CS 4510 Automata and Complexity April 17, 2024

Lecture 24: Karp-Lipton Theorems (Draft)

Lecturer: Abrahim Ladha *Scribe(s): Michael Wechsler*

Last lecture, we simply defined the polynomial hierarchy as an infinite set of dual classes which looks like this.

PICTURE

Today we will discuss how to actually put the polynomial hierarchy to use.

1 Natural Problems up the Hierarchy

[TBD, minimum formulas and other problems]

Theorem 1. For each i, Π_i, Σ_i have complete problems, complete for only that level.

Proof. Recall the definition of a complete problem. For A_k some Σ_k-complete problem, we know that $A_k \in \Sigma_k$ and $\forall L \in \Sigma_k$ that $L \leq_p A_k$. We prove that for each i that Π_i and Σ_i have such complete problems. The complete problem will simply be a generalization of SAT. Define $QSAT_i$ as

$$QSAT_i = \underbrace{\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \ldots}_{i} [\Phi]$$

where Φ is some CNF, and each x_i is just some polynomial number of variables. The language $QSAT_i$ consists of true quantified boolean formula, whose quantifiers are alternating, the first one is existential, and there are no more than i quantifiers. Observe that $QSAT_1$ is simply SAT and is NP-complete.

We will prove $QSAT_i$ is Σ_i-complete by showing it is in Σ_i and is Σ_i-hard.

First we prove that $QSAT_i$ is Σ_i-hard. Let $L \in \Sigma_i$. Then there exists a Σ_i machine to decide L. This machine M takes on the form

$$w \in L \iff \exists x_1 \forall x_2 ... Q x_i M(w, x_1, ..., x_i) \text{ accepts}$$

We may simply apply the Cook-Levin style construction to this machine to output a polynomial sized CNF Φ_M. The polynomial sized string input x_i to M corresponds to the polynomial amount of variables to the CNF defined to be x_i.

To prove that $QSAT_i \in \Sigma_i$, we simply may give a Σ_i machine. It quantifies appropriately over the variables and checks the CNF. □

2 Collapse

We proved the polynomial hierarchy has this infinite containment, but is it actually infinite? Are the containments strict? There are two possible models. One is that the polynomial hierarchy is actually infinite, and the containments are strict. This is the most believed and

accepted structure. The other is that the polynomial hierarchy *collapses*. After some level k that $\Sigma_{k+1} = \Sigma_k$. Everything above this level is equivalent in power to this last level.

[TBD why shouldn't the hierarchy collapse]

It turns out the polynomial hierarchy is quite delicate, and any peturbance to its structure will collapse it. We prove this in a few ways.

Theorem 2. If for any i that $\Sigma_i = \Pi_i$ then $\mathsf{PH} \subseteq \Sigma_i$. If there is a level of the polynomial hierarchy closed under complement then the polynomial hierarchy is not infinite, and collapses to that level.

Each Π_i, Σ_i behave like a struct or pillar, supporting the levels above them. Were it the case that two distinct pillars were the same, our tower collapses.

Proof. Suppose that for some i that $\Sigma_i = \Pi_i$. Then consider the next level, Σ_{i+1}

$$\Sigma_{i+1} = \exists\Pi_i = \exists\Sigma_i = \Sigma_i$$

Lets expand on these steps.

- $(\Sigma_{i+1} = \exists\Pi_i)$ This follows from the definition given previously.
- $(\exists\Pi_i = \exists\Sigma_i)$ Although we are assuming that $\Pi_i = \Sigma_i$, what we are really doing here is this. We know the logical definition of $\exists\Pi_i$ is

  $$w \in L \iff \exists x_1 \forall x_2 ... Q x_i M(w, x_1, ..., x_i) \text{ accepts}$$

  Note that the computation on $x_2, ..., x_i$ is a Π_i computation. We may replace it by an equivalent Σ_i one since we are assuming that $\Sigma_i = \Pi_i$.

  $$\exists x_1 \forall x_2 ... Q x_i M(w, x_1, ..., x_i) \text{ accepts} \iff \exists x_1 \exists x_2 ... Q x_i M'(w, x_1, ..., x_i) \text{ accepts}$$

- $(\exists\Sigma_i = \Sigma_i)$ We perform a quantifier compression, the same way we proved $\exists\exists\mathsf{P} = \exists\mathsf{P} = \mathsf{NP}$.

