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 P = 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 EXP = NEXP.

Solution:

1. Suppose we have a unary language U NP-complete. We then have a reduction R from SAT to U. $R(\phi)$ is computed in polynomial time, so we have p such that $|R(\phi)| \leq p(|\phi|)$. Basically, we can then use the self reducibility of SAT, but by cutting some recursions branching by using the fact that $R(\phi) = R(\psi)$ if and only if ϕ and ψ are both satisfiable or both un-satisfiable. We will write $\phi(t)$ where $t \in \{0,1\}^*$ to consider partial evaluation of ϕ where we substituted x_i with the truth value of t_i.
   This yields the algorithm, where n is the number of variables in ϕ :

```
Initialise hash table H
Sat(φ)
  if  |t| = n then return 'yes' if φ(t) has no clauses,
                               else return 'no'
  Otherwise, if R(φ(t)) ∈ H, then return H(R(φ(t)))
  Otherwise, return 'yes' if either Sat(φ(t0)) or Sat(φ(t1)).
      return no otherwise
     In both case, set H(R(φ(t)) to the answer
```

   There will be at most $p(n)$ different possibles values for the $R(\phi(t))$ (U is unary), so there will be at most $p(n)$ recursive call of the functions. And in every recursive call, we make a computation of R in time $p(n)$. So our algorithms runs in $O(p^2(n))$ wich is in P. Thus $SAT \in P$, and P = NP.

2. For a language L decided in time $T(n)$, we define $L_{pad} = \{1^{(x,10^{T(|x|)})}, x \in L\}$. Let $L \in NEXP$ recognized by N in time $T(n)$ exponential. We build $N' \in$ NP which recognizes L_{pad} :
   — On input 1^m, check the well-formdness to obtain $(x, 10^y) = m$
   — Simulate N on x for at most y step
   — Either return the result of N , or reject in case of time out.
   N' does recognizes L_{pad}, and it runs in polynomial times for the first step, and then y step for the second, with y being part of the input. Thus, $N' \in$ NP. But then by assumption, $L \in P$, and we have M a DTM which recognizes L_{pad} in polynomial time. We thus simply construct M' which is in exponential time, which given x computes $1^{(x,10^{T(|x|)})}$ and then simulate M with this input, and we are done.

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

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

Solution:

— $A \in \mathsf{NP}$: guess a number c, check that $f(c) = x$,i.e $c = f^{-1}(x)$, and finally, that $c < y$.

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

— $A \in P \Rightarrow f^{-1}$ computable in polynomial time

Exercise 3 : Prime Numbers

1. Show that `UNARY-PRIME` $= \{1^n \mid n$ is a prime number $\}$ is in P.
2. Show that `PRIME` $= \{p | p$ 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.

Solution:

1. For every $i < n$, we test if $i | n$
2. We guess the two factors
3. We guess the a, and then make $O(p)$ modulo exponentiation.

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

Solution:

1. The problem is in P as we can easily saturate until a fix point is reached. To show the hardness, we reduce Monotone Circuit Value, with gates with maximum two inputs. We are given a circuit $C = (V, E, label)$, and we define :

$$X = \{x^0, x^1 | x \in V\}$$

$$S = \{x^0 | x \in X \wedge label(x) = \perp\} \cup \{x^1 | x \in X \wedge label(x) = \top\}$$

$$x^i * y^j := \begin{array}{l} t^{i \wedge j} \text{ if } (x,t) \in E, (y,t) \in E, label(t) = \wedge \\ t^{i \vee j} \text{ if } (x,t) \in E, (y,t) \in E, label(t) = \vee \\ \text{undefined otherwise} \end{array}$$

Finally, we have :

$$v(x) = a \Leftrightarrow x^a \in Closure(S)(a \in \{0, 1\}$$

2. CKY is in polynomial time. For the hardness, we reduce from the previous problem. We are given $(X, S, x, *)$ and we define $G = (V, T, S, P)$ and $w \in T^*$ in the following way : w is the empty string, the set of variables $V = X$, there is only one terminal symbol, $T = \{a\}$, the initial variable is $S = \{x\}$, and the set of production is :

$$P := \{x \to yz : y * z = x\} \cup \{x \to \epsilon : x \in S\}$$

We then have :

$$x \in Closure(S) \Leftrightarrow \epsilon\ can\ be\ generated\ from\ x\ in\ G$$