Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
M. Ahmadi, P. Bamberger, H. Ghodselahi, Y. Maus, P. Schneider

Theoretical Computer Science - Bridging Course
Summer Term 2017
Exercise Sheet 8 - Sample Solution

Exercise 1: $\mathcal{NP}$ and Star Operation (5 points)

Show that $\mathcal{NP}$ is closed under the star operation.

*Remark 1: Let A be a language. The operation **star** $(\cdot^*)$ is defined as follows:*

$$A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A \text{ where } 1 \leq i \leq k\}.$$

*Remark 2: A collection of objects is **closed** under some operation if applying that operation (a finite number of times) to members of the collection returns an object still in the collection.*

Sample Solution

Let $A \in \mathcal{NP}$. Construct nondeterministic Turing machine M to decide A^* in nondeterministic polynomial time.

$M =$ "On input w :

1. Nondeterministically divide w into pieces $w = x_1 x_2 \dots x_k$.
2. For each x_i, nondeterministically guess the certificates that show $x_i \in A$.
3. Verify all certificates if possible, then *accept*. Otherwise, if verification fails, *reject*."

Exercise 2: The class $\mathcal{NPC}$ (8 points)

Let L_1, L_2 be languages (problems) over alphabets Σ_1, Σ_2. Then $L_1 \leq_p L_2$ (L_1 is polynomially reducible to L_2), iff a function $f : \Sigma_1^* \to \Sigma_2^*$ exists, that can be calculated in polynomial time and

$$\forall s \in \Sigma_1 : s \in L_1 \iff f(s) \in L_2.$$

Language L is called $\mathcal{NP}$-hard, if *all* languages $L' \in \mathcal{NP}$ are polynomially reducible to L, i.e.

$$L \ \mathcal{NP}\text{-hard} \iff \forall L' \in \mathcal{NP} : L' \leq_p L.$$

The reduction relation '$\leq_p$' is transitive ($L_1 \leq_p L_2$ and $L_2 \leq_p L_3 \Rightarrow L_1 \leq_p L_3$). Therefore, in order to show that L is $\mathcal{NP}$-hard, it suffices to reduce a known $\mathcal{NP}$-hard problem $\tilde{L}$ to L, i.e. $\tilde{L} \leq_p L$. Finally a language is called $\mathcal{NP}$-complete ($\Leftrightarrow$: $L \in \mathcal{NPC}$), if

1. $L \in \mathcal{NP}$ and
2. L is $\mathcal{NP}$-hard.

Show HittingSet $:= \{\langle \mathcal{U}, S, k\rangle \,|\, \text{universe } \mathcal{U} \text{ has subset of size} \leq k \text{ that } \textit{hits} \text{ all sets in } S \subseteq 2^{\mathcal{U}}\} \in \mathcal{NPC}$.[1]

Use that VertexCover $:= \{\langle G, k\rangle \mid \text{Graph } G \text{ has a } \textit{vertex cover} \text{ of size at most } k\} \in \mathcal{NPC}$.

*Remark: A **hitting set** $H \subseteq \mathcal{U}$ for a given universe $\mathcal{U}$ and a set $S = \{S_1, S_2, \ldots, S_m\}$ of subsets $S_i \subseteq \mathcal{U}$, fulfills the property $H \cap S_i \neq \emptyset$ for $1 \leq i \leq m$ (H 'hits' at least one element of every S_i).*
*A **vertex cover** is a subset $V' \subseteq V$ of nodes of $G = (V, E)$ such that every edge of G is adjacent to a node in the subset.*

Hint: For the poly. transformation ($\leq_p$) you have to describe an algorithm (with poly. run-time!) that transforms an instance $\langle G, k\rangle$ of VertexCover *into an instance $\langle \mathcal{U}, S, k\rangle$ of* HittingSet*, s.t. a vertex cover of size $\leq k$ in G becomes a hitting set of $\mathcal{U}$ of size $\leq k$ for S and vice versa(!).*

Sample Solution

We first show that hitting set belongs in $\mathcal{NP}$, by engineering a deterministic polynomial time verifier for it. Then we will prove that it is an $\mathcal{NP}$-hard problem, by reducing a known $\mathcal{NP}$-hard problem, vertex cover (as mentioned in the hint), to hitting set in polynomial time.

