Notes on Complexity Theory Last updated: September, 2011

Lecture 8

Jonathan Katz

1 The Polynomial Hierarchy

We have seen the classes $\mathcal{NP}$ and $\mathsf{co}\mathcal{NP}$, which are defined as follows:

$L \in \mathcal{NP}$ if there is a (deterministic) Turing machine M running in time polynomial in its first input, such that

$$x \in L \Leftrightarrow \exists w\, M(x, w) = 1.$$

$L \in \mathsf{co}\mathcal{NP}$ if there is a (deterministic) Turing machine M running in time polynomial in its first input, such that

$$x \in L \Leftrightarrow \forall w\, M(x, w) = 1.$$

It is natural to generalize the above; doing so allows us to capture problems that are "more difficult" than $\mathcal{NP} \cup \mathsf{co}\mathcal{NP}$. As an example, consider the language IND-SET:

$$\text{IND-SET} \stackrel{\text{def}}{=} \{(G, k) : G \text{ has an independent set of size} \geq k\}.$$

We know that IND-SET $\in \mathcal{NP}$; a certificate is just an independent set of vertices of size at least k (that the vertices are independent can easily be verified). How about the following language?

$$\text{MAX-IND-SET} \stackrel{\text{def}}{=} \{(G, k) : \text{the largest independent set in } G \text{ has size exactly } k\}.$$

This language does not appear to be in $\mathcal{NP}$: we can certify that some graph G has an independent set of size k, but how do we certify that this is the *largest* independent set in G? The language does not appear to be in $\mathsf{co}\mathcal{NP}$, either: although we could prove that $(G, k) \notin$ MAX-IND-SET if G happened to have an independent set of size *larger* than k, there is no easy way to prove that $(G, k) \notin$ MAX-IND-SET in case its largest independent set has size *smaller* than k.

As another example, consider the problem of CNF-formula minimization. A CNF formula ϕ on n variables naturally defines a function $f_\phi : \{0,1\}^n \to \{0,1\}$, where $f_\phi(x) = 1$ iff the given assignment x satisfies ϕ. Can we tell when a given formula is minimal? Consider the language

$$\text{MIN-CNF} \stackrel{\text{def}}{=} \{\phi : \text{no formula with fewer clauses than } \phi \text{ computes } f_\phi\}.$$

This language does not appear to be in $\mathcal{NP}$ or $\mathsf{co}\mathcal{NP}$, either. (Even if I give you a smaller formula ϕ' that computes f_ϕ, there is no obvious way for you to verify that fact efficiently.)

The above examples motivate the following definition:

Definition 1 *Let i be a positive integer. $L \in \Sigma_i$ if there is a (deterministic) Turing machine M running in time polynomial in its first input, such that*

$$x \in L \Leftrightarrow \underbrace{\exists w_1 \forall w_2 \cdots Q_i w_i}_{i \text{ times}}\, M(x, w_1, \ldots, w_i) = 1.$$

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

$L \in \Pi_i$ if there is a (deterministic) Turing machine M running in time polynomial in its first input, such that

$$x \in L \Leftrightarrow \underbrace{\forall w_1 \exists w_2 \cdots Q_i w_i}_{i \text{ times}} M(x, w_1, \ldots, w_i) = 1.$$

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

As in the case of $\mathcal{NP}$, we may assume without loss of generality that the w_i each have length polynomial in x. Note also the similarity to TQBF. The crucial difference is that TQBF allows an *unbounded* number of alternating quantifiers, whereas Σ_i, Π_i each allow (at most) i quantifiers. Since TQBF is PSPACE-complete, this implies that $\Sigma_i, \Pi_i \in$ PSPACE for all i. (One could also prove this directly, via a PSPACE algorithm just like the one used to solve TQBF.)

Returning to our examples, note that MAX-IND-SET $\in \Sigma_2$ since $(G, k) \in$ MAX-IND-SET iff there exists a set of vertices S such that for all sets of vertices S' the following (efficiently verifiable) predicate is true:

> $|S| = k$ and S is an independent set of G; moreover, either $|S'| \leq k$ or S' is not an independent set of G.

