# Advanced Complexity

## TD n°5

Charlie Jacomme

October 24, 2018

**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 $\phi$ and $\psi$ are both satisfiable or both un-satisfiable. We will write $\phi(t)$ where $t \in \{0,1\}^*$ to consider partial evaluation of $\phi$ where we substituted $x_i$ with the truth value of $t_i$.
   This yields the algorithm, where $n$ is the number of variables in $\phi$ :

```
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 $\texttt{UNARY-PRIME} = \{1^n \mid n$ is a prime number $\}$ is in $\mathsf{P}$.
2. Show that $\texttt{PRIME} = \{p|p$ is a prime number encoded in binary $\}$ is in $\mathsf{coNP}$.
3. We want to prove that $\texttt{PRIME}$ is in $\mathsf{NP}$. 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 $\mathsf{P}$-complete problems**

Show the following problems to be $\mathsf{P}$-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 $\mathsf{P}$ 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 :

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

**Exercise 5 : P-choice**

A language $L$ is said P-peek ($L \in \mathsf{P}p$) if there is a function $f : \{0,1\}^* \times 0,1^* \to \{0,1\}^*$ computable in polynomial time such that $\forall x, y \in \{0,1\}^*$ :

— $f(x, y) \in \{x, y\}$

— if $x \in L$ or $y \in L$ then $f(x, y) \in L$

$f$ is called the peeking function for $L$.

1. Show that $\mathsf{P} \subseteq \mathsf{P}p$
2. Show that $\mathsf{P}p$ is closed under complementary
3. Show that if there exist $L$ $\mathsf{NP}$-hard in $\mathsf{P}p$, then $\mathsf{P} = \mathsf{NP}$
4. Let $r \in [0, 1]$ a real number, we define $L_r$ as the set of words $b = b_1...b_n \in \{0;1\}^*$ such that $0, b_1...b_n \leq r$. Show that $L_r \in \mathsf{P}p$
5. Deduce that there exist a non-recursive language in $\mathsf{P}p$

**Solution:**

1. Let there be $A \in \mathsf{P}$. We set $f(x, y) = x$ if $x \in A$, and $f(x, y) = y$ otherwise.
2. Let there be $A \in \mathsf{P}p$ through $f$. Then, define $f'(x, y) = y$ if $f(x, y) = x$ and $f'(x, y) = x$ otherwise. $f'$ is then a peeking function for $A^c$ :
   — if $x \in A^c$ and $y \in A^c$, then $f'(x, y) = y \in A^c$
   — if $x \in A^c$ and $y \in A$, then $f(x, y) = y$ and $f'(x, y) = x \in A^c$
   — if $x \in A$ and $y \in A^c$, then $f(x, y) = x$ and $f'(x, y) = y \in A^c$ contains a language which is undecidable.
3. Let there be $A \in \mathsf{P}p$ through $f$ and $g$ a reduction from $SAT$ to $A$. Here is a polynomial algorithm for $SAT$ on input $\phi$ with $n$ variables, where we denote $\phi_0$ (resp. $\phi_1$) the formula $\phi$ in which the first variable is set to 0 (resp. 1).

