Computational Complexity

Take-home exam

Hand in via Canvas before Friday June 2, 2023, at 23:59
`https://canvas.uva.nl/courses/36220/assignments/418954`

Exercise 1 (*4pt; a: 1pt, b: 2pt, c: 1pt*).

(a) Prove that $\mathsf{BPP}^{\mathsf{BPP}} = \mathsf{BPP}$.

(b) Prove that $\mathsf{NP}^{\mathsf{BPP}} \subseteq \mathsf{BPP}^{\mathsf{NP}}$.

  – *Note:* this is a hard question.

(c) Prove that if $\mathsf{NP} \subseteq \mathsf{BPP}$, then $\Sigma_2^{\mathrm{p}} \subseteq \mathsf{BPP}$.

  – *Note:* for (c), you may use the statements of (a) and (b), even if you didn't manage to prove these.

Definition 1. Variable forgetting in propositional logic is defined as follows. Let φ be a propositional logic formula over the propositional variables X, and let $W \subseteq X$ be a subset of variables. The result of *forgetting W in φ* is a propositional logic formula ψ over the variables $X \setminus W$ such that for all truth assignments $\alpha : X \setminus W \to \{0, 1\}$ it holds that α makes ψ true if and only if α can be extended to a truth assignment $\beta : X \to \{0, 1\}$ such that β makes φ true. Note that such a formula ψ is not unique—there are other formulas ψ' that are logically equivalent, and thus also express the result of forgetting W in φ.

For example, consider the propositional formula $\varphi = (x_1 \to x_3) \wedge (x_2 \to \neg x_3)$ over the variables $X = \{x_1, x_2, x_3\}$, and let $W = \{x_3\}$. The formula $\psi = (\neg x_1 \vee \neg x_2)$ expresses the result of forgetting W in φ.

We say that a function f implements forgetting for propositional logic if for each propositional formula φ and each $W \subseteq \mathrm{Vars}(\varphi)$ it holds that $f(\varphi, W)$ is a propositional logic formula that expresses the result of forgetting W in φ.

Exercise 2 (*3pt; a: 1pt, b: 2pt*).

(a) Prove that if there is a function f that implements forgetting for propositional logic and that can be computed in polynomial time, then $\mathsf{P} = \mathsf{NP}$.

(b) Prove that if there is a function f that implements forgetting for propositional logic and that is of polynomial-size, then the Polynomial Hierarchy collapses. A function f is of *polynomial-size* if there exists some polynomial p such that $|f(x)| \leq p(|x|)$—i.e., the size of the result is upper bounded by a polynomial of the size of the input, but there are no restrictions on the time needed to compute the function (or whether it is computable at all).

  – *Hint:* use the fact that $\mathsf{NP} \subseteq \mathsf{P/poly}$ implies that $\mathsf{PH} = \Sigma_2^{\mathrm{p}}$.

  – *Hint:* for different values of $\ell \in \mathbb{N}$, consider the formula:

$$\varphi_\ell = \bigwedge_{1 \leq i \leq (2\ell)^3} (y_i \to c_i),$$

  where $c_1, \ldots, c_{(2\ell)^3}$ is an enumeration of all possible clauses of size 3 over the variables $x_1, \ldots, x_\ell$.

Definition 2. The decision problem 2-IN-5-SAT is defined as follows. The input is a propositional logic formula φ in CNF, over n variables and containing m clauses, where each clause consists of exactly 5 literals. A truth assignment α *exactly-2-in-5-satisfies* a clause c if there are exactly two literals in c that are made true by α (and thus three literals in c are made false by α). The question is to decide if there exists a truth assignment α that exactly-2-in-5-satisfies all clauses of φ.

Exercise 3 (*3pt; a: 1pt, b: 1pt, c: 1pt*)**.**

(a) Prove that 2-IN-5-SAT is not solvable in polynomial time, assuming $\mathsf{P} \neq \mathsf{NP}$.

(b) Let $\rho < 1$. A *ρ-approximation algorithm* for 2-IN-5-SAT takes an input for 2-IN-5-SAT and outputs a truth assignment α that exactly-2-in-5-satisfies at least $\rho \cdot u_\varphi$ clauses, where u_φ is the maximum number of clauses of φ that can be simultaneously exactly-2-in-5-satisfied.

Prove that there exists some $\rho < 1$ such that there is no polynomial-time ρ-approximation algorithm for 2-IN-5-SAT, assuming $\mathsf{P} \neq \mathsf{NP}$.

(c) Prove that 2-IN-5-SAT is not solvable in time $2^{o(n)} \cdot m^{O(1)}$, assuming the ETH.

Hint:

- The problem 1-IN-3-SAT is defined similarly—i.e., the input contains clauses of size three, a truth assignment α exactly-1-in-3-satisfies a clause c if it makes exactly one literal in c true, and the problem is to decide if there exists a truth assignment that exactly-1-in-3-satisfies all clauses.
- Consider the following reduction from 3SAT to 1-IN-3-SAT—which we state here without proving its correctness. (To be precise, this reduction assumes that all clauses are of size exactly three.)
  - Let $\varphi = c_1 \wedge \cdots \wedge c_m$ be an input for 3SAT. We will construct an input ψ for 1-IN-3-SAT by replacing each clause c_i by three new clauses $d_{i,1}, d_{i,2}, d_{i,3}$, as follows.
  - Let $c_i = (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$. Then $d_{i,1} = (\neg\ell_{i,1} \vee a_i \vee b_i)$, $d_{i,2} = (\ell_{i,2} \vee b_i \vee c_i)$ and $d_{i,3} = (\neg\ell_{i,3} \vee c_i \vee d_i)$, where a_i, b_i, c_i and d_i are fresh variables.
- Build forth on this reduction to construct a reduction from 3-SAT to 2-IN-5-SAT, and use this reduction to answer (a), (b) and (c).