15-414 — Bug Catching — Fall 2006

Instructor: Edmund Clarke

Teaching assistant: Himanshu Jain

Assignment 1

Due date: Tuesday, September 26, 2006

We will discuss homework related doubts on Tuesday, September 12, 2006

Some Reminders:

- Read the Policies section on the course web site before you start working on this assignment. If you have questions, contact the course staff.
- We are allowing handwritten solutions, although typeset ones are preferred. If you handwrite, WRITE CLEARLY, or we will revert to the old system of requiring you to typeset solutions.
- The cover page of your submission must clearly display the assignment number, your name, and your Andrew ID.

1 Tautologies

For each of the following formula report if it is a tautology or give an assignment which falsifies the formula.

[Hint: Try to solve these without constructing truth tables. Think about falsifying the formula.]

(a) $((q \rightarrow (p \vee (q \rightarrow p))) \vee \neg(p \rightarrow q)) \rightarrow p$

(b) $((a \rightarrow x) \wedge (a \Leftrightarrow \neg y) \wedge (y \Leftrightarrow (z \wedge w))) \rightarrow ((x \Leftrightarrow y) \rightarrow (z \rightarrow w))$

(c) $((p \rightarrow q) \wedge (q \Leftrightarrow T)) \rightarrow (p \Leftrightarrow T)$, where T denotes *true*.

2 Conversions to Normal forms without introducing new variables

(a) Convert following formula to CNF: $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$

(b) Convert following formula to CNF: $\neg(p \rightarrow (\neg(q \wedge (\neg p \rightarrow q))))$

(c) Convert following formula to DNF: $(x \vee y) \wedge (y \rightarrow \neg x) \wedge (z \Leftrightarrow \neg x)$

3 Convert to CNF by introducing new variables

(a) $(a \wedge b) \vee (c \wedge d) \vee (e \wedge f) \vee (g \wedge h)$

(b) $(x_1 \rightarrow (x_2 \rightarrow x_3)) \vee (x_4 \rightarrow x_5)$

4 Applying Davis-Putnam method

Check whether the following formulas are satisfiable by applying the Davis-Putnam rules. Also report a satisfying assignment when a formula is satisfiable.

(a) $(a \vee \neg b) \wedge (d \vee \neg e) \wedge (b \vee \neg c) \wedge (c \vee \neg d) \wedge (\neg a \vee z) \wedge e \wedge \neg z$

(b) $(\neg a \vee \neg b) \wedge (\neg a \vee \neg c) \wedge (\neg b \vee \neg c) \wedge (\neg d \vee \neg e) \wedge (\neg d \vee \neg f) \wedge (\neg e \vee \neg f) \wedge (a \vee d) \wedge (b \vee e) \wedge (c \vee f)$

5 Equivalence

(a) Let $\mathcal{V}_\phi(A)$ denote the value of a propositional formula A under an assignment ϕ. Show that if A is a well formed formula (wff) in which no propositional connective other than $\Leftrightarrow$ occurs, and ϕ is an assignment, then $\mathcal{V}_\phi(A) = F$ iff the number of occurrences of variables p in A such that $\phi(p) = F$ is odd. Note that F denotes $false$.

[Hint: Use induction on the structure of the formula (to be explained in class).]

(b) Prove that a wff of propositional calculus containing only the connective $\Leftrightarrow$ is a tautology iff each propositional variable occurs an even number of times.

[Hint: Use the part (a).]

6 Pigeonhole formulas

The *pigeonhole principle* states that if n pigeons are put into m pigeonholes, and if $n > m$, then at least one pigeonhole must contain more than one pigeon.

Given n pigeons and m pigeonholes, we wish to write a CNF formula G such that G is satisfiable iff each pigeon can be put in some pigeonhole and each pigeonhole has atmost one pigeon.

Let $x_{i,j}$ denote a propositional variable which is true when pigeonhole i contains pigeon j. We will give the formula for G when $n = 3$ and $m = 3$.

$E := (x_{1,1} \vee x_{2,1} \vee x_{3,1}) \wedge (x_{1,2} \vee x_{2,2} \vee x_{3,2}) \wedge (x_{1,3} \vee x_{2,3} \vee x_{3,3})$

$H_1 := (\neg x_{1,1} \vee \neg x_{1,2}) \wedge (\neg x_{1,1} \vee \neg x_{1,3}) \wedge (\neg x_{1,2} \vee \neg x_{1,3})$

$H_2 := (\neg x_{2,1} \vee \neg x_{2,2}) \wedge (\neg x_{2,1} \vee \neg x_{2,3}) \wedge (\neg x_{2,2} \vee \neg x_{2,3})$

$H_3 := (\neg x_{3,1} \vee \neg x_{3,2}) \wedge (\neg x_{3,1} \vee \neg x_{3,3}) \wedge (\neg x_{3,2} \vee \neg x_{3,3})$

$G := E \wedge H_1 \wedge H_2 \wedge H_3$

(a) What is the meaning of clause $(x_{1,3} \vee x_{2,3} \vee x_{3,3})$.

(b) What does E stand for?

(c) Explain the meaning of H_1, H_2, H_3.

(d) Is G satisfiable? If yes, give a satisfying assignment.

(e) Write G when $n = 3$ and $m = 2$. Is G satisfiable now?

7 Using a SAT solver

Most fast SAT solvers require the input formula in CNF. The input CNF formula is specified in the `DIMACS` format. Consider the following file `sample.cnf` in `DIMACS` format.

```
p cnf 4 5
1 0
2 -3 0
-4 -1 0
-1 -2 3 4 0
-2 4 0
```

The first line (p cnf x y) says that the input is a CNF formula containing `x` variables and `y` clauses. Our example has 4 variables (1, 2, 3, 4) and five clauses. The negation of a variable is denoted by putting a minus sign in front of the variable number. Each clause is described in a line terminated by a zero. Note that 0 cannot be used as a variable number. So `sample.cnf` denotes the following CNF formula: $1 \wedge (2 \vee \neg 3) \wedge (\neg 4 \vee \neg 1) \wedge (\neg 1 \vee \neg 2 \vee 3 \vee 4) \wedge (\neg 2 \vee 4)$.

Some publically available fast SAT solvers are `MiniSat, zChaff, siege`. For this assignment we will use the `MiniSat` SAT solver which was the fastest SAT solver in the SAT-competitions of 2005 and 2006. You can run `MiniSat` SAT solver simply by the following command:

```
MiniSat_v1.14_linux sample.cnf sample.result
```

The file `sample.cnf` is a description of a CNF formula in DIMACS format. `MiniSat` reports whether the given formula is (un)satisfiable in the file `sample.result`. If the formula is satisfiable, then a satisfying assignment is also written to `sample.result`.

You can find a binary of `MiniSat` solver for linux in `/afs/andrew.cmu.edu/usr21/himanshu/bin`. You need to include this binary in your path variable to call the `MiniSat` solver as shown above.

(a) The solution to Problem 2 (a), (b) involved generating CNF formulas. Express these formulas in DIMACS format. Run `MiniSat` SAT solver on these formulas and report whether the formulas are satisfiable or unsatisfiable.

(b) On the course webpage you will find five files in DIMACS format. Run `MiniSat` SAT solver on these files and report number of variables, number of clauses in each file, time taken by `MiniSat` to check satisfiability, and the result (SAT/UNSAT). We do not need satisfying assignment for satisfiable formulas. Report a timeout if `MiniSat` does not finish in 15 minutes.