Technische Universität München (I7) **Summer term 2016**
Prof. J. Křetínský / Dr. C.H. Broadbent **18. Juli 2016**

Solution

Computational Complexity – Homework 6

Discussed on 30.05.2016.

Exercise 6.1

You have seen that 2SAT is in **NL**. Show that 2SAT is also **NL**-hard.

Solution: Since *REACHABILITY* is NL-hard and we know that NL is closed under complement, it suffices to show that there exists a logspace reduction from $\overline{REACHABILITY}$ to 2SAT. Suppose that we are given a graph $\mathcal{G} = \langle V, E \rangle$, an initial vertex v_0 and a target vertex v_f. From this we assign a variable x_v to each node in V and then construct $\phi_{\mathcal{G}} := \bigwedge_{(v_1,v_2)\in E}(x_{v_1} \to x_{v_2})$ (where $x_{v_1} \to x_{v_2}$ is $\neg x_{v_1} \vee x_{v_2}$). Finally we take the result of the reduction to be $\psi_{\mathcal{G}} := x_{v_0} \wedge x_{v_f} \wedge \phi_{\mathcal{G}}$.

$\psi_{\mathcal{G}}$ is a 2SAT instance and can be constructed in logspace (in the size of the reachability problem instance). Indeed the construction can be carried out in constant space: we can reuse the node IDs as variable IDs and in particular $\phi_{\mathcal{G}}$ is just a rewriting of E (copying node IDs from a pairs (v_1, v_2) and adding the appropriate Boolean operators.

It just remains to check that v_f is NOT reachable from v_0 iff $\psi_{\mathcal{G}}$ is SAT. For this it suffices to show that (i) if a valuation satisfies $x_{v_0} \wedge \phi_{\mathcal{G}}$ it must set x_v to true for all v reachable from v_0, and (ii) if a node v is unreachable from v_0, then there exists a valuation satisfying $x_{v_0} \wedge \phi_{\mathcal{G}}$ that sets x_v to false for every unreachable node v.

To prove (i) argue by induction on the number of steps to reach v from v_0. To prove (ii) take the valuation that sets x_v to true if v is reachable and false otherwise. Assume for contradiction that this is not a satisfying valuation. Since v_0 is trivially reachable it follows that there is a clause $x_{v_1} \to x_{v_2}$ in $\phi_{\mathcal{G}}$ such that x_{v_1} is set to true but x_{v_2} is set to false. But if this clause exists, $(v_1, v_2) \in E$ and by the definition of valuation v_1 is reachable whilst v_2 is not, which is a contradiction.

Exercise 6.2

Show that deciding the inequivalence of context-free grammars over one-letter terminal alphabet is Σ_2^p-hard. You can make use of Σ_2^p-hardness of integer expression inequivalence.

What does it imply for the equivalence problem?

Exercise 6.3

Under the assumption that $3\text{S}{\scriptstyle\text{AT}} \leq_p \overline{3\text{S}{\scriptstyle\text{AT}}}$ show that $\mathbf{NP} = \mathbf{PH}$.

Solution: If $3\textsc{Sat} \leq_p \overline{3\textsc{Sat}}$, then $\mathbf{NP} = \text{co}\mathbf{NP}$, i.e., $\mathbf{\Sigma}_1^p = \mathbf{\Pi}_1^p$. Consider now any $L \in \mathbf{\Sigma}_2^p$. We have

$$x \in L \text{ iff } \exists u \in \{0,1\}^{p(|x|)} \forall u \in \{0,1\}^{q(|x|)} \, : \, M(x,u,v) = 1.$$

The language

$$L_1\{(x,u) \mid \forall v \, : \, M(x,u,v) = 1\}$$

is then in co$\mathbf{NP}$ and, thus, in $\mathbf{NP}$, i.e., we find a TM M' and a polynomial r, s.t.,

$$(x,u) \in L_1 \text{ iff } \exists v \in \{0,1\}^{r(|x|+|u|)} \, : \, M'(x,u,v) = 1.$$

As $|u| = p(|x|)$, we may assume that $|v| = r(|x|)$ by adjusting r.

Hence,

$$x \in L \text{ iff } \exists uv \in \{0,1\}^{p(|x|)+r(|x|)} \, : \, M'(x,uv) = 1,$$

i.e., $L \in \mathbf{NP}$.

So, $\mathbf{\Sigma}_2^p \subseteq \mathbf{NP} = \text{co}\mathbf{NP}$. Similarly, $\mathbf{\Pi}_2^p \subseteq \mathbf{NP} = \text{co}\mathbf{NP}$.

Using induction, one now shows that $\mathbf{NP} = \mathbf{PH}$.

Exercise 6.4

Apart from the certificate definition and the alternative bounded alternating Turing machine characterization, there is one more standard characterization of the polynomial hierarchy via *oracles*.

For a language L, an oracle machine M^L is a Turing machine which can moreover do the following kind of computation steps. It can write down a word w on a special tape and ask whether $w \in L$ and it immediately receives the correct answer. One can also talk about this machine even when the oracle is not specified, then we write $M^?$.

Example: In Exercise 3.4 (a), you have constructed an example of M^{SAT} where $M^?$ is a polynomial time TM.

- Prove or disprove: for every $M^?$, if $A \subseteq B$ then $\mathcal{L}(M^A) \subseteq \mathcal{L}(M^B)$.
- Prove or disprove: if $A \subseteq B$ then $\mathbf{P}^A \subseteq \mathbf{P}^B$ (as classes).

The polynomial hierarchy can be defined inductively setting $\Sigma_0^p = \Pi_0^p = \mathbf{P}$ and

$$\Sigma_{i+1}^p = \mathbf{NP}^{\Sigma_i^p}$$

$$\Pi_{i+1}^p = \textbf{co-NP}^{\Sigma_i^p}$$

where A^B is the set of decision problems solvable by a Turing machine in class A with an oracle for some complete problem in class B.

- Show this yields the same hierarchy as the original definition.

One can also define $\Delta_{i+1}^p = \mathbf{P}^{\Sigma_i^p}$ and show that $\Delta_{i+1}^p \subseteq \Sigma_{i+1}^p \cap \Pi_{i+1}^p$ and it contains all languages expressible as Boolean combinations (unions, intersections, complements) of languages of Σ_i^p and Π_i^p.

- What is the relationship of these classes to $\mathbf{DP} = \{L \mid \exists M, N \in \mathbf{NP} : L = M \setminus N\}$?