Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Emo Welzl, lecturer, and Timon Hertli, teaching assistant

Friday, April 19, 2013

Satisfiability of Boolean Formulas Spring 2013
Special Assignment Set 2

- The solution is due on **Friday, May 3, 2013, 10:15 (strict!)**. Please bring a print-out of your solution with you to the exercise session. If you cannot attend, you may alternatively send your solution as a PDF to `timon.hertli@inf.ethz.ch`. We will send out a confirmation that we have received your file. Make sure you receive this confirmation within the day of the due date, otherwise complain timely.
- Please solve the exercises carefully and then write a nice and complete exposition of your solution using a computer, where we strongly recommend to use LaTeX. A tutorial can be found at `http://www.cadmo.ethz.ch/education/thesis/latex`.
- For geometric drawings that can easily be integrated into LaTeX documents, we recommend the drawing editor IPE, retrievable at `http://ipe7.sourceforge.net/` in source code and as an executable for Windows.
- You are welcome to discuss the tasks with your colleagues, but we expect each of you to hand in your own, individual write-up. **You should not share your write-up or (even worse) your source code!**
- There will be three special assignments this semester. Each of them will be graded and the average grade will contribute 30% to your final grade.
- This is a theory course, which means: if an exercise does not explicitly say "you do not need to prove your answer" or "justify intuitively", then a formal proof is **always** required.
- As with all exercises, the material covered in special assignments is relevant for the final exam.

Exercise 1 (Strongly Satisfiable Formulas) *(40 points)*

Definition 1. *Let F be a boolean formula. A satisfying assignment α is called a* strongly satisfying assignment *of F (α* strongly satisfies *F) if any assignment β that differs from α in one variable (i.e. has Hamming distance 1) also satisfies F.*

If F has a strongly satisfying assignment, then F is called strongly satisfiable*.*

Solve the following tasks:

(a) Let F be a CNF formula. Prove that α strongly satisfies F *if and only if* α satisfies at least two literals of every clause $C \in F$.

(b) Let F be a strongly satisfiable (≤ 3)-CNF formula. Give a polynomial time algorithm that finds a satisfying assignment of F.

(c) Let F be a strongly satisfiable (≤ 4)-CNF formula. Give a randomized polynomial time algorithm that finds a satisfying assignment of F with probability at least $\frac{1}{2}$.

(d) Suppose you have an polynomial time algorithm $\mathcal{A}$ that finds a satisfying assignment of any strongly satisfiable boolean formula (not necessarily in CNF). Prove that in this case, $\mathcal{P} = \mathcal{NP}$.
HINT: Convert a CNF formula F into a boolean formula G (in polynomial time) s.t. if F is satisfiable then G is strongly satisfiable, and if F is not satisfiable then G is not satisfiable.

Exercise 2 (Monotone Formulas) *(60 points)*

Definition 2. *A boolean formula (not necessarily in CNF) is called* monotone *if it consists only of* positive *literals, constants* 0 *and* 1*, logical OR operators ($\vee$), logical AND operators ($\wedge$), but no negations,* negative *literals, or other logical operators.*

0, y and $x \vee (y \wedge (1 \vee y))$ are monotone formulas, while $\bar{x} \vee y$ and $x \to y$ are not monotone formulas.

Deciding whether a monotone formula is satisfiable is easy:

(a) Give a polynomial time algorithm that decides satisfiability of monotone formulas.

In this exercise we want to show that *counting* satisfying assignments of a monotone formula is as hard as counting satisfying assignments of a general boolean formula.

First we reduce counting satisfying assignments of a general formula to counting satisfying assignments of a (≤ 3)-CNF formula (and thus a CNF-formula).

(b) Argue that in Theorem 1.1 (from the lecture notes) the number of satisfying assignments are preserved.

Now we reduce counting satisfying assignments of a CNF formula to counting satisfying assignments of two monotone formulas.

(c) Let F be a CNF formula over V of the following form: Every clause C contains either only positive literals or only negative literals.

Construct in polynomial time (w.r.t. the total size of F) two monotone formulas G, H over V s.t. $|\mathrm{sat}_V(F)| = |\mathrm{sat}_V(G)| - |\mathrm{sat}_V(H)|$.

Note that we defined the notion $\mathrm{sat}_V(F)$ only for CNF formulas, but it extends to arbitrary formulas in the obvious way.
HINT: Let G be the part of F that contains only positive literals. Let H be such that its satisfying assignments are exactly those that satisfy G but not F. To build H remember De Morgan's laws and use the fact that H need *not* be in CNF.

(d) Let F be an arbitrary CNF formula over V. Construct in polynomial time two monotone formulas G, H over some variable set V' s.t. $|\mathrm{sat}_V(F)| = |\mathrm{sat}_{V'}(G)| - |\mathrm{sat}_{V'}(H)|$.
HINT: Replace every negative literal $\bar{x}$ of F by a new variable n_x, to obtain a monotone CNF formula over $2|V|$ variables. Add clauses that enforce $x \neq n_x$ and then use (c).

Now we show that it is sufficient to count satisfying assignments of *one* monotone formula.

(e) Let F be a monotone formula on n variables V. Give, for any $n' \geq n$, a monotone formula F' on n' variables V' with $|\mathrm{sat}_V(F)| = |\mathrm{sat}_{V'}(F')|$.

(f) Let F and G be monotone formulas over V_F and V_G, respectively. Construct in polynomial time a monotone formula H over some set V_H with

$$|\mathrm{sat}_{V_H}(H)| = |\mathrm{sat}_{V_F}(F)| \cdot 2^{|V_G|} + |\mathrm{sat}_{V_G}(G)| \cdot 2^{|V_F|} - |\mathrm{sat}_{V_F}(F)| \cdot |\mathrm{sat}_{V_G}(G)|.$$

HINT: The solution is very easy. Remember that you don't need to build a CNF formula.

(g) Let F and G be monotone formulas over V_F and V_G, respectively. Construct in polynomial time a monotone formula H s.t. from the number of satisfying assignments of H you can determine easily the number of satisfying assignments of F and G.
HINT: Tune the formulas by (e) and then use (f).