Complexité avancée - TD 9

Benjamin Bordais

December 09, 2020

We recall the definition of the Arthur-Merlin hierarchy.

Definition 1 *An* Arthur and Merlin triplet *is the data of* $(M, \mathcal{A}, D)$ *where* M *is a Merlin function, that is a function with the size of the output polynomial in the size of the input, possibly not computable, a randomized Turing machine* $\mathcal{A}$ *running in polynomial time and a language* $D \in \mathsf{P}$*. Then, for all* $w \in \{\mathsf{A}, \mathsf{M}\}^*$*, let us denote by* k *the number of times* A *appears in the word* w*. We consider the following algorithm induced by the word* w *(with* $n = |w|$ *and* $r_1, \ldots, r_k$ k *random tapes of size polynomial in* n*).*

$prot_w(M; x, r_1, \ldots, r_k)$:
$imp = x$
$i = 0$
for $j = 1, \ldots, n$:
  if $w_j = \mathsf{A}$ ***then*** $(i = i+1,\ q_j = \mathcal{A}(imp, r_i);\ imp = imp \ \# \ r_i \ \# \ q_j)$
  else $(y_j = M(imp);\ imp := imp \ \# \ y_j)$
accept if $(imp \in D)$, ***else reject***

We denote $prot[\mathcal{A}, M]_D(x, r_1, \ldots, r_k) = \top$ *if the previous algorithm accepts, otherwise* $prot[\mathcal{A}, M]_D(x, r_1, \ldots, r_k) = \bot$.

Recall the definition of the Arthur-Merlin hierarchy: $\mathsf{AM}[f]$ *for a proper function* f *denotes the class of languages* L *such that there exists an Arthur and Merlin triplet* $(M, \mathcal{A}, D)$ *such that for any* x *of size* n*, letting* $w \in \{\mathsf{A}, \mathsf{M}\}^{f(n)}$*:*

1. *Completeness: if* $x \in L$ *then* $Pr[prot_w[\mathcal{A}, M]_D(x, r_1, \ldots, r_k) = \top] \geq 2/3$
2. *Soundness: if* $x \notin L$ *then for any Merlin's function* M'*,* $Pr[prot_w[\mathcal{A}, M']_D(x, r_1, \ldots, r_k) = \bot] \geq 2/3$

Exercise 1 NP and BPP

- if $\mathsf{P} = \mathsf{NP}$ then $\mathsf{BPP} = \mathsf{P}$.
- if $\mathsf{NP} \subseteq \mathsf{BPP}$ then $\mathsf{AM} = \mathsf{MA}$ (you may use the fact (or even prove!) that $\mathsf{BPP}^{\mathsf{BPP}} = \mathsf{BPP}$).

Solution:

- In that case, $\mathsf{PH} = \mathsf{P}$, and therefore $\mathsf{P} \subseteq \mathsf{BPP} \subseteq \Sigma_2^p \subseteq \mathsf{P}$.
- We know that $\mathsf{BPP} = \mathsf{A} \subseteq \mathsf{MA} \subseteq \mathsf{AM} \subseteq \mathsf{BPP}^{\mathsf{NP}} \subseteq \mathsf{BPP}^{\mathsf{BPP}} = \mathsf{BPP}$. Therefore, if $\mathsf{NP} \subseteq \mathsf{BPP}$ we have $\mathsf{AM} = \mathsf{MA}$.

Exercise 2 AM with perfect soundness

Define AM_{ps} as AM with perfect soundness, that is, in the case $x \notin L$, for all Merlin's function, the probability to reject is equal to 1. Show that $\mathsf{AM}_{ps} = \mathcal{C} \subseteq \mathsf{AM}$, where $\mathcal{C}$ is a known complexity class.

