Assignment 5
(I can't get no) Satisfaction

15-414: Bug Catching: Automated Program Verification

Due Friday, March 22, 2024
70 pts

This assignment is due on the above date and it must be submitted electronically on Gradescope. Please carefully read the policies on collaboration and credit on the course web pages at http://www.cs.cmu.edu/~15414/assignments.html.

What To Hand In

You should hand in the following files on Gradescope:

- Submit the file `asst5.zip` to Assignment 5 (Code). You can generate this file by running `make handin`. This will include your solution `baby-sat.mlw` and the proof session in `baby-sat/`.
- Submit a PDF containing your answers to the written questions to Assignment 5 (Written). You may use the file `asst5-sol.tex` as a template and submit `asst5-sol.pdf`. You can generate this file by running `make sol` (assuming you have pdflatex in your system).

  Make sure your session directories and your PDF solution files are up to date before you create the handin file.

Using LaTeX

We prefer the answer to your written questions to be typeset in LaTeX, but as long as you hand in a readable PDF with your solutions it is not a requirement. We package the assignment source `asst5.tex` and a solution template `asst5-sol.tex` in the handout to get you started on this.

1 Encodings (20 pts)

Task 1 (10 pts). Consider the following CNF formula:

$$\begin{array}{ll} \neg x_1 \vee \neg x_2 & C_1 \\ x_1 \vee x_2 & C_2 \\ \neg x_1 \vee \neg x_3 & C_3 \\ x_1 \vee x_3 & C_4 \\ \neg x_2 \vee \neg x_3 & C_5 \\ x_2 \vee x_3 & C_6 \end{array}$$

Show that this formula is unsatisfiable by using resolution to derive the empty clause. You should present your derivation as done in the Lecture Notes 14 (page 5).

Task 2 (10 pts). Consider a 2-coloring problem for integers between 1 and 5 such that for every integer solution $a + b = c$ with $1 \leq a < b < c \leq n$ holds that a, b, and c do not have the same color. Note that the possible sums with numbers 1 to 5 under these conditions are:

- 1 + 2 = 3
- 1 + 3 = 4
- 1 + 4 = 5
- 2 + 3 = 5

Write a propositional encoding for this problem. How you choose to encode colors and integers (i.e., unary or binary) is your choice, but be sure to explain how each propositional variable should be interpreted, as well as the rationale for the clauses in your formula.

2 Baby SAT steps (50 pts)

In this assignment, we will explore simple operations that can be performed over formulas in the conjunctive normal form before we build our first verified SAT solver. You may write auxiliary predicates or functions beyond the ones provided in `baby-sat.mlw`.

Consider the following types that define a variable (var), literal (lit: which is define as a positive (e.g. x_1) or negative variable ($\neg x_1$)), clause, cnf formula, and valuation. Assume that the variables range from 0 to $nvars - 1$ and that the cnf formulas have 0 or more variables.

```
type var = int
type lit = { var : var ; sign : bool }
type clause = list lit
type cnf = { clauses : array clause ; nvars : int }
type valuation = array bool
```

An example of how a CNF is represented using this type is provided Figure 1.

Note that for a clause to be satisfied, there exists at least one literal in the clause that is satisfied. A positive literal is satisfied if the variable is assigned *true*, while a negative literal is satisfied if the variable is assigned *false*. For the formula in Figure 1, a valuation (also called *interpretation* in some future lecture notes) is represented as an array of booleans whose i^{th} component is the value of x_i. For this formula, the valuation $[\top, \bot, \top, \bot]$ would satisfy all clauses (this valuation would assign $x_0 = \top$, $x_1 = \bot$, $x_2 = \top$, $x_3 = \bot$).

```
{ nvars = 4;
  clauses = [
    [ {var=3; sign=false} ];
    [ {var=0; sign=true}; {var=2; sign=false}; {var=3; sign=true} ];
    [ {var=1; sign=false}; {var=2; sign=true} ] ] }
```

Figure 1: Representation of the formula $\neg x_3 \wedge (x_0 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2)$.

Task 3 (5 pts). Write data structure invariants for the type `cnf`.

Task 4 (10 pts). Specify and implement a function `eval_clause` that takes a valuation ρ and a clause c as its arguments and returns `true` if c is true for valuation ρ and `false` otherwise.

Task 5 (15 pts). Specify and implement a function `eval_cnf` that takes a valuation ρ and a formula cnf in conjunctive normal form as its arguments and returns `true` if cnf is true for valuation ρ and `false` otherwise.

Pure Literals

Any variable that only appears in either positive or negative literals is called *pure*, and their corresponding variables can always be assigned in a way that satisfies the literal. Thus, they do not constrain the problem in a meaningful way, and can be assigned without making a choice. This is called *pure literal elimination* and is one type of simplification that can be applied to CNF formulas. Consider the following CNF formula:

$$\underbrace{(x_1 \vee x_2)}_{C_0} \wedge \underbrace{(\neg x_1 \vee x_2)}_{C_1} \wedge \underbrace{(x_1 \vee \neg x_2 \vee x_3)}_{C_2} \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{C_3}$$

Notice that x_3 appears only as a positive literal in this formula. Hence, we can assign x_3 to *true* and satisfy the literal. This procedure will simplify the above formula into:

$$\begin{aligned}
&(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_1 \vee x_3) \\
\leftrightarrow\ &(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee \top) \wedge (\neg x_1 \vee x_1 \vee \top) \\
\leftrightarrow\ &(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)
\end{aligned}$$

Note that if a formula is satisfiable and if a literal l is pure, then it is always possible to have an interpretation that satisfies the literal, i.e., assigns l to *true* if l is positive or to *false* if l is negative.

Task 6 (20 pts). Specify and implement a function `pure_literal` that takes a formula *cnf* in conjunctive normal form and a literal l as its arguments and returns `true` if l is a pure literal and `false` otherwise.