CS 4510 Automata and Complexity 4/8/2024

Lecture 21: Relativization

Lecturer: Abrahim Ladha *Scribe(s): Samina Shiraj Mulani*

1 Introduction

We give strong evidence that solving $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ is a very hard problem. We do not know how to solve the problem, but to a deep extent, we know how *not* to solve it. We know that there can be no easy proof of the question.

2 Oracles

The oracle of Delphi was like a witch or a shaman. You would bring her gifts and she would answer your questions. There would be no provided explanation. She is an antiquated version of a magic eight ball. An oracle machine has some similar mysticism.

Definition 2.1 (Oracle Machine)**.** An oracle machine is a fictional Turing Machine with an additional tape. It writes down a query on this special tape and issues an instruction. Then the tape clears and all that is left is the answer, a 1 or 0. This occurs in unit time. We formalize an oracle as a language A, and the oracle machine M^A.

This oracle machine M^A can test membership to language A in unit time. Many natural things we have discussed appear to be representable in an oracle way. Nondeterminism was originally formulated like an oracle. Both many-one and polytime reductions could be formalized in an oracle fashion. The study of oracle machines was intended to study how easy or hard certain problems can become relative to others. If you get instantaneous access to a specfic language, what can you do? What power do you gain?

Definition 2.2 (Relativized Complexity Class)**.** For a class C and language A, we let C^A be the languages decidable by C-machines with oracle access to A.

For example, consider the structure of $\mathsf{P}^{\mathsf{SAT}}$. Certainly, any oracle machine can ignore its oracle, so $\mathsf{P} \subseteq \mathsf{P}^{\mathsf{SAT}}$. Also notice that all of NP is deterministic polytime computable relative to SAT. Since SAT is NP-complete under polynomial time reduction, $\mathsf{NP} \subseteq \mathsf{P}^{\mathsf{SAT}}$. Certainly $\mathsf{P}^{\mathsf{SAT}}$ is very different than P. In fact, $\mathsf{coNP} \subseteq \mathsf{P}^{\mathsf{SAT}}$, as $\mathsf{TAUT} \in \mathsf{P}^{\mathsf{SAT}}$. Nondeterminism is a biased, onesided power. You can guess something only if it exists. Meanwhile an oracle will tell you yes/no just as strongly. We won't explain why, but $\mathsf{NP}^{\mathsf{SAT}}$ is actually bigger than NP as well.

As a second example, Consider P^A if $A \in \mathsf{P}$. Certainly $\mathsf{P} \subseteq \mathsf{P}^A$. We may prove $\mathsf{P}^A \subseteq \mathsf{P}$ by simulation. Replace each oracle call with a polynomial time algorithm to decide A. This is just a polynomial time algorithm and the result follows.

We don't really care about comparing two classes, one with and one without the oracle. We care about a "relativized world". One in which every machine has oracle access.

Definition 2.3 (Relativized World)**.** A world relative to oracle A is one in which every machine has access to the same oracle. Every complexity class is a relativized one.

For some fixed A, what does the world look like? What is the relationship between $\mathsf{L}^A, \mathsf{P}^A, \mathsf{NP}^A, \mathsf{PSPACE}^A$ and so on. How does this relativized world differ from our own? For each fixed A, there exists an entire separate world with its own language and rules and relationships. What theorems that we have proven in our world hold in all worlds?

3 Time Hierarchy Theorem

Now we prove a weaker form of the time-hierarchy theorem. A hierarchy is like a ladder, we are able to prove that more asymptotic time gives more power. The strongest form of the theorem says

$$\mathsf{TIME}(o(f(n)/\log f(n)) \subsetneq \mathsf{TIME}(f(n))$$

We prove a weaker form of the theorem to demonstrate the same technique.

Theorem 1 (Time Hierarchy Theorem)**.** $\forall k\ \mathsf{TIME}(n^k) \subsetneq \mathsf{TIME}(n^{k+1})$

Proof. The containment is obvious, so we only need to show existence of a language computable in time $O(n^{k+1})$ but not in time $O(n^k)$. We will diagonalize over all languages decidable in time n^k and make sure our language is decidable in time n^{k+1}.
Let $M_1, M_2, ...$ be an enumeration of the Turing machines in $\mathsf{TIME}(n^k)$. Construct a Turing machine D as follows.