Solution: We have $\mathsf{NP} \subseteq \mathsf{AM}$. In addition, for $L \subseteq \mathsf{AM}_{ps}$ we have a Arthur-Merlin triplet $(M, \mathcal{A}, D)$ from the definition of AM_{ps}. Now, let L' be the language decided by the non-deterministic polynomial time algorithm on an input x that guesses r and y (of size bounded by the execution time of $\mathcal{A}$ on x and size of the output of the function M, respectively) and returns return $x\#r\#\mathcal{A}(x,r)\#y \in D$. It follows that:

- if $x \in L$ then $Pr_r[prot[\mathcal{A}, M]_D(x,r) = \top] \geq 1 - 1/2^n$. Hence, there exists y and r such that $x\#r\#\mathcal{A}(x,r)\#y \in D$, therefore $x \in L'$.
- if $x \notin L$, then for all Merlin function M', $Pr_r[prot[\mathcal{A}, M]_D = \bot] = 1$. Therefore, for all r, y, we have $x\#r\#\mathcal{A}(x,r)\#y \notin D$, hence $x \notin L'$.

That is $L = L'$ and $L \in \mathsf{NP}$. That is, $\mathsf{AM}_{ps} \subseteq \mathsf{NP}$.

Furthermore, for $L \in \mathsf{NP}$, there exists $L' \in \mathsf{P}$ and a polynom p such that $L = \{x \mid \exists y,\ |y| \leq p(|x|),\ (x,y) \in L'\}$. Now, we set $D = \{y\#x\#\# \mid (x,y) \in L'\} \in \mathsf{P}$. Then the language L is recognized by the Arthur-Merlin triplet $(M, \mathcal{A}, D)$ for M sending the non deterministic choice associated to L and for a lazy Arthur $\mathcal{A}$.

Exercise 3 BPP-completeness? – A follow up

Recall the exercise from TD07:

1. Show that the language $L_{\mathsf{NP}} = \{(M, x, 1^t) \mid M \text{ accepts on input } x \text{ in time at most } t\}$, where M is the code of a non-deterministic Turing machine, x an input of M and t a natural number, is NP-complete.
2. Let now L_{BPP} be the language of words $(M, x, 1^t)$ where M designates the encoding of a probabilistic Turing machine and x a string on $M's$ alphabet such that M accepts x in at most t steps, for at least $2/3$ of the possible random tapes of size t.

   Is L_{BPP} BPP-hard? Is it in BPP ?

It is straightforward to prove that L_{BPP} is BPP-hard, however, it is not known if it is in BPP. To this day, no BPP-complete problem is known. This can be circumvent with promise problem. However, promise problems also ensure counter intuitive properties.

Definition 2 *A promise problem L is a pair $(L_{yes}, L_{no}) \subseteq (\{0,1\}^*)^2$ such that $L_{yes} \cap L_{no} = \emptyset$. The set $L_{yes} \cup L_{no}$ is called the* promise.

Definition 3 *A promise problem $L = (L_{yes}, L_{no})$ is Karp-reducible to the promise problem $L' = (L'_{yes}, L'_{no})$ if there exists a polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that:*

- *if $x \in L_{yes}$, then $f(x) \in L'_{yes}$;*
- *if $x \in L_{no}$, then $f(x) \in L'_{no}$.*

Definition 4 *A promise problem $L = (L_{yes}, L_{no})$ is Cook-reducible to the promise problem $L' = (L'_{yes}, L'_{no})$ if there exists polynomial time Turing machine $M^{L'}$ with an oracle in L' such that:*

- *if* $x \in L_{yes}$*, then* $M^{L'}(x) = \top$*;*
- *if* $x \in L_{no}$*, then* $M^{L'}(x) = \bot$*.*

Note that the correctness of the answer of the oracle is only guaranteed if the query is in the promise of the language L'*.*

We can now define the alternative to BPP with promise problems.