```
For i from 1 to n do
  if f(g(φ0), g(φ1)) = g(φ0) then φ ← φ0
  else φ ← φ1
Accept iff φ = ⊤
```

4. $f(x, y) = min(x, y)$ is a valid selection function for $L_r$
5. $\mathsf{P}p$ is not countable as it contains $L_r$ for any $r \in [0, 1]$. Thus, $\mathsf{P}p$ contains a language which is not decidable.

**Exercise 6 : Complete problems for levels of PH**

Show that the following problem is $\Sigma_k^P$-complete (under polynomial time reductions).

$\Sigma_k\mathtt{QBF}$ : • INPUT : A quantified boolean formula $\psi := \exists X_1 \forall X_2 \exists ... Q_k X_k \phi(X_1, ..., X_k)$, where $X_1, ..X_k$ are $k$ disjoint sets of variables, $Q_k$ is the quantifier $\forall$ if $k$ is even, and the quantifier $\exists$ if $k$ is odd, $\phi$ is a boolean formula over variables $X_1 \cup \cdots \cup X_k$ ;

— QUESTION : is the input formula true ?

Define a similar problem $\Pi_k\mathtt{QBF}$ such that $\Pi_k\mathtt{QBF}$ is $\Pi_k^P$-complete.

**Solution:**

— If we are given some $X_1, ..., X_k$, we can check in polynomial time if $\phi(X_1, ..., X_k)$ is true. Thus, it is in $\Sigma_k^P$.

— Let there be $A \in \Sigma_k^P$. A can be expressed as follows :

$$x \in A \Leftrightarrow \exists y_1 \in \{0,1\}^{p|(x)|} \forall y_2 \in \{0,1\}^{p|(x)|} ... Q_k y_k \in \{0,1\}^{p|(x)|} (x, y_1, ..., y_k) \in B$$

with $B \in P$.

Let us assume that $Q_k = \exists$, the other case can be done in a similar fashion. Now, $\exists y_k \in \{0,1\}^{p|(x)|}(x, y_1, ..., y_k) \in B$ is in $\mathsf{NP}$, so by Cook's theorem, we have $\phi$ such that :

$$\exists y_k \in \{0,1\}^{p|(x)|}(x, y_1, ..., y_k) \in B \Leftrightarrow \exists z, \phi_{x,y_1,...,y_{k-1}}(z)$$

By inspecting Cook's proof, we can modify $\phi$ such that the input tape $x, y_1, ..., y_{k-1}$ appear as variables in $\phi$. We thus have

$$\exists y_k \in \{0,1\}^{p|(x)|}(x, y_1, ..., y_k) \in B \Leftrightarrow \exists z, \phi(x, y_1, ..., y_{k-1}, z)$$

And finally :

$$x \in A \Leftrightarrow \exists y_1, \forall y_2, ... \forall y_{k-1} \exists z, \phi(x, y_1, ..., y_{k-1}, z)$$

**Exercise 7 : Oracle machines**

Let $O$ be a language. A Turing machine with oracle $O$ is a Turing machine with a special additional read/write tape, called the oracle tape, and three special states : $q_{query}, q_{yes}, q_{no}$. Whenever the machine enters the state $q_{query}$, with some word $w$ written on the oracle tape, it moves **in one step** to the state $q_{yes}$ or $q_{no}$ depending on whether $w \in O$.

We denote by $\mathsf{P}^O$ (resp. $\mathsf{NP}^O$) the class of languages decided in polynomial time by a deterministic (resp. non-deterministic) Turing machine with Oracle $O$. Given a complexity class $\mathcal{C}$, we define $\mathsf{P}^{\mathcal{C}} = \bigcup_{O \in \mathcal{C}} \mathsf{P}^O$ (and similarly for $\mathsf{NP}$).

1. Prove that for any $\mathcal{C}$-complete language $L$, $\mathsf{P}^{\mathcal{C}} = \mathsf{P}^L$ and $\mathsf{NP}^{\mathcal{C}} = \mathsf{NP}^L$.
2. Show that for any language $L$, $\mathsf{P}^L = \mathsf{P}^{\bar{L}}$ and $\mathsf{NP}^L = \mathsf{NP}^{\bar{L}}$.
3. Prove that if $\mathsf{NP} = \mathsf{P}^{SAT}$ then $\mathsf{NP} = \mathsf{coNP}$.

