CS 4510 Automata and Complexity 4/17/2023

Lecture 20: Relativization

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

1 Diagonalization

First we begin with a review of diagonalization. Diagonalization is a proof technique. You order some elements, construct a "fixed point", or a "diagonal", and take its opposite, reaching some kind of contradiction. We will reprove the halting problem, but this time, using a functional notation.
Let $\varphi_1, \varphi_2, ...$ be an enumeration of the recursive functions (these correspond to Turing machines which are allowed to loop). We prove that there is no total computable function h -

$$h(x,y) = \begin{cases} 1 & \varphi_x(y) \text{ halts} \\ 0 & \varphi_x(y) \text{ loops} \end{cases}$$

Assume to the contrary that h is total and computable. Consider the function d:

$$d(x) = \begin{cases} 1 & \text{if } h(x,x) = 0 \\ \text{loops} & \text{if } h(x,x) = 1 \end{cases}$$

Certainly d exists since h does. d is a recursive function, so there must exist some i such that $d = \varphi_i$. Consider d on its own index i.

$$d(i) = 1 \iff h(i,i) = 0 \iff \varphi_i(i) \text{ loops} \iff d(i) \text{ loops} \tag{1}$$

$$d(i) \text{ loops} \iff h(i,i) = 1 \iff \varphi_i(i) \text{ halts} \iff d(i) \text{ halts} \tag{2}$$

We arrive at a contradiction. There cannot exist a total recursive function for h.

2 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. We show

$$\forall k \ \mathsf{TIME}(n^k) \subsetneq \mathsf{TIME}(n^{k+1})$$

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 which run in time n^k and make sure our language runs 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.

3 Remarks

Both of these proofs had the same structure. We interacted with other machines but only in a black-box way. We simulated them only to disagree with their output. We did nothing with M except run it. We did not deal with any of the internal mechanics of computation. We can separate the decidable from the undecidable and an infinite hierarchy of deterministic time classes. Note that such a proof could also separate P from EXP. Could we use such techniques to separate P from NP? Turns out, no. Thats the point of today's lecture.

4 Relativization

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. Antiquated version of a magic eight ball. An oracle machine has some similar mysticism. It is a 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.

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. We won't explain why, but $\mathsf{NP}^{\mathsf{SAT}}$ is actually bigger than NP.

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. For some fixed A, what does the world look like? What is the relationship between $L^A, \mathsf{P}^A, \mathsf{NP}^A, \mathsf{PSPACE}^A$ 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. We can relate these worlds to our own.
We say a proof "relativizes" if you can copy paste it from our world to the relativized world with only a minor modification.

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^A on input w_i
- Compute $n = |w_i|$
- Simulate* ${M_i}^A$ on w_i for n^k steps
- **if** ${M_i}^A$ accepts w_i **then**
  - reject
- **end if**
- **if** ${M_i}^A$ 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. Could there exist a proof of diagonalization for $\mathsf{P} \neq \mathsf{NP}$? It might look like:

Algorithm 4

Enumerate P machines $M_0, M_1, ...$
D on input w_i
- Simulate M_i on w_i
- Return opposite

Somehow show $L(D) \in \mathsf{NP}$

Algorithm 5

Enumerate P^A machines $M_0, M_1, ...$
D^A on input w_i
- Simulate* ${M_i}^A$ on w_i
- Return opposite

Somehow show $L(D^A) \in \mathsf{NP}^A$

If a relativizing proof exists in our world, we say it relativizes to all worlds. If $\mathsf{P} \neq \mathsf{NP}$ in our world, and it is provable in this way, then $\forall A\ , \mathsf{P}^A \neq \mathsf{NP}^A$. It holds in all relativized worlds. However, if there exists A such that $\mathsf{P}^A = \mathsf{NP}^A$, then there exists a world whose version of the problem is $\mathsf{P}^A = \mathsf{NP}^A$, so there cannot exist a relativizing proof that $\mathsf{P} \neq \mathsf{NP}$ from our own. Similarly if $\exists B$, a relativized world where $\mathsf{P}^B \neq \mathsf{NP}^B$, then there cannot exist a relativizing proof that $\mathsf{P} = \mathsf{NP}$.We show two oracles A, B such that

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

Since there exists two 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!!!!!

5 A World Where its True

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

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

Relative to TQBF, every PSPACE language is decidable in nondeterministic polynomial time. The second containment holds by Savitch's theorem. The final one holds similarly to why $\mathsf{NP} \subseteq \mathsf{P}^{\mathsf{SAT}}$. We observe then that relative to $A = \mathsf{TQBF}$ that $\mathsf{NP}^A = \mathsf{P}^A$.

6 A World Where its False

Showing oracle B such that $\mathsf{P}^B \neq \mathsf{NP}^B$ will be much harder. Ironically, we will construct 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? For correctness, we will require 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 in P^B.
Let L_B = set of strings such that there is some string of the same length in B. We don't say which string to require making 2^n queries.

$$L_B = \{w \mid \exists\, x \in B \text{ with } |x| = |w|\}$$

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 construct B in a sequence of steps such that ${M_i}^B$ does not decide L_B is ensured in stage i. At each stage, only a finite amount of strings have been decided to be in B and decided

to not be in B.
Suppose we are stage i. Let w be the largest string in B. Choose n such that $2^n > n^i$ and $n > |w|$. 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. 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, it will prophesize no, 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. If ${M_i}^B$ accepted 1^n, all other strings of length n are declared not to be in B. If ${M_i}^B$ rejected 1^n, declare one string of length n to be in B. Therefore $L({M_i}^B) \neq L_B \notin \mathsf{P}^B$.
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 $\mathsf{P}^B \neq \mathsf{NP}^B$.

7 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. 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.