Definition 5 *Let* BPP_{prm} *be the set of promise problems* $L = (L_{yes}, L_{no})$ *such that there exists a probabilistic Turing machine* M *running in polynomial time such that:*

- *if* $x \in L_{yes}$*, then* $Pr_r[M(x, r) = \top] \geq 2/3$*;*
- *if* $x \in L_{no}$*, then* $Pr_r[M(x, r) = \top] \leq 1/3$*.*

1. Exhibit a BPP_{prm}-complete problem (for Karp reductions).
2. Define analogously to BPP_{prm} the classes NP_{prm} and coNP_{prm}.
3. Give a NP_{prm}-complete problem (for Karp-reduction).
4. Prove that if L is Karp-reducible to L' and L' is Cook-reducible to L'' then L is Cook reducible to L''.
5. Prove that the following problem xSAT is in $\mathsf{NP}_{prm} \cap \mathsf{coNP}_{prm}$ and is NP_{prm}-hard for Cook reductions:
   - $L_{yes} = \{(\varphi_1, \varphi_2) \mid \varphi_1 \in \mathsf{SAT},\ \varphi_2 \notin \mathsf{SAT}\}$
   - $L_{no} = \{(\varphi_1, \varphi_2) \mid \varphi_1 \notin \mathsf{SAT},\ \varphi_2 \in \mathsf{SAT}\}$

Solution:

1. The following promise problem
   - $L_{yes} = \{(M, x, 1^t) \mid M$ accepts x in at most t steps with probablity at least $2/3\}$
   - $L_{no} = \{(M, x, 1^t) \mid M$ accepts x in at most t steps with probablity at most $1/3\}$

   is BPP_{prm}-complete.
2. **Definition 6** *Let* NP_{prm} *(resp.* coNP_{prm}*) be the set of promise problems* $L = (L_{yes}, L_{no})$ *such that there exists a non-deterministic Turing machine* M *running in polynomial time such that:*
   - *if* $x \in L_{yes}$*, then there exists an accepting run (resp. all runs are accepting) of* M *on input* x*;*
   - *if* $x \in L_{no}$*, then all runs are rejecting (resp. there exists a rejecting run) of* M *on input* x*.*
3. The promise problem that is equivalent to SAT is:
   - $L_{yes} = \{\varphi \mid \varphi$ is satisfiable$\}$
   - $L_{no} = \{\varphi \mid \varphi$ is not satisfiable$\}$

It is straightforwardly in NP_{prm}. It is in addition NP_{prm}-hard for Karp reductions by using the same reduction used to prove that SAT is NP-hard.

4. Let us assume that $L = (L_{yes}, L_{no})$ is Karp-reducible to the promise problem $L' = (L'_{yes}, L'_{no})$ and consider the associated function f and that $L' = (L'_{yes}, L'_{no})$ is Cook-reducible to $L'' = (L''_{yes}, L''_{no})$. Let us prove that L is also Cook reducible to L''. Consider the deterministic polynomial time Turing machine M for the reducibility between L' and L''. Then, we construct M' that just computes $f(x)$ on an input x and then simulates M on $f(x)$ (of polynomial size). Then:

   - if $x \in L_{yes}$, then $f(x) \in L'_{yes}$ and $M^{L''}(f(x))$ returns $\top$;
   - if $x \in L_{no}$, then $f(x) \in L'_{no}$ and $M^{L''}(f(x))$ returns $\bot$;

5. One can consider a non-deterministic machine that, on an input (φ_1, φ_2) guesses a valuation and checks that it satisfies φ_1 (resp. $\neg\varphi_2$). By definition of the semantics of NP_{prm} and coNP_{prm}, we have $\mathsf{xSAT} \in \mathsf{NP}_{prm} \cap \mathsf{coNP}_{prm}$. Let us now prove that it is NP_{prm}-hard under Cook-reductions. To do so, let us reduce the promise version of the problem SAT from question 3 that is NP_{prm}-hard under Karp-reductions. From question 4, this will show that xSAT is NP_{prm}-hard under Cook reductions. Consider an input φ. Consider the following polynomial time algorithm with an oracle to xSAT:

```
a(φ):
        if φ = True:
        then accept;
        elif φ = False:
        then reject;
        else
        let x in Var(s);
        if xSAT[φ[x ← True], φ[x ← False]] = ⊤:
        then a(φ[x ← True])
        else a(φ[x ← False])
```

