Advanced Complexity

TD n°5

Charlie Jacomme

October 11, 2017

Exercise 1 : Unary Languages

1. Prove that if a unary language is NP-complete, then $\mathsf{P} = \mathsf{NP}$.
   Hint : consider a reduction from SAT to this unary language and exhibit a polynomial time recursive algorithm for SAT
2. Prove that if every unary language in NP is actually in P, then $\mathsf{EXP} = \mathsf{NEXP}$.

Exercise 2 : On the existence of one-way functions

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

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

Exercise 3 : Prime Numbers

1. Show that `UNARY-PRIME` $= \{1^n \mid n \text{ is a prime number }\}$ is in P.
2. Show that `PRIME` $= \{p | p \text{ is a prime number encoded in binary }\}$ is in coNP.
3. We want to prove that `PRIME` is in NP. Use the following characterization of prime numbers to formulate a non-deterministic algorithm runing in polynomial time.
   A number p is prime if and only if there exists $a \in [2, p-1]$ such that :
   (a) $a^{p-1} \equiv 1[p]$, and
   (b) for all q prime divisor of $p-1$, $a^{\frac{p-1}{q}} \not\equiv 1[p]$
   To prove that your algorithm runs in polynomial time, you can admit that all common arithmetical operations on $\mathbb{Z}/p\mathbb{Z}$ can be performed in polynomial time.

Exercise 4 : Some P-complete problems

Show the following problems to be P-complete :

1. — INPUT : A set X, a binary operator $*$ defined on X, a subset $S \subset X$ and $x \in X$
   — QUESTION : Does x belongs to the closure of S with respect to $*$?
   Hint : for the hardness, reduce from Monotone Circuit Value
2. — INPUT : G a context-free grammar, and w a word
   — QUESTION : $w \in \mathcal{L}(G)$?
   Hint : for the hardness, reduce from the previous problem