Assignment 1

CSC 2512—Fall 2015

Out: Oct 13th, 2015
Due: Monday Oct 26th (email me your write-up)
Worth 15% of your final mark.

1 Questions

Q1. 4/10 From the backtracking based DPLL algorithm one can obtain both a resolution refutation if the input is UNSAT and also a satisfying assignment if the input is SAT. With the resolution based DP algorithm a resolution refutation can be extracted when the input is UNSAT, but can we also extract a satisfying assignment when the input is SAT?

Describe how you would modify DP so that a satisfying assignment can be extracted from a SAT input.

Q2. 6/10 You can down the minisat SAT from the https://github.com/niklasso/minisat (see also the minisat web site http://minisat.se/). Modify the solver so that instead of learning 1-UIP clauses it learns All-UIP clauses. Use some problems from 2007 Sat competition (industrial track) to test how well minisat does All-Decision clauses vs. its standard 1-UIP clauses. You can obtain a full selection of these problems from the url http://www.cril.univ-artois.fr/SAT07/benchs/industrial.tar, but you do not need to use all these problems just use some of them so that do some experiments that can be completed in a reasonable amount of time.

If you are using a Mac you should be able to make minisat by removing the --static linker flag, and executing make r. That is, make the release version not the dynamic version (although there is certainly some way of making a dynamically linked version on a mac).

The routine you will want to modify is analyze contained in minisat/core/Solver.cc The first do–while loop computes the initial 1-UIP clause (you should not need to modify the simplify clause part after this). The main things to know about the code are the following:

- CRef is a clause reference. There is a global variable ca which is indexed by a clause reference (ca[CRef]) to obtain a clause. The clause can itself be indexed like a vector or array to obtain the literals in it.
- reason() returns the reason for an implied literal. These reasons are stored indexed by the variable of the literal (obtained by var()). The reason is a clause reference, and it can be the undefined reference (equal to CRef_Undef). If the reason is CRef_Undef then the literal was a decision literal.
- Once the clause c is obtained from a reason() by indexing into ca it should be noted that c[0] is the asserted literal. (This is why the main for loop starts with j=1 rather than j=0).
The code uses a vector `seen` to mark literals on the trail that are contained in the conflict clause. The code is clever in that it uses `seen` to flag literals that need to be resolved or added to the final learnt clause (`out_learnt`). By moving up the trail (using `index`) and using `seen` flags it can do resolution steps without ever adding duplicates literals to `out_learnt`.

If you modify this mechanism you need to be sure that you remember all of the `seen` flags that need to be cleared. The code does this by adding the literals whose `seen` flags remain to be cleared to `analyze_toclear`.

Implement the changes, do some experiments, draw up some graphs and submit a 1 or 2 page answer (including the graphs) explaining your experiments and summarizing the key insights from your results.

Please feel free to ask me for more guidance in understanding the minisat code.