CAS CS 535: Complexity Theory

Lecturer: Mark Bun | Fall 2023

Lecture Notes 11:

More on Alternation, Time-Space Tradeoffs

Reading.

- Arora-Barak § 5.3-5.4

Last time: $\mathbf{PH}$ via oracles, alternation

We've now seen (at least?) four equivalent characterizations of each level of the polynomial hierarchy. For example, the class $\mathbf{\Sigma_2^p}$ can be described in any of the following ways:

1. $\exists\forall\mathbf{P}$
2. The class of languages L such that there exist polynomials p, q and a poly-time deterministic TM M such that
$$x \in L \iff \exists u \in \{0,1\}^{p(|x|)} \forall v \in \{0,1\}^{q(|x|)} M(x,u,v) = 1.$$
3. $\mathbf{NP}^{\mathbf{NP}} = \mathbf{NP}^{\mathsf{SAT}}$
4. $\cup_{c=1}^{\infty} \mathbf{\Sigma_2 TIME}(n^c)$

Here, recall that $\mathbf{\Sigma_2 TIME}(T(n))$ is the class of languages decidable by an alternating TM that starts in a $\vee$ state, and alternates at most once on every computation branch.

1 Unbounded Alternations

At the end of last class, we started talking about alternating TMs with an unbounded number of alternations. We defined

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

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: $\mathbf{AL} = \mathbf{P}$, $\mathbf{AP} = \mathbf{PSPACE}$, $\mathbf{APSPACE} = \mathbf{EXP}$. We'll do one of these today.

Theorem 1. $\mathbf{AP} = \mathbf{PSPACE}$.

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:

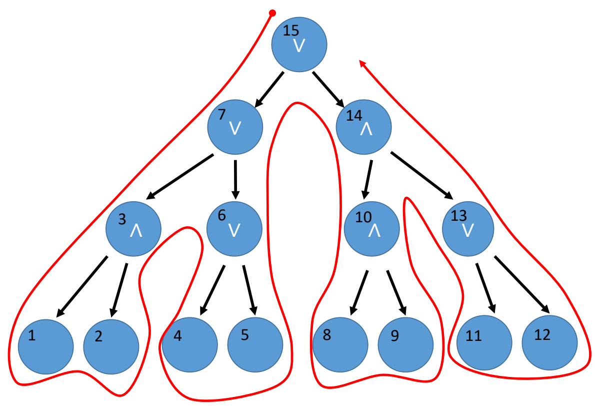
On input QBF $\Psi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \varphi$:

If $n = 0$, evaluate φ

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

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

$\mathbf{AP} \subseteq \mathbf{PSPACE}$. 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 $O(T(n))$. $\square$

2 Time-Space Tradeoffs

For all we know...

1. SAT *could* have a linear time algorithm, i.e., SAT $\in \mathbf{DTIME}(n)$.
2. SAT *could* have a logspace algorithm, i.e., SAT $\in \mathbf{L}$.
3. Or both! I.e., SAT $\in \mathbf{DTIME}(n) \cap \mathbf{L}$.

We believe these are probably not the case, but are very far from proving so. However, what we *can* show is that SAT cannot be solved by an algorithm that simultaneously runs in low time and in low space.

Definition 2. For functions $T(n)$ and $S(n)$, define $\mathbf{TISP}(T(n), S(n))$ to be the class of languages decidable by TMs running in both time $T(n)$ and space $S(n)$.

Make sure you understand the difference between $\mathbf{TISP}(T(n), S(n))$ and $\mathbf{DTIME}(T(n)) \cap \mathbf{SPACE}(S(n))$!