Guess and Check: Given a finite set $\mathcal{U}$, a collection S of subsets of $\mathcal{U}$, a positive integer k and a finite set H as a certificate, the following deterministic polynomial time verifier for hitting set confirms in polynomial time that $(\mathcal{U}, S)$ has a hitting set of size at most k. Let λ be the sum of the sizes of all the subsets S_i in S and δ the size of $\mathcal{U}$. Note that we can check if A is a subset of B with the following brute-force algorithm: $\forall a \in A$ check if $\exists b \in B : a = b$ which needs $\mathcal{O}(|A| \cdot |B|)$ comparisons. We can check if H is a subset of $\mathcal{U}$ that has at most k elements with $\mathcal{O}(k \cdot \delta)$ comparisons and if it contains at least one element from each subset S_i in the collection S, with $\mathcal{O}(\lambda \cdot k)$ comparisons. We accept iff both checks are true. These two checks are obviously equivalent to the problem's definition, so hitting set has a polynomial time verifier. Therefore it belongs in $\mathcal{NP}$.

Polynomial Reduction of VertexCover **to** HittingSet**:** We will create a polynomial time reduction from vertex cover to hitting set, proving that since vertex cover is $\mathcal{NP}$-hard, hitting set must also be $\mathcal{NP}$-hard.
The reduction takes as input an undirected graph $G = (V, E)$, where V is a set of nodes and E a set of edges defined over those nodes, as well as a positive integer k and outputs the set V, the collection $E = \{e_1, e_2, \ldots, e_n\}$ of subsets of V and the positive integer k. We claim the following equivalence holds:

"G has a vertex cover of size at most k" $\Leftrightarrow$ "(V, E) has a hitting set of size at most k"

Here is the proof:

"G has a vertex cover of size at most k" $\Leftrightarrow$
$\exists V' \subseteq V : |V'| \leq k$ and $\forall$ edge $e_i = \{u_i, v_i\} \in E, u_i \in V'$ or $v_i \in V' \Leftrightarrow$
$\exists V' \subseteq V : |V'| \leq k$ and $\forall$ subset e_i in collection E $\exists c \in e_i : c \in V' \Leftrightarrow$
"(V, E) has a hitting set of size at most k"

This reduction takes time linear to the size of the input (all it does is copy the input to the output), therefore polynomial. Also, as we showed, it is correct. Therefore, hitting set is at least as hard as vertex cover and since vertex cover is $\mathcal{NP}$-hard, so is hitting set.
One might notice that this reduction was rather straightforward. This makes sense, since vertex cover is a special version of hitting set, where each subset S_i in the collection S has exactly two elements of $\mathcal{U}$. Obviously, no problem can be harder than its generalization and since vertex cover is $\mathcal{NP}$-hard, hitting set (as a generalization of vertex cover) must also be $\mathcal{NP}$-hard.

[1] The power set $2^{\mathcal{U}}$ of some ground set $\mathcal{U}$ is the set of *all subsets* of $\mathcal{U}$. So $S \subseteq 2^{\mathcal{U}}$ is a collection of subsets of $\mathcal{U}$.

Exercise 3: Complexity Classes: Big Picture (2+3+2 points)

(a) Why is $\mathcal{P} \subseteq \mathcal{NP}$?

(b) Show that $\mathcal{P} \cap \mathcal{NPC} = \emptyset$ if $\mathcal{P} \neq \mathcal{NP}$.
Hint: Assume that there exists a $L \in \mathcal{P} \cap \mathcal{NPC}$ and derive a contradiction to $\mathcal{P} \neq \mathcal{NP}$.

(c) Give a Venn Diagram showing the sets $\mathcal{P}, \mathcal{NP}, \mathcal{NPC}$ for both cases $\mathcal{P} \neq \mathcal{NP}$ and $\mathcal{P} = \mathcal{NP}$.
Remark: Use the results of (a) and (b) even if you did not succeed in proving those.

Sample Solution

(a) If $L \in \mathcal{P}$ there is a deterministic Turing machine that decides L in polynomial time. Then $L \in \mathcal{NP}$ simply by definition since a deterministic Turing machine is a special case of a non-deterministic one.

