**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, May 9, 2014

# *Satisfiability of Boolean Formulas Spring 2014*
# *Special Assignment Set 3*

- The solution is due on **Friday, May 23, 2014, 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 (PPZ and Unit Clause Propagation) *(50 points)*

Remember the PPZ algorithm from the lecture notes on page 136. PPZ determines the value of a variable as follows: If $\{x\} \in F$, $\alpha(x)$ is set to 1; if $\{\bar{x}\} \in F$, $\alpha(x)$ is set to 0; otherwise $\alpha(x)$ is guessed uniformly at random.

In this exercise we consider a more powerful strategy to determine the value of $x$. For this, look again at unit clause reduction on page 79 (we ignore the second part of the output). Obviously if $\square \in \mathrm{uc}(F^{[x \mapsto b]}, \mathrm{vbl}(F) \setminus \{x\})$, then to possibly obtain a satisfying assignment, we have to set $x$ to $1 - b$.

Making use of this observation we modify ppz by replacing the condition $\{x\} \in F$ with $\square \in \mathrm{uc}(F^{[x \mapsto 0]}, \mathrm{vbl}(F) \setminus \{x\})$ and by replacing the condition $\{\bar{x}\} \in F$ with $\square \in \mathrm{uc}(F^{[x \mapsto 1]}, \mathrm{vbl}(F) \setminus \{x\})$. Call the resulting algorithm **ppz_imp**. Do the same replacements in encode on page 133 and call the resulting algorithm **encode_imp**.

(a) Argue that **ppz_imp** runs in polynomial time and that the probability that **ppz_imp** finds a satisfying assignment of a satisfiable ($\leq k$)-CNF formula over $n$ variables is at least $2^{-n+n/k}$ (that is, **ppz_imp** doesn't perform worse than ppz).

In the following subtasks let $F$ be an ($\leq k$)-CNF formula over $n$ variables $V$ with *exactly one* satisfying assignment $\alpha$ (on $V$). For (b)-(f) consider the case $k = 3$; for (g) consider general $k \geq 3$.

For $x \in V$, define guessed($x$) to be the probability over $\pi \in_{\text{u.a.r.}} \mathcal{S}_V$ that in **encode_imp**$(\alpha, F, V, \pi)$, in the step of $x$, $\alpha(x)$ is added to the string $\epsilon$.

(b) Let $x \in V$. Suppose $x$ has two (different) critical clauses, or a critical ($\leq 2$)-clause. Show that guessed($x$) $\leq \frac{7}{12}$.

(c) Let $x \in V$. Suppose there is $y \in V$ such that $x$ has a critical 3-clause $C_x$ with $y \in \mathrm{vbl}(C_x)$, and $y$ has a critical clause $C_y$ with $x \notin \mathrm{vbl}(C_y)$. Show that guessed($x$) $\leq \frac{37}{60}$.

(d) Let $x \in V$. Suppose $x$ has exactly one critical clause $C_x$, and this clause contains another variable $y$ (as a dissatisfied literal), and all critical clauses for $y$ contain the variable $x$. Show that in this case guessed($x$) $\leq \frac{7}{12}$.
HINT: What happens if we flip both $x$ and $y$ in $\alpha$?

(e) Prove that **ppz_imp** finds the satisfying assignment of $F$ with probability at least $2^{-\frac{37}{60}n}$.

(f) The analysis of ppz also works for multiple satisfying assignments. Where does the same approach fail here?

(g) Adapt (b)-(e) for general $k \geq 3$, i.e. use the same ideas to get corresponding bounds (the bounds will depend on $k$).
NOTE: Feel free to solve (b)-(e) directly for general $k$.

## Exercise 2 (An Interesting Way to Count 2-SAT) *(50 points)*

Let $F = \{C_1, C_2, \ldots, C_m\}$ be an $(\leq 2)$-CNF formula $F$ with $m$ clauses over $n$ variables $V$. Assume that both $F$ and its clauses are ordered. In this exercise we devise an interesting way to count the number of satisfying assignments of $F$.

