

# Computational Complexity

September 27 2006

**ad previous lesson:** Fortnow's theorem was published in the paper *Time-Space Trade-Offs for Satisfiability* (Buhrman, Fortnow, '96). For some other reading material, see the new textbook by Sanjeev Arora and Boaz Barak at <http://www.cs.princeton.edu/theory/complexity/>.

## 1 Other Space-Time Trade-Offs

**Proposition 1.1.** (70')  $\text{TIME}(t) \subseteq \text{SIZE}(O(t \log t))$

**Corollary 1.2.**  $\text{SAT} \notin \text{NL} \cap \text{TIME}(n^{1+o(1)})$

**Corollary 1.3.**  $\text{SAT} \notin \text{TISP}(n^{1+o(1)}, O(\log n))$

**Theorem 1.4.** (Bryan Williams, 2005)  $\text{SAT} \notin \text{TISP}(n^{1.7..}, \text{polylog } n)$

**Theorem 1.5.** (60')  $\text{PALINDROME} \notin \text{TISP}(n^{2-\varepsilon}, \text{polylog } n)$  (but this holds only for TM model, not for RAM, unlike the previous claim)

## 2 Some (A)PSPACE-complete Problems

**Generalized geography game:** Given a directed graph  $G$ , and a pebble put in one of its vertices. Players 1 and 2 alternate in moves, and they should always move a pebble along a directed edge. Who closes a cycle (undirected), loses. Let  $(G, P)$  be an ordered pair defining the starting configuration (graph, pebble). Let

$$\text{GG} = \{(G, P) \mid \text{player 1 has a winning strategy}\}.$$

**Theorem 2.1.** *Deciding GG is a PSPACE-complete problem.*

*Proof.* Clearly  $\text{GG} \in \text{AP} = \text{PSPACE}$ . Let us prove that  $\text{QBF} \leq_L \text{GG}$ , thus GG is PSPACE-complete, since QBF is PSPACE complete. For any given perfectly alternating quantified boolean formula in the form

$$F = \forall x_1 \exists x_2 \dots \forall x_{2n-1} f(x_1, \dots, x_{2n-1}),$$

where  $f$  is a DNF formula, there's a mapping  $g(F) = G$  that assigns a gadget to every quantifier at the beginning. The sequence of gadgets we connect to the "star" that computes  $f$ . The leaves

of this star are labeled by terms. We connect every term to the gadgets corresponding to the literals contained in it, where we always choose the positive resp. negative vertex for the positive resp. negative literal. Playing the false/true game on  $F$ , player 1 wins if it is false. We show, that this happens if and only if player 2 is forced to close a cycle playing GG on  $G$ .

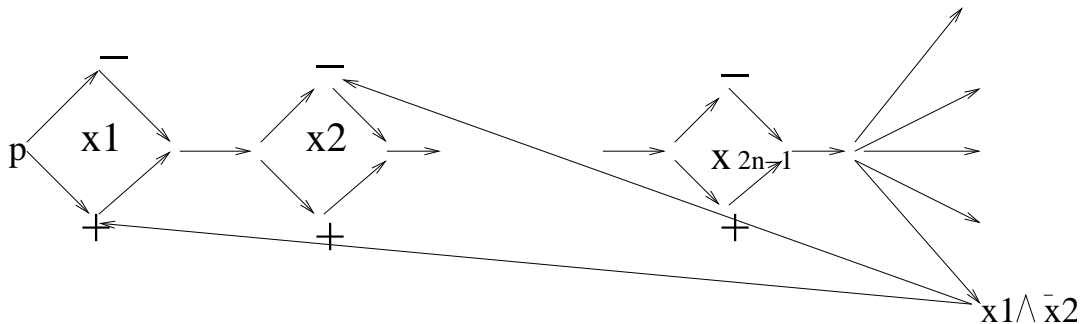


Figure 1:  $(G,P)$

If  $F$  is false (player 1 wins there), then since the first move (picking a term) in the star computing  $f$  is up to the second player, he certainly picks one, that is not true under the evaluation that was assigned to the variables while moving along the gadgets (since only then DNF is false). That means that this term contains a literal which is false. Let us denote it by  $u_i \in \{x_i, \bar{x}_i\}$ . Player 1 can now move the pebble along the edge, connecting this term with the  $i$ -th gadget, namely to the opposite vertex than the one contained in the path made so far, thus he doesn't close a cycle. Now player 2 is forced to close the cycle.

In the opposite direction, the existence of a winning strategy of player 1 on the given  $(G, P)$  proves that the corresponding QBF is false.  $\square$

**Proposition 2.2.**  $\text{SUCCINCT CIRCUIT VALUE} \leq_L \text{GEN. CHECKERS}$ , thus  $\text{GEN. CHECKERS}$  is  $\text{APSPACE=EXP-complete}$ .

