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 Robin Moser, teaching assistant

Tuesday, March 13, 2012

Satisfiability of Boolean Formulas Spring 2012
Special Assignment Set 1

- The solution is due on **Tuesday, April 3, 2012**. Please bring a print-out of your solution with you to the lecture. If you cannot attend (and please only then), you may alternatively send your solution as a PDF to `robin.moser@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.
- 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 (Improving Branching Algorithms) *(50 Points)*

In this task, we are interested in improving on the result of Exercise 1.20. To begin with, study Appendix A.2 in order to familiarize yourself with the way linear recurrence relations can be resolved. Include Footnote 2 in your studies: it explains how the existence of a solution can be proved which you will need to know in order to derive the following technical tool.

(a) Suppose we have a recursive algorithm A running on CNF formulas whose running time we would like to bound. The algorithm performs some polynomial-time basic operations and then invokes itself recursively multiple times. Let $r, s \geq 1$ be integers and $c_{ij} \geq 0$ integers for $1 \leq i \leq r$, $1 \leq j \leq s-1$ satisfying $\sum_{j=1}^{s-1} c_{ij} > 1$ for all i. Suppose we know that for every input formula F on $n > s$ variables, there exists $i \in \{1..r\}$ such that the algorithm makes exactly c_{ij} recursive calls on formulas with $n-j$ variables for $j \in \{1..s-1\}$. Prove that algorithm A then runs in time $\mathcal{O}(\sigma^n \cdot \mathrm{poly}(|F|))$ on any formula F on n variables where

$$\sigma := \max \left\{ \lambda \in (1,\infty) \;\middle|\; \exists i : \lambda^s = \sum_{j=1}^{s-1} c_{ij} x^{s-j} \right\}.$$

We now return to the generalization of Exercise 1.20.

(b) Give a polynomial time algorithm which, on input F a (≤ 3)-CNF where every variable appears at most once, outputs the number of satisfying assignments.

(c) Design a recursive branching algorithm which, on input F a (≤ 3)-CNF on n variables, outputs the number of satisfying assignments in time $\mathcal{O}(1.776^n \cdot \text{poly}(|F|))$.

HINT: If the formula is such that you can neither use (b) nor other obvious special cases, then it can select a pair of 3-clauses which overlap in at least one variable. In the latter case, there are several possibilities how the two clauses can overlap. Come up with some branching rule for each case. In the end, use (a) in order to establish the total running time of your algorithm.

Exercise 2 (A Local Lemma for Inhomogeneous Formulas) *(50 Points)*

We have studied the Lovász Local Lemma in Chapter 2 and have found that k-CNF formulas in which every clause has a neighborhood of no more than 2^{k-2} other clauses are always satisfiable. This version of the Local Lemma requires all clauses to have exactly the same size. There are more general versions that allow the clauses to be of varying sizes. Smaller clauses are harder to satisfy so if a neighborhood contains smaller clauses it must contain less clauses. Here is one of the more general versions of the Local Lemma.

Theorem 1. *Let F be a CNF formula for which there exists a mapping $\mu : F \to (0,1)$ associating a number with every clause such that*

$$\forall C \in F \ : \ 2^{-|C|} \ \leq \ \mu(C) \cdot \prod_{D \in \Gamma_F(C)} (1 - \mu(D)) \, .$$

Then F is satisfiable.

The proof of this version works the very same way we proved the simpler symmetric version in the lecture notes, it just involves longer expressions, so this is not overly interesting and we do not ask you to do this here (do it if you want to check whether you have understood the proof in the lecture notes). Instead, we want to derive a condition that is more intuitive to read.

Theorem 2. *If F is any (≥ 3)-CNF formula such that for all clauses $C \in F$, we have*

$$\sum_{D \in \Gamma_F(C)} \frac{1}{2^{|D|}} \leq \frac{1}{4},$$

then F is satisfiable[1]*.*

(a) Show that Theorem 2.3 (in the lecture notes) is implied by Theorem 2 (on this sheet).

(b) Use Theorem 1 to prove Theorem 2.
HINT: Try μ-values of magnitude $\mu(C) = \mathcal{O}(2^{-|C|})$ and use for example an estimate of the type $\forall x \in (0,a) : 1 - x > e^{-bx}$ for suitable constants $0 < a < 1$ and $1 < b < 2$. You will need to do some parameter optimization to make this work.

We now seek to make Theorem 2 algorithmic as well. There is a way to prove that Theorem 1 can already be made constructive, from which the algorithmic version of Theorem 2 readily follows. But this algorithmic proof[2] is considerably more involved and less intuitive than the one we presented in Chapter 2*. However, if we are okay with a very slight weakening of the statement, we can adapt the proof in Chapter 2* so as to cater for inhomogeneous formulas. Our goal is the following version.

[1] note how nicely this compares to the homogeneous and inhomogeneous bounds on the total number of clauses we can have in a formula while still guaranteeing satisfiability that we have looked at as warm-up exercises at the beginning of Chapter 2

[2] if you are interested in seeing this, it was part of last semester's APC course and we are always happy to provide you with a copy; but this is not mandatory material for this course

Theorem 3. *If F is any (≥ 3)-CNF formula such that for all clauses $C \in F$, we have*

$$\sum_{D \in \Gamma_F^+(C)} \frac{1}{2^{|D|}} \leq \frac{1}{8},$$

then F is satisfiable and there exists a randomized algorithm to find a satisfying assignment which runs in expected polynomial time.

Note the differences: we sum over the *inclusive* instead of the exclusive neighborhood and we have lowered the constant by a factor of 2.

We establish this theorem by means of the following proof program. Recall that a code $\mathcal{C} \subseteq \{0,1\}^*$ is called *prefix-free*, if there are no two codewords $v, w \in \mathcal{C}$ such that v is a prefix of w.

(c) Show: If $(r_1, r_2, \ldots, r_t)$ are positive integers such that

$$\sum_{i=1}^{t} 2^{-r_i} \leq 1,$$

then there exists a prefix-free code $\mathcal{C}$ with $|\mathcal{C}| = t$ such that $\mathcal{C} = \{w_1, w_2, \ldots, w_t\}$ with $|w_i| = r_i$ for all $1 \leq i \leq t$.

(d) Establish Theorem 3 by adapting[3] the proof in Chapter 2* using the result from (c).

[3]adapting here means: you do not have to copy the whole chapter, just write what you do *differently*, but make sure that you highlight *everything* that needs to be changed