We showed that the next level $\Sigma_{i+1} = \Sigma_i$, a single level collapse. Why does this imply that all of PH collapses? Observe that for $k > i$

$$\Sigma_k = \underbrace{\exists\forall ... \forall\exists\forall}_{k-i}\Sigma_i = \exists\forall ... \forall\exists \underbrace{\forall\Pi_i}_{\Pi_i} = \underbrace{\exists\forall ... \forall\exists}_{k-i-1}\Pi_i = \exists\forall ... \forall \underbrace{\exists\Sigma_i}_{\Sigma_i} = \underbrace{\exists\forall ... \forall}_{k-i-2}\Sigma_i = ... = \Sigma_i$$

Repeatedly swap Σ_i with Π_i, compress a quantifier, then swap back. □

Theorem 3. If any two levels of PH are equal then PH collapses to that level.

The main motivating study of the polynomial hierarchy was to ask, if $\mathsf{P} = \mathsf{NP}$, then what else is in P? We prove a less general, but more informative idea.

Corollary 4. If $\mathsf{P} = \mathsf{NP}$, then $\mathsf{PH} \subseteq \mathsf{P}$.

Proof. Suppose that $\mathsf{P} = \mathsf{NP}$. Then since P is closed under complement, we know that $\mathsf{P} = \mathsf{coNP}$ as well, so $\mathsf{NP} = \mathsf{coNP}$, or that $\Pi_1 = \Sigma_1$. We may apply the previous result to solve then that PH collapses to Σ_1. □

Note that this proof also works for when $\mathsf{P} \neq \mathsf{NP}$ but $\mathsf{NP} = \mathsf{coNP}$.

3 Implications

We are interested in the relationships between PH, and its the other more algorithmically defined complexity classes.

Theorem 5. PH collapses if and only if there is a PH-complete problem. Here, we mean complete for the entire class.

Proof. Suppose that there was a PH-complete problem A. It would be the case that $A \in \mathsf{PH}$ and $\forall L \in \mathsf{PH}$ that $L \leq_p A$. Since this complete problem is in PH, it resides at some level of PH, suppose then $A \in \Sigma_k$. Since PH is closed under polytime reduction, this would imply that $\mathsf{PH} \subseteq \Sigma_k$.

To prove the other way, suppose that the polynomial hierarchy collapses to some Σ_k. We know that each level has a complete problem $QSAT_k$, which now serves as a complete problem for all of PH. □

You may observe that since $\mathsf{PH} \subseteq \mathsf{AP} = \mathsf{PSPACE}$ that $\mathsf{PH} \subseteq \mathsf{PSPACE}$. You may even conjecture that $\mathsf{PH} = \mathsf{PSPACE}$. In some sense this stretches PH upward. But ironically, this weakens it as then PH collapses.

Corollary 6. $\mathsf{PH} = \mathsf{PSPACE} \implies$ PH collapses to some level.

Proof. Since PSPACE has a complete problem, namely TQBF, then this would also be a complete problem for PH. We may apply the previous theorem to infer a collapse. □

It is then believed that $\mathsf{PH} \subsetneq \mathsf{PSPACE}$.

Theorem 7. If PH is infinite then $\mathsf{P} \neq \mathsf{PSPACE}$

We can prove it with something even weaker than if the polynomial hierarchy is infinite, we need to only suppose there exists any two levels which are strictly not equal $\Sigma_i \subsetneq \Sigma_j$

Proof.

$$\mathsf{P} \subseteq \Sigma_i \subsetneq \Sigma_j \subseteq \mathsf{PSPACE} \implies \mathsf{P} \neq \mathsf{PSPACE}$$

□

4 Karp-Lipton Theorems

We show relationship among uniform and non-uniform complexity classes. Since we may hope that PH is infinite, demonstration of its collapse leads to the negation of the assumption being plausible.

