Assignment 1

due Monday, September 16, 2024

The homework is due at 6pm on Monday, September 16, 2024. Each student has to complete the homework assignment on their own. Please submit your homework via Gradescope, see https://www.gradescope.com/courses/862231. Your submission can be a combination of code **with comments**, PDFs, **clearly readable** scans of handwritten answers, and scripts or screenshots of solver runs.

The questions below are mostly encoding questions. Encoding tools, such as PySAT, are allowed for Questions 4 and 5. We prefer answers that consist of a generator that produces the requested DIMCAS file in a common programming language, such as Python or C(++). Alternatively, you can submit the encoding answers as a LaTeX document. However, Questions 4(b), 4(c), 5(b), and 5(c) can only be solved using a generated DIMACS file.

The maximum number of regular points for this assignment is 50: the 30 points of Question 1, 2, and 3 + either 20 points of Question 4 or 20 points of Question 5(a) and (b). Additionally, 10 bonus points can be earned in Question 5(c).

Question 1

Consider the following conjunction of clauses:

$$
\begin{aligned}
1 &: (a \vee b \vee \overline{c}) \\
2 &: (\overline{a} \vee \overline{b} \vee c) \\
3 &: (b \vee c \vee \overline{d}) \\
4 &: (\overline{b} \vee \overline{c} \vee d) \\
5 &: (a \vee c \vee d) \\
6 &: (\overline{a} \vee \overline{c} \vee \overline{d}) \\
7 &: (\overline{a} \vee b \vee d)
\end{aligned}
$$

(a) [8 points] Derive all possible unit clauses via resolution. In each step, print the step number (starting with 8) followed by the resolvent and the two clause indices used for resolution. E.g., the first steps could be

$$
\begin{aligned}
8 &: (a \vee b \vee \overline{d}) \quad 1 + 3 \\
9 &: (b \vee \overline{c} \vee \overline{d}) \quad 6 + 8
\end{aligned}
$$

Hint: work toward producing smaller clauses, in which case 15 resolution steps are enough.

Question 2 (no encoding tools allowed)

Consider the following constraint: $x_1 + x_2 + x_3 + x_4 \leq 1$.

(a) [6 points] Express this constraint in negation normal form (NNF) using each literal (i.e., $\overline{x}_1$, $\overline{x}_2$, $\overline{x}_3$, and $\overline{x}_4$) at most twice and without using auxiliary variables.

(b) [6 points] Apply the Tseitin transformation to the answer of (a) to turn it into CNF. For this question it is required to use auxiliary variables. You are allowed to use the optimizations discussed in class.

Question 3 (no encoding tools allowed)

Consider graph $G = (V, E)$ with $V = \{u, v, w, x, y\}$ and $E = \{(u, v), (v, w), (w, x), (x, y), (u, y)\}$.

(a) [6 points] Encode whether G can be colored with two colors. Test whether this formula is satisfiable using a SAT solver.

(b) [4 points] Construct a symmetry-breaking predicate that breaks the color symmetry.

Question 4 (answer this question or Question 5)

(a) [10 points] Consider a $n \times m$ grid of squares and all possible rectangles within the grid whose length and width are at least 2. Encode whether there exists a coloring of the grid using three colors so that no such rectangle has the same color for its four corners. (Hint: The encoding requires two types of constraints. First, each square needs to have at least one color. Second, if four squares form the corners of a rectangle, then they cannot have the same color.)

```
0 0 1 1 2 2 0 1 2
2 0 0 1 1 2 2 0 1
1 2 0 0 1 1 2 2 0
0 1 2 0 0 1 1 2 2
2 0 1 2 0 0 1 1 2
2 2 0 1 2 0 0 1 1
1 2 2 0 1 2 0 0 1
1 1 2 2 0 1 2 0 0
0 1 1 2 2 0 1 2 0
```

(b) [5 points] Solve the encoding for a 10×10 grid using a SAT solver and decode the solution into a valid coloring. Show the output of the SAT solver and a valid 3-coloring similar to the one above of the 9×9 grid.

(c) [5 points] Solve the encoding for a 9×12 grid using a SAT solver and decode the solution into a valid coloring. Show the output of the SAT solver and a valid 3-coloring similar to the one above of the 9×9 grid.

Question 5 (answer this question or Question 4)

An *almost square* is a $n \times (n + 1)$ rectangle. One can cover the almost square 4×5 using the smallest three almost squares: 1×2, 2×3, and 3×4. A solution is shown below.

```
1 1 3 3 3
2 2 3 3 3
2 2 3 3 3
2 2 3 3 3
```

(a) [10 points] Encode whether the smallest k almost squares can cover an almost square. A satisfying assignment of the encoding should represent a covering. In case the smallest k almost squares don't add up to an almost square, the encoding should simply print a formula with only the empty clause.

(b) [10 points] Solve the encoding for the smallest 8 almost squares, which can cover the almost square 15×16, and decode the solution into a valid cover. Show the output of the SAT solver and valid cover similar to the one above of the 4×5 grid.

(c) [Bonus: 10 points] Construct a compact encoding for the smallest 20 almost squares, which can cover the almost square 55×56. Auxiliary variables are useful to reduce the size of the encoded formula. Bonus points are awarded for reasonably small encodings: 2 points for less than 3 million clauses; 4 points for less than 2 million clauses; 6 points for less than a million clauses; and 8 points for less than half a million clauses. All 10 points are awarded for any encoding for which you can show that a SAT solver can find a satisfying assignment. Warning: this problem is challenging.