Complexity Theory

Homework Sheet 1 - Solutions

February 16, 2017

Exercise 1. For the following pairs of functions and relations (i.e. $\mathcal{O}, o, \omega, \Omega, \Theta$), prove for the two relations at each pair whether they hold or do not hold.

1.	$f(n) = n^{\log n}$	$g(n) = 2^{(\log n)^3}$	$g \in \Omega(f)$?	$f \in \Theta(g)$?
2.	$f(n) = \log n$	$g(n) = n$	$g \in \omega(f)$?	$f \in \mathcal{O}(g)$?
3.	$f(n) = n^5$	$g(n) = 100n^5$	$f \in o(g)$?	$g \in \Theta(f)$?

Exercise 2. By the fundamental theorem of arithmetic, any natural number x can be uniquely written as a product

$$x = p_1 \cdot p_2 \cdots p_k$$

with $p_1, \ldots, p_k$ prime numbers such that $p_i \leq p_j$ if $i < j$. This yields a function

$$f : \mathbb{N} \to \{0,1\}^* : x \mapsto \langle p_1, p_2, \ldots, p_k \rangle,$$

which maps a natural number x to a binary encoding of the prime factorization of x. See section 0.1 of the book for more explanation of the notation used here.

(a) Using the fact that there is a polynomial-time algorithm for testing primality,[1] show that deciding whether $z = f(x)$, when given x and z as input, can be done in polynomial time.

(b) Show that the set

$$\textsc{Factorization} = \{\langle x, i \rangle \mid \text{the } i\text{-th bit of } f(x) \text{ is } 1\}$$

is in **NP**. Here $\langle x, i \rangle$ is a binary encoding of the integers x, i.

(c) Show: if Factorization is **NP**-complete, then **NP** = **coNP**.
Hint: First show that Factorization is in **coNP**.

(d) Define

$$\textsc{Composite} = \{\langle x \rangle \mid x \in \mathbb{N} \text{ has at least two prime factors}\}.$$

Show: Composite is **NP**-complete if and only if **P** = **NP**.

[1]Which has been an open problem for a very long time, but solved in 2002 by Agrawal, Kayal and Saxena, see `http://en.wikipedia.org/wiki/AKS_primality_test`.

Solution. **(a)** Define a Turing machine M to do the following on input x, z in this order:

- Reject if z is not a binary encoding of a tuple $z = \langle a_1, ..., a_k \rangle$
- Reject if $a_i > a_{i-1}$ for some $i = 2, ..., k$.
- Test the primality of all a_i and reject if any of the a_i is not prime.
- Compute $y = a_1 \cdot ... \cdot a_k$ and accept if and only if $x = y$

All four steps take polynomial time (polynomial in size of the input x, z), and M accepts if and only if z is equal to $f(x)$ (i.e. the unique factorization of x).

(b) We have to show that there is a polynomial time Turing machine M that, for every $x \in \{0,1\}^*$ satisfies

$$x \in \textsc{Factorization} \iff \exists u \in \{0,1\}^{p(|x|)} \quad M(x,u) = 1$$

where p is some fixed polynomial. Define M to do the following, in this order:

- Reject if x is not of the form $x = \langle x', i \rangle$
- Reject if u is not of the form $u = \langle a_1, ..., a_m \rangle$
- Check if $u = f(x')$ using **(a)** in polynomial time.
  - If $u \neq f(x')$ then reject.
  - If $u = f(x')$ then accept iff the ith bit of u is 1.[2]

By definition, the only way to make M accept is by using a witness u that is equal to $f(x')$ and that has the ith bit equal to 1. Such a u exists if and only if $x \in \textsc{Factorization}$: if x is not in $\textsc{Factorization}$, then there is no u that can make M accept. Furthermore, the length of u is a polynomial of $|x|$ since every prime factor's representation is at most $|x|$ bits and there are at most $|x|$ prime factors, so $p(|x|) \in \mathcal{O}(|x|^2)$.

