Theory of Computer Science

G. Röger — University of Basel
Spring Term 2019 — Computer Science

Exercise meeting 10 — Solutions

Exercise 10.1

The following statements are all wrong. In each case, explain in 1–2 sentences why the statement is wrong and what a correct version would be.

(a) To show that a problem X is NP-complete, it suffices to show that $X \in \text{NP}$ and $X \leq_{\text{p}} Y$ for some NP-complete problem Y.

Solution:

For X to be NP-complete, it has to be in NP and it has to be NP-hard. One way to show NP-hardness is to reduce another NP-hard problem to X (the NP-hardness then follows with the transitivity of reductions, see exercise 13.2 (a)). In this exercise the direction of the reduction is wrong: it should be $Y \leq_{\text{p}} X$ instead of $X \leq_{\text{p}} Y$.

(b) There is an NP-complete problem X that can be solved with an efficient deterministic algorithm, even if there is none for SAT.

Solution:

If there is such an algorithm for X, then SAT can be efficiently solved by reducing SAT to X. This is possible, since SAT $\in$ NP and since X is NP-hard. For every NP-complete problem X there are only two possible cases: either there are efficient algorithms for X and SAT ($\text{P} = \text{NP}$) or for none of them ($\text{P} \neq \text{NP}$).

(c) For every NP-hard problem X: $X \leq_{\text{p}}$ SAT.

Solution:

There are problems that are NP-hard but not in NP. Since SAT is NP-hard, all problems $X \in \text{NP}$ have $X \leq_{\text{p}}$ SAT. But NP-hard problems outside of NP cannot be reduced to SAT (otherwise they would be in NP).

(d) If there is a problem $X \in \text{P}$ such that $X \leq_{\text{p}} Y$ for some NP-complete problem Y then $\text{P} = \text{NP}$.

Solution:

Due to $\text{P} \subseteq \text{NP}$ it is true for *all* problems $X \in \text{P}$ and *all* NP-hard (and therefore also all NP-complete) problems Y that $X \leq_{\text{p}} Y$. Equality $\text{P} = \text{NP}$ would follow from $Y \leq_{\text{p}} X$ because then X was NP-complete and in P.