Complexité avancée - TD 11

Benjamin Bordais

January 6, 2021

Definition 1 (Probabilitically Checkable Proofs (PCP)) *A Turing machine with direct access is a Turing machine with:*

- *a special state, called the* reading state*,*
- *a reading oracle,*
- *two special working tapes, called the* direct access tape*, and the* address tape*.*

The machine never reads directly the content of the direct access tape (in the sense that the normal transitions of the machine are independent of the content of the direct access tape). This tape is only accessed via the reading oracle in the following way: when the machine goes in the reading state, the content of the address tape is interpreted as the binary representation of a position i of the direct access tape. The reading oracle then provides in one step, the symbol in position i of the direct access tape. (You can assume this symbol is stored in the control state, or in a special output tape of the reading oracle.)

A $\mathsf{PCP}(R(n), Q(n), T(n))$-verifier *is a probabilistic Turing machine with direct access to a tape called the* proof tape *over alphabet* $\{0,1\}$*. On input x of size n and proof tape content π, the machine uses $R(n)$ random bits and works in the following three phases:*

1. *It first computes $Q(n)$ positions $p_1, \dots p_{Q(n)}$ (in binary) in polynomial time in n, and with no calls to the reading oracle (i.e. these positions are only a function of x and the random tape content).*
2. *Then it makes $Q(n)$ calls to the reading oracle, to retrieve the symbols of the proof tape π in positions $p_1, \dots p_{Q(n)}$.*
3. *Finally, it computes a boolean value (either accept or reject) in time $T(n)$ and with no calls to the reading oracle (i.e. the answer computed in this phase is only a function of x, the random tape content, and the symbols $\pi[p_1], \dots \pi[p_{Q(n)}]$).*

The class $\mathsf{PCP}(R(n), Q(n), T(n))$ *is the set of languages L such that there exists a* $\mathsf{PCP}(R(n), Q(n), T(n))$*-verifier V such that:*

- *if $x \in L$, there exists a proof $\pi \in \{0,1\}^*$ such that $Pr_r[V(x, \pi, r)$ rejects $] = 0$;*
- *if $x \notin L$, then for all $\pi \in \{0,1\}^*$ $Pr_r[V(x, \pi, r)$ accepts $] \leq 1/2$.*

Where the probability is computed over all random tape contents r of size $R(n)$.

Exercise 1 PCP witnessing

Let $\mathsf{PCP}'(k \cdot \log\ n, Q(n), T(n))$ be defined as $\mathsf{PCP}(k \cdot \log\ n, Q(n), T(n))$ except that only proofs π of size $n^k \cdot Q(n)$ are considered, and addresses computed by the verifier have $\log(n^k \cdot Q(n))$ bits. Prove that $\mathsf{PCP}(k \cdot \log\ n, Q(n), T(n)) = \mathsf{PCP}'(k \cdot \log\ n, Q(n), T(n))$.

Solution:

Straightforwardly, we have $\mathsf{PCP}(k \cdot \log\ n, Q(n), T(n)) \supseteq \mathsf{PCP}'(k \cdot \log\ n, Q(n), T(n))$. Consider now a $\mathsf{PCP}(k \cdot \log\ n, Q(n), T(n))$-verifier V. On an input x of size n, for any random tape content of size $k \cdot \log\ n$, at most $Q(n)$ different positions are queried. If we consider the set $Pos(x) = \{i \mid \exists r \in \{0,1\}^{k \cdot \log\ n},\ \exists k \leq Q(n),\ p_k = i \text{ on random tape content } r \text{ on input } x\}$ of all positions of the proof tape used by the verifier V on input x, we have $|Pos(x)| \leq n^k \cdot Q(n)$. Hence, one can construct an injective function $f_x : Pos(x) \mapsto \{0, \ldots, n^k \cdot Q(n) - 1\}$ in polynomial time ($Q(n)$ is polynomial since we have to able to compute $Q(n)$ positions in polynomial time) and a $\mathsf{PCP}'(k \cdot \log\ n, Q(n), T(n))$-verifier V' that simulates V and instead of querying and using tape content of position $i \in Pos(x)$ on input x, it queries and uses position $f_x(i) \in \{0, \ldots, n^k \cdot Q(n) - 1\}$. The languages accepted by V and V' are the same since the function f_x is injective. The proof tape content for V' only needs to have $\log(n^k \cdot Q(n))$ bits.

Exercise 2 PCP and non-deterministic classes

Prove that, with $R(n) = \Omega(log\ n)$, we have $\mathsf{PCP}(R(n), Q(n), T(n)) \subseteq \mathbf{NTIME}(2^{O(R(n))} \cdot Q(n) \cdot (T(n) + poly(n)))$.