Algorithm 1

D on input w_i
- Compute $n = |w_i|$
- Simulate M_i on w_i for n^k steps
- **if** M_i accepts w_i **then**
  - reject
- **end if**
- **if** M_i rejects w_i **then**
  - accept
- **end if**

Notice D on input w_i returns $1 - M_i(w_i)$. So $\nexists j$ such that $L(D) = L(M_j)$, so $L(D) \notin \mathsf{TIME}(n^k)$. Since it differs from every n^k-time machine, there is no n^k time algorithm to decide $L(D)$. What is the cost of simulation? Turns out this is complicated but we can safely upper bound that simulation of M_i for a single step takes at most $O(n)$ steps for the simulator. Since $n^k n = n^{k+1}$, we conclude $L(D) \in \mathsf{TIME}(n^{k+1})$. □

Two quick comments, first, the n^k machines are not enumerable, so this is only a rough proof idea. There is a fix around that, but it can be quite messy. Second, notice that the

tightness of our hierarchy depends on the complexity of simulation. This can vary actually from this or that Turing machine formalization. The fact there is a hierarchy at all is the take-away, rather than the specific hierarchy itself.

It turns out the time hierarchy theorem is true in every single relativized world.

Theorem 2 (Relativized Time Hierarchy Theorem)**.** Every relativized world has a time hierarchy theorem. For A any oracle that $\forall k\ \mathsf{TIME}(n^k)^A \subsetneq \mathsf{TIME}(n^{k+1})^A$.

Proof. We simply copy and paste the proof of the time hierarchy theorem and make minor adjustments to the simulation. We give an oracle machine D^A so that D^A diagonalizes against all n^k oracle machines but runs in time n^{k+1}.

Algorithm 2

D on input w_i
- Compute $n = |w_i|$
- Simulate M_i on w_i for n^k steps
- **if** M_i accepts w_i **then**
  - reject
- **end if**
- **if** M_i rejects w_i **then**
  - accept
- **end if**

Algorithm 3

$D^{\mathbf{A}}$ on input w_i
- Compute $n = |w_i|$
- Simulate* ${M_i}^A$ on w_i for n^k steps
- **if** ${M_i}^{\mathbf{A}}$ accepts w_i **then**
  - reject
- **end if**
- **if** ${M_i}^{\mathbf{A}}$ rejects w_i **then**
  - accept
- **end if**

In our world, D simulates M_i. In the relativized world, D^A simulates ${M_i}^A$. If ${M_i}^A$ makes an oracle call to A, D^A simulates this instruction of ${M_i}^A$ by calling its own oracle. Here the simulation only has this slight difference.

□

4 Relativization

Notice that since the proof of the time hierarchy theorem only interacted with computation in a black box way, we were able to modify the proof in an extremely simple way so that it held for all oracles, so that every relativized world had its own time hierarchy theorem.

Definition 4.1 (Relativizing Proof)**.** We say a proof relativizes if you may copy-paste from our world to all worlds so that it holds for any oracle. A proof of some complexity statement C relativizes if only slight modification to the proof can be made to demonstrate $\forall O[C^O]$.

Relativization is a specific kind of generalization. Determining if a proof relativizes is not a formal notion, but a sort of vibes based informal one. Which proofs have we done appear to relativize? Certainly every proof by diagonalization appears to relativize. Which proofs have we done so that appear to relativize and which ones don't appear to?

Probably relativizes:

- existence of undecidable languages
- time hierarchy theorem
- Ladner's theorem
- Savitch's theorem

- $\mathsf{NP}^A = \mathsf{NP}^A_v$
- $\mathsf{P}^A \neq \mathsf{EXP}^A$
- $P \subseteq \mathsf{NP} \subseteq PSPACE$
- and more...

Probably doesn't relativize:

- Cook-Levin
- TQBF is PSPACE-complete
- and more?

For the results which probably don't relativize, it is unclear where we could "stick the oracle". Most of the proofs we have done so far appear to relativize, they involve simulations of one machine by another in a purely black box way. The internals of computation are not involved. We did nothing with the machines except run them. Again, if a proof relativizes or not is an informal notion, but it is unambiguous.