Theorem 8 (Karp-Lipton)**.**

$$\mathsf{NP} \subseteq \mathsf{P/poly} \implies \mathsf{PH} \subseteq \Pi_2 \cup \Sigma_2$$

If SAT has a polynomial sized circuit family, then the polynomial hierarchy collapses to its second level.

Originally Karp and Lipton proved it by a collapse to the third level, but Sipser improved it to the second level. The proof idea is to show $\Pi_2 \subseteq \Sigma_2$. Conversion of any $\forall\exists$-sentence to a $\exists\forall$-sentence means for any sentence higher in the hierarchy, we can repeatedly alternate and compress quantifiers until a sentence with many quantifiers is left with only two.

Proof. Let $\mathsf{NP} \subseteq \mathsf{P/poly}$, then SAT has a polynomial sized circuit family, $\{C_0, C_1, \ldots\}$. Each polynomial sized circuit C_n takes as input an n-variable formula and outputs a single bit for yes/no if the input formula was satisfiable. By a decision-to-search transformation, there exists a circuit family $\{C'_0, C'_1, \ldots\}$. Where each C'_n outputs n bits for not just if it was satisfiable or not, but the satisfying assignment itself. Since each C_n is polynomially-sized, so is each C'_n. This decision-to-search transformation should be believable, but to give you an example suppose we had a circuit to say if some formula Φ was satisfiable or not. If x_1 is the first variable of Φ, then $\Phi \wedge x_1$ is a formula which is satisfiable if and only if Φ was satisfiable, with $x_1 = 1$. We can play a hotter/colder[1] game with the circuit families to infer not just if a formula was satisfiable, but what the actual satisfying assignment was. This decision-to-search transformation will incur only a polynomial overhead.

Let $L \in \Pi_2$. Then

$$w \in L \iff \forall x \exists y M(w, x, y) \text{ accepts}$$

We may convert M to a CNF Φ_M using a Cook-Levin style construction. Note since M runs in a polynomial number of steps, Φ_M is polynomially sized, and its construction takes polynomial time. Now, notice there is a k such that $C'_k(\Phi_M, w, x) = y$. The CNF Φ_M takes on several input variables which correspond to w, x, y. Since SAT has a polynomial sized circuit family, there exists a polysized circuit to search for this witness y instead of quantifying over it. By conversion of our machine into a CNF, we may use the polynomial sized circuit family to compute the witness which would bring the machine to accept! We can replace the existential quantification of x_2 with a computation of C'_k. So our definition of L has an equivalent statement:

$$\forall x \exists y M(w, x, y) \text{ accepts} \iff \forall x M(w, x, C'_k(\varphi_M, w, x)) \text{ accepts} \tag{1}$$

Now how do we determine C'_k? Since it is polynomial sized, we simply may quantify over it. Use existential quantification to guess the C'_k circuit. This circuit must exist by assumption that $SAT \in \mathsf{P/poly}$.

$$\forall x \exists y M(w, x, y) \text{ accepts} \iff \exists C'_k \forall x_1 M(w, x, C'_k(\varphi_M, w, x)) \text{ accepts} \tag{2}$$

Note that this is an equivalent Σ_2 statement. We converted a Π_2 sentence into a Σ_2 one! $L \in \Pi_2 \Rightarrow L \in \Sigma_2 \Rightarrow \Pi_2 \subseteq \Sigma_2$. We observe that if $\mathsf{NP} \subseteq \mathsf{P/poly}$, then the polynomial hierarchy does collapse to its second level. □

The technique developed here is called self-reducibility. In the case you may solve SAT efficiently, you can use this as a primitive to perform witness extraction. There are more applications of it.

[1] hotter/colder aka binary search over the 2^n possible assignments

Theorem 9 (Meyer)**.**

$$\mathsf{EXP} \subseteq \mathsf{P/poly} \implies \mathsf{EXP} \subseteq \Sigma_2$$

If EXP has polynomial sized circuits then EXP is contained in the second level of the polynomial hierarchy.

