Complexité avancée - TD 10

Benjamin Bordais

December 16, 2020

Exercise 1 A little come back to P and RP

We define a random language A by setting that each word $x \in \{0,1\}^*$ is in A with probability $1/2$. Show that almost surely (on the probabilistic choice on the language A) we have $\mathsf{P}^A = \mathsf{RP}^A$.

Hint: *Fix an $\epsilon > 0$ and an enumeration $(M_i)_{i\in\mathbb{N}}$ of probabilistic Turing machine running in polynomial time with an oracle. Exhibit deterministic polynomial time Turing machines $(N_i)_{i\in\mathbb{N}}$ such that the probability (over the random language considered) that there is one i such that M_i and N_i do not coincide is lower than $C \cdot \epsilon$ for a constant C. You may use the language A as a random bit generator.*

Solution:

For all languages A, we have $\mathsf{P}^A \subseteq \mathsf{RP}^A$. Now, consider the other inclusion. Let $(M_i)_{i\in\mathbb{N}}$ be an enumeration of probabilistic Turing machine running in polynomial time with an oracle. Fix an $\epsilon > 0$. Let us denote by $(M'_i)_{i\in\mathbb{N}}$ the Turing machine that executes $i + 2\cdot n + \log\epsilon$ (with n the size of the input) time the machine M_i, to get the probability of error $\leq \epsilon \cdot 2^{-i-2n}$ (if the machine M_i has a behavior as in RP) and let us denote by t'_i its execution time. Now, we consider the deterministic machine N_i^A that simulates the execution of the machine $M_i'^A$ by replacing random bits read by the call to the oracle machine on (arbitrary) words of size bigger that t'_i (therefore, which are not used by $M_i'^A$). Since A is random, so are the bits from the oracle and they also are independent. Note that all the machines N_i run in polynomial time. Then, if we consider M_i^A with the acceptance condition of type RP^A and which then recognizes the language $L(M_i^A)$, for all x of size n we have (since either of the two machine may make a mistake):

$$Pr_A[N_i^A(x) \neq [x \in L(M_i^A)]] \leq \epsilon \cdot 2^{-i-2n}$$

By summing over all such x:

$$Pr_A[\exists x \in \{0,1\}^n,\ N_i^A(x) \neq [x \in L(M_i^A)]] \leq \epsilon \cdot 2^{-i-n}$$

Now, we sum over all such n:

$$Pr_A[\exists x \in \{0,1\}^*,\ N_i^A(x) \neq [x \in L(M_i^A)]] \leq 2 \cdot \epsilon \cdot 2^{-i}$$

Finally, we sum over all such i:

$$Pr_A[\exists i,\ \exists x \in \{0,1\}^*,\ N_i^A(x) \neq [x \in L(M_i^A)]] \leq 4 \cdot \epsilon$$

Therefore, we have $Pr_A[\mathsf{RP}^A \subsetneq \mathsf{P}^A] \leq 4 \cdot \epsilon$. This holds for all $\epsilon > 0$. That is, almost surely, P^A and RP^A coincide.

Exercise 2 Multi-Prover Protocol

Definition 1 *Let $P_1, \dots, P_k$ be infinitely powerful machines whose output is polynomially bounded. Let V be a probabilistic polynomial-time machine. V is called the* verifier*, and $P_1, \dots, P_k$ are called the* provers.

A round of a multi-prover interactive protocol on input x consists of an exchange of messages (i.e. words over a given alphabet) between the verifier and the provers, and works as follows:

- *The verifier V is executed on an input consisting of x, the history of all previous messages exchanged with all provers (both sent and received messages), and a random tape content of size polynomial in $|x|$. The output of the verifier is computed in time polynomial in $|x|$, and consists of messages to some or all of the provers.*
- *Each message q_i sent from the verifier to prover P_i is followed by an answer a_i, of size polynomial in $|x|$, sent from the prover P_i to the verifier. The answer a_i is computed by P_i on input consisting of x and the history of all messages previously exchanged between the verifier and the prover P_i (and only P_i).*
- *Alternatively the verifier may decide not to produce messages, and terminates the protocol by either accepting or rejecting, based on the input x and the history of all previous messages exchanged with all provers.*

You can view the protocol as executed by the verifier sharing communication tapes with each P_i, where different provers P_i and P_j (for $i \neq j$) have no tapes they can both access, besides the input tape. In a round the verifier stores each message q_i to prover P_i on the i-th communication tape, shared between the prover and P_i. The answer of P_i is put on tape i as well. The verifier has access to the input and all communication tapes, while each prover P_i has access only to the input and tape i.

$P_1, \dots, P_k$ and V form a multi-prover interactive protocol *for a language L if the execution of the protocol between V and $P_1, \dots P_k$ terminates after a polynomial number of rounds (in the size of the input x) and:*

- *if $x \in L$, then $Pr[(V, P_1, \dots, P_k) \text{ accepts } x] > 1 - 2^{-n}$;*
- *if $x \notin L$, then for all provers $P'_1, \dots, P'_k$, $\quad Pr[(V, P'_1, \dots, P'_k) \text{ accepts } x] < 2^{-n}$;*

where q is a polynomial and the probability is computed over all possible random choices of V.

