CAS CS 535: Complexity Theory

Lecturer: Mark Bun Fall 2023

Lecture Notes 9:

Immerman-Szelepcsényi Theorem, Polynomial Hierarchy

Reading.

- Arora-Barak § 4.3, 5.1-5.2

Last time: Logspace Computation
Recall our canonical $\mathbf{NL}$-complete problem

$$\mathsf{PATH} = \{\langle G, s, t\rangle \mid \text{ Digraph } G \text{ has a path from } s \text{ to } t\}$$

and the certificate-verifier view of the complexity class $\mathbf{NL} = \mathbf{NSPACE}(\log n)$.

Theorem 1. *A language $A \in \mathbf{NL}$ if and only if there exists a logspace TM V with a read-once "certificate tape" and a polynomial p such that for all $x \in \{0,1\}^*$,*

$$x \in A \iff \exists u \in \{0,1\}^{p(|x|)} \quad M(x,u) = 1.$$

Here, x is given to M on its input tape while u is given on its certificate tape.

1 Immerman-Szelepcsényi Theorem: $\mathrm{NL} = \mathrm{coNL}$

Today, we'll prove a remarkable result about logspace computation, which one can view as the space-bounded analog of $\mathbf{NP} = \mathbf{coNP}$.

Theorem 2. $\mathbf{NL} = \mathbf{coNL}$.

The way we'll prove this is by showing that the $\mathbf{coNL}$-complete problem $\overline{\mathsf{PATH}}$ is also contained in $\mathbf{NL}$. Pause for a moment to think about how remarkable that is. While it's straightforward to check the existence of a path with a logspace verifier, this is saying that there's a similarly efficient way to certify the *non*-existence of such a path.

Here's the intuition for the proof. Say I want to convince you that in a digraph G with n vertices, there is no path from s to t. I can do this by convincing you of the following two statements:

1. There are exactly m_n distinct vertices reachable from s by paths of length $\leq n$.

2. The destination vertex t is *not* one of those m_n vertices.

Now how would I actually convince you of these statements? Let's start with the second one. I can do that by showing you m_n vertices *other* than t which are reachable from s by paths of length $\leq n$. Now how about the first one? The idea we'll use for this one is "inductive counting." For each $k = 0, \dots, n-1$, I'll show you that "if there are m_k vertices reachable by paths of length $\leq k$, then there are m_{k+1} vertices reachable by paths of length $\leq k+1$."

Proof. Let G be a graph with n vertices. For each $i = 0, \ldots, n$, let C_i be the set of vertices reachable from s within $\leq i$ steps. Then

$$\langle G, s, t\rangle \in \overline{\mathsf{PATH}} \iff t \notin C_n \iff \exists m_0, \ldots, m_n \text{ s.t. } \begin{cases} |C_0| = m_0 (= 1) \\ \text{If } |C_0| = m_0 \text{ then } |C_1| = m_1 \\ \vdots \\ \text{If } |C_{n-1}| = m_{n-1} \text{ then } |C_n| = m_n \\ \text{If } |C_n| = m_n \text{ then } t \notin C_n. \end{cases}$$

That is, certifying all of the statements on the right is equivalent to certifying the statement on the left. Let us now see how to describe and verify these certificates.

1. $|C_0| = m_0$. This one is easy. Since $|C_0| = \{s\}$, the only possibility is to take $m_0 = 1$ which the verifier can immediately check.

2. $|C_n| = m_n \implies t \notin C_n$. Define u_v to be a certificate for the fact that there is a path from s to v of length $\leq n$ (i.e., the list of vertices along this path). Take the certificate to be $(u_{v_1}, u_{v_2}, \ldots, u_{v_{m_n}})$ where the vertices are sorted so that $v_j < v_{j+1}$ and $t \neq v_j$ for every j.

   To check this certificate, scan it once while checking that a) the vertices are indeed sorted, b) m_n distinct vertices are hit, c) t does not appear in the list, and d) the individual path certificates all check out.