   Note that if $\mathsf{xSAT}[\varphi[x \leftarrow \text{True}], \varphi[x \leftarrow \text{False}]] = \top$, the only thing we can deduce is that $[\varphi[x \leftarrow \text{True}], \varphi[x \leftarrow \text{False}]] \notin L_{no}$. However, this is sufficient to only consider the case where x is set to True since this means that either φ is not satisfiable, φ is satisfiable regardless of the choice for x or it is satisfiable only if x is set to True. Then, the reasoning is similar for the other case where $\mathsf{xSAT}[\varphi[x \leftarrow \text{True}], \varphi[x \leftarrow \text{False}]] = \bot$. It follows that this algorithm finds a satisfying valuation of φ if and only if it is satisfiable. Hence, we have exhibited a Cook reduction from the promise version of SAT to xSAT.

Exercise 4 The PP class

This is the same exercise as last week. The only new question is question 4.

The class PP is the class of languages L for which there exists a polynomial time probabilistic Turing machine M such that:

- if $x \in L$ then $Pr[M(x, r) \text{ accepts }] > \frac{1}{2}$
- if $x \notin L$ then $Pr[M(x, r) \text{ accepts }] \leq \frac{1}{2}$

Also define $\mathsf{PP}_<$ as the class of languages L for which there exists a polynomial time probabilistic Turing machine M such that:

- if $x \in L$ then $Pr[M(x,r) \text{ accepts }] > \frac{1}{2}$
- if $x \notin L$ then $Pr[M(x,r) \text{ accepts }] < \frac{1}{2}$

1. Show that $\mathsf{BPP} \subseteq \mathsf{PP}$ and $\mathsf{NP} \subseteq \mathsf{PP}$;
2. Show that $\mathsf{PP} = \mathsf{PP}_<$ and that PP is closed under complement;
3. Consider the decision problem MAJSAT:
   (a) Input: a boolean formula ϕ on n variables
   (b) Output: the (strict) majority of the 2^n valuations satisfy ϕ.

   Show that $\mathsf{MAJSAT} \in \mathsf{PP}$. In fact, MAJSAT is PP-complete.

   One may also consider the decision problem MAXSAT:
   (a) Input: a boolean formula ϕ on n variables, a number K
   (b) Output: more than K valuations satisfy ϕ.

   Show that MAXSAT is also PP-complete (to prove that $\mathsf{MAXSAT} \in \mathsf{PP}$ one may reduce MAXSAT to MAJSAT).
4. The class $\#\mathsf{P}$ is the class of **functions** $f : \Sigma^* \to \mathbb{N}$ for which there exists a relation $R \subseteq \Sigma^* \times \Sigma^*$ and a polynomial p such that:
   (a) for every $x, y \in \Sigma^*$, $R(x,y)$ implies $|y| < p(|x|)$
   (b) $R \in \mathsf{P}$
   (c) for every x, $f(x) = |\{y \mid R(x,y)\}|$

   The function $f(x)$ counts the number of words y such that $(x,y) \in R$. In fact $\#\mathsf{P}$ is as powerful as PP, however this class cannot be compared directly since $\#\mathsf{P}$ is a class of functions. In fact, we need to use oracle machines. Specifically, for a function $f \in \#\mathsf{P}$, a Turing machine can use as oracle the function f that, when a word u is written on the oracle tape, writes in constant time $f(u)$ in binary in that oracle tape. Then, $\mathsf{P}^{\#\mathsf{P}} = \cup_{f \in \#\mathsf{P}} \mathsf{P}^f$. The class P^{PP} is defined as usual.

