CS 6810: Theory of Computing **Spring 2026**

Lecture 6: February 10, 2026

Lecturer: Eshan Chattopadhyay *Scribe: Gordon Mei*

1 Introduction

In previous classes, we introduced the idea of polynomial time hierarchy, generalizing the definitions of **P**, **NP**, **coNP**. We recall the following definitions.

Definition 1.1 (Polynomial Hierarchy)**.** *The* polynomial hierarchy *consists of complexity classes* Σ_i^P *and* Π_i^P *defined inductively:*

$$\begin{aligned}\Sigma_0^P &= \Pi_0^P = \mathbf{P}\\ \Sigma_i^P &= \exists \Pi_{i-1}^P \quad \textit{for } i \geq 1\\ \Pi_i^P &= \forall \Sigma_{i-1}^P \quad \textit{for } i \geq 1\end{aligned}$$

Theorem 1.2 (Polynomial Hierarchy via Oracle Machines)**.**

$$\begin{aligned}\Sigma_i^P &= \mathbf{NP}^{\Sigma_{i-1}^P} \quad \textit{for } i \geq 1\\ \Pi_i^P &= \mathbf{coNP}^{\Pi_{i-1}^P} \quad \textit{for } i \geq 1\end{aligned}$$

We illustrate this theorem by proving the case $i = 2$. The general case follows by a similar inductive argument.

Claim 1.3. $\mathbf{\Sigma_2^P}$ *is equivalent to* $\mathbf{NP}^{SAT}$*.*

Proof. We prove both directions.

($\Sigma_2^P \subseteq \mathbf{NP}^{\text{SAT}}$): It suffices to show that Σ_2-SAT, a complete problem for Σ_2^P, can be solved by a nondeterministic Turing machine with oracle access to SAT in polynomial time.

Recall that in Σ_2-SAT, we are given a boolean formula $\phi(x, y)$ and asked whether:

$$\exists x \, \forall y \, . \, \phi(x, y)$$

where x and y are sets of variables $x_1, \ldots, x_n$ and $y_1, \ldots y_m$.

Let our NDTM $N^{\mathbf{SAT}}$ operate as follows.

1. Nondeterministically guess an assignment for $x = x_1, \ldots, x_n$.
2. Query the oracle: $\exists y \, \neg\phi(x, y) \in$ SAT?
3. If the oracle returns NO (meaning $\forall y \, \phi(x, y)$ holds), then accept. Otherwise, reject.

Then, our TM accepts iff there exists some x such that $\forall y \, \phi(x, y)$ holds, which matches the definition of Σ_2-SAT. Then, $\Sigma_2^P \subseteq \mathbf{NP}^{\text{SAT}}$ holds.

($\mathbf{NP}^{\text{SAT}} \subseteq \Sigma_2^P$): Suppose that some language L can be solved by an NDTM N^{SAT} in polynomial time. We want to show that $L \in \Sigma_2^P$. We recall that $\Sigma_2^P = \exists\Sigma_1^P = \exists\forall\mathbf{P}$.

On input x, the machine N^{SAT} accepts x iff there exists a sequence of non-deterministic choices and correct oracle answers that make N^{SAT} accept x. More precisely, there exists a sequence of choices $c_1, c_2, \dots c_l$ and oracle queries $\psi_1, \dots, \psi_k$ with answers $a_1, a_2, \dots, a_k$ such that on input x, if the machine N^{SAT} uses choices $c_1, \dots c_l$ and receives a_i as an answer to its ith query, then two things occur:

1. N^{SAT} reaches an accepting state.
2. All answers $a_1, \dots, a_k$ are correct.

If ψ_i is the ith oracle query that N^{SAT} makes and receives answer a_i, we can state (2) as follows:

- If $a_i = 1$, then there exists assignment u_i such that $\psi_i(u_i) = 1$.
- If $a_i = 0$, then for every assignment v_i, $\psi_i(v_i) = 0$.

We note that the first condition uses an existential quantifier (we can guess witnesses u_i for satisfiable queries), while the second uses a universal quantifier (we must verify that no witness exists for unsatisfiable queries). This naturally leads to a $\Sigma_2^P = \exists\forall\mathbf{P}$ formulation.

$$\begin{aligned} x \in L \iff \exists\, c_1, \dots, c_l, \psi_1, \dots, \psi_k, a_1, \dots, a_k, u_1, \dots, u_k\ \forall v_1, \dots, v_k \text{ such that} \\ N \text{ accepts } x \text{ using choices } c_1, \dots, c_l \text{ and answers } a_1, \dots, a_k \text{ AND} \\ \forall i \in [k] \text{ if } a_i = 1 \text{ then } \psi_i(u_i) = 1 \text{ AND} \\ \forall i \in [k] \text{ if } a_i = 0 \text{ then } \psi_i(v_i) = 0 \end{aligned}$$

Since each condition can be verified in polynomial time (by simulating N^{SAT} and checking satisfying assignments), this implies that $L \in \Sigma_2^P$. □

Remark 1.4 (Alternating Turing Machines)**.** *The polynomial hierarchy can also be characterized using alternating Turing machines with polynomial-bounded alternations.*

2 Space Complexity

We now turn our attention to space complexity, which looks into the memory requirements of computational tasks. With space complexity bounds, we restrict number of tape cells that can be used in our TM calculations.

