

# Computational Complexity Theory - Course Notes

November 29, 2006

## 1 Previous Lecture

We defined the classes  $MA$  (*Merlin-Arthur*) and  $AM$  (*Arthur-Merlin*) as follows:

- A language  $L$  is in  $MA$  if and only if there exists a language  $L' \in P$  such that:

$$\begin{aligned}x \in L &\Rightarrow \exists^p y : \forall^p z : \langle x, y, z \rangle \in L' \\x \notin L &\Rightarrow R^p z : Pr[\exists^p y : \langle x, y, z \rangle \in L'] \leq \frac{1}{2}\end{aligned}$$

- A language  $L$  is in  $AM$  if and only if there exists a language  $L' \in P$  such that:

$$\begin{aligned}x \in L &\Rightarrow \forall^p z : \exists^p y : \langle x, y, z \rangle \in L' \\x \notin L &\Rightarrow R^p z : Pr[\exists^p y : \langle x, y, z \rangle \in L'] \leq \frac{1}{2}.\end{aligned}$$

where  $\exists^p y$  is short-hand for  $\exists y \in \{0, 1\}^p$  for some polynomial  $p$  and similar for  $\forall^p$  and  $R^p$ .

We observed that

$$NP \subseteq MA \subseteq AM \subseteq \prod_2^p \text{ and } MA \subseteq \sum_2^p$$

and similar to Meier's theorem (that  $PSPACE \subseteq P/poly \Rightarrow PSPACE = \sum_2^p$ ) we have

**Theorem 1.** If  $PSPACE \subseteq P/poly$  then  $PSPACE = MA$

*Proof.* Since  $MA \subseteq IP = PSPACE$  we only need to prove that  $PSPACE \subseteq P/poly$  implies  $PSPACE \subseteq MA$  and do so by giving a two-player game:

Given a *QBF* formula  $f$ , Player 1 presents a circuit encoding the computation of the optimal prover in Shamir's protocol ( $IP \subseteq PSPACE$ ) for this prover when given  $f$ . Thus, the input to the circuit is the conversation that have occurred between the prover and verifier up to a point, and the output is the prover's reply at that point of the protocol. Player 2 supplies a random bitstring, and the referee emulates Shamir's protocol using circuits (as prover) and the random bits.  $\square$

It is noted that in the above proof the prover doesn't have to participate when the referee decides and can be seen as some advanced alien life form that leaves a powerful circuit behind before leaving.

## 2 PCP Theorem

The PCP theorem tells us that mathematical proofs (i.e. *NP witnesses*) can be encoded in such a way that only a constant number of bits of the proof need to be read in order to be convinced of correctness. A probabilistically checkable proof system

$$PCP[r(n), q(n)]$$

is the set of languages where membership can be decided with only a small probability of false positives by a Turing machine  $V$  (*the verifier*) running in polynomial time, and using  $r(n)$  random bits, and reading only  $q(n)$  bits of the proof of length  $2^{r(n)}$ . These restrictions imply that we need a Turing machine model with random access to the tape, but instead of introducing a new model, we use oracle machines where the proof is given to the verifier as a (finite<sup>1</sup>) oracle, i.e. the TM asks questions of the form “what is bit 42 of the proof?”. One way to relate the three strings  $x$ ,  $y$  and  $z$ , is to say that it is expensive to access  $y$  as opposed to  $x$  and  $z$ .

More formally, a language  $L$  is in  $PCP[r(n), q(n)]$  if and only if there exists a polynomial time oracle Turing machine  $V$  such that

$$\begin{aligned} x \in L &\Rightarrow \exists y \in \{0, 1\}^{2^{r(n)}} : \forall z \in \{0, 1\}^{r(n)} : V^y \text{ accepts } \langle x, z \rangle \\ x \notin L &\Rightarrow \forall y \in \{0, 1\}^{2^{r(n)}} : Pr[V^y \text{ accepts } \langle x, z \rangle] \leq \frac{1}{2}. \end{aligned}$$

where  $V$  only makes  $q(n)$  queries to the oracle, i.e. it only reads  $q(n)$  bits of the proof  $y$ .

**Theorem 2.**  $PCP[r(n), q(n)] \subseteq NTIME(2^{O(r(n))})$

*Proof.* The non-deterministic Turing machine can guess  $y$  and then enumerate all bit strings  $z$  of length  $r(n)$  and verify that  $V^y$  accepts for all  $\langle x, z \rangle$ .  $\square$

**Theorem 3.**  $P \subseteq PCP[O(\log n), O(1)] \subseteq NP$

Actually, as we will show below, the *PCP-theorem* tells us that  $PCP[O(\log n), O(1)] = NP$ . It is noted that it is not trivial to find examples of languages in  $PCP[O(\log n), O(1)]$ .

*Proof.* For  $P \subseteq PCP[O(\log n), O(1)]$ , the verifier  $V$  can do all the computation by it-self since  $V \in P$  by definition. For  $PCP[O(\log n), O(1)] \subseteq NP$ , we have  $2^{\log n} = n^k$  for some  $k$ , and so the result follows from Theorem 2.  $\square$

**Theorem 4 (PCP Theorem).**  $NP = PCP[O(\log n), O(1)]$

The proof of Theorem 4 is rather long so instead we prove the following, less ambitious, result:

**Lemma 5.**  $NP \subseteq PCP[\text{polylog} n, \text{polylog} n]$

Note that I will only give a sketch of the proof here as the details can be found in the note by Peter Bro Miltersen. The proof uses arithmetization and hence follows the lines of the proof of Shamir’s theorem that  $IP = PSPACE$ .

