Komplexitätstheorie 2004
Problemset 3

December 15, 2004

Problem 1 – Warm Up

Prove that **NL** = **NP** implies **NL** = **PH**.

Problem 2 – Functional Composition of FL-Computations

Let f and g be two functions which are computable within logarithmic space, i.e., $f, g \in \mathbf{FL}$. Prove that $f \circ g \in \mathbf{FL}$. Finally, argue that logspace reductions are transitive, i.e., if $A \leq_L B$ and $B \leq_L C$ then $A \leq_L C$ where $\leq_L$ denotes logspace reducibility.

Problem 3 – Fuzzy Logic

The syntactical structure of Gödel logic is same as in the case of ordinary Boolean logic. However, the semantics are different. Given a formula ϕ over a set of variables $X = \{x_1, \ldots, x_n\}$, we define

- an assignment to the variables X as a function $\tau : X \to [0, 1]$.
- and the evaluation function $\mathsf{m}(\phi, \tau)$ where ϕ is a formula over the variables X and τ is an assignment to the variables in X. We set
  - $\mathsf{m}(x_i, \tau) = \tau(x_i)$ with $x_i \in X$.
  - $\mathsf{m}(\sigma \to \rho, \tau) = \begin{cases} 1 & : \ \mathsf{m}(\sigma) \leq \mathsf{m}(\rho) \\ \mathsf{m}(\rho) & : \ \text{otherwise} \end{cases}$
  - $\mathsf{m}(\sigma \vee \rho, \tau) = \max(\mathsf{m}(\sigma), \mathsf{m}(\rho))$
  - $\mathsf{m}(\sigma \wedge \rho, \tau) = \min(\mathsf{m}(\sigma), \mathsf{m}(\rho))$

  Using $\neg\phi$ as shortcut for $\phi \to 0$ we find $\mathsf{m}(\neg\sigma, \tau) = \begin{cases} 1 & : \ \mathsf{m}(\sigma) = 0 \\ 0 & : \ \text{otherwise} \end{cases}$

We call the satisfiability problem in this logic GödelSat.

- Prove that the restricted version of GödelSat where we allow only assignments of the form $\tau : X \to \{0, 1/2, 1\}$ is **NP**-hard.
- Prove that the restricted version of GödelSat is in **NP**.
- Prove that general GödelSat is in **NP**.

Problem 4 – P ≠ NP is not enough

Proving **P** ≠ **NP** would be a major break-through. However, even after such a proof a lot of questions would remain open which are of central interest. In particular, we will look at the complexity of FACTORING which is the computational problem of finding the prime factors for any given positive integer.

- Define the decision problem FBIT with $\langle x, p \rangle \in$ FBIT iff x are a positive integer such that the pth bit of the largest prime-factor of x is set to 1.
- Define PRIMES as the language of all prime numbers – it is known that PRIMES ∈ **NP** ∩ **coNP**[1].

1. Prove the following statement: **P** = **NP** ∩ **coNP** implies FACTORING ∈ **FP** (**FP** is the class of functions which are computable within polynomial time).
   (a) Prove FBIT ∈ **NP** and FBIT ∈ **coNP** separately by using the assumption PRIMES ∈ **P**. (Hint: The two machines to prove this will be *very* similiar)
   (b) Conclude that PRIMES ∈ **P** implies FBIT ∈ **NP** ∩ **coNP** .
   (c) Prove that **P** = **NP** ∩ **coNP** implies FACTORING ∈ **FP**.
2. Based on this: What is the relationship of the statements **P** ≠ **NP**, **P** ≠ **NP** ∩ **coNP** and FACTORING ∉ **FP**?

Problem 5 – Upward Translations

Prove that **P** = **NP** implies **EXP** = **NEXP**.

Problem 6 – DSPACE(n) ≠ NP

Although we are currently unable to prove that either **PSPACE** ≠ **NP** or **PSPACE** = **NP**, we can show the following statement:

$$\mathbf{DSPACE}(n) \neq \mathbf{NP}$$

Note, that the inequality is the only relationship between **DSPACE**(n) and **NP** that we are able to prove.

Prove that the two classes are different. Hint: In the lecture we proved that **NP** is closed under log-space reductions. Show that **DSPACE**(n) is not closed under log-space reductions and conclude that the two classes are in fact different.

Can you apply the same proof-technique to other classes? For example can you prove that

- **E** ≠ **EXP**
- **E** ≠ **PSPACE**

[1] In fact, a more recent result shows PRIMES ∈ **P**. Beforehand, it was known that PRIMES ∈ **ZPP**, i.e., the class of randomized polynomial time algorithms, which are expected to produce a definite result within a constant number of trials.

- ...

The definitions are: $\mathbf{EXP} = \bigcup_{c=1}^{\infty} \mathbf{DTIME}(2^{n^c})$ and $\mathbf{E} = \bigcup_{c=1}^{\infty} \mathbf{DTIME}(2^{cn})$.

Can you generalize? Argue why **P** is defined as the set of all decision problems which are solvable within *polynomial* time. Also, explain why **EXP** is preferred over **E**.

A note on $\mathbf{DSPACE}(n) \neq \mathbf{NP}$: By the same argument, you can separate $\mathbf{NSPACE}(n)$ from **NP** and **P** and other classical classes. The classes of nondeterministic linear-space Turing machines coincides with the those languages that are recognizable by context sensitive grammars. Thus, the class of context sensitive languages is different from **P**, **NP**, **PSPACE** ....

Problem 7 – HornSat $\in$ P

HornSat is another restriction of Sat. An instance of HornSat contains only clauses which contain at most one positive literal ($x \vee \neg y \vee \neg z$ is a Horn-clause, but $x \vee y \vee \neg z$ is not a Horn clause). HornSat is the problem of deciding whether such an instance is satisfiable or not.

Prove HornSat $\in$ **P**.

Problem 8 – FSat $\in \mathbf{FP}^{\mathbf{NP}}$

FSat is the problem of determining whether a CNF-formula is satisfiable, *and* if so, of computing such a satisfying assignment. Prove that a machine which runs in polynomial time and has access to an **NP**-oracle can solve this problem.