   Prove that: $\mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\#\mathsf{P}}$.
5. Show that $\mathbf{MA} \subseteq \mathsf{PP}$.

Solution:

1. - A language $L \in \mathsf{BPP}$ is recognized by a PTM M such that if $x \in L$ then $Pr[M(x,r) \text{ accepts }] \geq \frac{2}{3}$ and if $x \notin L$ then $Pr[M(x,r) \text{ accepts }] \leq \frac{1}{3}$. It follows that $L \in \mathsf{PP}$.
   - The class PP is closed under logspace reduction. It suffice to show that $\mathsf{SAT} \in \mathsf{PP}$. Consider now a probabilistic Turing machine with an input that is a formula φ. According to the first bit of the random tape, it either accepts or reads what remains of the random tape for a valuation and accepts if and only if it satisfies φ. Then, if $\varphi \in \mathsf{SAT}$, we have $Pr[M(x,r) \text{ accepts }] > \frac{1}{2}$, otherwise $Pr[M(x,r) \text{ accepts }] = \frac{1}{2}$.

2. Trivially, we have $\mathsf{PP}_< \subseteq \mathsf{PP}$. Now, consider $L \in \mathsf{PP}$ and its associated Turing machine M running in polynomial time p. Without loss of generality, we assume that the alphabet of the random tape is of size 2, hence the probability of a random word for M on an input x such that $|x| = n$ is $2^{-p(n)}$. Therefore, if $x \in L$ then $Pr[M(x,r) \text{ accepts }] \geq \frac{1}{2} + \frac{1}{2^{p(n)}}$. Now, we construct another Turing machine M' that runs M on an input. If M would reject, M' rejects too, and if M would accept then M' rejects with probability $\frac{1}{2^{p(n)}}$ (for instance, by reading a word in the random tape of length $p(n)$and accepting only if there are only 0s). Then:

   - if $x \in L$: $Pr[M(x,r) \text{ accepts}] \geq (\frac{1}{2} + \frac{1}{2^{p(n)}}) \cdot (1 - \frac{1}{2^{p(n)}}) = \frac{1}{2} + \frac{1}{2^{p(n)+1}} - \frac{1}{2^{2 \cdot p(n)}} > \frac{1}{2}$
   - if $x \notin L$: $Pr[M(x,r) \text{ accepts}] \leq \frac{1}{2} \cdot (1 - \frac{1}{2^{p(n)}}) < \frac{1}{2}$

   That is, $L \in \mathsf{PP}_<$. The stability under complement then follows by inverting the accepting and rejecting states.

3. A probabilistic Turing machine that checks that a valuation read on the random tape satisfies the formula decides MAJSAT for PP. Then, MAJSAT can be reduced to MAXSAT in logarithmic space as one has to write on the output tape the number $2^{n-1} + 1$ in binary, which consists in a 1, $n-2$ 0s and then a 1 which only requires a counter in binary up to $n-2$. Therefore, MAXSAT is also PP-hard. Let us now show that $\mathsf{MAXSAT} \in \mathsf{PP}$. To do so, let us reduce MAXSAT to MAJSAT. Consider an instance (φ, i) of MAXSAT with $0 \leq r_1 < r_2 < \ldots < r_k \leq n$ such that $2^n - i = 2^{n-r_1} + \ldots + 2^{n-r_k}$ (the values $n - r_j$ refers to the 1s in the binary decomposition of $2^n - i$). Let us denote $x_1, \ldots, x_n$ the variables of φ. Then, we consider the formula ψ as:

$$\begin{aligned}\psi = &(x_1 \wedge \ldots \wedge x_{r_1}) \\ &\vee (\neg x_1 \wedge \ldots \wedge \neg x_{r_1} \wedge x_{r_1+1} \wedge \ldots \wedge x_{r_2}) \\ &\vee \cdots \\ &\vee (\neg x_1 \wedge \ldots \wedge \neg x_{r_{k-1}} \wedge x_{r_{k-1}+1} \wedge \ldots \wedge x_{r_k})\end{aligned}$$