Proof. Let $L \in \mathsf{EXP}$. Then there is a machine to decide L which runs in 2^{n^k} time for some k. Let the configurations of this exponential time machine be $z_0, ..., z_{2^{n^k}}$ Note that $w \in L \iff \forall i\ z_i$ can satisfy some easily checkable conditions. For example, z_0 is initial, $z_{2^{n^k}}$ is accepting, and so on. Let T be some machine which checks the configurations syntatically. Then

$$w \in L \iff T(w, z_0, ..., z_{2^{n^k}}) \text{ accepts}$$

If $\mathsf{EXP} \subseteq \mathsf{P/poly}$ then there exists a polynomial sized circuit family to compute z_i given i. Let this circuit family be denoted by C. We may replace z_i with $C(i)$. Then replace the sequence of configurations by quantifying over every i.

$$w \in L \iff \forall i T'(w, C(i)) \text{ accepts}$$

How do we determine the circuit family C then? Since it is polynomially sized, we again may simply quantify over it.

$$w \in L \iff \exists C \forall i T'(w, C(i)) \text{ accepts}$$

This is a Σ_2 statement so we see that EXP collapses to Σ_2. □

Meyer's theorem is a good demonstration of when upper bounds can imply lower bounds.

Theorem 10. If $\mathsf{P} \neq \mathsf{NP}$ then $\mathsf{EXP} \neq \mathsf{P/poly}$

Proof. Suppose $\mathsf{P} = \mathsf{NP}$. Then we know that $\mathsf{PH} = \mathsf{NP} = \mathsf{P}$. If $\mathsf{EXP} \subseteq \mathsf{P/poly}$ then $\mathsf{EXP} \subseteq \Sigma_2 \subseteq P$, or that $\mathsf{EXP} = \mathsf{P}$, contradicting the time hierarchy theorem. □

5 Kannan's Theorem

Theorem 11 (Kannan)**.** There is a language

$$L_k \in \Sigma_2 \setminus \mathsf{SIZE}(n^k)$$

First, this does not say that $\Sigma_2 \notin \mathsf{P/poly}$. It constructs a different language L_k for each polynomial n^k. Our proof of the weak time hierarchy theorem proved that $\mathsf{TIME}(n^k) \subsetneq \mathsf{TIME}(n^{k+1})$ but this doesn't show $\mathsf{P} \neq \mathsf{P}$. $\mathsf{PH} \neq \mathsf{P/poly}$ is an open question that this does not resolve. Although it does prove $L_1 \notin \mathsf{SIZE}(n), L_2 \notin \mathsf{SIZE}(n^2), ...$, this does not rule out that it could be the case that $L_k \in \mathsf{SIZE}(n^{2k})$ for example.

First we prove a related statement two steps up the hierarchy.

Theorem 12 (Also Kannan)**.** There is a language

$$L_k \in \Sigma_4 \setminus \mathsf{SIZE}(n^k)$$

Proof. We proceed by diagonalization. We construct a Σ_4 alternating machine which uses no more than four alternating quantifiers. These quantifiers will diagonalize against every $O(n^k)$ sized circuit family to ensure none of them are correct. By construction, the language decided by this machine will be in Σ_4.

Algorithm 1 Σ_4 machine for L_k

on input w, let $n = |w|$
$\exists$ C^* circuit of size $\leq n^{2k+5}$
    $\forall$ C' circuits of size $\leq n^{k+1}$
        $\exists$ y a string of length n
            $C^*(y) \neq C'(y)$
    $\forall$ C circuits of size $\leq n^{k+1}$
        $\exists$ C_0 circuit of size $\leq n^{k+1}$
            $\forall$ z a string of length n
                $C_0(z) = C(z)$