Solution:

Consider a $\mathsf{PCP}(R(n), Q(n), T(n))$-verifier V. With the same idea than for the previous question, we assume without loss of generality that every position p_i queried by the verifier V is at most $2^{R(n)} \cdot Q(n)$ (in time $2^{R(n)} \cdot Q(n)$, we can construct a table to associate with each initial position the corresponding position lower than $2^{R(n)} \cdot Q(n)$). Consider now the non-deterministic algorithm that, on an input x of size n guesses $2^{R(n)} \cdot Q(n)$ bits (and stores them). Then, it enumerates all possible random tape content of size $R(n)$ and simulates the execution of V with the bits guessed as the content of the proof tape needed while maintaining a counter c that corresponds to the number of accepting random tapes. The algorithm accepts if and only if $c > 1/2^{R(n)-1}$. This runs in time $2^{O(R(n))} \cdot Q(n) \cdot (T(n) + poly(n))$ (the final $poly(n)$ comes from the polynomial time taken in the first step, to compute the $Q(n)$ position) and accepts if and only if there is a proof tape content leading to acceptance.

Exercise 3 Known classes

Prove the following statements:

$$\bigcup_{c \in \mathbb{N},\ T(n) \text{ a polynomial}} \mathsf{PCP}(0, c \cdot \log\ n, T(n)) = \mathsf{P}$$

$$\bigcup_{R(n),\ T(n) \text{ polynomials}} \mathsf{PCP}(R(n), 0, T(n)) = \mathsf{coRP}$$

$$\bigcup_{Q(n),\ T(n) \text{ polynomials}} \mathsf{PCP}(0, Q(n), T(n)) = \mathsf{NP}$$

(In fact $\bigcup_{T(n) \text{ a polynomial}} \mathsf{PCP}(O(\log n), O(1), T(n)) = \mathsf{NP}$ (this is known as the PCP theorem).)

Solution:

- The direct inclusion comes from the fact that a polynomial time algorithm can simulate all the different possible calls to the proof content tape (there are polynomially many as an exponential of a logarithm) and check that there exists one that leads to acceptance (note that the calls do not depend on any random bit). The reverse inclusion is straightforward: one can simulate a polynomial time algorithm by just ignoring the randomness and the calls to the proof tape.
- This is by definition.
- The direct inclusion is straightforward. As for the reverse inclusion, the number of non deterministic calls of a non deterministic Turing machine may depend on the result of the queries considered, however in any case there are polynomially many. Hence, it suffice to consider as many non deterministic bits as the worst case execution time of the Turing machine and simulate the execution of the non-deterministic Turing machine.

Exercise 4 Graph non-ismorphism

Show that $\overline{\textbf{ISO}} \in \mathsf{PCP}(p(n), 1, c)$ for some polynomial p and constant c.

Solution:

Consider a pair of graph (G_0, G_1) with n vertices. In the proof tape, the verifier V expects, for each graph H with n vertices, that $\pi[H] = b \in \{0, 1\}$ where H is isomorphic to G_b (if H are isomorphic to neither or both G_0 and G_1, the value of $\pi[H]$ is not specified) (that is, to an exponential number (in n) of natural indexes corresponds an adjacency matrix of an n vertices graph). Then, a verifier V randomly picks $b \in \{0, 1\}$ and a permutation ν of the vertices, computes $H = \nu(G_b)$ accordingly, queries $\pi[H]$ and checks that $b = \pi[H]$.

Then, if G_0 and G_1 are not isomorphic, an honest proof tape will always lead to acceptance whereas, if they are isomorphic, the content of $\pi[H]$ is not specified and there is at most probability $1/2$ of acceptance.

Exercise 5 PCP, MIP and NEXPTIME

Recall the definition of MIP from the previous exercise sheet (where we can assume that the probability of acceptance is equal to 1 when $x \in L$).

Prove that

$$\bigcup_{R(n),Q(n),T(n) \text{ polynomials}} \mathsf{PCP}(R(n), Q(n), T(n)) \subseteq \textbf{MIP} \subseteq \textbf{NEXPTIME}$$

(The last inclusion was proved in the previous exercise sheet. In fact, we have an equality.)

Remark. Indeed **MIP** and this version of PCP *coincide* with **NEXPTIME**, but you are not required to prove the opposite inclusions.

Solution:
With the definition of MIP with oracle, this is straightforward: every call to the proof content tape can be simulated by calls to an oracle which are (as in this case) independent from one another (which is different from calls to a prover from an interactive protocol).

Exercise 6 Polynomial Identity Testing

This was already given in the previous exercise sheet.

An n-variable *algebraic circuit* is a directed acyclic graph having exactly one node with out-degree zero, and exactly n nodes with in-degree zero. The latter are called *sources*, and are labelled by variables $x_1, \dots x_n$; the former is called the *output* of the circuit. Moreover each non-source node is labelled by an operator in the set $\{+, -, \times\}$, and has in-degree two.

This can be seen with an array $(s_1, \dots, s_n, g_1, \dots, g_m)$ (the number of nodes), with first the n sources and then the m internal nodes (or gates) where an input of a gate g_i can either be a source s_j or another gate g_k with $k < i$.

An algebraic circuit defines a function from $\mathbb{Z}^n$ to $\mathbb{Z}$, associating to each integer assignment of the sources the value of the output node, computed through the circuit. It is easy to show that this function can be described by a polynomial in the variables $x_1, \dots x_n$. Algebraic circuits are indeed a form of implicit representation of multivariate polynomials. Nevertheless algebraic circuits are more compact than polynomials.

