

# Computational Complexity Theory - Course Notes

September 29, 2006

## 1 Previous lecture

### 1.1 The polynomial hierarchy (PH)

We defined the *polynomial hierarchy*  $PH = \cup_{i \geq 0} \Sigma_i^p$  and noted the following properties:

- $\Sigma_0^p = \Pi_0^p = P$
- $\Sigma_1^p = NP$
- $\Pi_1^p = coNP$
- $\Sigma_{i+1}^p = NP \Sigma_i^p$  for  $i \geq 1$
- $\Pi_{i+1}^p = coNP \Sigma_i^p$  for  $i \geq 1$

### 1.2 Characterizations

First characterization:

$$\Sigma_i^p = \left\{ L \mid \begin{array}{l} \text{language } L \text{ is decided by an ATM running in polynomial-time with } i \text{ half-moves} \\ \text{(i.e. } i - 1 \text{ alternations) with player E (player 1) making the first move} \end{array} \right\}$$

$$\Pi_i^p = \left\{ L \mid \begin{array}{l} \text{language } L \text{ is decided by an ATM running in polynomial-time with } i \text{ half-moves} \\ \text{(i.e. } i - 1 \text{ alternations) with player A (player 2) making the first move} \end{array} \right\}$$

and second characterization:

$$L \in \Sigma_i^p \iff \forall x \in L : \exists y : \langle x, y \rangle \in L' \in \Pi_{i-1}^p \text{ where } |y| \leq p(|x|)$$

$$L \in \Pi_i^p \iff \forall x \in L : \forall y : \langle x, y \rangle \in L' \in \Sigma_{i-1}^p \text{ where } |y| \leq p(|x|).$$

where it is seen that languages in, say  $\Sigma_2^p$ , can be expressed on the form “ $\exists x \forall y \dots$ ”:

$$x \in L \iff \{x \mid \exists y : \forall z : \text{ such that TM } M(x, y, z) = 1\}$$

and similarly languages in, say  $\Pi_2^p$  can be expressed on the form “ $\forall x \exists y \dots$ ”:

$$x \in L \iff \{x \mid \forall y : \exists z : \text{ such that TM } M(x, y, z) = 1\}$$

## 2 Fragility of the polynomial hierarchy

**Proposition 1.** For  $i \geq 0$  we have

1.  $\Sigma_i^p \subseteq \Sigma_{i+1}^p$
2.  $\Pi_i^p \subseteq \Sigma_{i+1}^p$
3. If  $\Sigma_j^p = \Sigma_i^p$  then  $\Pi_j^p = \Sigma_i^p$  for all  $j \geq i$ .

**Theorem 2.** If  $\Sigma_i^p = \Pi_i^p$  for some  $i$  then  $\Sigma_j^p = \Pi_j^p = \Sigma_i^p$  for all  $j \geq i$ , i.e. the polynomial hierarchy collapses to the  $i$ th level:  $PH = \Sigma_i^p$ .

*Proof.* By Proposition 1 it is sufficient to show  $\sum_j^p = \sum_i^p$  for  $j \geq i$ . Doing induction in  $j$  we assume  $\sum_i^p = \prod_i^p$  and show  $\sum_{i+1}^p = \sum_i^p$ . By Proposition 1 it is enough to show  $\sum_{i+1}^p \subseteq \sum_i^p$ .

We use the second characterization of  $\sum_i^p$ :

$$x \in L \iff \exists y : \langle x, y \rangle \in L' \text{ where } |y| \leq p(|x|).$$

for  $L \in \sum_{i+1}^p$  and  $L' \in \prod_i^p$ . But by assumption  $\prod_i^p = \sum_i^p$  and so  $L' \in \sum_i^p$ . Again, using the second characterization:

$$\langle x, y \rangle \in L' \iff \exists z : \langle \langle x, y \rangle, z \rangle \in L'' \text{ where } |z| \leq q(|\langle x, y \rangle|)$$

for  $L'' \in \prod_{i-1}^p$ . Combining these results we get:

$$\begin{aligned} x \in L &\iff \exists y : \langle x, y \rangle \in L' \\ &\iff \exists y : \exists z : \langle \langle x, y \rangle, z \rangle \in L'' \\ &\iff \exists (y, z) : \langle x, y, z \rangle \in L'' \end{aligned}$$

and hence  $L \in \sum_i^p$  as required (note that the length of  $(y, z)$  is at most polynomial since the composition of the two polynomials  $p$  and  $q$  is a polynomial).  $\square$

**Corollary 3.** If  $NP = coNP$  then  $NP = PH$ . If  $P = NP$  then  $P = PH$ .

*Proof.* For the first part we have from Section 1.1 that  $NP = \sum_1^p$  and  $coNP = \prod_1^p$ . The second part follows from the first part in that if  $P = NP$  then  $NP = coNP$  (since  $P$  is closed under complementation).  $\square$

**Corollary 4.** If  $\sum_j^p \neq \prod_j^p$  for some  $j$  then  $\sum_i^p \neq \prod_i^p$  for all  $i \leq j$ .

**Remark 5.** If  $NP \neq coNP$  then *FACTORING* is not NP-complete (even relative to Cook-reductions). If *PH* does not collapse then *GRAPH ISOMORPHISM* is not NP-complete.

