Logical reasoning and programming, lab session 2

(September 30, 2024)

2.1 Use `http://fmv.jku.at/limboole/` on $\varphi = (a \to (c \wedge d)) \vee (b \to (c \wedge e))$.

2.2 Derive the empty clause from $\{\{\overline{a}, b\}, \{\overline{b}, c\}, \{a, \overline{c}\}, \{a, b, c\}, \{\overline{a}, \overline{b}, \overline{c}\}\}$ using resolution.

2.3 A clause $c_1 = \{l, l_1, \ldots, l_n\}$ is blocked in φ by l if for every clause $c_2 \in \varphi$ such that $\overline{l} \in c_2$ the resolvent of c_1 and c_2 is a tautology. Prove that if a clause d is blocked (by a literal) in ψ, then $\psi \in \mathtt{SAT}$ iff $\psi \setminus d \in \mathtt{SAT}$.

2.4 Let φ be a formula in CNF such that it contains only Horn clauses; they contain at most one positive literal. Show that SAT for φ is decidable in polynomial time.[1]

Hint: Perform all the unit propagations first.

2.5 Decide the satisfiability of

$$\{\{\overline{p}, r, s, t\}, \{\overline{r}, s, t\}, \{\overline{p}, r, \overline{s}\}, \{p, q\}, \{p, \overline{q}\}, \{\overline{p}, \overline{t}\}, \{\overline{r}, \overline{s}, t\}\}$$

by DPLL. Use the order of branching: p, q, r, $\ldots$.

2.6 Check the details of two watched literals, for example, these slides illustrate the data structure.

2.7 Formulate graph coloring (a vertex coloring) as a SAT problem. Namely, given a graph G, does G admit a proper vertex coloring with k colors?

Discuss various possibilities how to formulate the problem. Moreover, are really all the constraints necessary?

2.8 Check this video, where zChaff colors the McGregor graph.

2.9 Express the pigeonhole principle in propositional logic. Namely, define a propositional formula PHP_n^{n+1}, which says that you have $n+1$ pigeons, n holes, every pigeon has a hole, and no two pigeons sit in the same hole.

2.10 Try PySAT (or for example PicoSAT/pycosat) on PHP_n^{n+1} for small values of n. What is the maximal value of n for which you can solve PHP_n^{n+1} in one minute?

2.11 If you want to play with SAT solving a bit, then a standard exercise is to formalize Sudoku as a SAT problem and hence produce a Sudoku solver. Write a program that generates a problem specification in the DIMACS format in such a way that it is possible to specify an input (a partially completed grid) by appending[2] clauses saying which variables are true. Some input is available from here, where each line has format XYZ meaning there is Z in cell (X, Y).

By the way, is it possible to obtain also a generator of Sudoku puzzles this way?

---

[1] In fact, it is possible to improve the algorithm in such a way that it works in linear time.
[2] Note that this changes the number of clauses, a parameter specified in the DIMACS format.