5 Contradictory Relativizations

As we may relativize proofs in our world to all worlds, we may relate the truth of other worlds back to our own. Lets take the contrapositive of the definition of a relativizing proof

$$\begin{aligned} \text{proof of } C \text{ relativizes} &\implies \forall O[C^O] \text{ is true} \\ \exists A, B \text{ with } C^A \text{ true and } C^B \text{ false} &\implies \text{no relativizing proof exists of } C \end{aligned}$$

Could there exist a relativizing proof of $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$? Suppose maybe it could, by diagonalization. It might look like this. If it was a relativizing proof, then its counterpart would also exist.

Algorithm 4

Enumerate P machines $M_0, M_1, ...$
D on input w_i
  Simulate M_i on w_i
  Return opposite
Conclude $L(D) \notin \mathsf{P}$
Somehow show $L(D) \in \mathsf{NP}$

Algorithm 5

Enumerate P^A machines $M_0^A, M_1^A, ...$
D^A on input w_i
  Simulate* ${M_i}^A$ on w_i
  Return opposite
Conclude $L(D^A) \notin \mathsf{P}^A$
Somehow show $L(D^A) \in \mathsf{NP}^A$

If such a proof existed to prove $\mathsf{P} \neq \mathsf{NP}$, then it would be true that for all oracles O that $\mathsf{P}^O \neq \mathsf{NP}^O$. We give an oracle A so that $\mathsf{P}^A = \mathsf{NP}^A$. Then no such relativizing proof can exist to show that $\mathsf{P} \neq \mathsf{NP}$. Could there exist a relativizing proof that $\mathsf{P} = \mathsf{NP}$? Then it would be true that for all oracles O that $\mathsf{P}^O = \mathsf{NP}^O$. We give an oracle B such that $\mathsf{P}^B \neq \mathsf{NP}^B$. We show two oracles A, B such that

$$\mathsf{P}^A = \mathsf{NP}^A \qquad\qquad \mathsf{P}^B \neq \mathsf{NP}^B$$

Since there exists two relativized worlds, one where $\mathsf{P} = \mathsf{NP}$ and one where $\mathsf{P} \neq \mathsf{NP}$, no proof of $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ in our world can generalize to all worlds. Our demonstration of contradictory

relativizations will prove that there is no relativizing proof of P vs NP in our world! You cannot use diagonalization to separate P from NP!!!!!

6 A World Where its True

Theorem 3. There is a world relative to oracle A where $\mathsf{P}^A = \mathsf{NP}^A$

Proof. We choose A to elevate $\mathsf{P}^A, \mathsf{NP}^A$ to the same class where non-determinism gives no power. Certainly $\forall A\ \mathsf{P}^A \subseteq \mathsf{NP}^A$ so we show for some A that $\mathsf{NP}^A \subseteq \mathsf{P}^A$. We will use space complexity. Let $A = \mathsf{TQBF}$. Then

$$\mathsf{NP}^A = \mathsf{NP}^{\mathsf{TQBF}} \subseteq \mathsf{NPSPACE} = \mathsf{PSPACE} \subseteq \mathsf{P}^{\mathsf{TQBF}} = \mathsf{P}^A$$

The containments may be obvious but we expand on each.

- $\mathsf{NP}^{\mathsf{TQBF}} \subseteq \mathsf{NPSPACE}$ Consider an NP machine with oracle access to TQBF. We can simulate this machine on an NPSPACE machine. Simply replace the oracle by our linear space algorithm, and simulate the rest of the computation identically. The nondeterminism will simulate the nondeterminism.
- $\mathsf{PSPACE} = \mathsf{NPSPACE}$ This follows immediately from Savitch's theorem.
- $\mathsf{PSPACE} \subseteq \mathsf{P}^{\mathsf{TQBF}}$ Recall that TQBF is PSPACE-complete. We gave a polynomial time reduction for any $L \in \mathsf{PSPACE}$ that $L \leq_p \mathsf{TQBF}$. We simulate a PSPACE machine on a P^{TQBF} machine as follows. Given the description of the polynomial space machine and its input, we apply the reduction to get some quantified formula Φ. We query the oracle and accept/reject appropriately based on its answer. This reduction takes polytime, and there is only one query, so this machine runs in polytime.

