CSE200: Complexity theory
Nondeterminism

Shachar Lovett

December 9, 2019

1 Nondeterministic TM, space and time complexity

Recall that NP is the class of languages, for which a "solution" can be verified in polynomial time. More generally, we can define nondeterministic time classes for any time bound. We first need to define what is a nondeterministic Turing machine (NTM).

Definition 1.1 (Nondeterministic Turing machine)**.** *A nondeterministic Turing machine is the same as a standard TM, except that it has two transition functions δ_1, δ_2. At each step, it is allowed to "guess" which one to use. This defines many possible "pathes" of computation (for t steps, it has 2^t possible pathes). An NTM accepts an input x if*

(i) It terminates on all pathes of the computation.

(ii) There exists a path on which it reaches the accept state.

Definition 1.2 (Nondeterministic time complexity)**.** *Let $T : \mathbb{N} \to \mathbb{N}$ be a time bound. A language L is in nondeterministic time T if there exists a NTM M that computes L, such that on input x, all branches of computation of M terminate in at most $O(T(|x|))$ many steps. The class NTIME(T) is the class of all such languages.*

Definition 1.3 (Nondeterministic space complexity)**.** *Let $S : \mathbb{N} \to \mathbb{N}$ be a space bound. A language L is in nondeterministic space S if there exists a NTM M that computes L, such that on input x, all branches of computation of M terminate and used at most $O(S(|x|))$ space. The class NSPACE(S) is the class of all such languages.*

We would discuss nondeterministic space in detail when we dive into space complexity. For now, lets focus on nondeterministic time.

Lemma 1.4. *$NP = \cup_{c \geq 1} NTIME(n^c)$.*

Proof. We defined NP as the class of languages that can be verified in polynomial time. We need to show that this is equivalent to the existence of a poly-time NTM that computes L. In one direction, assume L is computed by a NTM M. Then the witness for input x will be

the sequence of steps that M makes that accept x. Concretely, if M makes $t \leq O(T(|x|))$ steps then $y \in \{0,1\}^t$. Clearly, one can verify that M accepts x on the path described by y in poly-time. For the other direction, assume that there exists a poly-time TM M' such that $x \in L \iff \exists y, |y| \leq |x|^{O(1)}, M'(x,y) = 1$. We can construct a NTM M that first guesses y and writes it down, and then runs M' on x, y. □

2 The class coNP

The class NP corresponds to problems which can be verified efficiently .The class coNP is the opposite - problems which can be refuted efficiently. Formally, for a language $L \subset \{0,1\}^*$ let $L^c = \{x \in \{0,1\}^* : x \notin L\}$ be the complement. Then

$$\text{coNP} = \{L : L^c \in \text{NP}\}.$$

For example,

$$\text{UNSAT} = \{\text{CNF formulas with no satisfying assignment}\}$$

is in coNP.

Definition 2.1 (coNP-hard anc coNP-complete languages)**.** *A language L is coNP-hard if for any $L' \in coNP$ it holds that $L' <_p L$. If in addition $L \in coNP$ then L is said to be coNP-complete.*

Theorem 2.2. *UNSAT is coNP-complete.*

Proof. Fix some efficient encoding of CNF formulas in $\{0,1\}^*$ which is onto. Then UNSAT $=$ SAT^c. If L is in coNP then $L^c \in$ NP, hence $L^c <_p$ SAT, hence $L <_p$ UNSAT. □

3 The polynomial hierarchy

Let $\mathcal{C}$ be a family of languages (like P, NP, EXP). We define $\exists_p \mathcal{C}$ to be the set of languages L for which there exists $c > 0$ and a language $L' \in \mathcal{C}$ such that

$$x \in L \iff \exists y, |y| \leq |x|^c, (x,y) \in L'.$$

Define $\forall_p \mathcal{C}$ to be the set of languages L for which there exists $c > 0$ and a language $L' \in \mathcal{C}$ such that

$$x \in L \iff \forall y, |y| \leq |x|^c, (x,y) \in L'.$$

We would often shorthand these as $\exists \mathcal{C} = \exists_p \mathcal{C}$ and $\forall \mathcal{C} = \forall_p \mathcal{C}$.