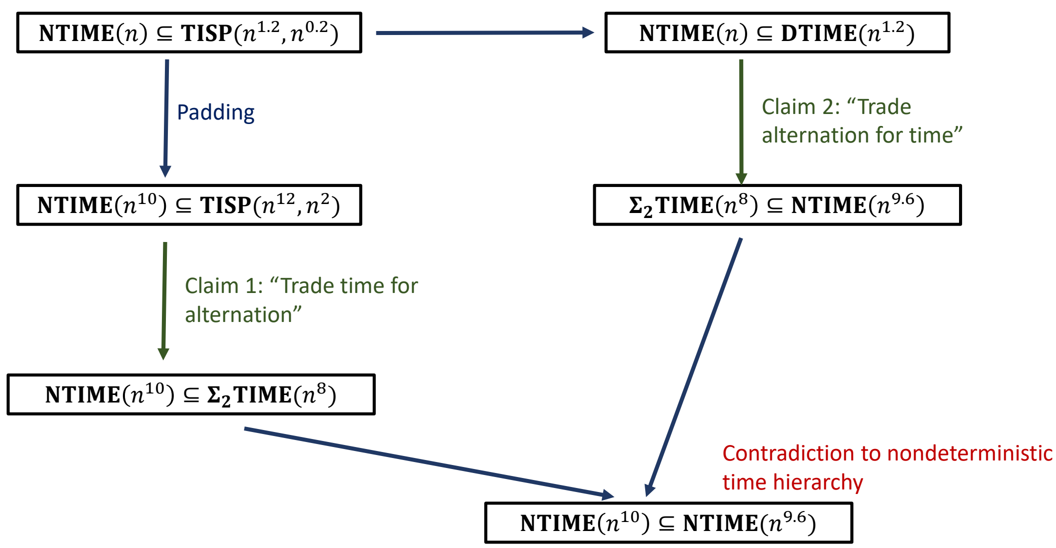
Curiously, while this is just a statement about the deterministic time/space complexity of a specific combinatorial problem, the proof crucially makes use of alternations.

Theorem 3.

$$\mathsf{SAT} \notin \mathbf{TISP}(n^{1.1}, n^{0.1}).$$

We won't prove this for SAT directly. What we'll actually show is that there exists a language $L \in \mathbf{NTIME}(n)$ such that $L \notin \mathbf{TISP}(n^{1.2}, n^{0.2})$. The statement about SAT follows from a refinement of the Cook-Levin Theorem which says that an arbitrary language $L \in \mathbf{NTIME}(T(n))$ can be reduced to SAT with only a quasi-linear blowup, i.e., each instance in L maps to a formula of size $O(T(n) \log T(n))$.

Here's the gameplan for the proof. Assume for the sake of contradiction that $\mathbf{NTIME}(n) \subseteq \mathbf{TISP}(n^{1.2}, n^{0.2})$. Our goal will be to derive a contradiction to the nondeterministic time hierarchy theorem as follows.



Claim 4. $\mathbf{TISP}(n^{12}, n^2) \subseteq \mathbf{\Sigma_2 TIME}(n^8)$.

Proof. Suppose L is decided by a TM M running in time $\leq Kn^{12}$ and space $O(n^2)$ for some constant K. Then

$$\begin{aligned} x \in L &\iff \exists \text{ a path from } C_{\text{start}} \text{ to } C_{\text{acc}} \text{ in } G_{M,x} \text{ of length } \leq Kn^{12} \\ &\iff \exists \text{ configurations } C_0, \ldots, C_{n^6} \text{ such that } C_0 = C_{\text{start}}, C_{n^6} = C_{\text{acc}} \\ &\qquad \text{and there is a path from each } C_{i-1} \text{ to } C_i \text{ of length } \leq Kn^6. \end{aligned}$$

Now observe that:

- The sequence of configurations $C_0, \ldots, C_{n^6}$ has description length $O(n^6) \cdot O(n^2) = O(n^8)$, since each configuration can be described in $O(n^2)$ bits.

- By simulation via the UTM, it's possible to check if C_{i-1} can reach C_i within Kn^6 steps using time $O(n^7)$.

Thus, the following Σ_2 type TM decides L in time $O(n^8)$:
On input x:
Existentially guess configurations $C_0, \dots, C_{n^6}$
Universally guess an index $i \in 1, \dots, n^6$
Check $C_0 = C_{\text{start}}$, $C_{n^6} = C_{\text{acc}}$, and that C_{i-1} leads to C_i in $\leq Kn^6$ steps. □

Claim 5. *If* $\mathbf{NTIME}(n) \subseteq \mathbf{DTIME}(n^{1.2})$*, then* $\mathbf{\Sigma_2 TIME}(n^8) \subseteq \mathbf{NTIME}(n^{9.6})$.

Proof. Let $L \in \mathbf{\Sigma_2 TIME}(n^8)$. Then there exists an $O(n^8)$-time TM M (where runtime is measured as a function of $|x|$) and constant c such that

$$\begin{aligned} x \in L &\iff \exists u \in \{0,1\}^{c|x|^8} \forall v \in \{0,1\}^{c|x|^8} M(x,u,v) = 1 \\ &\iff \exists u \in \{0,1\}^{c|x|^8} \text{ s.t. } \langle x, u \rangle \notin R, \end{aligned}$$

where the helper language R is defined as

$$R = \{\langle x, u \rangle \mid \exists v \in \{0,1\}^{c|x|^8} M(x,u,v) = 0\}.$$

Note that $R \in \mathbf{NTIME}(n) \subseteq \mathbf{DTIME}(n^{1.2})$ by assumption, so there is some deterministic TM D deciding R in time $O(n^{1.2})$.