In this case, we denote $L \in \mathsf{MIP}_k$. The number of provers k need not be fixed and may be a polynomial in the size of the input x. We say that $L \in \mathsf{MIP}$ if $L \in \mathsf{MIP}_{p(n)}$ for some polynomial p. Clearly $\mathsf{MIP}_1 = \mathsf{IP} = \mathsf{PSPACE}$ (as you will see in the lecture), but allowing more provers makes the interactive protocol model potentially more powerful.

1. Let M be a probabilistic polynomial-time Turing machine with access to an oracle. A language L is accepted by M iff:
   - if $x \in L$, then there exists an oracle O s.t. M^O accepts x with probability greater than $1 - 2^{-n}$;
   - if $x \notin L$, then for any oracle O', $M^{O'}$ accepts x with probability lower than 2^{-n}.

Show that $L \in \mathsf{MIP}$ if and only if L is accepted by a probabilistic polynomial time oracle machine.

2. Show that $\mathsf{MIP} = \mathsf{MIP}_2$ (assuming we can use error-reduction).

3. Show that $\mathsf{MIP} \subseteq \mathsf{NEXP}$ (this is, in fact, an equality. It can be shown by using the same kind of idea (but more involved) that was used to prove that $\mathsf{IP} = \mathsf{PSPACE}$).

Solution:

1. Suppose that $L \in \mathsf{MIP}$ with a verifier V. We define the probabilistic polynomial time Turing machine M that simulates V. However, M can only call an oracle that only gives yes-or-no answers, not a polynomially-bounded size response like a prover. Therefore, we call an oracle to get each bit of the response of the prover. That is, when V sends a message to a prover, M asks the query $(x, i, j, l, q_{i,1}, \ldots, q_{i,j})$ to the oracle, which, in turn, will be used as the l-th bit of the j-th message of V sent to prover i with $q_{i,1}, \ldots, q_{i,j}$ the first j message sent from the verifier to prover i. Then, M accepts iff the verifier V does. We get:

   - If $x \in L$, then the oracle O that faithfully simulates the calls to the different provers by sending the corresponding bits ensures that the probability of acceptance is greater than $1 - 2^{-n}$ (since $L \in \mathsf{MIP}$).
   - Suppose now that $x \notin L$ and that there exists an oracle O' such that the probability of acceptance by $M^{O'}$ is at least $1 - 2^{-n}$. Then, we can construct the prover $(P_i)_i$ by using the oracle O' (the same way M does, by calling it bit by bit). It would follow that the probability of acceptance is at least $1 - 2^{-n}$, hence the contradiction since $L \in \mathsf{MIP}$.

   Suppose now that L is accepted by an oracle probabilistic polynomial-time Turing machine M running in time n^c (with n the size of the input). We want to simulate the oracle with provers. If we use only one prover to simulate the oracle, then the answer to a query may depend on previous queries, which is different from the specification of an oracle. However, we could use one prover per query, so that no prover has an history of previous exchanges. However, if we want to prove that the probability of accepting if below 2^{-n} when $x \notin L$, then we have to use the hypothesis that for any oracle, the probability of error is below 2^{-n}. But we do not know which prover is used on a given query since this may depend on the probabilistic tape. A solution may be to consider the majority of the choice of all provers used, but in that case the probability that at least one prover does not agree with the majority on at least one query is very high. To circumvent this phenomenon, for each call to the oracle we use several provers and require that they are all unanimous on the answer. More, formally, we use $2 \cdot n^{c+1}$ provers. We consider a verifier V that first chooses randomly (and uniformly) an ordering of these provers. Then, the verifier V simulates M and, each time M makes a query to the oracle, V asks the question to the next $2n$ provers. If they are unanimous, V proceeds with the simulation of M with the common answers of the provers as answer of the oracle, otherwise it rejects. Then, V accepts iff M does (assuming that all queries to the oracle passed successfully). Note that indeed $2 \cdot n^{c+1}$ provers suffice.

   - If $x \in L$, there exists an oracle O such that the probability of acceptance is at least $1 - 2^{-n}$, hence if the provers faithfully simulates the answer of the oracle O, then the probability of acceptance will be the same.

- Suppose now that $x \notin L$. Consider any provers $P_1, \ldots, P_{2 \cdot n^{c+1}}$ and the oracle O' that answers like the majority of the $2 \cdot n^{c+1}$ provers. Now, there are two possibilities: either all oracle queries in the simulation is consistent with O', or there is at least one difference. In the first case, the probability of acceptance is less than 2^{-n} (by definition of the acceptance condition of a probabilistic Turing machine). In the other case, the probability of acceptance is bounded by the probability that at least one oracle query is inconsistent with O' and it did not reject immediatly. That is, there exists a sequence of $2 \cdot n$ provers that are unanimously inconsistent with O' on a query. However, by definition of O', at least half of the provers are consistent with it. Therefore, the probability (for a fixed sequence of $2 \cdot n$ provers) that the provers are unanimously inconsistent is lower than 2^{-2n}. By summing over all n^c queries, it follows that the probability that there is at least one inconsistency not rejected is lower than $n^c \cdot 2^{-2n}$. Overall, we have $Pr[\text{accept}] < 2^n + n^c \cdot 2^{-2n} < 2^{-n+1}$ for n large enough.