*Proof.* It is enough to show that  $3SAT \in PCP[\text{polylog} n, \text{polylog} n]$ .

The proof is presented as an interactive proof and it is argued that the prover can be replaced with a finite string. This string will be a table of answers, indexed by any question the verifier could ever ask the prover, and thus, oracle access to the string can replace interaction with the prover, yielding the desired result.

The central part of the proof is that, given a formula  $f \in 3SAT$ , the statement *a satisfies f*, can be expressed by an arithmetic expression. If we assume, without loss of generality, that  $f$  has  $n$  variables and  $n$  clauses for  $n = 2^m$  we have *a satisfies f* if and only if

$$\sum_{c \in \{0, 1\}^m} \sum_{v_1, v_2, v_3 \in \{0, 1\}^m} \prod_{i=1}^3 h_i(c, v_i) \left( s_i(c) - a(v_i) \right)^2 = 0$$

---

<sup>1</sup>The oracle can only give an answer for a finite number of questions.

where  $a(\cdot)$ ,  $h_i(\cdot)$  and  $s_i(\cdot)$  are defined as

$$a : \{0, 1\}^m \rightarrow Z$$

and is the arithmetization of a satisfying assignment

$$h_i : \{0, 1\}^m \times \{0, 1\}^m \rightarrow Z$$

and  $h_i(c, v) = 1$  if variable  $v$  is the  $i$ 'th variable in clause  $c$ , and 0 otherwise

$$s_i : \{0, 1\}^m \rightarrow Z$$

and  $s_i(c) = 0$  if the  $i$ 'th variable of clause  $c$  occurs negated (in  $f$ ), and 1 otherwise

The expression can be interpreted as:  $\sum_c$  is “for every clause”,  $\sum_{v_i}$  is “for every variable in clause”,  $h_i(c, v_i)$  captures the current variable (1 if  $v_i$  is the variable addressed in this term, 0 otherwise) and  $s_i(c) - a(v_i)$  captures that the Boolean value assigned in  $a$  must match that of  $f$ , i.e. if  $\bar{v}_i$  in  $f$  then  $\bar{v}_i = 1$  in  $a$ . Also, if the values of  $a$  is only 0 or 1 then the expression counts the number of clauses *not* satisfied by the assignment  $a$  (the domain of  $a(\cdot)$  is  $Z$  but note that if other values than 0 or 1 are assigned to the variables of  $a$  then the prover only makes life more difficult for himself).

From Smolensky we know that for any function  $h : \{0, 1\}^m \rightarrow \{0, 1\}$  there exists a polynomial  $\tilde{h}$  of degree  $\leq m$  such that  $h \equiv \tilde{h}$  on  $\{0, 1\}^m$  and  $\tilde{h}$  is multilinear (i.e. not quadratic etc.) and unique.  $\tilde{h}$  is called the multilinear extension of  $h$ . Therefore,  $h_i(\cdot)$ ,  $s_i(\cdot)$  and  $a(\cdot)$  in the expression can be replaced by their multilinear extensions given a polynomial with degree at most  $12m$  (3 from  $\prod_i$ , 2 from  $h_i$  and 2 from  $(\dots)^2$ ):

$$\sum_{c \in \{0, 1\}^m} \sum_{v_1, v_2, v_3 \in \{0, 1\}^m} \prod_{i=1}^3 \tilde{h}_i(c, v_i) (\tilde{s}_i(c) - \tilde{a}(v_i))^2 = 0$$

The protocol continues as in Shamir's proof, eliminating variables until  $\prod_{i=1}^3 \tilde{h}_i(c, v_i) (\tilde{s}_i(c) - \tilde{a}(v_i))^2$  is reached. However, the verifier cannot be sure that  $\tilde{a}$  is a low-degree polynomial, i.e. that it is actually a correct tabulation of some multilinear polynomial, but can only be convinced that it is close to a low-degree polynomial by using a multilinearity test. Let  $T$  be a multilinearity test so that if  $\tilde{a}$  is multilinear then  $T(\tilde{a})$  always accepts and if  $\tilde{a}$  is not close in relative Hamming distance (i.e. it differs in at most  $\frac{1}{10}$  of the bits) then  $T(\tilde{a})$  rejects with probability  $\geq \frac{m-1}{m}$ . By the Schwarz-Zippel theorem, if  $\tilde{a}$  is close to multilinear polynomial  $a^*$  then this  $a^*$  is uniquely defined<sup>2</sup>.

How to do multilinearity test:

1. Pick random variable (there are  $m$ )
2. Pick random constants for all of the other variables
3. Query for all values of the chosen variables
4. Accept if this is arithmetic progression
5. Reject if it is not

In the end, if there exists a satisfying assignment to  $f$  then the verifier can be convinced of this by inspecting only  $O(\log n)$  bits of the proof, and if there is no satisfying assignment then the overall probability of fooling the verifier is  $\leq \frac{1}{2}$ . This completes the proof that  $3SAT \in PCP[O(\log n), O(\log n)]$  and hence  $NP \subseteq PCP[O(\log n), O(\log n)]$ . □

---

<sup>2</sup> $\tilde{a}$  can be seen as what is *supposed* to be there while  $a^*$  is what is actually there.