(It is easier to express the above in English as: "there exists a set S of k vertices, such that for all sets S' containing more than k vertices, S is an independent set and S' is not an independent set". But one has to be careful to check that anytime the quantification is limited [e.g., by quantifying over all sets of size greater than k, rather than all sets], the limitation is efficient to verify.) We also have MAX-IND-SET $\in \Pi_2$ since we can swap the order of quantifiers in this case (since S' does not depend on S, above). Turning to the second example, note MIN-CNF $\in \Pi_2$ since $\phi \in$ MIN-CNF iff for all formulas ϕ' that are smaller than ϕ there exists an input x for which $\phi(x) \neq \phi(x')$. Here, however, we cannot just swap the order of quantifiers (do you see why?), and so we do not know, or believe, that MIN-CNF $\in \Sigma_2$.

We make some relatively straightforward observations, leaving their proofs as exercises.

- $\Sigma_1 = \mathcal{NP}$ and $\Pi_1 = \text{co}\mathcal{NP}$.
- For all i, we have $\text{co}\Sigma_i = \Pi_i$ (and $\text{co}\Pi_i = \Sigma_i$).
- For all i, we have $\Sigma_i, \Pi_i \subseteq \Pi_{i+1}$ and $\Sigma_i, \Pi_i \subseteq \Sigma_{i+1}$.

The *polynomial hierarchy* PH consists of all those languages of the form defined above; that is, $\text{PH} \stackrel{\text{def}}{=} \bigcup_i \Sigma_i = \bigcup_i \Pi_i$. Since $\Sigma_i \subseteq$ PSPACE for all i, we have PH $\subseteq$ PSPACE. Each Σ_i (resp., Π_i) is called a *level* in the hierarchy.

As in the case of $\mathcal{NP}$ and $\text{co}\mathcal{NP}$, we believe

Conjecture 1 *$\Sigma_i \neq \Pi_i$ for all i.*

If the above conjecture is not true, then the polynomial hierarchy *collapses* in the following sense:

Theorem 2 *If $\Sigma_i = \Pi_i$ for some i, then* PH $= \Sigma_i$.

Proof We show that for all $j > i$, we have $\Sigma_j = \Sigma_{j-1}$. Let $L \in \Sigma_j$. So there is a polynomial-time Turing machine M such that

$$x \in L \Leftrightarrow \exists w_j \forall w_{j-1} \cdots Q w_1 \; M(x, w_j, \ldots, w_1) = 1,$$

where we have numbered the variables in descending order for clarity in what follows. Assume $j - i$ is even. (A symmetric argument works if $j - i$ is odd.) Define language L' as

$$(x, w_j, \ldots, w_{i+1}) \in L' \Leftrightarrow \exists w_i \forall w_{i-1} \cdots Q w_1 \; M(x, w_j, \ldots, w_1) = 1,$$

and note that $L' \in \Sigma_i$. By the assumption of the theorem, $L' \in \Pi_i$ and so there is some machine M' running in time polynomial in $|x| + |w_j| + \cdots + |w_{i+1}|$ (and hence polynomial in $|x|$) such that

$$(x, w_j, \ldots, w_{i+1}) \in L' \Leftrightarrow \forall w'_i \exists w'_{i-1} \cdots Q' w'_1 \; M'(x, w_j, \ldots, w_{i+1}, w'_i, \ldots, w'_1) = 1.$$

(Note that the $\{w'_i, \ldots, w'_1\}$ need not have any relation with the $\{w_i, \ldots, w_1\}$.) But then

$$\begin{aligned} x \in L &\Leftrightarrow \exists w_j \forall w_{j-1} \cdots \forall w_{i+1} \;\; (x, w_j, \ldots, w_{i+1}) \in L' \\ &\Leftrightarrow \exists w_j \forall w_{j-1} \cdots \forall w_{i+1} \left[\forall w'_i \exists w'_{i-1} \cdots Q' w'_1 \;\; M'(x, w_j, \ldots, w'_1) = 1 \right] \\ &\Leftrightarrow \exists w_j \forall w_{j-1} \cdots \forall w_{i+1} w'_i \exists w'_{i-1} \cdots Q' w'_1 \;\; M'(x, w_j, \ldots, w'_1) = 1, \end{aligned}$$

and there are only $j - 1$ quantifiers in the final expression. Thus, $L \in \Sigma_{j-1}$. ■

