CAS CS 535: Complexity Theory

Lecturer: Mark Bun — Fall 2023

Lecture Notes 10:

PH via oracles, alternation

Reading.

- Arora-Barak § 5.2, 5.3, 5.5

Last time: $\mathbf{NL} = \mathbf{coNL}$, Polynomial Hierarchy

Definition 1. Define

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

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.

It's helpful to unpack some lower levels of the hierarchy explicitly. For instance, $\mathbf{\Sigma_2^p}$ is the class of languages L such that there exist polynomials p and q and a poly-time TM M such that

$$x \in L \iff \exists u \in \{0,1\}^{p(|x|)} \forall v \in \{0,1\}^{q(|x|)} M(x,u,v) = 1.$$

Definition 2. 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 3. *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 4. *If for some level* i*, we have* $\mathbf{\Sigma_i^p} \subseteq \mathbf{\Pi_i^p}$*, then* $\mathbf{PH} = \mathbf{\Sigma_i^p} = \mathbf{\Pi_i^p}$*. (If this happens, we say "*$\mathbf{PH}$ *collapses to the* i*'th level.")*

Proof. First observe that if $\mathbf{\Sigma_i^p} \subseteq \mathbf{\Pi_i^p}$, then $\mathbf{\Pi_i^p} = \mathsf{co}\mathbf{\Sigma_i^p} \subseteq \mathsf{co}\mathbf{\Pi_i^p} \subseteq \mathbf{\Sigma_i^p}$, so we actually have the equality $\mathbf{\Sigma_i^p} = \mathbf{\Pi_i^p}$.

Now $\mathbf{\Sigma_{i+1}^p} = \exists\mathbf{\Pi_i^p} = \exists\mathbf{\Sigma_i^p} = \mathbf{\Sigma_i^p} = \mathbf{\Pi_i^p}$ 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,

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

   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 5. *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 $\mathbf{PH} \subseteq \mathbf{\Sigma_i^p}$. □

1 PH via Oracles

Recall that for a language A, the class $\mathbf{NP}^A$ is the class of languages decidable in poly-time by an NTM with oracle access to A.

Theorem 6. $\mathbf{\Sigma_2^p} = \mathbf{NP}^{\mathsf{SAT}}$.

Proof. As usual, there are two directions to show.

$\mathbf{\Sigma_2^p} \subseteq \mathbf{NP}^{\mathsf{SAT}}$. It's enough to show that the complete problem $\Sigma_2\text{-SAT} \in \mathbf{NP}^{\mathsf{SAT}}$. This is because $\mathbf{NP}^{\mathsf{SAT}}$ is closed under poly-time reductions. Recall that

$$\Sigma_2\text{-SAT} = \{\varphi \mid \exists u \forall v \varphi(u, v) = 1\}.$$

So the following poly-time NTM (with access to a SAT oracle) decides this language:

On input φ:
  Nondeterministically guess an assignment b to u
  Query the SAT oracle on $\overline{\varphi(b, v)}$
  Accept if unsatisfiable (i.e., $\forall v \varphi(b, v) = 1$).

$\mathbf{NP}^{\mathsf{SAT}} \subseteq \mathbf{\Sigma_2^p}$. Let $L \in \mathbf{NP}^{\mathsf{SAT}}$ be decided by a poly-time oracle NTM N running in time $T(n)$. Let $m = T(|x|)$. As a first attempt, we'd like to use the observation that

$$x \in L \iff \exists b_1, \ldots, b_{T(|x|)} N^{\mathsf{SAT}}(x) \text{ accepts on nondeterministic choices } b.$$

However, it's not really clear how to check the condition on the right, even with a universal quantifier, since N might issue an adaptive sequence of oracle queries and responses.

To overcome this, we'll offload more work to the outer existential quantifier by using it to guess the entire transcript of N's computation. Then we'll use it again, together with the inner universal quantifier, to check that this transcript is correct.

In more detail, observe that

$$x \in L \iff \exists C_1, \ldots, C_m \text{ s.t. } \begin{cases} C_1 = C_{\text{start}}, \\ C_i \to C_{i+1} \forall i = 1, \ldots, m-1, \text{ and} \\ C_m = C_{\text{accept}}. \end{cases}$$

Here, each C_i represents a configuration of $N^{\mathsf{SAT}}(x)$'s computation.

So how do we check that $C_i \to C_{i+1}$?

1. If it's a "normal" transition, juts check (in poly-time) consistency with N's transition function.
2. If C_i queries the SAT oracle on some formula φ_i, receiving answer $a_i = 1$ in C_{i+1}, then check "$\exists u_i \varphi(u_i)$."
3. If C_i queries the SAT oracle on φ_i receiving answer $a_i = 0$, check "$\forall v_i \overline{\varphi(v_i)}$."