An algebraic circuit C is said to be *identically zero* if it evaluates to zero for all possible integer assignments of the sources.

The **Polynomial identity** problem is as follows:

- Input: An algebraic circuit C
- Ouput: C is identically zero

1. Show that if the variables x may range from 0 to $X \in \mathbb{N}$, then the maximum (absolute) value of a cricuit with m internal gates is X^{2^m} and show that this maximum value can achieved (this justifies the sentence "Algebraic circuits are more compact than polynomials").

2. Show that Polynomial identity is in coRP (note that it is not known whether Polynomial identity is in P).

   Hint: you may need the following statements

- **Schwartz-Zippel lemma** If $p(x_1, \dots x_n)$ is a nonzero polynomial with coefficients in $\mathbb{Z}$ and total degree at most d, and $S \subseteq \mathbb{Z}$, then the number of roots of p belonging to S^n is at most $d \cdot |S|^{n-1}$.
- **Prime number theorem** There exists a known integer $X_0 \geq 0$ such that, for all integers $X \geq X_0$, the number of prime numbers in the set $[1..2^X]$ is at least $\frac{2^X}{X}$.

Solution:

1. We can prove this result by induction on the internal gates.

2. First, note that for all polynomial q, we have $\mathsf{coRP} = \mathsf{coRP}(1 - \frac{1}{q(n)})$ (the error can be exponentially small or polynomially large).

   Consider now a circuit with n sources and m internal gates. The straightforward idea would be to use the Schwartz-Zippel lemma: We pick n numbers $(x_1, \ldots, x_n)$ at random between 1 and $10 \cdot 2^m$, compute the output y and accept iff $y = 0$. In this case, we obtain:

   - If $p(x_1, \ldots, x_n)$ is identically 0, then $Pr[y = 0] = 1$;
   - If $p(x_1, \ldots, x_n)$ is not, then by using the Schwartz-Zippel lemma for $S = \{1, \ldots, 10 \cdot 2^m\}$ and the polynom p of degree at most 2^m, we get:

$$Pr[y = 0] \leq \frac{2^m \cdot |S|^{m-1}}{|S|^m} = \frac{2^m}{|S|} = \frac{1}{10}$$

   However, this does not work since y may be equal to $(10 \cdot 2^m)^{2^m}$ which cannot be represented in polynomial time. Hence, we will do the computation modulo a given k chosen at random between 1 and 2^{2m}. Now, all along the computation, $y \mod k$ is at most 2^{2m} which can be represented an used in computations in polynomial time. Now, we have:

   - If $p(x_1, \ldots, x_n)$ is identically 0, then $Pr_{x,k}[y = 0[k]] \leq P[y = 0] = 1$;
   - If $p(x_1, \ldots, x_n)$ is not, we have:

     $$Pr[y = 0[k]] = Pr[y = 0[k] \mid y = 0] \cdot Pr[y = 0] + Pr[y = 0[k] \mid y \neq 0] \cdot Pr[y \neq 0]$$

     We have already proven that $Pr[y = 0] \leq \frac{1}{10}$. Hence, we have:

$$Pr[y = 0[k]] \leq \frac{1}{10} + Pr[y = 0[k] \mid y \neq 0] \cdot \frac{9}{10}$$

     Let us now bound the probability $Pr[y = 0[k] \mid y \neq 0]$. Let us denote by K_y the set of prime numbers that do not divide y. Note that if $k \in K_y$, then assuming $y \neq 0$ we have $y \neq 0[k]$. By the prime number theorem, for m large enough, there is at least $\frac{2^{2m}}{2m}$ prime numbers lower than 2^{2m}. Furthermore, y has at most $\log y = 2^m(\log 10 + m) \leq \frac{2^{2m}}{4m}$ prime divisors (since $4m \cdot (\log 10 + m) \leq 2^m$, for m large enough). Hence, $|K_y| \geq \frac{2^{2m}}{4m}$. Therefore:

$$Pr[y \neq 0[k] \mid y \neq 0] \geq Pr[k \in K_y] = \frac{|K_y|}{2^{2m}} \geq \frac{1}{4m}$$

     It follows that:

$$Pr[y = 0[k]] \leq \frac{1}{10} + (1 - \frac{1}{4m}) \cdot \frac{9}{10} = 1 - \frac{1}{40m/9}$$

     That is, $\mathsf{Polynomialidentity} \in \mathsf{coRP}$

Exercise 7 A general note on self-reducibility

Define a language L to be *downward-self-reducible* if there is a polynomial-time Turing Machine R such that for any x of length n, $R^{L_{n-1}}(x) = L(x)$, where L_k denotes an oracle that decides L on input of size at most k. Prove that if L is such a language, then $L \in \mathsf{PSPACE}$.

Solution:

Consider such a language L and the corresponding Turing machine R. On an input x, we can build a polynomial space algorithm that runs R and, each time a query to the oracle is made, the algorithm solves recursively the problem for instance of smaller length. If we denote by p the execution time of R, we can prove by induction that the space taken is bounded by $\sum_{i=1}^{n} p(i)$ by using a separate working tape for computing the calls to the oracle.