Definition 2.1 (Space-bounded Computation)**.** *Let $S : \mathbb{N} \to \mathbb{N}$. We define* $\mathbf{DSPACE}(S(n))$ *(respectively,* $\mathbf{NSPACE}(S(n))$*) as all languages computable by a multitape TM M (resp. a non-deterministic TM M) using $O(S(n))$ space. That is, for all inputs x, the number of work tape cells used by M is at most $c \cdot S(|x|)$ for some constant c. The input and output tapes are not counted in the number of cells used.*

Definition 2.2 (Space Complexity Classes)**.** *Analogous to* $\mathbf{P}$ *and* $\mathbf{NP}$*, we define:*

$$\mathbf{PSPACE} = \bigcup_{c>0} \mathbf{DSPACE}(n^c)$$

$$\mathbf{NPSPACE} = \bigcup_{c>0} \mathbf{NSPACE}(n^c)$$

Definition 2.3 (Other Complexity Classes)**.** *Analogously, we define the logarithmic-space classes:*

$$\mathbf{L} = \mathbf{DSPACE}(\log n)$$

$$\mathbf{NL} = \mathbf{NSPACE}(\log n)$$

Open problem 2.4. *Is* $\mathbf{L} = \mathbf{NL}$*?*

Observation 2.5. *We have the following containments:*

1. $\mathbf{P} \subseteq \mathbf{PSPACE}$*, since a Turing machine running in polynomial time can use at most polynomial space.*
2. $\mathbf{NP} \subseteq \mathbf{NPSPACE}$*, by a similar argument.*
3. $\mathbf{PSPACE} \subseteq \mathbf{NPSPACE}$*, since every deterministic machine is also nondeterministic.*

We now show that polynomial space is contained in exponential time.

Theorem 2.6.

$$\mathbf{PSPACE} \subseteq \mathbf{EXP} = \bigcup_{c>0} DTIME\left(2^{n^c}\right)$$

To prove this, we use the notion of a configuration graph.

Definition 2.7 (Configuration Graph)**.** *Let M be a deterministic (or nondeterministic) Turing machine and $x \in \{0,1\}^*$ be an input string.*

The configuration graph *$G_{M,x}$ is a directed graph whose vertices are all possible configurations of M on input x that are reachable from the start configuration, denoted as C_{start}.*

A configuration consists of:

- *the current state of M,*
- *the contents all work tapes,*
- *the head positions of all work tapes.*

There is a directed edge from configuration C_1 to configuration C_2 if M can move from C_1 to C_2 in one transition.

Observe:

- If M is a deterministic TM, then $G_{M,x}$ has an out degree of at most 1.
- If M is a nondeterministic TM, then $G_{M,x}$ has an out degree of at most 2.

If M uses space $S(n)$, then each configuration can be described using $O(S(n))$ bits. Then, the total number of configurations is at most $2^{O(S(n))}$.

Proof. We now prove Theorem 2.6. Suppose a language $L \in \mathbf{PSPACE}$. Then there exists deterministic TM M deciding L using space n^c.

For input x, the configuration graph $G_{M,x}$ has at most $2^{O(n^c)}$ nodes. Given two nodes C_i, C_j of $G_{M,x}$, We can determine whether an edge between configurations C_i and C_j exists in polynomial time. Thus, we can construct the configuration graph in exponential time.

Since $x \in L$ if and only if there is a path from C_{start} to C_{accept}, we can perform a DFS or BFS over all configurations. Since the graph has at most $2^{O(n^c)}$ nodes, this search runs in time $2^{O(n^c)}$.

Therefore, $L \in \mathbf{EXP}$, and hence $\mathbf{PSPACE} \subseteq \mathbf{EXP}$. □

A similar argument with a nondeterministic TM shows that **NPSPACE** $\subseteq$ **EXP**.

Theorem 2.8 (Savitch's Theorem)**.** *Nondeterminism does not add power in polynomial space.*

$$\mathbf{PSPACE} = \mathbf{NPSPACE}$$

Open problem 2.9. *We know that*

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P}$$

but it remains open whether any of these containments are strict.