The size C^* is quantified over is n^{2k+5} because this is sufficiently larger than n^{k+1}. Just enough to guarantee it can decide a language not decided by n^{k+1} sized circuits. The first line ensures it is decided by a circuit family of size n^{2k+5}. The second, third, and fourth lines ensure it cannot be decided by a circuit family of size n^{k+1}. This is the diagonalization step. Lines 5-8 ensure that our Σ_4 machine is forced to simulate C^* on every input of length n. Let the language decided by this machine be L_k. Since this is a $\exists\forall\exists\forall$ computation, we see $L_k \in \Sigma_4$. Since this computation diagonalizes against every circuit family of size n^{k+1}, we see that $L_k \notin \mathsf{SIZE}(n^k)$. Therefore, $L_k \in \Sigma_4 \setminus \mathsf{SIZE}(n^k)$. □

Lets make some observations on this proof. Notice how the diagonalization occured, it was not by construction of a table. Second is how powerful a few quantifiers can be. If you consider NP to be an unreasonable class, I wonder what you may think of Σ_4.

It is unlikely this diagonalization could be done with fewer quantifiers, but we can finish Kannan's theorem nonconstructively.

Proof. We prove that there exists $L \in \Sigma_2 \setminus \mathsf{SIZE}(n^k)$. We proceed nonconstructively. We have two cases:

- $SAT \notin \mathsf{P/poly}$. Then $SAT \notin \mathsf{SIZE}(n^k)$, so $L = SAT$
- $SAT \in \mathsf{P/poly}$. By the Karp-Lipton theorem, $\Sigma_4 \subseteq \Sigma_2$, so $L_k \in \Sigma_2 \setminus \mathsf{SIZE}(n^k)$ so $L = L_k$

$L = SAT$ or $L = L_k$. Which one? We don't know! But its one of them! Either way, we have established that such a language must exist. □

[graph isomorphism]

6 Further Study

I want to conclude with some advice on how to self study complexity theory. Your journey doesn't have to end here if you don't want it to. First, finish the Sipser book. It does not contain everything, but it does contain the best proofs of what it does cover. I wish it had a second volume. It's coverage of randomness, interaction, cryptography, and more may surprise you. Before you go further, you should definitely finish Sipser. After you finish Sipser, go through the first six chapters of the Arora-Barak book. All the other chapters (7+) cover an incredible breadth of material, and good pointers to other sources. These may include communication complexity, quantum complexity, the complexity of counting, and so on. Each of these chapters deserves (and has) their own books, but it's an introduction to these theories. You need to know what the things you don't know are called in order to google and learn them. Then go through Goldreich's and Papadimitriou's books. Goldreich has 400 pages of incredible notes. Finally, I recommend the books *The Nature of Computation* and Wigderson's *Mathematics and Computation*. Both of these are light on proofs, as a tradeoff for coming with incredible wisdom. If you want a proof, use the other books. If you want to what a proof means, use Wigderson's book. Of course, you may use me as a resource. If you have any questions, or come across anything in your own independent study, I would be happy to help and answer. Thank you for taking my class, I had a lot of fun.

- Introduction to the Theory of Computation, Michael Sipser 2012
- Computational Complexity: A Modern Approach, Sanjeev Arora and Boaz Barak, 2009
- Computational Complexity: A Conceptual Perspective, Oded Goldreich, 2008
- Computational Complexity, Christos Papadimitriou, 1994
- Mathematics and Computation, Avi Wigderson, 2019
- Goldreich's encyclopedic lecture notes. 375 pages across two semesters. An invaluable resource from 1999.
  `http://gen.lib.rus.ec/book/index.php?md5=fbb240574e6f5059fccdce95fab0ff38`
- Hatami's notes from 2022, also insanely useful.
  `https://www.cs.mcgill.ca/~hatami/comp531-F2022/files/Lectures.pdf`

Each level is defined only using finitely many quantifiers, so it would appear that $\mathsf{PH} \subseteq \mathsf{PSPACE}$ since $TQBF$ is a PSPACE-complete problem. If that containment is strict, it is also a an open problem. We would hope to show it is since $\mathsf{P} \subseteq \mathsf{PH} \subsetneq \mathsf{PSPACE} \Rightarrow \mathsf{P} \neq \mathsf{PSPACE}$.