**Solution:**

1. We do the proof for $\mathsf{NP}$. Let $B \in \mathsf{NP}^{\mathcal{C}}$, we have $N$ a polynomial NTM for $B$ with an oracle $C$, $C \in \mathcal{C}$. We also have a polynomial reduction $f$ such that : $x \in \mathcal{C} \Leftrightarrow f(x) \in A$. We build $N'$ for $B$ with oracle A, by simulating $N$ and replacing a call $u \in C$? with a call $f(u) \in A$?. $f$ is polynomial, so we are still in $\mathsf{NP}$, which concludes the proof.
2. We simply have to swap the states $q_{yes}$ and $q_{no}$ in the computation.
3. $\mathsf{P}^{SAT}$ is a deterministic class, so it is closed by complementation, so if $\mathsf{NP} = \mathsf{P}^{SAT}$, $\mathsf{coNP} = \mathsf{NP}$
4.

**Exercise 8 : Collapse of PH**

1. Prove that if $\Sigma_k^P = \Sigma_{k+1}^P$ for some $k \geq 0$ then $\mathsf{PH} = \Sigma_k^P$. (Remark that this is implied by $\mathsf{P} = \mathsf{NP}$).
2. Show that if $\Sigma_k^P = \Pi_k^P$ for some $k$ then $\mathsf{PH} = \Sigma_k^P$ (*i.e.* $\mathsf{PH}$ collapses).
3. Show that if $\mathsf{PH} = \mathsf{PSPACE}$ then $\mathsf{PH}$ collapses.
4. Do you think there is a polynomial time procedure to convert any QBF formula into a QBF formula with at most 10 variables ?

**Solution:**

1. We assume that $\Sigma_k^P = \Sigma_{k+1}^P$ for some $k \geq 0$ , we prove by induction that $\forall t \geq k, \Sigma_k^P = \Sigma_j^P$, For $j = i$, it is directly correcT. For $j > i$, $\Sigma_j^P = \mathsf{NP}^{\Sigma_{j-1}^P} = \mathsf{NP}^{\Sigma_i^P}$ by induction, and thus $\Sigma_j^P = \Sigma_{i+1}^P$ . By hypothesis, we then have $\Sigma_j^P = \Sigma_i^P$
2. With the previous question, we just have to prove that $\Sigma_k^P = \Sigma_{k+1}^P$ .
   Let there be $A \in \Sigma_{k+1}^P$. A can be expressed as follows :
   $$x \in A \Leftrightarrow \exists y_1 \in \{0,1\}^{p|(x)|} \forall ... Q_{k+1} y_{k+1} \in \{0,1\}^{p|(x)|} (x, y_1, ..., y_{k+1}) \in B$$
   with $B \in \mathsf{P}$.
   On input, $(x, y_1)$, decide if $\forall y_2 \in \{0,1\}^{p|(x)|} ... Q_{k+1} y_{k+1} \in \{0,1\}^{p|(x)|} (x, y_1, ..., y_{k+1}) \in B$ is a problem in $\Pi_k^P = \Sigma_k^P$. We can thus rewrite it as, with $C \in \mathsf{P}$ :
   $$\exists y_2 \in \{0,1\}^{p|(x)|} ... \overline{Q}_{k+1} y_{k+1} \in \{0,1\}^{p|(x)|} (x, y_1, ..., y_{k+1}) \in C$$
   Finally :
   $$x \in A \Leftrightarrow \exists y_1, y_2 \in \{0,1\}^{p|(x)|} \forall ... \overline{Q}_{k+1} y_{k+1} \in \{0,1\}^{p|(x)|} (x, y_1, ..., y_{k+1}) \in B$$
   with $B \in \mathsf{P}$. And this is the expression of a problem in $\Sigma_k^P$. Finally, $\Sigma_k^P = \Sigma_{k+1}^P$.
3. If $\mathsf{PH} = \mathsf{PSPACE}$, then QBF is in $\Sigma_k^P$ for some $k$. But QBF is a complete problem for $\mathsf{PSPACE}$, and thus $PH$. Let there be $B \in \mathsf{PH}$, it can be reduced to $QBF \in \Sigma_k^P$, so $B \in \Sigma_k^P$, and $\mathsf{PH} = \Sigma_k^P$
4. It is unlikely that $\mathsf{PH}$ collapses, and the statement would imply the previous question.

**Exercise 9 : Relativization**

Show that there is an oracle $O$ such that $\mathsf{P}^O = \mathsf{NP}^O$.

**Solution:**

We have $\mathsf{PSPACE} \subseteq \mathsf{NP}^{\mathsf{PSPACE}}$, we must show the converse. Let there be $N$ a polynomial NTM with oracle $A \in \mathsf{PSPACE}$. We can simulate $N$ in $\mathsf{PSPACE}$ on input $x$ by :

— enumerate all possible path of $N^A(x)$

— For each of them, compute the oracle calls

— accept if one of the path accepts.

Each path is in polynomial size, thus the enumeration is, and the oracle calls are $\mathsf{PSPACE}$. We do have $\mathsf{NP}^{\mathsf{PSPACE}} \subseteq \mathsf{PSPACE}$.