A similar argument gives:

Theorem 3 *If $\mathcal{P} = \mathcal{NP}$ then $\mathsf{PH} = \mathcal{P}$.*

Each Σ_i has the complete problem Σ_i-SAT, where this language contains all true expressions of the form $\exists w_1 \forall w_2 \cdots Q w_i \; \phi(w_1, \ldots, w_i) = 1$ for ϕ a boolean formula in CNF form. However, if PH has a complete problem, the polynomial hierarchy collapses: If L were PH-complete then $L \in \Sigma_i$ for some i; but then every language $L' \in \Sigma_{i+1} \subseteq \mathsf{PH}$ would be reducible to L, implying $\Sigma_{i+1} = \Sigma_i$. Since PSPACE has complete languages, this indicates that PH is strictly contained in PSPACE.

1.1 Alternating Turing Machines

The polynomial hierarchy can also be defined using *alternating Turing machines*. An alternating Turing machine (ATM) is a non-deterministic Turing machine (so it has two transition functions, one of which is non-deterministically chosen in each step of its computation) in which every state is labeled with either $\exists$ or $\forall$. To determine whether a given ATM M accepts an input x, we look at the configuration graph of $M(x)$ and recursively mark the configurations of $M(x)$ as follows:

- The accepting configuration is marked "accept".
- A configuration whose state is labeled $\exists$ that has an edge to a configuration marked "accept" is itself marked "accept".
- A configuration whose state is labeled $\forall$, both of whose edges are to configurations marked "accept," is itself marked "accept".

$M(x)$ accepts iff the initial configuration of $M(x)$ is marked "accept".

We can define classes $\text{ATIME}(t(n))$ and $\text{ASPACE}(s(n))$ in the natural way. We can also define $\Sigma_i\text{TIME}(t(n))$ (resp., $\Pi_i\text{TIME}(t(n))$) to be the class of languages accepted by an ATM running in time $O(t(n))$, whose initial state is labeled $\exists$ (resp., $\forall$), and that makes at most $i-1$ alternations between states labeled $\exists$ and states labeled $\forall$. Note that $L \in \Sigma_2\text{TIME}(t(n))$ iff

$$x \in L \Leftrightarrow \exists w_1 \forall w_2 \; M(x, w_1, w_2),$$

where M is a deterministic Turing machine running in time $t(|x|)$. (The situation for space-bounded ATMs is tricker, since the length of the certificates themselves must be accounted for.) Given the definitions, it should not be surprising that $\Sigma_i = \bigcup_c \Sigma_i\text{TIME}(n^c)$ and $\Pi_i = \bigcup_c \Pi_i\text{TIME}(n^c)$. We also have $\mathsf{PSPACE} = \bigcup_c \text{ATIME}(n^c)$ (this follows readily from the fact that TQBF is PSPACE-complete).

Defining the polynomial hierarchy in terms of ATMs may seem more cumbersome, but has some advantages (besides providing another perspective). Perhaps the main advantage is that it allows for studying more refined notions of complexity, e.g., $\Sigma_i\text{TIME}(n^2)$ vs. $\Sigma_i\text{TIME}(n)$.

It is known that

Theorem 4

- $\text{NSPACE}(s(n)) \subseteq \text{ATIME}(s(n)^2) \subseteq \text{SPACE}(s(n)^2)$ *for time/space-constructible* $s(n) \geq n$.
- $\text{ASPACE}(s(n)) = \text{TIME}(2^{O(s(n))})$ *for time/space-constructible* $s(n) \geq \log n$.