As hinted, we have as a hypothesis that the hierarchie does not collapse.

### 3 PH and small circuits for NP-complete languages

**Theorem 6** (Karp-Lipton). If  $NP \subseteq PSIZE$  then  $\sum_3^p \subseteq \sum_2^p$ .

*Proof.* Given a language  $L \in \sum_3^p$  we construct a  $\sum_2^p$  game deciding  $L$ . We first reduce  $L$  to a QBF formula with three quantifiers and use the polynomial sized circuits for *SAT* to solve this formula.

On input  $x$ :

- Player 1 writes down circuits  $C_1, C_2, \dots, C_{p(|x|)}$  solving *SAT* with size polynomial in input  $x$ ; loses if he cannot do that. This step corresponds to the  $\exists$ -part of the  $\sum_2^p$  language.
- If one of these circuits are not correct, Player 2 must prove this using *CORRECT-CIRCUIT-FOR-SAT* (which is in *coNP*); he wins if he can. This step corresponds to the  $\forall$ -part of the  $\sum_2^p$  language.

Using the (correct) circuits for *SAT*, a deterministic polynomial time TM can now evaluate the QBF formula by eliminating the three blocks of quantifiers as done in Fortnow's theorem. Assuming both play optimal, the TM declare Player 1 as the winner if the QBF formula is true and Player 2 as the winner otherwise.  $\square$

### 4 Relativizable proof techniques

**Definition 7.** The relativized complexity classes

$$\begin{aligned} P^A &= \{L \mid L \text{ is decided by } D^A \text{ in polynomial time}\} \\ NP^A &= \{L \mid L \text{ is decided by } N^A \text{ in polynomial time}\} \end{aligned}$$

where  $D^A$  is a deterministic TM with access to an oracle  $A$  and  $NP^A$  is a similar nondeterministic TM.

We say that a proof technique relativizes if it works with oracles, e.g.  $DTIME^A(t) \subsetneq DTIME(pt)$ ,  $P^A \subsetneq EXP^A$  and  $NP^A \subsetneq P^A/poly$  (if  $PH^A$  does not collapse). However, as shown below, it cannot be used to settle the  $P$  vs.  $NP$  question.

**Theorem 8.** There exist oracles  $A$  and  $B$  such that  $P^A = NP^A$  and  $P^B \neq NP^B$ .

*Proof.* For oracle  $A$  we want an oracle from a place where the (presumable) power  $NP$  has over  $P$  is of no use. Selecting  $A = HALTING$  or  $A = QBF$  we have such an oracle, where the latter holds because  $QBF$  is  $PSPACE$ -complete and hence can simulate the execution of any nondeterministic TM.

The proof for oracle  $B$  is a bit more tricky. The idea is to pick an oracle that amplifies the differences between  $P$  and  $NP$ , i.e. an oracle that takes full advantage of nondeterministic TMs ability to guess, while utilizing that deterministic TMs in  $P$  have polynomial bounded running time.

We pick the oracle  $B$  at random and show that the probability of  $P^B \neq NP^B$  is 1. By picking  $B$  at random the TM has no way (negligible probability) of being able to decide  $B$  without using the oracle.

Given  $B$  we can construct the function  $f_B : \{0,1\}^n \rightarrow \{0,1\}^n$  and so having  $B$  is the same as having  $f_B$ . We define the language  $L_B = \{x|x \in \{0,1\}^n \text{ and there is a } y \text{ such that } f_B(y) = 0^n\}$ , which is in  $NP^B$  as a nondeterministic TM can guess  $y$  and ask the oracle if  $f_B(y) = 0^n$ . However, for random oracle,  $L_B \notin P^B$  with probability 1: the deterministic TM  $D$  trying to decide  $L_B$  runs in polynomial time and hence can only ask at most polynomial questions to the oracle (with each answer equally likely). If  $n$  is large enough the probability of  $D$  deciding  $L_B$  goes to zero.  $\square$

This result gave birth to the *random oracle hypothesis*:  $\mathcal{C}_1^A \neq \mathcal{C}_2^A$  if and only if  $\mathcal{C}_1 \neq \mathcal{C}_2$  for random oracle  $A$ .