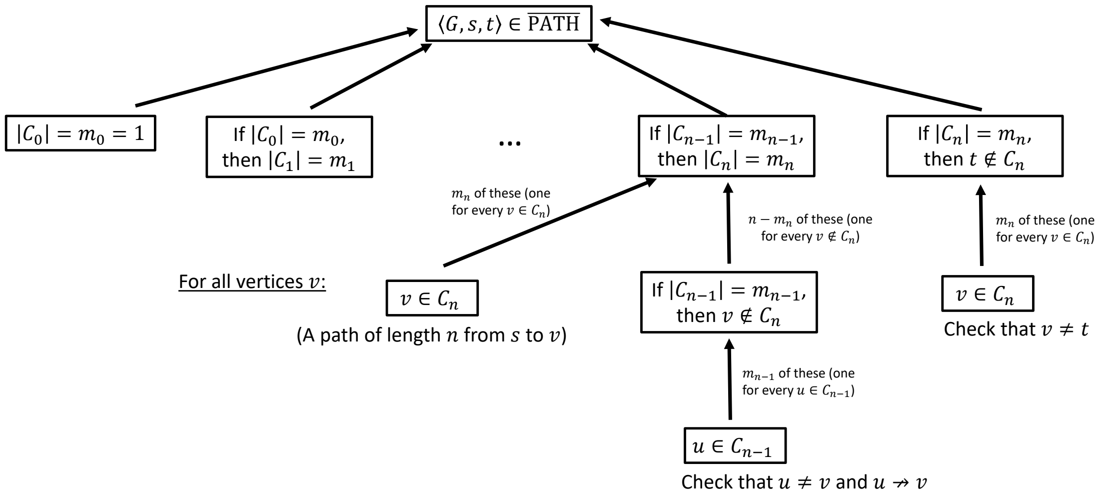
3. $|C_k| = m_k \implies |C_{k+1}| = m_{k+1}$. Here, the certificate will take the form $(w_1, w_2, \ldots, w_n)$ where each w_i itself is a certificate for either "$v_i \in C_{k+1}$" or "If $|C_k| = m_k$, then $v_i \notin C_{k+1}$." This suffices because the verifier can count to ensure there are exactly m_{k+1} "$\in C_{k+1}$" certificates while verifying each individual certificate.

   Now let's see what these individual certificates look like.

   (a) To certify $v_i \in C_{k+1}$: Just exhibit a path from s to v_i of length $\leq k + 1$.

   (b) To certify "If $|C_k| = m_k$, then $v_i \notin C_{k+1}$": Similar to Case 2, let u_v be a certificate for the fact that there is a path from s to v of length $\leq k$. Our certificate is now $(u_{v_1}, \ldots, u_{v_{m_k}})$, where again the vertices are sorted so that every $v_j < v_{j+1}$. Checking this certificate is similar, but now we check that for every target vertex v_j in this list, there is no edge $v_j \to v_i$.

□



2 Polynomial Hierarchy

So far, our zoo of major complexity classes looks like

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP} \subseteq \mathbf{NEXP}.$$

Today, we'll peer into the space between $\mathbf{NP}$ and $\mathbf{PSPACE}$ guided by (at least) two motivations:

1. The complexity of some problems seems to lie strictly between $\mathbf{NP}$ and $\mathbf{PSPACE}$. Can we classify these problems?
2. What exactly are we capable of solving if $\mathbf{P} = \mathbf{NP}$?

Let's start with an example.

Example 3. Given a DNF formula φ, is there a "small" DNF ψ computing the same function?

$$\begin{aligned}\textsf{SMALL-EQ-DNF} &= \{\langle \varphi, k\rangle \mid \exists \text{ a DNF } \psi \text{ of size } \leq k \text{ s.t. } \psi \equiv \varphi\} \\ &= \{\langle \varphi, k\rangle \mid \exists \text{ a DNF } \psi \text{ of size } \leq k \text{ s.t. } \forall x, \psi(x) = \varphi(x)\}.\end{aligned}$$

So what's interesting about this problem?

1. It seems harder than $\mathbf{NP}$ problems. The "obvious" certificate ψ looks like it needs exponential time to check (i.e., testing all possible assignments x).
2. Nevertheless, if $\mathbf{P} = \mathbf{NP}$, we can solve this problem efficiently. To see this, define the auxiliary language

$$\textsf{EQ-DNF} = \{\langle \varphi, \psi\rangle \mid \forall x, \psi(x) = \varphi(x)\}.$$

If $\mathbf{P} = \mathbf{NP}$, then $\mathbf{P} = \mathbf{coNP}$, so $\textsf{EQ-DNF} \in \mathbf{P}$. But now this implies that $\textsf{SMALL-EQ-DNF} \in \mathbf{NP} = \mathbf{P}$! This is because we can write

$$\textsf{SMALL-EQ-DNF} = \{\langle \varphi, k\rangle \mid \exists \text{ a DNF } \psi \text{ of size } \leq k \text{ s.t. } \langle \varphi, \psi\rangle \in \textsf{EQ-DNF}\}.$$

So we can take ψ as a certificate and verify it in poly-time.

So again, our goal is to study problems like this in a more systematic way. To do so, we'll start by defining some operations that allow us to build new complexity classes from old.

Definition 4. Let $\mathbf{C}$ be complexity class. Define

$\exists\mathbf{C}$**:** A language $L \in \exists\mathbf{C}$ if there exists a language $R \in \mathbf{C}$ and a polynomial p such that $x \in L \iff \exists u \in \{0,1\}^{p(|x|)}$ s.t. $(x, u) \in R$.

$\forall\mathbf{C}$**:** A language $L \in \forall\mathbf{C}$ if there exists a language $R \in \mathbf{C}$ and a polynomial p such that $x \in L \iff \forall u \in \{0,1\}^{p(|x|)}$ s.t. $(x, u) \in R$.

Example 5. $\exists\mathbf{P} = \mathbf{NP}$. What about $\forall\mathbf{P}$? $\exists\exists\mathbf{P}$? $\exists\forall\mathbf{P}$?

