}$.

Recall the language SAT, consisting of all satisfiable boolean formula. Consider a time-constructable function $H(n)$. We define the language $\mathrm{SAT_H}$, consisting of all satisfiable boolean formula 'padded' by $n^{H(n)}$ 1's.

Definition 2.2.

$$\mathrm{SAT_H} = \left\{ \psi 0 1^{n^{H(n)}} : \psi \in \mathrm{SAT}, n = |\psi| \right\}$$

Observe that when H grows fast enough, $\mathrm{SAT_H} \in \mathbf{P}$. For example, if $H(n) = n$, the input size is (2^n). Hence, we can check whether an input is in $\mathrm{SAT_H}$ by simply evaluating the boolean expression on all possible combinations of truth values of the literals, which would take at most 2^n time which is polynomial in input size.

On the other hand, consider the case when $H(n)$ is bounded above by some constant c for every n. In that case, $\mathrm{SAT_H}$ is **NP**-complete. In our proof of the theorem, we choose a function H that lies in between these two extremes. It tends to infinity with n, so that $\mathrm{SAT_H}$ is not **NP**-complete but it also grows slowly enough to ensure that $\mathrm{SAT_H} \notin P$. We define the function H recursively as follows. We assume the existence of a natural labelling scheme from the integers to the set of all descriptions of Turing machines using the universal Turing machine described earlier, and denote by M_i the TM labelled by the integer i.

Definition 2.3. *$H(n)$ is the smallest natural number $i \leq \log \log n$ such that for every $x \in \{0,1\}^*$, with $|x| \leq \log n$, the Turing machine M_i with input x halts within $i|x|^i$ steps and outputs 1 if and only if $x \in \mathrm{SAT_H}$. If no such Turing machine exists, $H(n)$ is defined to be $\log \log n$.*

As the definition of H depends on H itself, we need to check that it is not cyclic. The definition of $H(n)$ only involves checking the output of Turing machines on strings of length at most length $\log n$, and hence the value of H on n only depends on values of H on integers less than $\log n$. Remember that the goal is to show that $\mathrm{SAT_H} \notin \mathbf{P}$. We now show that this is equivalent to showing that $H(n)$ is bounded.

Lemma 2.4. $\mathrm{SAT_H} \in \mathbf{P}$ *if and only if* $H(n) < c$ *for all natural numbers* n*, where* c *is an absolute constant*

Proof. If $\mathrm{SAT_H} \in \mathbf{P}$, then there exists a Turing machine M which solves $\mathrm{SAT_H}$ in at most cn^c steps for some constant c. Now, as there are infinite representations for every Turing machine, there exists a natural number $k > c$ such that $M = M_k$. This machine M_k halts within kn^k steps. Hence, by the definition of $H(n)$, if $n \geq 2^{2^k}$, $H(n) \leq k$.

We now prove the converse. As $H(n)$ is upper bounded by a constant c, this means that there exists $1 \leq i \leq c$ such that $H(n) = i$ for infinitely many i. We claim that the Turing machine M_i solves $\mathrm{SAT_H}$ in a polynomial number of steps.

To prove that, we assume that there exists $x \in \{0,1\}^*$ on which M_i does not halt after $i|x|^i$ steps. Consider any $n > 2^{|x|}$. If $H(n) = i$, this means that M_i must halt on input x in $i|x|^i$ steps. Hence, this implies that for all $n \geq 2^{|x|}$, $H(n) \neq i$ which is a contradiction to the fact that $H(n) = i$ infinitely often. □

A natural corollary of this theorem is that if $\mathrm{SAT_H}$ is not in $\mathbf{P}$, then it must be an unbounded function.

Corollary 2.5. *If* $\mathrm{SAT_H} \notin \mathbf{P}$*, this means that* $\lim_{n\to\infty} H(n) = \infty$

We now have all the necessary tools to prove Ladner's theorem. It is clear that $\mathrm{SAT_H} \in \mathbf{NP}$. The certificate would simply consist of a satisfying assignment to the formula, and the verifier can evaluate the formula on the given assignment and then verify if the padding consists of the correct amount of 1's in time polynomial in input size.

We now show that $\mathrm{SAT_H} \notin \mathbf{P}$. By Lemma 2.4, $\mathrm{SAT_H} \in \mathbf{P}$ implies that $H(n) \leq c$ for all n. This means that any $x \in \mathrm{SAT_H}$ consists of a satisfiable formula ψ followed by at most $|\psi|^c$ 1's. We can reduce a SAT instance to an instance of $\mathrm{SAT_H}$ by simply padding $H(n)$ 1's, and because $H(n) \leq c$, the blowup of the input size is at most polynomial. As $\mathrm{SAT} \leq_P \mathrm{SAT_H}$, this implies that there exists a Turing machine that solves SAT in polynomial time, which is a contradiction to our assumption that $\mathbf{P} \neq \mathbf{NP}$.

We now show that $\mathrm{SAT_H}$ is not $\mathbf{NP}$-complete. If $\mathrm{SAT_H}$ is $\mathbf{NP}$-complete, then there must exist a polynomial time reduction from SAT to $\mathrm{SAT_H}$. that maps a boolean formula ψ of size n to a $\mathrm{SAT_H}$ instance of size n^c for a constant c. This instance must be of the form $\phi 01^{|\phi|^{H(|\phi|)}}$ and $n^c = |\phi| + |\phi|^{H(|\phi|)}$. As $\mathrm{SAT_H} \notin \mathbf{P}$, $\lim_{n\to\infty} H(n) = \infty$ by Corollary 2.5. Hence, the padding is superpolynomial in size of ϕ. This implies that $\lim_{n\to\infty} |\phi|/n = 0$, as otherwise, the right hand side of this equation would grow much faster than the left hand side. Hence, we have in effect reduced the problem of solving the SAT instance ψ to solving the SAT instance $|\phi|$ where $\frac{|\phi|}{|\psi|} = o(|\psi|)$. Hence, by applying this reduction at most polynomially many times, we can obtain a SAT instance of constant size, which can be solved in constant time. Hence, this implies that $\mathrm{SAT} \in \mathbf{P}$ which is a contradiction.