**McMaster University**
**Department of Computing and Software**
**Dr. W. Kahl**

**FP 2006**
**Assignment 3**

# CAS 743 — Functional Programming

6 February 2006

## SAT Solving

The task is easily summarised: Use Haskell to implement a SAT solver that accepts problems in DIMACS .cnf format, and test it at least on the uf20-91 benchmarks from the following web page:

http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/benchm.html

In detail:

(a) Make sure you understand the SAT problem and at least section 2.1 of the distributed format discription (check for example http://en.wikipedia.org/wiki/Boolean_satisfiability_problem).

(b) Design and implement datatypes to represent propositional formulae.

(c) Implement a simple, "obviously correct" function that transforms arbitrary formulae into disjunctive normal forms. (As a SAT solving algorithm, this is not particularly sophisticated, but it has the advantage that it is relatively simple…)

(d) Implement a function to read conjunctive normal forms from files in the .cnf format described in section 2.1 of the format discription.

(e) Define appropriate output (normally using the format of section 2.4) and driver functions for the following tasks:

  (1) Produce **any** solution (simple satisfiability problem).

  (2) Produce **all** solutions (as represented by the DNF).

(f) Test and benchmark against at least the `uf20-91` benchmarks, and document the outcome.

(g) Also write a **solution checker**, i.e., a function that reads both a problem file and a solution file, and checks whether the variable assignments contained in the solution file really are solutions the given problem (without calculating the DNF again — this should work in polynomial time!).

(h) Discuss possible optimisations with respect to CNF/DNF formula representations and the DNF algorithm. Select some optimisation to implement, and compare the benchmark results.