We need to reduce that probability even further. Hence, we consider V' that simulates V three times in a row and answers according to the majority. In that case:

- If $x \in L$, we have:

$$Pr[\text{accept}] > (1 - 2^{-n})^3 + 3(1 - 2^{-n})^2 \cdot 2^{-n} > 1 - 3 \cdot 2^{-2n} > 1 - \cdot 2^{-n}$$

- If $x \notin L$, we have:

$$Pr[\text{accept}] < (2^{-n+1})^3 + 3(2^{-n+1})^2 \cdot (1 - 2^{-n+1}) < 2^{-n}$$

2. Consider a language $L \in \mathsf{MIP}$ with an arbitrary number of provers $(P_i)_i$ (but polynomially bounded in the size of the input) with a verifier V. We want to simulate what happens with these provers with only two provers. The idea is to use one prover to simulate the calls to all these provers and then simulating V with these calls. However, since these queries are made to a single prover, in the simulation, the provers simulated may interact with each other (which is not allowed in a MIP protocol), hence we check with the second prover that the answer for a given prover P_i can be obtained with a single prover that does not interact with the other. However, we can only do one call to that second prover, otherwise the second call would have the information of the first call (i.e. there would an interaction between provers). Therefore, we randomly choose the prover P_i to check, and we use error reduction.

   More formally, assume that the verifier V uses k provers in time n^c on an input x. We consider a verifier V' generating a random word r of length n^c and sending it to the first prover. This prover answers with the complete interaction of all k provers with the verifier V over the whole computation of V on x with the random word r. Then, V' simulates V with the given interaction with the provers. If V rejects, so does V'. Otherwise, it randomly picks a number j between 1 and k and simulates the complete interaction with prover P_j with queries to the second prover (while indicating the number j in the exchange). If it differs from what was sent by the first prover, V' rejects, otherwise it accepts. Then, we have:

   - If $x \in L$, then the two provers can just faithfully simulate the other provers and get a probability greater then $1 - 2^{-n}$.
   - If $x \notin L$, either the first prover simulates faithfully the other provers (in which case, the probability of accepting is below 2^{-n}), otherwise at least one prover

cannot have this interaction without exchanging with other provers (in which case, the probability to reject is at least $1/k$ if the prover chosen at random is the faulty one). Overall, the probability to accept is lower than:

$$Pr[\text{accept}] < 2^{-n} + (1 - 1/k)$$

We conclude by using error reduction with k^2 rounds (where, at each round, one new random word is chosen and therefore the knowledge of the previous rounds is not an issue). Note this is possible since k is bounded polynomially in n.

3. Consider a language L decided by an oracle probabilistic machine M running in time n^c. This machine makes at most n^c calls to the oracle. Note that it is not enough to only guess the result of the n^c queries, and then counting the number of accepting runs over all possible random word of the appropriate size (which would yield a polynomial space algorithm) since the calls to the oracle may depend on the random bits read. Hence, we consider a non-deterministic exponential time machine M' that guesses the oracle O, or more precisely that guesses the answer of the oracle O to all possible $2^{n^c+1} - 1$ queries that the machine M can make (since the size of a query is at most n^c). Then, for a word r of size n^c, denote f the function such that $f(x, O, r) = 1$ if the simulation of M on the input x with r used as random tape accepts, and 0 otherwise. Then, le machine M' accepts x iff:

$$S = \sum_{r \in \{0,1\}^{n^c}} f(x, O, r) \geq 2^{n^c - 1}$$

Then, if $x \in L$, there exists an oracle O such that $S > (1 - 2^n) \cdot 2^{n^c} \geq 2^{n^c-1}$ and if $x \notin L$, for all oracle O', we have $S < 2^{-n} \cdot 2^{n^c} \leq 2^{n^c-1}$. Therefore, $L \in \mathsf{NTIME}(2^{O(n^c)}) \subseteq \mathsf{NPTIME}$.

Exercise 3 Polynomial Identity Testing

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 $x_1, \ldots x_n$; 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 $(s_1, \ldots, s_n, g_1, \ldots, g_m)$ (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 $x_1, \ldots x_n$. 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 eroor 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 $(x_1, \dots, x_n)$ at random between 1 and $10 \cdot 2^m$, compute the output y and accept iff $y = 0$. In this case, we obtain:

   - If $p(x_1, \dots, x_n)$ is identically 0, then $Pr[y = 0] = 1$;
   - If $p(x_1, \dots, x_n)$ is not, then by using the Schwartz-Zippel lemma for $S = \{1, \dots, 10 \cdot 2^m\}$ 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 $p(x_1, \dots, x_n)$ is identically 0, then $Pr_{x,k}[y = 0[k]] \leq P[y = 0] = 1$;
   - If $p(x_1, \dots, x_n)$ 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 $y \neq 0[k]$. 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, $\mathsf{Polynomialidentity} \in \mathsf{coRP}$