### 3 Cook Reduction

**Definition 3.1.** Oracle Turing Machine (OTM) with an oracle  $A$  is a DTM with an extra tape - oracle tape, and three extra states: ASK, YES, NO. If it is in ASK state, it submits the whole input tape to the oracle, changes the state according to the answer, and erases its tape.

**Definition 3.2.**  $\text{DTIME}^A(t)$  is a class of languages decidable by OTM with an oracle  $A$  in polytime.

**Definition 3.3.**  $\text{P}^A = \bigcup_{p \text{ poly}} \text{DTIME}^A(p)$

**Definition 3.4.**  $L_1 \leq_{\text{cook}} L_2$  if  $L_1 \in \text{P}^{L_2}$ .

**Remark P** is closed under Cook reduction.

**Proposition 3.5.**  $\text{NP} \cap \text{coNP}$  is closed under Cook reduction.

*Proof.* Let  $L_1 \in \text{NP} \cap \text{coNP}$ . Then for any machine  $M^{L_1}$  running polynomially on input  $x$  guess the computation of the machine. Verify *yes* and *no* answers efficiently, using that  $L_1 \in \text{NP} \cap \text{coNP}$ . Thus, the language decided by  $M^{L_1}$  machine is in  $\text{NP} \cap \text{coNP}$ .  $\square$

**Corollary 3.6.** *If  $\text{NP} \neq \text{coNP}$ , then factoring is not NP-complete, even using Cook reduction.*

*Proof.* Then  $\text{NP} \cap \text{coNP}$  is closed under Cook reduction, while NP is not. Factoring is in  $\text{NP} \cap \text{coNP}$ .  $\square$

## 4 Polynomial Hierarchy

**Definition 4.1.** *If  $\mathcal{C}$  is a class of languages, then*

$$P^{\mathcal{C}} := \bigcup_{L \in \mathcal{C}} P^L.$$

**Definition 4.2.** (Polynomial Hierarchy)

$$\begin{aligned} \Delta_0^P &= \Sigma_0^P = \Pi_0^P = P \\ \Sigma_1^P &= \text{NP} \\ \Pi_1^P &= \text{coNP} \\ \Delta_{i+1}^P &= P^{\Sigma_i^P} \\ \Sigma_{i+1}^P &= \text{NP}^{\Sigma_i^P} \\ \Pi_{i+1}^P &= \text{coNP}^{\Sigma_i^P}, \quad i \geq 1 \end{aligned}$$

**Remarks**

$$\begin{aligned} \Sigma_1^P &= \text{NP} \\ \Pi_1^P &= \text{coNP} \\ \Pi_i^P &= \text{co}\Sigma_i^P \\ \Sigma_i^P &\subseteq \Delta_{i+1}^P \subseteq \Sigma_{i+1}^P \\ \Pi_i^P &\subseteq \Delta_{i+1}^P \subseteq \Pi_{i+1}^P \end{aligned}$$

**Proposition 4.3.**  $\bigcup_{i \in \mathbb{N}} \Delta_i^P \subseteq \text{PSPACE}$

*Proof.*  $\text{NP}^{\text{PSPACE}} = \text{PSPACE}$   $\square$

**Theorem 4.4.**

$$\begin{aligned} \Sigma_i^P &= \{L \mid L \text{ is decided by an ATM in polytime and } i \text{ halfmoves (player 1 begins)}\} \\ \Pi_i^P &= \{L \mid L \text{ is decided by an ATM in polytime and } i \text{ halfmoves (player 2 begins)}\} \end{aligned}$$

*Proof.*  $\supseteq$ : induction on  $i$

$\subseteq$ : induction on  $i$ , induction step goes as follows

?  $\text{NP}^{\text{NP}} \subseteq \{L \mid L \text{ is decided by an ATM in polytime and 2 moves, pl. 1 begins}\}$

player 1: writes down the transcript of  $\text{NP}^{\text{SAT}}$  on  $x$ , for every *yes* oracle-answer gives a satisfying assignment

player 2: checks oracle answers - for *no* by guessing, for *yes* according to the given assignment

That describes the game.  $\square$

**Another view**

$$\begin{aligned}\Sigma_i^{\text{P}} &= \{\underbrace{\exists \forall \dots}_i f\} \\ \Pi_i^{\text{P}} &= \{\underbrace{\forall \exists \dots}_i f\},\end{aligned}$$

where  $f$  is a formula in either CNF or DNF form according to the last quantifier.