Putting everything together, we thus have

$$x \in L \iff \exists C_1, \ldots, C_m \exists u_1, \ldots, u_m \forall v_1, \ldots, v_m ((a_i \wedge \varphi(u_i)) \vee (\overline{a_i} \wedge \overline{\varphi(v_i)})) \wedge \text{[poly-time consistency checks]}.$$

□

In general, we have $\mathbf{\Sigma_{i+1}^p} = \mathbf{NP}^{\Sigma_i\text{-}\mathsf{SAT}}$.

Definition 7. For complexity class $\mathbf{C}$, define $\mathbf{NP}^{\mathbf{C}} = \bigcup_{L \in \mathbf{C}} \mathbf{NP}^L$.

Note that if $\mathbf{C}$ has a complete problem A (under poly-time reductions, then $\mathbf{NP}^{\mathbf{C}} = \mathbf{NP}^A$. Thus,

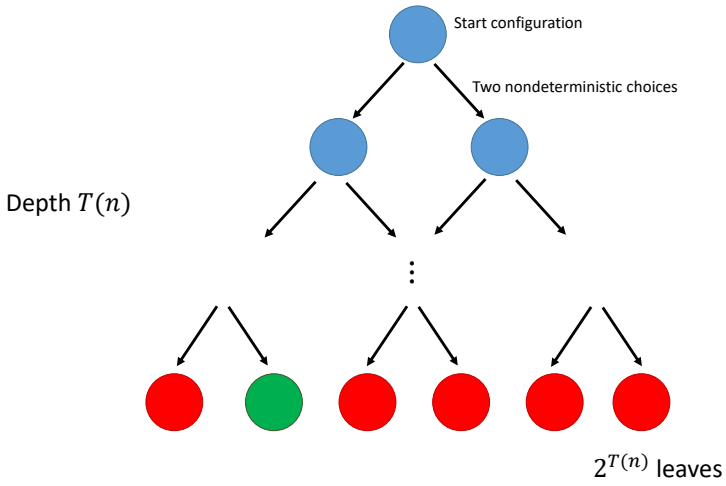
$$\mathbf{NP}^{\mathbf{NP}} = \mathbf{NP}^{\mathsf{SAT}} = \mathbf{\Sigma_2^p}.$$

Similarly, $\mathbf{\Sigma_3^p} = \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$ etc.

2 Alternating Turing Machines

Alternating TMs generalize NTMs to allow for a mixture of existential and universal "guessing." While, like NTMs, they don't represent a realistic model of computation, they are helpful for "programming" natural algorithms for certain problems and give illuminating alternative characterizations of the complexity classes we've studied.

To motivate the definition, let's recall the tree view of plain old nondeterministic computation. When running an NTM N on an input x, it starts in some initial configuration. Then in every time step, it has a choice of two transition functions to follow. The NTM accepts if there exists a leaf in this tree representing an accepting configuration.



Similarly, one can define a coNTM, except that it accepts whenever all leaves are in accepting configurations.

Another way to think about this is that in an NTM, all nodes in the tree of computation paths are labeled by $\vee$ ("existential guessing"). The interpretation here being that the computation as a whole accepts if at least one of the outgoing paths leads to acceptance. On a coNTM, think of all nodes as being labeled by $\wedge$ ("universal guessing"), with the computation accepting if both outgoing paths lead to acceptance.

Now a question you might ask is: What happens if we mix $\wedge$ and $\vee$ labels? This gives us alternating TMs.

Definition 8 (Informal)**.** An alternating TM is like an NTM, but before taking each transition, it may choose whether to interpret the branch as existential ($\vee$) or universal ($\wedge$).

See Arora-Barak for how to formalize this definition. Basically, each *state* in an alternating TM is labeled with either $\vee$ or $\wedge$, governing how to interpret the transitions from that state as above. To determine wither an alternating TM accepts an input, imagine writing down the configuration graph and use the state labels to backpropagate the acceptance criteria up to the start configuration.

A useful perspective on alternating TMs comes from parallel computation. On each transition, you can think of an alternating TM as spawning two processes. It can then choose to accept overall if either:

$\vee$: At least one process accepts

$\wedge$: Both processes accept.