(c) Use Definition 2.20 in the book for **coNP**. To show that $\textsc{Factorization}$ is in **coNP**, we need to define a polynomial time Turing machine M that, for every $x \in \{0,1\}^*$ satisfies

$$x \in \textsc{Factorization} \iff \forall u \in \{0,1\}^{p(|x|)} \quad M(x,u) = 1$$

where p is some fixed polynomial. Notice the change of $\exists$ to $\forall$ compared to the **NP** case. Define M to do the following, in this order:

- Reject if x is not of the form $x = \langle x', i \rangle$
- **Accept** if u is not of the form $u = \langle a_1, ..., a_m \rangle$
- Check if $u = f(x')$ using **(a)** in polynomial time.
  - If $u \neq f(x')$ then **accept**.
  - If $u = f(x')$ then accept iff the ith bit of u is 1.[3]

The bold words mark the changes compared to the machine in exercise **(b)**. Let us check why this machine satisfies the requirement. If $x \in \textsc{Factorization}$ then $x = \langle x', i \rangle$ and the ith bit of $f(x')$ is 1. For any $u \in \{0,1\}^{p(|x|)}$, whether it is $f(x')$ or not, we see that

[2]So reject if i is larger than the size of u.

[3]Reject if i is larger than the size of u.

$M(x, u)$ will accept. If $x \notin$ FACTORIZATION then either x is not of the form $\langle x', i\rangle$ or the ith bit of $f(x')$ is 0. If x is not of the form $\langle x, i\rangle$ then M rejects and if it is of that form then there is a u that makes M reject, because M will reject if $u = f(x')$. The requirement is satisfied so FACTORIZATION $\in$ **coNP**.

If FACTORIZATION is **NP**-complete then for any $L \in$ **NP**, we have $L \leq_p$ FACTORIZATION, so $L \in$ **coNP** because FACTORIZATION $\in$ **coNP**. (A **coNP** verifier for L can first do the reduction in polynomial time and then run the verifier for FACTORIZATION). We therefore have **NP** $\subseteq$ **coNP**(1). By definition of **coNP** we have $L \in \mathbf{coNP} \overset{(2)}{\iff} \bar{L} \in \mathbf{NP}$. Therefore:

$$L \in \mathbf{coNP} \overset{(2)}{\implies} \bar{L} \in \mathbf{NP} \overset{(1)}{\implies} \bar{L} \in \mathbf{coNP} \overset{(2)}{\implies} L \in \mathbf{NP},$$

so **coNP** $\subseteq$ **NP** and we conclude **NP** $=$ **coNP**.

(d) A number is composite iff it is not prime[4]. Since primality testing is in P, we have COMPOSITE $\in$ **P** as we can flip the output of the turing machine for primality testing. If COMPOSITE is **NP**-complete then for all $L \in$ **NP** we have $L \leq_p$ COMPOSITE so then $L \in$ **P**. This holds for any L so we have **NP** $\subseteq$ **P** and hence **P** $=$ **NP**. For the other implication, assume **P** $=$ **NP**. In this case, COMPOSITE is **NP**-complete because any problem in **NP** can be computed in polynomial time and then mapped to a yes-instance or no-instance of COMPOSITE, in polynomial time. (For example, the polynomial time reduction can simply compute the result and output '4' (composite) if the result is true and output '3' (not composite) if the result is false.)

Exercise 3. A given function $f : \{0,1\}^* \to \{0,1\}^*$ is called *honest* if there is some real constant $c \geq 0$ such that $|f(x)|^c > |x|$ for all x, where $|x|$ is the length of the bitstring x. We say the *inverse* of a function f is polynomial-time computable if there is a Turing machine that always halts in polynomial time and that given an input $y \in \{0,1\}^*$ computes and outputs an $x \in \{0,1\}^*$ such that $f(x) = y$ or outputs NONE if no such x exists.

(a) Show that if **P** $=$ **NP** then for every honest function that is polynomial-time computable, the inverse is also polynomial-time computable.

(b) Prove the converse of the previous statement, i.e., show that if every honest, polynomial-time computable function has a polynomial-time computable inverse, then **P** $=$ **NP**. Hint: Which function would you have to invert to find the witness you are searching for?

Together, the result is sometimes known as the cryptographic theorem:

Theorem 1. **P** $=$ **NP** *if and only if every honest, polynomial-time computable function has a polynomial-time computable inverse.*

[4]Where the number 1 is an exception if you do not consider it prime, but a Turing machine can simply check for this and handle it separately.

