Logical reasoning and programming, lab session 4

(October 11, 2021)

4.1 Given the formula $\{\{p_1, p_2\}, \{\overline{p_1}, p_3\}, \{p_2, \overline{p_3}\}, \{\overline{p_2}, \overline{p_4}\}, \{\overline{p_3}, p_4\}\}$, what clause will a CDCL solver learn first if it begins by deciding that p_1 is true?

4.2 How many symmetries does your formulation of PHP_n^{n+1} have?

4.3 We can define the lexicographic order on two bit vectors $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, denoted $x_1 \ldots x_n \leq_{lex} y_1 \ldots y_n$, as follows

$$\bigwedge_{i=1}^{n} ((\overline{x_i} \vee y_i \vee \overline{a_{i-1}}) \wedge (\overline{x_i} \vee a_i \vee \overline{a_{i-1}}) \wedge (y_i \vee a_i \vee \overline{a_{i-1}})),$$

where $\overline{a_0}$ is always false, using new auxiliary variables $a_0, a_1, \ldots, a_{n-1}, a_n$.

(a) What is the purpose of auxiliary variables?

Hint: When is it necessary to satisfy $x_i \leq y_i$?

(b) Why is $\overline{a_0}$ always false and hence useless?

(c) Why can we replace $(\overline{x_n} \vee y_n \vee \overline{a_{n-1}}) \wedge (\overline{x_n} \vee a_n \vee \overline{a_{n-1}}) \wedge (y_n \vee a_n \vee \overline{a_{n-1}})$ just by $(\overline{x_n} \vee y_n \vee \overline{a_{n-1}})$? Hence we need only $3n-2$ clauses and $n-1$ auxiliary variables (a_n is also useless).

(d) How does the meaning of the formula change if you replace $(\overline{x_n} \vee y_n \vee \overline{a_{n-1}})$ by $(\overline{x_n} \vee \overline{a_{n-1}}) \wedge (y_n \vee \overline{a_{n-1}})$?

4.4 How can we exploit the lexicographic order to decrease the number of symmetries in PHP_n^{n+1}?

Hint: Order hole-occupancy or pigeon-occupancy vectors.

4.5 A very nice symmetry breaker for PHP_n^{n+1} is based on columnwise symmetry, namely we can add the following clauses

$$p_{i(i+1)} \vee \overline{p_{ij}}$$

for $1 \leq i < j \leq n$, where p_{kl} means that pigeon k is in hole l, for $1 \leq k \leq n+1$ and $1 \leq l \leq n$. Why?

4.6 Try PicoSAT/pycosat and PySAT on PHP_n^{n+1} with various symmetry breakers.

4.7 Symmetry breaking and PHP_n^{n+1} (cont'd). For details see Knuth's TAOCP on satisfiability or slides Symmetry in SAT: an overview.

4.8 Try BreakID.

4.9 There are various encodings of cardinality constraints, discuss sequential counter and bitwise encodings. You can find further examples in this presentation, this presentation, or in PySAT.

4.10 Formulate the software package upgradability as a MaxSAT problem.

4.11 Try CBMC on this example. You can also try this program. For details, see these lecture notes.