Jochen Hoenicke
Tanja Schindler

29.10.2019
submit until 05.11.2019, 14:15

Tutorials for Decision Procedures
Exercise Sheet 2

Exercise 1: Normal Forms 4 Points

Convert the following formulae into disjunctive normal form (DNF) and into conjunctive normal form (CNF).

(a) $\neg(P \to Q) \vee (P \wedge Q)$

(b) $(P \wedge Q) \to (P \vee Q)$

(c) $(P \vee (Q \to P)) \wedge Q$

(d) $P \leftrightarrow (P \vee Q)$

Exercise 2: Equisatisfiability and DNF 4 Points

We have seen that converting a formula into an equisatisfiable formula into CNF can be done efficiently. Now show the following:

> There is a polynomial algorithm to convert a formula into an equisatisfiable formula in *DNF* if and only if $P = NP$.

Hint: Show that there is an efficient algorithm to decide satisfiability of a formula in DNF. How can this be used to solve the exercise?

Exercise 3: DPLL with Learning 4 Points

Apply the DPLL with Learning algorithm from the lecture to check satisfiability of the following formulae. You can apply the rules in the order you find most convenient. Use the rule-based notation.

(a) $\{\{A, B, C\}\{A, \bar{C}\}\{\bar{A}, D\}\{\bar{A}, E\}\{B, \bar{D}, \bar{E}\}\}$

(b) $\{\{A, B, C\}\{A, \bar{C}\}\{\bar{A}, D\}\{\bar{A}, E\}\{B, \bar{D}, \bar{E}\}\{\bar{B}\}\}$

(c) $\{\{A, B\}, \{C, D\}, \{E, F\}, \{\bar{A}, \bar{C}\}, \{\bar{A}, \bar{E}\}, \{\bar{C}, \bar{E}\}, \{\bar{B}, \bar{D}\}, \{\bar{B}, \bar{F}\}, \{\bar{D}, \bar{F}\}\}$

Bonus Exercise 4: Sudoku Generator 4 Bonus Points

A Sudoku is a $n^2 \times n^2$ matrix whose elements are labelled with numbers from 1 to n^2. The figure on the right shows an example for $n = 3$. Every row and every column contains each number. Additionally, every $n \times n$-submatrix (there are n^2 of this) contains each number.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

- Encode the Sudoku rules into boolean contraints.

  Hint: Introduce n^2 boolean variables per field (that means you need n^6 variables overall). For instance boolean variable `v_x_y_i` should have the semantics "the field at position (x,y) is filled out with number i".

- Write a program (in a not-too-exotic language of your choice) which takes an integer n as argument and generates an SMT-LIB file with all necessary constraints for a Sudoku of size n.

- Use an SMT-LIB compliant SMT solver (e.g. Z3 or SMTInterpol which are linked at the lecture website) to generate Sudokus. On the lecture website there is a commented example script encoding the formula from Slide 28 in the slide set on propositional logic.

- Experiment with different numbers for n and measure the run time of your program and of the solver.

Please submit your program source code and a summary of your experiments.