CS510 Assignement #3 (Due 3/26 before class)

March 12, 2015

1 Proof

(20p) Prove the following sequents:

(a) $p \to q,\ r \to s \ \vdash\ p \lor r \to q \lor s$

(b) $(p \land q) \lor (p \land r) \ \vdash\ p \land (q \lor r)$

(c) $\vdash\ \neg p \to (p \to (p \to q))$

(d) $q \ \vdash\ (p \land q) \lor (\neg p \land q)$ using LEM

(e) $p \land q \ \vdash\ \neg(\neg p \lor \neg q)$

2 Validity

(15p) Determine the validity of the following semantic entailment by constructing the truth table.

$$\neg r \to (p \lor q),\ r \land \neg q \ \models\ r \to q$$

3 CNF and SAT solving

(25p)

`If I study, then I will not fail basket weaving 101. If I do not play cards to often, then I will study. I failed basket weaving 101. Therefore, I played cards too often.`

(a) Model the above statement to a formula. Turn the formula to its CNF form to decide validity. (15p)

(b) Decide its validity by formulating it as a satisfiability problem and solving it. Please show the parse tree and the value assignment process. (10p)

4 SAT solving

(20p)

(a) Find a formula such that the cubic SAT solving algorithm cannot decide its satisfiability, that is, it fails to converge on a deterministic value for any unmarked node. Please do not use examples from the text book. (10p)

(b) Devise a try-and-backtrack SAT solving algorithm that operates on formulas with $\neg$ and $\wedge$. You can use the linear algorithm as a primitive to construct your algorithm, e.g. *linear_solve()* means performing the linear inference as much as possible. (10p)

5 The DIMACS Format

(10p) 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: $1 \wedge (2 \vee \neg 3) \wedge (\neg 4 \vee \neg 1) \wedge (\neg 1 \vee \neg 2 \vee 3 \vee 4) \wedge (\neg 2 \vee 4)$.

Express the following clause set in the `DIMACS` format. Use the variable name i in the `DIMACS` format for the variable named x_i in the the above clauses. (For example, use 4 for x_4.)

$$\omega_1 = x_1 \vee x_3 \vee x_4 \qquad \omega_3 = \neg x_1 \vee x_2 \vee x_3$$
$$\omega_2 = x_1 \vee \neg x_2 \vee x_3 \qquad \omega_4 = x_1 \vee \neg x_2 \vee x_4$$
$$\omega_5 = x_2 \vee x_4 \qquad \omega_6 = x_3 \vee x_4$$

6 Using a SAT solver

(10p)

Some publicly available fast SAT solvers are `MiniSat`, `zChaff`, `siege`. For this assignment you will install and 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:

```
/path/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`.

Run `MiniSat` on the `DIMACS` file you made from the previous problem. What is the output stored in results file?