Complexité avancée - TD 4

Benjamin Bordais

October 14, 2020

Exercise 1 A translation result

Show that if $\mathsf{P} = \mathsf{PSPACE}$, then $\mathsf{EXPTIME} = \mathsf{EXPSPACE}$.

Solution:

In any case, we have $\mathsf{EXPTIME} \subseteq \mathsf{EXPSPACE}$. Let us assume that $\mathsf{PSPACE} = \mathsf{P}$ and let us show that $\mathsf{EXPSPACE} \subseteq \mathsf{EXPTIME}$. Let $L_1 \in \mathsf{EXPSPACE}$ be a language accepted by a Turing machine M_1 running in 2^{n^c} space, for some $c \geq 1$. We define:

$$L_2 = \{(x, 1^{2^{|x|^c}}) \mid x \in L_1\}$$

A Turing machine M_2 which launches M_1 on x for an input $w = (x, 1^{2^{|x|^c}})$ (after checking the size of w) accepts L_2 and runs in space $O(|w|)$. Hence, $L_2 \in \mathsf{PSPACE} \subseteq \mathsf{P}$. Therefore, there exists a Turing machine M_3 running in polynomial time accepting L_2. Now, consider a Turing machine M_4 that, on an input x, computes $w = (x, 1^{2^{|x|^c}})$ in exponential time and then launches M_3. Then, M_4 accepts L_1 and runs in exponential time. That is, $L_1 \in \mathsf{EXPTIME}$ and $\mathsf{EXPSPACE} \subseteq \mathsf{EXPTIME}$.

Exercise 2 Unary languages

Recall that a *unary* language is any language over a one-letter alphabet.

1. Prove that if a unary language is NP-complete, then $\mathsf{P} = \mathsf{NP}$.
2. Prove that if every unary language in NP is actually in P, then $\mathsf{EXP} = \mathsf{NEXP}$.

Solution:

1. Consider a unary language L (say on the alphabet $\Sigma = \{1\}$) that is NP-complete and a polynomial time reduction tr ensuring $\phi \in \mathsf{SAT} \Leftrightarrow tr(\phi) \in L$. There is $a, c \geq 1$ such that we have $|tr(\phi)| \leq a \cdot |\phi|^c$ for all ϕ. We design a polynomial time algorithm that solves SAT. Consider a SAT formula ϕ. For a variable x appearing in ϕ, we denote by $\phi[x \leftarrow \mathsf{True}]$ the (simplification of the) formula ϕ where x is set to True (and similarly $\phi[x \leftarrow \mathsf{False}]$). Note that $|\phi[x \leftarrow \mathsf{True}]| \leq |\phi|$ and $|\phi[x \leftarrow \mathsf{False}]| \leq |\phi|$

   We maintain a list l of pairs $(tr(\varphi), \varphi)$ such that ϕ is satisfiable if and only if one of the formula of l is satisfiable while ensuring $|l| \leq 2 \times a \cdot |\phi|^c$ at all time. Initially, we set $l = \{(tr(\phi), \phi)\}$. Then, we loop over the variables $x_1, \ldots, x_n$ of ϕ and, at each iteration dealing with a variable x_i for some $1 \leq i \leq n$, we proceed in two steps:

   - for every pair $p = (tr(\varphi), \varphi)$ in l, we add $(tr(\varphi[x_i \to \mathsf{True}]), \varphi[x_i \to \mathsf{True}])$ and $(tr(\varphi[x_i \to \mathsf{False}]), \varphi[x_i \to \mathsf{False}])$ and we remove p.

- for all $1 \leq k \leq a \cdot |\phi|^c$, we keep (at most) one pair of the shape $(1^k, \varphi)$ and remove the other from l.

By construction, at the end of each iteration, we have $|l| \leq a \cdot |\phi|^c$ because, for all formula φ on which tr is applied, we have $tr(\varphi) \in \{1^k \mid 1 \leq k \leq a \cdot |\phi|^c\}$. Therefore, l is of size at most $2 \cdot a \cdot |\phi|^c$ (this upper bound may be achieved at the end of the first step). Furthermore, if at the beginning of an iteration we have the equivalence that ϕ is satisfiable if and only if one of the formula of l is satisfiable, we still have it at the end of the iteration. Indeed, it is straightforward that this holds at the end of the first step. Furthermore, if $tr(\varphi) = tr(\varphi')$ for two formulas φ and φ', then $\varphi \in \mathsf{SAT} \Leftrightarrow \varphi' \in \mathsf{SAT}$. It follows that the property still holds at the end of the second step and at the end of the iteration. Then, once these iterations are over, the final step consist in checking that the list l contains a pair $(1^k, \mathsf{True})$ for some k. The algorithm we described runs in polynomial time and decides SAT. Therefore $\mathsf{SAT} \in \mathsf{P}$.

Exercise 3 Regular language

Let REG denote the set of regular/rational languages.

1. Show that for all $L \in \mathsf{REG}$, L is recognized by a TM running in space 0 and time $n + 1$.
2. Exhibit a language recognized by a TM running in space $\log n$ and time $O(n)$ that is not in REG.

Solution:

1. Consider $L \in \mathsf{REG}$. It is recognized by a finite automaton $\mathcal{A}$. We consider the TM with the same states than $\mathcal{A}$ that, on an input w, simulates the execution of w on $\mathcal{A}$ and accepts if $\mathcal{A}$ does. This TM does not consumes any space and runs in time $n + 1$ (the $n + 1$-th step reads the first blank after the input and accepts/rejects).
2. The language $L = \{a^n \cdot b^n \mid n \geq 0\}$ is not regular and can be recognized by a TM that counts the number of a with a binary counter, decrements it for each b seen and accepts if, at the end of the word, the counter equal 0.

Exercise 4 On the existence of one-way functions

A one-way function is a bijection f from k-bit integers to k-bit integers such that f is computable in polynomial time, but f^{-1} is not. Prove that if there exist one-way functions, then

$$A \stackrel{\text{def}}{=} \{(x, y) \mid f^{-1}(x) < y\} \in (\mathsf{NP} \cap \mathsf{coNP}) \backslash \mathsf{P}\,.$$

Solution:

1. $A \in \mathsf{NP}$: consider a Turing machine that, on an input $w = (x, y)$, guesses a number c (with $|c| = |x|$) and checks in polynomial time that $f(c) = x$ and $c < y$. This non-deterministic TM runs in polynomial time and accepts the language A.

2. $A \in \mathsf{coNP} \Leftrightarrow \{(x, y) \mid f^{-1}(x) \geq y\} \in \mathsf{NP}$, which we solve as previously.

3. Assume that $A \in P$. Then we build a Turing machine running in polynomial time that computes f^{-1}: On an input x such that $|x| = n$, there is 2^n possibility for the value of $f^{-1}(x)$. We consider a TM that proceeds by a dichotomic search on the possible values v of $f^{-1}(x)$ until it finds some v with $(x, v-1) \in A$ and $(x, v) \notin A$ and deduce $f^{-1}(x) = v - 1$. Since at most n tests are necessary and each test is polynomial time, this TM runs in polynomial time.

Exercise 5 Too fast!

Show that $\mathsf{ATIME}(\log\ n) \neq \mathsf{L}$.

Solution: When considering $\mathsf{ATIME}(\log\ n)$, we do not even have the time to read the full input. So any language which is in L and needs for the input to be completely read will yield the result. For instance, one may use the palindromes language, or 0^n on a two letter alphabet, or 0^{2^k} on a one letter alphabet.