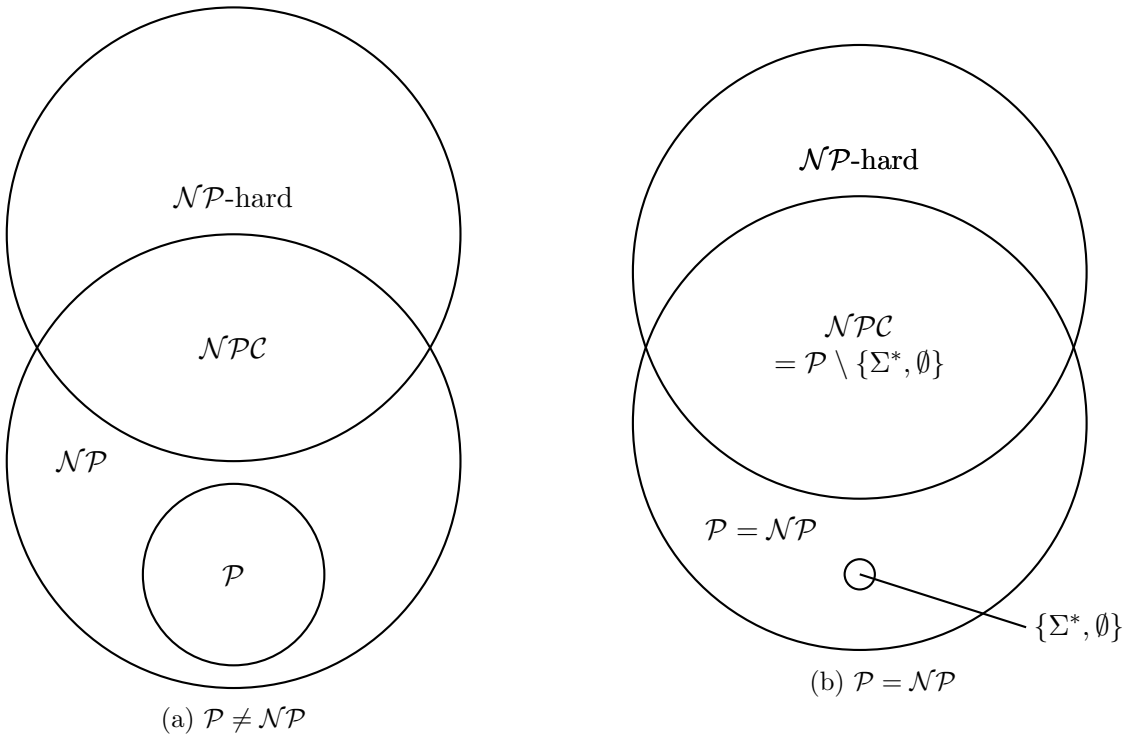
(b) As the hint suggests we assume that there is a language L which is $\mathcal{NP}$-complete and simultaneously solvable in polynomial time by a Turing machine. We use this language L to show that $\mathcal{NP} \subseteq \mathcal{P}$, which together with (a) implies $\mathcal{NP} = \mathcal{P}$, i.e., a contradiction to our premise $\mathcal{NP} \neq \mathcal{P}$. Hence L cannot exist if $\mathcal{NP} \neq \mathcal{P}$.

So let $L' \in \mathcal{NP}$. We want to show that L' is in $\mathcal{P}$ to obtain the contradiction. Since L is also $\mathcal{NP}$-hard, we can solve the decision problem L' via L by using the polynomial reduction $L' \leq_p L$. In particular for any string $s \in L'$ we have the equivalency $s \in L' \iff f(s) \in L$, where f is induced by the reduction.

We construct a Turing machine for L' that runs in poly. time. For instance s it first computes $f(s)$ in polynomial time and then uses the Turing machine for L as a subroutine to return the answer of $f(s) \in L$ in polynomial time. In total, we require only polynomial time to decide $s \in L'$ which means $L' \in \mathcal{P}$.

(c) See Figure 1. For the case $\mathcal{P} = \mathcal{NP}$, the notion of $\mathcal{NP}$-hardness becomes utterly meaningless since the class $\mathcal{NP}$ can be polynomially reduced to every other language except Σ^* and $\emptyset$. In order to show that $L' \leq_p L$ for an $L \neq \Sigma^*, \emptyset$ and for all $L' \in \mathcal{NP} = \mathcal{P}$, we need show that there is a polynomially computable mapping f such that $\forall s \in \Sigma^* :\ s \in L' \Leftrightarrow f(s) \in L$.

But such a mapping f always exists for $L \neq \Sigma^*, \emptyset$. We simply have to use a known 'yes-instance' $y \in L$ and a 'no-instance' $n \notin L$. Then we define for $s \in \Sigma^*$ that $f(s) := y$ if $s \in L'$ and $f(s) := n$ if $s \notin L'$. This obviously fulfills the above equivalency. Moreover f is polynomially computable since we can find out whether $s \in L'$ in polynomial time.



(a) $\mathcal{P} \neq \mathcal{NP}$

(b) $\mathcal{P} = \mathcal{NP}$

Figure 1: Venn-Diagram of the Language classes $\mathcal{P}, \mathcal{NP}, \mathcal{NPC}, \mathcal{NP}$-hard.