Logical reasoning and programming, lab session II

(September 30, 2019)

II.1 Produce a formula in CNF which is equivalent to

$$\varphi = (a \to (c \wedge d)) \vee (b \to (c \wedge e)).$$

Then use the Tseytin transformation to produce a formula in CNF which is equisatisfiable to φ.

II.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.

II.3 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?

II.4 Define constraints *at least one* and *at most one* in CNF and discuss various variants of them.

II.5 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 linear time.

Hint: Do all the unit propagations first.

II.6 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, ...

II.7 Decide the satisfiability of

$$\{\{p_1, p_{13}\}, \{\overline{p_1}, \overline{p_2}, p_{14}\}, \{p_3, p_{15}\}, \{p_4, p_{16}\},$$
$$\{\overline{p_5}, \overline{p_3}, p_6\}, \{\overline{p_5}, \overline{p_7}\}, \{\overline{p_6}, p_7, p_8\}, \{\overline{p_4}, \overline{p_8}, \overline{p_9}\}, \{\overline{p_1}, p_9, \overline{p_{10}}\},$$
$$\{p_9, p_{11}, \overline{p_{14}}\}, \{p_{10}, \overline{p_{11}}, p_{12}\}, \{\overline{p_2}, \overline{p_{11}}, \overline{p_{12}}\}\}$$

by CDCL using the first UIP. Use the order of branching: p_1, p_2, p_3, ...

II.8 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[1] clauses saying which variables are true. You can use MiniSat or pycosat and some input is available from here.

You can try various cardinality constraints, e.g., the one based on binary (bitwise) encoding that requires $\mathcal{O}(n \log n)$ clauses and $\mathcal{O}(\log n)$ new variables.

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

[1] Note that this changes the number of clauses, a parameter specified in the DIMACS format.