Proof See [2]. ■

We now show an application of ATMs to proving a result about standard non-deterministic machines. Let $\text{TIMESPC}(t(n), s(n))$ be the class of languages that can be decided by a (deterministic) Turing machine M that runs in time $O(t(n))$ and uses space $O(s(n))$. (Note that $\text{TIMESPC}(t(n), s(n))$ is not the same as $\text{TIME}(t(n)) \cap \text{SPACE}(s(n))$.)

Theorem 5 $\mathsf{SAT} \notin \text{TIMESPC}(n^{1.1}, n^{0.1})$.

Proof We show that $\text{NTIME}(n) \not\subseteq \text{TIMESPC}(n^{1.2}, n^{0.2})$, which implies the theorem because (by the strong version of the Cook-Levin theorem proved in [1]) deciding membership of $x \in \{0,1\}^n$ in some language $L \in \text{NTIME}(n)$ can be reduced to solving SAT on an instance of length $O(n \log n)$.

We begin with the following claim (which is of independent interest).

Claim 6 $\text{TIMESPC}(n^{12}, n^2) \subseteq \Sigma_2\text{TIME}(n^8)$.

Proof Let $L \in \text{TIMESPC}(n^{12}, n^2)$, and let M be a machine deciding L in time $O(n^{12})$ and space $O(n^2)$. Considering the configuration graph of $M(x)$, we see that $M(x)$ accepts iff there exist a sequence of configurations $c_0, \ldots, c_{n^6}$ such that (1) c_0 is the initial configuration and c_{n^6} is the (unique) accepting configuration, and (2) for all i, machine $M(x)$ goes from configuration c_i to configuration c_{i+1} in at most $O(n^6)$ steps. Each configuration can be specified using $O(n^2)$ bits, and so the sequence $c_1, \ldots, c_{n^6}$ can be written down in $O(n^8)$ time. Moreover, existence of a path of length $O(n^6)$ from c_i to c_{i+1} can be verified in $O(n^8)$ time. ■

The above claim was unconditional. The next claim uses the assumption that $\text{NTIME}(n) \subseteq \text{TIMESPC}(n^{1.2}, n^{0.2})$.

Claim 7 *If* $\text{NTIME}(n) \subseteq \text{TIMESPC}(n^{1.2}, n^{0.2})$*, then* $\Sigma_2\text{TIME}(n^8) \subseteq \text{NTIME}(n^{9.6})$.

Proof Since $\text{TIMESPC}(n^{1.2}, n^{0.2}) \subseteq \text{TIME}(n^{1.2})$ and $\text{TIME}(s(n))$ is closed under complementation, the condition of the claim implies $\text{coNTIME}(n) \subseteq \text{TIME}(n^{1.2})$. If $L \in \Sigma_2\text{TIME}(n^8)$ then

$$x \in L \Leftrightarrow \exists w_1 \forall w_2 \; M(x, w_1, w_2) = 1,$$

where M is a deterministic machine running in time $O(|x|^8)$. But then

$$L' \stackrel{\text{def}}{=} \{(x, w_1) : \forall w_2 \; M(x, w_1, w_2)\} \subseteq \text{coNTIME}(n^8) \subseteq \text{TIME}\left(\left(n^8\right)^{1.2}\right) = \text{TIME}(n^{9.6})$$

where n denotes the length of x. If we let M' be a deterministic machine deciding L', where $M(x, w_1)$ runs in time $O(|x|^{9.6})$, then

$$x \in L \Leftrightarrow \exists w_1 \; M'(x, w_1) = 1$$

and hence $L \in \text{NTIME}(n^{9.6})$. ■

We now put everything together to prove the theorem. Assume toward a contradiction that $\text{NTIME}(n) \subseteq \text{TIMESPC}(n^{1.2}, n^{0.2})$. Then

$$\text{NTIME}(n^{10}) \subseteq \text{TIMESPC}(n^{12}, n^2) \subseteq \Sigma_2\text{TIME}(n^8) \subseteq \text{NTIME}(n^{9.6}),$$

using Claim 6 for the second inclusion, and Claim 7 for the last inclusion. But this contradicts the non-deterministic time-hierarchy theorem. ■

References

[1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[2] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *J. ACM* 28(1): 114–133, 1981.