Solution. **(a)** Let f be an honest function ($|f(x)|^c > |x|$) that is polynomial-time computable. Define the language L as the image of f, so $L = \{y \mid \exists x : f(x) = y\}$. Define a Turing machine M that gets inputs x, y and outputs 1 if and only if $f(x) = y$ by computing $f(x)$ in polynomial time and comparing it to y. We have $L \in \mathbf{NP}$ because M is a polynomial time verifier for L:

$$y \in L \iff \exists x \in \{0,1\}^{p(|y|)} \quad M(x, y) = 1,$$

where the polynomial p is given by $p(|y|) = |y|^c$. If $\mathbf{P} = \mathbf{NP}$ then $L \in \mathbf{P}$ so there is a polynomial-time machine that decides L. This polynomial-time machine solves the decision problem but this can be converted to a machine that finds the certificate by theorem 2.18 (page 55). This machine is a polynomial time Turing machine that computes the inverse of f so we conclude that the inverse of f is polynomial-time computable.

Alternative solution: Alternatively, without using theorem 2.18 directly, one can construct a slightly different language L' that allows us to compute the certificate by basically doing to construction of theorem 2.18 manually. Consider $L' = \{\langle y, x'\rangle \mid \exists x : f(x'x) = y\}$ where $x'x$ is the concatenation of x' and x. A polynomial-time verifier for L' can simply concatenate x' with the certificate u and check if $f(x'u) = y$. Note that there is an x that starts with x' (i.e. x agrees on the first $|x'|$ many bits with x') such that $f(x) = y$ if and only if there is a certifcate u for $\langle y, x'\rangle$. By a similar argument as above we have $L' \in \mathbf{NP}$. If $\mathbf{P} = \mathbf{NP}$ then $L' \in \mathbf{P}$ so there is a polynomial-time machine $M_{L'}$ that decides L'. Now we can use $M_{L'}$ that solves the decision problem L' to construct a polynomial-time machine M that finds the certificate. The machine $M(y)$ simply calls $M_{L'}(\langle y, 0\rangle)$ and $M_{L'}(\langle y, 1\rangle)$. If both reject then $M(y)$ outputs NONE. If one of them accepts (or even both) then M knows the first bit of x and can call $M_{L'}$ again to find out the second bit and so on untill it learned the full certificate which is the inverse of f. This is called prefix search.

(b) Let $L \in \mathbf{NP}$, then there exists a polynomial-time Turing machine M and a polynomial p such that $x \in L$ if and only if $\exists u \in \{0,1\}^{p(|x|)}$ such that $M(x, u) = 1$. Define the function $f : \{0,1\}^* \to \{0,1\}^*$ as

$$f(\langle x, u\rangle) = \begin{cases} \langle 1, x\rangle & M(x,u) = 1 \\ \langle 0, x\rangle & M(x,u) = 0 \end{cases}.$$

Note that $|\langle x, u\rangle| = \mathcal{O}(|x| + p(|x|))$ and $|f(\langle x, u\rangle)| = |\langle b, x\rangle| = |x| + 1$ (where $b \in \{0,1\}$). Because p is a polynomial there exists a c such that $|x| + p(|x|) < (|x| + 1)^c$, where c is independent of x. This shows that the function f is honest and since M is polynomial-time computable, so is f.

Now assume that f has a polynomial-time computable inverse. Consider the TM N that, on input x, computes the inverse of f on $\langle 1, x\rangle$. If the result of that computation is NONE then N outputs 0 and otherwise N outputs 1. Note that N decides L because $x \in L$ if and only if there there exists a u such that $f(\langle x, u\rangle) = \langle 1, x\rangle$. So the inverse of $\langle 1, x\rangle$ exists if and only if $x \in L$. Since the computation of the inverse is polynomial-time by assumption, N is a polynomial-time Turing machine that decides L. Therefore $L \in \mathbf{P}$ and we conclude $\mathbf{P} = \mathbf{NP}$.

Exercise 4. Show that $\mathbf{NP} \subseteq \mathbf{EXP}$.