Suppose we are given an assignment $\beta$ on $W \subseteq V$ and we want to count the number of satisfying assignments of $F^{[\beta]}$ on $(V \setminus W)$. We do this using the following (very wasteful) algorithm $\mathcal{A}(F, \beta)$:

If $\square \in F^{[\beta]}$, return 0. If $\beta$ satisfies $F$ (that is, $F^{[\beta]} = \{\}$), return $2^{|V|-|W|}$.

Otherwise, let $C_i$ be the first clause in $F$ (w.r.t. the fixed order specified on $F$) that is not satisfied by $\beta$. Let $C_i' := C_i^{[\beta]}$. For all $2^{|C_i'|}$ assignments $\beta'$ on $\mathrm{vbl}(C_i')$, call $\mathcal{A}(F, \beta \cup \beta')$; then return the sum of the results.

Let $d(F, \beta)$ the maximum recursion depth of $\mathcal{A}(F, \beta)$: 0 if $\mathcal{A}$ does not make recursive calls; 1 plus the maximum of $d(F, \beta \cup \beta')$ otherwise.

(a) Argue that $\mathcal{A}(F, \beta) = |\mathrm{sat}_{V \setminus W}(F^{[\beta]})|$ and that $\mathcal{A}$ runs in time at most $\min\{4^{d(F,\beta)}, 2^{|V \setminus W|}\}\mathrm{poly}(n)$.

(b) Define the following map $m$ from $\beta$ with $d(F, \beta) > 0$: As in the algorithm, let $C_i$ be the first clause in $F$ not satisfied by $\beta$ and $C_i' := C_i^{[\beta]}$. Choose $\beta'$ on $\mathrm{vbl}(C_i')$ such that all literals of $C_i'$ evaluate to false. Let $m(\beta)$ be the pair of the assignment $\beta \cup \beta'$ and the information which literals of $C_i$ are in $C_i'$ (there are 3 possibilities: first, second, or all). Show that $m$ is injective.

(c) For given $n$, what $l$ maximizes $\binom{n}{l}2^l$?

(d) Suppose $l \geq \frac{9}{10}n$ (this condition is not tight). Let $B(l, s)$ be the set assignments $\beta$ on some $W \in \binom{V}{l}$ with $d(F, \beta) > s$.

For $\beta \in B(l, s)$ do the following: As $d(F, \beta) > s$, for some choices of the algorithm we will reach recursion depth $s + 1$. Consider the first time this happens; at this time $s + 2$ calls are still active; the call of $\mathcal{A}$ we are currently in and $s+1$ previous calls that lead to this call. For the first $s$ previous calls, look at the $s$ clauses $C_i'$ that lead to a recursive call. Let $\gamma$ be the assignment that dissatisfies all these $s$ clauses $C_i'$ (on the corresponding variables). Additionally for each of the $s$ clauses $C_i'$, write down which literals of $C_i$ are in $C_i'$, and which literals of $C_i'$ are chosen to be satisfied by the algorithm in the branch we are currently in.

Show that from $\beta \cup \gamma$ and the additional information you can get back $\beta$. Conclude that $|B(l, s)| \leq n \cdot \binom{n}{l+s}2^{l+s} \cdot 3^s \cdot 3^s$.
HINT: Try to apply the findings of (b) iteratively.

(e) Set $l = \lceil \frac{99}{100}n \rceil$ and $s = \frac{1}{1000}n$. Pick the set $W$ uniformly at random from $\binom{V}{l}$. Then pick an assignment $\beta \in_{\text{u.a.r.}} \{0,1\}^W$. Show that the probability that $\beta \in B(l, s)$ is in $O(2^{-\epsilon' n})$ for some $\epsilon' > 0$.

(f) Use the previous subtask to give an algorithm for counting the number of satisfying assignments of an $(\leq 2)$-CNF formula $F$ on $n$ variables in *expected* time $O(2^{(1-\epsilon)n})$ for some $\epsilon > 0$.

NOTE: This algorithm might seem pointless. However for larger $k$ it is the fastest (w.r.t. *expected* time) algorithm for $k$-SAT counting.

Also note that unlike the randomized SAT algorithms you have seen, this algorithm will always output the correct answer (we have a so-called Las Vegas algorithm).