Definition 6. Define the classes $\mathbf{\Sigma_2^p} = \exists\forall\mathbf{P}, \mathbf{\Sigma_3^p} = \exists\forall\exists\mathbf{P}, \mathbf{\Sigma_4^p} = \exists\forall\exists\forall\mathbf{P}, \ldots$. In general,

$$\mathbf{\Sigma_i^p} = \exists\forall\exists\ldots Q_i\mathbf{P}$$

where $Q_i = \exists$ if i is odd and $Q_i = \forall$ if i is even.

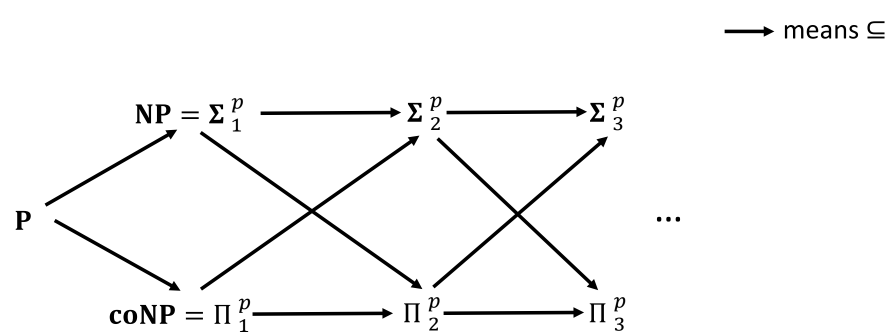
Similarly, define

$$\mathbf{\Pi_i^p} = \forall\exists\forall\ldots Q_i\mathbf{P}$$

where $Q_i = \forall$ if i is odd and $Q_i = \exists$ if i is even.

Here are some basic observations about these classes:

- $\mathbf{\Sigma_1^p} = \mathbf{NP}$.
- $\mathbf{\Pi_1^p} = \mathbf{coNP}$.
- For every i, we have $\mathbf{\Sigma_i^p}, \mathbf{\Pi_i^p} \subseteq \mathbf{\Sigma_{i+1}^p} \cap \mathbf{\Pi_{i+1}^p}$.



Definition 7. The polynomial hierarchy is defined as

$$\mathbf{PH} = \bigcup_{i=1}^{\infty} \mathbf{\Sigma_i^P} = \bigcup_{i=1}^{\infty} \mathbf{\Pi_i^P}$$

Theorem 8. *If* $\mathbf{P} = \mathbf{NP}$*, then* $\mathbf{PH} = \mathbf{P}$.

Proof idea. It suffices to show that if $\mathbf{P} = \mathbf{NP}$, then $\mathbf{\Sigma_i^P} \in \mathbf{P}$ for every i. We immediately have $\mathbf{\Sigma_1^P} = \mathbf{NP} = \mathbf{P}$, and therefore also that $\mathbf{coNP} = \mathbf{P}$. Now observe that $\mathbf{\Sigma_2^P} = \exists\forall\mathbf{P} = \exists\mathbf{coNP} = \exists\mathbf{P} = \mathbf{NP} = \mathbf{P}$. And so on, by induction. □

Theorem 9. *If* $\mathbf{\Sigma_i^P} = \mathbf{\Pi_i^P}$*, then* $\mathbf{PH} = \mathbf{\Sigma_i^P}$*. (If this happens, we say "*$\mathbf{PH}$ *collapses to the* i*'th level.")*

Proof. $\mathbf{\Sigma_{i+1}^P} = \exists\mathbf{\Pi_i^P} = \exists\mathbf{\Sigma_i^P} = \mathbf{\Sigma_i^P}$ and so on. □

A widely believed conjecture (generalizing $\mathbf{P} \neq \mathbf{NP}$) is that the polynomial hierarchy does not collapse. Some more observations:

1. $\mathbf{\Sigma_i^P}$ is closed under poly-time reductions $\leq_p$: That is, if $A \leq_p B$ and $B \in \mathbf{\Sigma_i^P}$, then $A \in \mathbf{\Sigma_i^P}$.
2. $\mathbf{\Sigma_i^P}$ has complete problems. For example,

   $$\Sigma_i\text{-}\mathsf{SAT} = \{\text{TQBFs of the form } \exists x^{(1)}\forall x^{(2)} \ldots \varphi(x^{(1)}, x^{(2)}, \ldots, x^{(i)})\},$$

   where each $x^{(j)}$ denotes a block of variables, is $\mathbf{\Sigma_i^P}$-complete.

A natural question you might ask is: Does $\mathbf{PH}$ itself have complete problems? The answer is "probably not."

Theorem 10. *If* $\mathbf{PH}$ *has a complete problem, then* $\mathbf{PH}$ *collapses.*

Proof. Suppose L is $\mathbf{PH}$-complete. Then $L \in \mathbf{\Sigma_i^P}$ for some level i. On the other hand, for every language $A \in \mathbf{PH}$, we have $A \leq_p L$, so $A \in \mathbf{\Sigma_i^P}$. Hence $\mathbf{PH} \subseteq \mathbf{\Sigma_i^P}$. □