Theorem Proving: Propositional Logic

1.7.04.2013

1. Using DPLL algorithm with learning, refute or find models for the following clause sets:
   
   (a) \{A, B\}, \{\neg A, B\}, \{\neg B\}.
   
   (b) \{A, B\}, \{\neg A, B\}, \{A, \neg B\}, \{\neg A, \neg B\}.
   
   (c) \{\neg A, B\}, \{A, C, B\}, \{B, \neg C\}.
   
   (d) \{A, B, E\}, \{\neg A, C\}, \{\neg B, D, E\}, \{\neg A, \neg C\}, \{\neg B, \neg D\}, \{\neg E\}.
   
   (e) \{P, Q\}, \{Q, R\}, \{P, R\}, \{\neg P, \neg Q\}, \{\neg R, \neg Q\}, \{\neg R, \neg P\}.

2. Specify the basic multiplication block in propositional logic:

\[
(C_{\text{out}}, S_{\text{out}}) = A_i B_j + C_{\text{in}} + S_{\text{in}}.
\]

3. Specify a four bit multiplier with eager carry propagation.
   
   Test on a few examples, using sat solver 2007, that the specification does indeed define multiplication.

4. Specify a four bit multiplier with lazy carry propagation. (Be careful not to use overlapping variables with the eager carry propagation.)
   
   Test on a few examples, using sat solver 2007, that the specification does indeed define multiplication.
   
   If you are brave you may also write a program that generates specifications of arbitrary size.

5. Specify a formula that states equivalence of the two multiplication circuits, and prove it, using sat solver 2007. You may also use minisat. This prover is much much quicker, but it has the disadvantage that the DIMACS format is difficult to read, and it seems to be unable to output proofs.

6. Inside the package for sat solver 2007 is a file magic4. Show that the sat solver can find a magic square of size four in a couple of minutes.

   (a) Does there exist a magic square of size 4 with 1 in the corner? If yes, then find one.
(b) Does there exist a magic square of size 4 with 2 in the corner?
(c) Does there exist a magic square of size 4 with 0 and 1 in opposite corners? If yes, then find one.
(d) Does there exist a magic square of size 4 with 0 and 1 in adjacent corners? If yes, then find one.

(You can use `magic_dimacs.cpp` to generate DIMACS format. You have to add the first line, of form `p cnf <nr-of-literals> <nr-of-clauses>` by yourself. `Minisat` is much better than `satsolver`, but you will have difficulties reading the output.)