□

7 A World Where its False

Theorem 4. There is a world relative to oracle B where $\mathsf{P}^B \neq \mathsf{NP}^B$

Proof. Showing oracle B such that $\mathsf{P}^B \neq \mathsf{NP}^B$ will be much harder. Ironically, we will construct B by diagonalization. We cannot prove $\mathsf{P} \neq \mathsf{NP}$ by diagonalization, but we can prove $\mathsf{P}^B \neq \mathsf{NP}^B$ by diagonalization. We want to show a language exists which could not be in P^B. How we will show it cannot be done in polynomial number of steps? We will define our language such that any correct algorithm to decide it must make an exponential number of oracle queries. Since each query takes unit time, an exponential number of queries implies that the machine to decide this language must take exponential time, and and thus could not have been a P^B machine. Let L_B contain the string 1^n if there is a string in B or length n.

$$L_B = \{1^n \mid B \cap \{0,1\}^n \neq \emptyset\}$$

Let ${M_1}^B, {M_2}^B, ...$ be an ordering of the oracle machines of P^B. Lets even suppose they are weakly sorted to guarantee ${M_i}^B$ halts in time n^i on all inputs. We will construct B by diagonalization (and therefore L_B) so that no polytime oracle machine may correctly decide L_B. We proceed in a sequence of stages. At each stage, a finite number of strings have been prophesized to be in B and not in B. In stage i, we will ensure that machine M_i^B is incorrect. At stage zero, we begin with $B = \emptyset$.

Suppose we are stage i. Let w be the largest string in B. Choose n such that $2^n > n^i$ and $n > |w|$. Note that n is chosen so that none of the 2^n strings of length n have been prophesized to be in B (yet). We will increase the knowledge about B such that ${M_i}^B$ accepts $1^n \iff 1^n \notin L_B$.
Run ${M_i}^B$ on 1^n and record all its oracle queries. On its query to oracle B, if it has been queried by that string before, the oracle will respond consistently. If B has not seen the string before, the oracle will prophesize no; that string will be defined to not be a member of B.

- If ${M_i}^B$ accepted 1^n, all other strings of length n are declared not to be in B. This ensures that $1^n \in L(M_i^B) \implies 1^n \notin L_B$.
- If ${M_i}^B$ rejected 1^n, declare one string of length n that M_i^B did not have time to query to be in B. Since ${M_i}^B$ runs in time n^i, it does not have time to query B on all 2^n strings of length n, so such a string must exist that it did not query. This ensures that $1^n \notin L(M_i^B) \implies 1^n \in L_B$.

Since for every i there is an n such that $1^n \in L_B \iff 1^n \notin L(M_i^B)$. Since this is done for all polytime oracle machines, none of them can decide L_B so B is defined so that $L_B \notin \mathsf{P}^B$.

Why is $L_B \in \mathsf{NP}^B$? Rather than testing all 2^n strings against the oracle, nondeterministically guess the right one to test the oracle against. This takes unit time on an NP^B machine but exponential time on a P^B machine. Since $L_B \in \mathsf{NP}^B \setminus \mathsf{P}^B$, we observe $\mathsf{P}^B \neq \mathsf{NP}^B$. □

8 Frustration. Coping. Crying.

This result has set the stage for the next half-century of complexity research. Any proof which could resolve P vs NP would genuinely have to use new techniques, ones which do not relativize. We weren't even sure at the time if such techniques existed! The last fifty years has seen attempts trying to bend the rules. To list a few:

- Randomness: What if SAT is decidable in polytime by an algorithm which returns the assignment correctly only two thirds of the time? Maybe the deterministic requirement of the algorithm is too stringent.
- Approximation: What if there exists an algorithm for SAT to satisfy a majority of the clauses in polytime, but this last stretch to all clauses requires exponential? Maybe the correctness requirement of the algorithm is too stringent.
- Circuits: Proofs using circuits do not appear to relativize. Does there exist a super polynomial lower bound on circuit size for SAT? Maybe the uniformity requirement of the algorithm is too stringent.

All of these areas have been good at asking questions and bad at giving answers. Randomized or approximate polytime algorithms for SAT can imply the existence of deterministic polytime algorithms for SAT, making these rather hard as well. The fact that $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ has no relativizing proof is called the relativization barrier. This is not the only barrier the problem admits! Complexity theory has gone through several historical eras. We are further from answering P vs NP than when the question was conjectured. The relativization barrier was just the first hurdle, we would hit many many more. Truly, there is no harder problem. No problem has produced more corpses than P vs NP.