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, March 22, 2013

Satisfiability of Boolean Formulas Spring 2013
Special Assignment Set 1

- The solution is due on **Friday, April 12, 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.
- **You need to solve only 3 out of 4 exericses (each exercise is 30 points). Only these 3 exericses will be relevant for your grade.** You can solve all 4 exercises and we correct them, **but you have to tell us beforehand which ones should count for your grade! If you do not tell us, we will count Exercise 1, 2, and 3.** The exercises might be of varying difficulty.
- 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 (Partial Satisfaction) *(30 points)*

Let F be a CNF formula with the following properties:

(i) $\square \notin F$

(ii) For $x \in \mathrm{vbl}(F)$: $\{\bar{x}\} \notin F$, i.e. 1-clauses do not have negative literals.

(iii) For $x \in \mathrm{vbl}(F)$: If $\{x\} \in F$ then any clause containing the literal $\bar{x}$ must have size at least 5. (Formally: If $\{x\} \in F$, then for all $C \in F$ with $\bar{x} \in C$ we have $|C| \geq 5$)

Note that x is a variable, so we treated positive and negative literals quite differently!

Answer the following questions (a)-(d):

(a) For which $k \in \mathbb{N}_0$ is F guaranteed to be k-satisfiable?

(b) Give a CNF formula F with the above properties s.t. all assignments on $\mathrm{vbl}(F)$ satisfy at most a $\frac{3}{4}$-fraction of the clauses.

(c) Prove that for any CNF formula F with the above properties there exists an assignment that satisfies at least a $\frac{3}{4}$-fraction of the clauses. Does this also hold for weighted CNF formulas?

Suppose we replace condition (iii) by the following condition (iii') (the 5 is replaced by a 4):

(iii') For $x \in \text{vbl}(F)$: If $\{x\} \in F$ then any clause containing the literal $\bar{x}$ must have size at least **4**. (Formally: If $\{x\} \in F$, then for all $C \in F$ with $\bar{x} \in C$ we have $|C| \geq \mathbf{4}$)

(d) Give a weighted CNF formula (F, μ) with the properties (i), (ii), (iii') where no assignment satisfies a $\frac{3}{4}$-fraction of the clauses.
HINT: Build a counterexample similar to the formulas used in the proof showing that Theorem 2.4 is tight. You need to give only one counterexample, but you need to show that all assignments are bad. You are allowed to use a computer algebra tool (e.g. Mathematica). In this subtask, you are allowed to give a plot as "proof".

Exercise 2 (Factoring Using SAT) *(30 points)*

The integer factorization problems asks, for an non-prime integer $x \in \mathbb{N}$, to find integers $a, b \geq 2$ s.t. $a \cdot b = x$. Suppose you have an algorithm that decides satisfiability of a (≤ 3)-CNF formula over n variables in time $f(n)$. We would like to use this algorithm to factor an integer.

(Factoring is considered a hard problem, but it's not known to be NP-hard. In particular, on quantum computers there exists an efficient algorithm, but it's not known if good enough quantum computers can be built to factor large numbers.)

Let $a, b, c \in \{0, 1, \ldots, 2^n - 1\}$ and suppose there are Boolean variables a_i, b_i, c_i (for $i = 1, \ldots, n$). Denote by $\alpha(a, b, c)$ the assignment that assigns to a_i, b_i and c_i the binary representations of a, b, c in the sense that $a = \sum_{i=0}^{n-1} a_i \cdot 2^i$ etc.

(a) Give a 3-CNF formula F over the variables a_i, b_i, c_i (for $i = 1, \ldots, n$) and $2n$ additional variables s.t. $F^{[\alpha(a,b,c)]}$ is satisfiable if and only if $a + b = c$ (modulo 2^n).
HINT: How would you do addition by hand? Start with a 4-CNF formula with n additional variables. It should be straightforward to find a 3-CNF formula with $3n$ additional variables. How to reduce the number of additional variables to $2n$?

(b) Give a 3-CNF formula F over the variables a_i, b_i, c_i (for $i = 1, \ldots, n$) and (at most) $4n^2$ additional variables s.t. $F^{[\alpha(a,b,c)]}$ is satisfiable if and only if $a \cdot b = c$ (modulo 2^n).
HINT: How would you do multiplication by hand?

(c) Suppose you have an algorithm $\mathcal{A}$ that decides satisfiability of a (≤ 3)-CNF formula over n variables in time $f(n)$.

Give an algorithm $\mathcal{B}$ that, given as input an n-bit integer $x \geq 2$, runs in time $O(f(10n^2) \cdot poly(n))$ and does the following: If x is prime, it outputs "prime". If x is not prime, it outputs integers $a, b \geq 2$ s.t. $a \cdot b = x$.
HINT: Use (b), and run $\mathcal{A}$ several times. How can you ensure that $a \neq 1$ and $b \neq 1$?

Exercise 3 (SAT Algorithms) *(30 points)*

Let F be a CNF formula. Denote by $l(F) := \sum_{C \in F} |C|$ the *length* of F, i.e. the total number of occurrences. In this exercise, we consider algorithms for SAT where the running time depends on $l(F)$.

(a) Give an algorithm *deciding* satisfiability of F in time $O(2^{\frac{1}{2}l(F)}\text{poly}(l(F)))$.

(b) Give an algorithm *counting* the number of satisfying assignments of F in time $O(2^{\frac{1}{2}l(F)}\text{poly}(l(F)))$.
HINT: Use a similar approach as in (a), but you might need to be more careful.

(c) Prove the following for any $x \in \mathrm{vbl}(F)$:

Let F_x be the set of all clauses of F containing the literal x. Let $F_{\bar{x}}$ be the set of all clauses of F containing literal $\bar{x}$. Let F_R be the set of all resolvents obtained by resolution with one clause from F_x and the other from $F_{\bar{x}}$.

Then F is satisfiable if and only if $(F \setminus (F_x \cup F_{\bar{x}})) \cup F_R$ is satisfiable.

(d) Use (c) to conclude again that resolution is complete (i.e. prove again Theorem 1.5 using (c)).

(e) Give an algorithm *deciding* satisfiability of F in time $O(2^{\frac{1}{3}l(F)}\mathrm{poly}(l(F)))$.
HINT: Use (c).

Exercise 4 (k-CNF vs. $(\leq k)$-CNF) *(30 points)*

Let $k \in \mathbb{N}$ be a constant. Suppose you have an algorithm deciding satisfiability of k-CNF formulas in time $O(f(n))$ where n is the number of variables. We call CNF formulas F and G *SAT-equivalent* if either both are satisfiable or both are unsatisfiable.

(a) Let F be a $(\leq k)$-CNF formula over n variables. Give a SAT-equivalent k-CNF formula on $n+k$ variables.

(b) Give a $(\leq k)$-CNF formula over n variables that has no SAT-equivalent k-CNF formula on $n+k-1$ variables.
HINT: How many variables does a k-CNF formula need to be unsatisfiable?

(c) Argue that $n^k \in O(f(n))$.

(d) Suppose you have an algorithm deciding satisfiability of $(\leq k)$-CNF formulas over n variables in time $g(n)$. Give an algorithm that decides satisfiability of $(\leq k)$-CNF formulas over $n+1$ variables in time $2g(n) + O(n^k)$.

(e) Give an algorithm deciding satisfiability of $(\leq k)$-CNF formulas in time $O(f(n))$.
HINT: Use (a), then (d) together with the algorithm deciding satisfiability of k-CNF formulas, and then (c).

This shows that when we want to give an algorithm that decides k-SAT (without any other restriction on the formulas), smaller clauses don't cause any problems.