Lemma 3.1. *NP $= \exists P$ and coNP $= \forall P$.*

Proof. This is by definition. A language L is in NP if there exists a poly-time Turing machine V such that

$$x \in L \iff V(x, y) = 1.$$

Assume V runs in time n^c. Consider the language $L' = \{(x, y) : V(x, y) = 1\}$. It is in P since V is poly-time computable. Hence

$$x \in L \iff \exists y, |y| \leq |x|^c, (x, y) \in L'.$$

Similarly, if such an L' exists we can define V to be the Turing machine deciding L'. □

This allows us to define a hierarchy of complexity classes. Define

$$\Sigma_2 = \exists\forall \mathrm{P}, \quad \Pi_2 = \forall\exists \mathrm{P}$$

and generally

$$\Sigma_i = \exists\Pi_{i-1}, \quad \Pi_i = \forall\Sigma_{i-1}.$$

For example, $L \in \Sigma_3$ if

$$L = \{x : \exists y \forall z \exists w, M(x, y, z, w) = 1\}$$

where M is poly-time computable. We define the polynomial hierarchy as

$$\mathrm{PH} = \cup_{i \geq 1} \Sigma_i = \cup_{i \geq 1} \Pi_i.$$

As an example to problems that seem to be outside NP or coNP, but in PH, consider the problem of formula minimization - given a CNF formula, find a minimal formula equivalent to it. We can define it via the language:

$$\text{MIN-CNF} = \{(\varphi, 1^k) : \varphi \text{ is a CNF, and there is a CNF } \psi \text{ of size } \leq k \text{ computing } \varphi\}$$

Claim 3.2. *MIN-CNF is in* Σ_2.

Proof. To verify that $(\varphi, 1^k) \in$ MIN-CNF we need to provide a CNF formula ψ of size $\leq k$ such that $\psi \equiv \varphi$. To verify this we need to check all inputs. Hence

$$(\varphi, 1^k) \in MINCNF \iff \exists \psi (|\psi| \leq k) \forall x, \varphi(x) = \psi(x).$$

□

4 Collapses: what if P=NP?

We believe that P $\neq$ NP and NP $\neq$ coNP, and moreover that all the classes Σ_i, Π_i are distinct. The following theorem gives some intuition why we believe so. At a high level, if P=NP then the polynomial hierarchy collapses.

Theorem 4.1. *If $P = NP$ then $PH = P$.*

Proof. Assume P = NP. We will prove by induction on i that $\Sigma_i, \Pi_i = \mathrm{P}$.

The base case if $i = 1$. By assumption we have $\Sigma_1 = \mathrm{NP} = \mathrm{P}$. Recall that $\Pi_1 = \mathrm{coNP}$ is the class of languages L for which $L^c \in \mathrm{NP}$. But if NP = P then we can compute L^c in poly-time, and hence compute L in poly-time. So also $\Pi_1 = \mathrm{P}$.

Assume we proved the theorem for i and we want to prove it for $i + 1$. We will prove it for Σ_{i+1}, where the proof for Π_{i+1} is analogous. To recall, a language $L \in \Sigma_{i+1}$ if there exists a language $L' \in \Pi_i$ and $c > 0$ such that

$$x \in L \iff \exists y, |y| \leq |x|^c, (x, y) \in L'.$$

By the induction hypothesis, $L' \in \mathrm{P}$. So there exists a poly-time TM M' computing it. But then $L \in \mathrm{NP}$, which we assume equals P. So we proved that $\Sigma_{i+1} = \mathrm{P}$. □

We get similar collapses of PH if other classes collapse.

Theorem 4.2. *If NP = coNP then PH = NP.*

Proof. We will prove that $\Sigma_i = \Pi_i = \mathrm{NP}$ for all $i \geq 1$. The base case of $i = 1$ is our assumption. Assume by induction we proved it for i, and we want to prove it for $i + 1$. We will prove it for Σ_{i+1}, where the proof for Π_{i+1} is analogous.

Let $L \in \Sigma_{i+1}$. By definition, there exists a language $L' \in \Pi_i$ and $c > 0$ such that

$$x \in L \iff \exists y, |y| \leq |x|^c, (x, y) \in L'.$$

By induction, $L' \in \mathrm{NP}$. So there exists a language $L'' \in \mathrm{P}$ and $c' > 0$ such that

$$(x, y) \in L' \iff \exists z, |z| \leq |(x, y)|^{c'}, (x, y, z) \in L''.$$

Combining these together gives (for $c'' = c \cdot c'$) that

$$x \in L \iff \exists y, z, |y| \leq |x|^c, |z| \leq |x|^{c''}, (x, y, z) \in L''.$$

We can view (y, z) as a single "solution" that the verifier L'' checks. As $L'' \in \mathrm{P}$ we get that $L \in \mathrm{NP}$, as claimed. □

More generally, following the same proof idea we get the following theorem.

Theorem 4.3. *For any $i \geq 1$, if $\Sigma_i = \Pi_i$ then PH = Σ_i.*