Thus, we obtain the following NTM for the language L:
On input x:
Nondeterministically guess $u \in \{0,1\}^{c|x|^8}$
Run $D(x,u)$ and flip the answer.
The runtime of this NTM is $O(n^8) + O((n^8)^{1.2}) = O(n^{9.6})$. □

2.1 Different Time-Space Tradeoffs

1. The same proof gives the following. For every $\varepsilon > 0$, there exists $\delta > 0$ such that $\mathbf{NTIME}(n) \not\subseteq \mathbf{TISP}(n^{1+\delta}, n^{1-\varepsilon})$

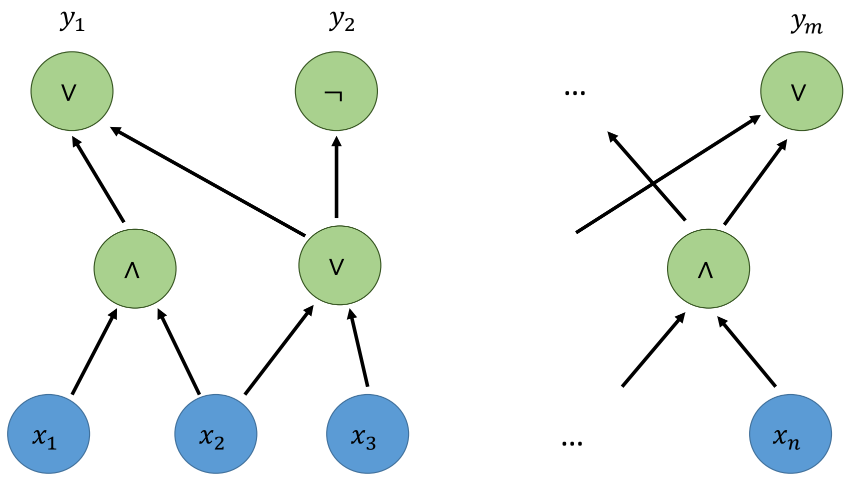
2. At the other extreme, we also have that for every $\varepsilon > 0$, there exists $\delta > 0$ such that $\mathbf{NTIME}(n) \not\subseteq \mathbf{TISP}(n^{2\cos(\pi/7)-\varepsilon}, n^{\delta})$, where $2\cos(\pi/7) \approx 1.8019$. This record is due to Williams from around 2007, and was discovered by computer search. Moreover, this search provides evidence that this bound is optimal for "alternation-trading" proofs.

3 Boolean Circuits

A Boolean circuit is a directed, acyclic graph with

- n sources representing inputs,
- m sinks representing outputs,
- non-input vertices ("gates") labeled by $\vee$, $\wedge$, or $\neg$,

- fan-in (in-degree) and fan-out (out-degree) of 1 or 2 on gates.



To evaluate a circuit on an input $x \in \{0,1\}^n$, evaluate the intermediate gates recursively until values are derived at the output gates.

A circuit defines a function $C_n : \{0,1\}^n \to \{0,1\}^m$. Its size, denoted $|C_n|$, is the number of vertices.

Definition 6. A $T(n)$-size circuit family is an infinite sequence of circuits $C = \{C_n\}_{n=1}^{\infty}$ such that $|C_n| \leq T(n)$ for every n.

We say that C decides a language L if for every n and every $x \in \{0,1\}^n$, we have $x \in L \iff C_n(x) = 1$.

Some motivation for studying circuits:

- Circuits more closely model computer hardware (silicon chips) than TMs, and also turn out to be useful for modeling parallel computation.
- It's often easier to reason about circuits than it is to reason about TMs. For example, much of the power of the Cook-Levin Theorem comes from how it translates questions about arbitrary NTMs into questions about CNF formulas (a restricted class of circuits).
- There's a close connection between circuit complexity and oracle complexity, based on fruitful analogies between TM classes and classes of circuits (e.g., $\mathbf{NP} \approx$ DNF, $\mathbf{coNP} \approx$ CNF, $\mathbf{PH} \approx \mathbf{AC}^0$). Much of what we know about oracle classes comes from circuit complexity. And much of what we know about circuit complexity comes from asking questions about oracles.