   We can see there are exactly 2^{n-r_j} valuations satisfying the j-th line of ψ. With the negation at beginning of the lines, no valuation satisfies two lines of ψ. Therefore, there are exactly $2^{n-r_1} + \ldots + 2^{n-r_k} = 2^n - i$ valuations satisfying ψ. Consider now a fresh variable y and the formula: $\varphi' = (y \wedge \varphi) \vee (\neg y \wedge \psi)$. Then, we have φ' computable in polynomial time from φ and φ is satisfied by more than i valuations if and only if φ' is satisfied by more than $2^n - i + i = 2^{n+1}/2$ valuations, that is half of valuations, i.e. $\varphi \in \mathsf{MAXSAT} \Leftrightarrow \varphi' \in \mathsf{MAJSAT}$.

4. Let $A \in \mathsf{PP}$. Consider the probabilistic Turing machine M as in the definition of PP and the polynom p that bounds the running time of M. Let us define the function $f : \Sigma^* \to \mathbb{N}$ by $f(x) = |\{r \in \Sigma^{p(|x|)} \mid M(x,r) \text{ accepts}\}|$. By definition, we have $f \in \#\mathsf{P}$. Furthermore, $x \in A \Leftrightarrow f(x) > \frac{|\Sigma|^{p(|x|)}}{2}$. Hence, for a Turing machine T_A running in polynomial time with oracle calls to A, we can construct a Turing machine T_f running in polynomial time with oracle calls to f that simulates T_A with each call to the oracle A replaced by a call to the oracle function f, and then comparing the result with $\frac{|\Sigma|^{p(|x|)}}{2}$ which can be done in polynomial time. Hence, $\mathsf{P}^{\mathsf{PP}} \subseteq \mathsf{P}^{\#\mathsf{P}}$.

   Now, let $f \in \#\mathsf{P}$ and R and p as in the definition of $\#\mathsf{P}$. For x in Σ^*, let us denote by R_x the set $\{y \in \Sigma^{p(|x|)} \mid (x,y) \in R\}$. It ensures $|R_x| = f(x)$. Consider now

the set $B = \{(x, z) \in \Sigma^* \times \mathbb{N} \mid f(x) \geq z\}$ (with z written in binary). Let us prove that $B \in \mathsf{PP}$. Consider now the following randomized polynomial time algorithm on an input x: according to the first random bit, either it accepts with probability $1 - \frac{z-1}{|\Sigma|^{p(|x|)}}$ (for instance by reading $p(|x|)$ bits and rejecting iff the word obtained is one of the first $z-1$ word in lexicographic order, which can be done in time polynomial in the number of bits necessary to encode z) or it chooses randomly (and uniformly) an instance $y \in \Sigma^{p(|x|)}$ of R_x and accepts if $(x, y) \in R$. Then, if we denote by q the probability of accepting, it ensures:

$$q = \frac{1}{2} \cdot (1 - \frac{z-1}{|\Sigma|^{p(|x|)}}) + \frac{1}{2} \cdot \frac{|R_x|}{|\Sigma|^{p(|x|)}}$$

Hence:

$$f(x) = |R_x| \geq z \Leftrightarrow q > \frac{1}{2}$$

Therefore, $B \in \mathsf{PP}$. Now, for a Turing machine T_f running in polynomial time with oracle calls to f, we can construct a Turing machine T_B running in polynomial time with oracle calls to B that simulates T_f with each call to the oracle f replaced by binary search with calls to the oracle B (note that the number of steps in the binary search in polynomial as $f(x) \leq |\Sigma|^{p(|x|)}$ for all $x \in \Sigma^*$). Hence, $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{P}^{\mathsf{PP}}$.