Example 9. Recall the problem

$$\text{SMALL-EQ-DNF} = \{\langle \varphi, k\rangle \mid \exists \text{ DNF } \psi, |\psi| \leq k, \quad \forall x \varphi(x) = \psi(x)\}.$$

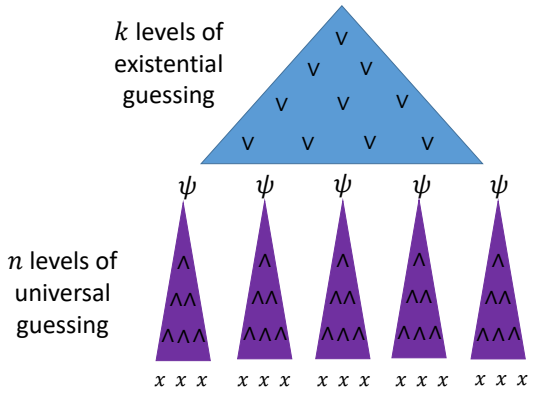
The following alternating TM decides this language:

On input $\langle \varphi, k\rangle$:

Existentially guess a DNF ψ of size $\leq k$

Universally guess an input x

Accept iff $\psi(x) = \varphi(x)$.



2.1 Bounded Alternations

Definition 10. An ATM is of Σ_i type if its start state is labeled $\vee$ and it alternates $\leq i-1$ times on every computation branch. (Similarly for Π_i type, but with $\vee$ replaced with $\wedge$)

Define the class

$$\mathbf{\Sigma_i\text{-}TIME}(T(n)) = \{L \mid L \text{ is decidable by a } \Sigma_i \text{ type ATM in } O(T(n)) \text{ steps}\}.$$

ATMs with bounded alternations give us yet another characterization of the polynomial hierarchy.

Fact 11.

$$\mathbf{\Sigma_i^P} = \bigcup_{c=1}^{\infty} \mathbf{\Sigma_i\text{-}TIME}(n^c)$$

2.2 Unbounded Alternations

Define the classes

$$\mathbf{ATIME}(T(n)) = \{L \mid L \text{ is decidable by an ATM in } O(T(n)) \text{ steps}\}$$
$$\mathbf{ASPACE}(T(n)) = \{L \mid L \text{ is decidable by an ATM in } O(T(n)) \text{ space}\}.$$

These naturally give rise to alternating analogs of the main classes we've studied, e.g., $\mathbf{AL}$, $\mathbf{AP}$, $\mathbf{APSPACE}$. What do we know about these alternating classes? It turns out we know exactly what they are: $\mathbf{AL} = \mathbf{P}$, $\mathbf{AP} = \mathbf{PSPACE}$, $\mathbf{APSPACE} = \mathbf{EXP}$.

Theorem 12. $\mathbf{AP} = \mathbf{PSPACE}$.

Proof. As always, there are two things to show.

$\mathbf{PSPACE} \subseteq \mathbf{AP}$. Since $\mathbf{AP}$ is closed under poly-time reductions, it suffices to show that the $\mathbf{PSPACE}$-complete problem TQBF is in $\mathbf{AP}$. Here's the alternating algorithm:

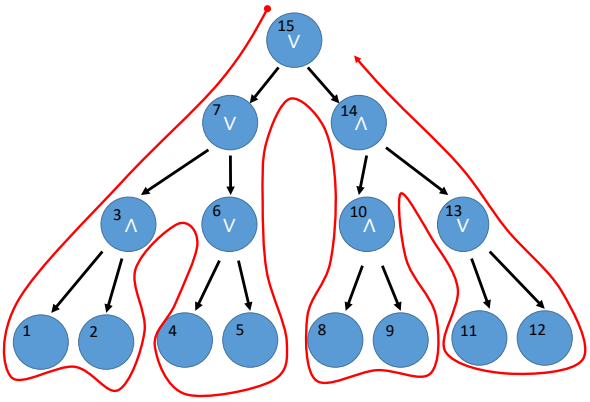
On input QBF $\Psi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \varphi$:

If $n = 0$, evaluate φ

If $Q_1 = \exists$: Existentially guess x_1, and recurse on $\Psi|_{x_1}$.

If $Q_1 = \exists$: Universally guess x_1, and recurse on $\Psi|_{x_1}$.

AP $\subseteq$ PSPACE. We'll actually show the more general statement that an ATM running in time $T(n)$ can be simulated by a deterministic TM running in space $O(T(n))$. Recall that to simulate an alternating TM M on an input x, we could materialize the tree of possible computations and propagate the acceptance criteria up from the leaves to the root. Unfortunately, if our ATM runs in time $T(n)$, this tree has size roughly $2^{T(n)}$. So instead, the idea will be to simulate this evaluation while only constructing nodes as we need them. More specifically, we'll do a depth-first post-order traversal to evaluate the nodes in the tree.



At any point in the traversal, one needs enough working space to simulate one branch of the computation, which takes $O(T(n))$ space, plus maintain the identity of the current working path from the tree root, which takes another $O(T(n))$ space. So the total space usage of this algorithm is $O(T(n))$. □