5. Consider the characterization **MA** of exercise 1. Let $L \in \mathbf{MA}$ and the corresponding Arthur-Merlin triplet $(M, \mathcal{A}, D)$. Here, once y is fixed (which is the result of the Merlin map M whose size is bounded by the polynom p), we can repeat the experience – that is, iterate $36 \cdot q(|x|) \cdot \log(2)$ calls to the Arthur probabilistic Turing machine – and use majority voting and the Chernoff bound to have the error rate below $1/2^{q(|x|)}$ for all polynom q. This new probabilistic Turing machine works in polynomial time, specifically $36 \cdot q(|x|) \cdot \log(2) \cdot p(|x|)$, which is also the length of the random tape used by this new Turing machine on an input (x, y) of size $|x| + p(|x|)$. Let $q = p + 2$, $u = 36 \cdot q \cdot \log(2)$, and $D'_x = \{(r_1, \ldots, r_{u(|x|)}, y) \mid |y| = p(|x|) \wedge \forall i,\ |r_i| = p(|x|)$ and the majority of the r_i ensure $(x, r_i, y) \in D\}$. Note that deciding if $(r, y) \in D'_x$ can be done in polynomial time. Then,

   - $x \in L \Rightarrow \exists y \in \{0,1\}^{p(|x|)},\ Pr_{r \in \{0,1\}^{u(|x|) \cdot p(|x|)}}[(r, y) \in D'_x] \geq 1 - 1/2^{p(|x|)+2}$
   - $x \notin L \Rightarrow \forall y \in \{0,1\}^{p(|x|)},\ Pr_{r \in \{0,1\}^{u(|x|) \cdot p(|x|)}}[(r, y) \in D'_x] \leq 1/2^{p(|x|)+2}$

   Let us now consider the size of the set D'_x. If $x \in L$, we have:

$$\sum_{y \in \{0,1\}^{p(|x|)}} \sum_{r \in \{0,1\}^{u(|x|) \cdot p(|x|)}} [(r, y) \in D'_x] \geq 2^{u(|x|) \cdot p(|x|)} \cdot (1 - 1/2^{p(|x|)+2}) \geq 2^{u(|x|) \cdot p(|x|) - 1}$$

   If $x \notin L$, we have:

$$\sum_{y \in \{0,1\}^{p(|x|)}} \sum_{r \in \{0,1\}^{u(|x|) \cdot p(|x|)}} [(r, y) \in D'_x] \leq 2^{p(|x|)} \cdot 2^{u(|x|) \cdot p(|x|)} \cdot (1/2^{p(|x|)+2}) = 2^{u(|x|) \cdot p(|x|) - 2}$$

   Therefore, we have $x \in L \Leftrightarrow |D'_x| \geq 2^{u(|x|) \cdot p(|x|) - 1}$.

   Consider now the following randomized polynomial time algorithm on an input x: according to the first random bit, either it accepts with probability $1 - 1/2^{p(|x|)+1}$ (for instance by reading $p(|x|) + 1$ bits and rejecting iff all are 0s) or it chooses

randomly (and uniformly) an instance (r, y) of D'_x and accepts if $(r, y) \in D'_x$. Then, if we denote by p_x the probability of accepting, it ensures:

$$p_x = \frac{1}{2} \cdot (1 - \frac{1}{2^{p(|x|)+1}}) + \frac{1}{2} \cdot \frac{|D'_x|}{2^{p(|x|)+u(|x|)\cdot p(|x|)}}$$

Hence:

$$x \in L \Leftrightarrow \frac{|D'_x|}{2^{p(|x|)+u(|x|)\cdot p(|x|)}} \geq \frac{1}{2^{p(|x|)+1}} \Leftrightarrow p_x \geq \frac{1}{2}$$

It follows that $L \in \mathsf{PP}$ (since PP is closed under complement, we can assume, in the definition of PP that if $x \in L$, then the probability to accept is $\geq 1/2$ and if $x \notin L$, then the probability is $< 1/2$).

Exercise 5 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 does 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 (M'_i) 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:

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

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