Lesson 1 – Exercises

Static Program Analysis and Constraint Solving
Master's Degree in Formal Methods in Computer Science
Year 2021/22

1. Given the following set of clauses:

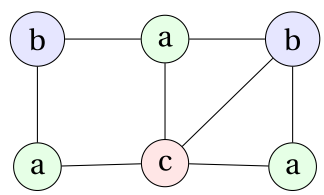
$$\begin{aligned} c_1 &\equiv \neg p_5 \\ c_2 &\equiv p_5 \vee \neg p_6 \\ c_3 &\equiv p_1 \vee p_2 \\ c_4 &\equiv \neg p_1 \vee p_3 \\ c_5 &\equiv \neg p_3 \vee p_2 \\ c_6 &\equiv \neg p_2 \vee p_4 \\ c_7 &\equiv \neg p_4 \vee \neg p_2 \end{aligned}$$

   (a) Apply the CDCL algorithm to check its satisfiability.

   (b) Write an SMT-LIB script that checks the satisfiability of this set and run it via Z3.

   (c) Z3 provides an API that allows one to interoperate with several languages and platforms: C, C++, Java, .NET, Python, etc. Use your preferred programming language to check the satisfiability of the set given above.

2. Assume a graph G. A *graph coloring* is an assignment of colors from a set K to the vertices of G, such that there are no adjacent vertices in G sharing the same color. For example, if $K = \{a, b, c\}$, the following is a valid colored graph:



   In your favorite programming language, implement a function, method, or procedure that, given a graph G and a positive number n, generates an SMT-LIB script that checks whether the graph can be colored with n colors. You can use only quantifier-free propositional logic.

3. We are given a set T of teachers and a set S of subjects. Every teacher $t \in T$ can teach a subset of those subjects. We denote by $S(t)$ the set of subjects which t may teach. We want to know whether, given a number k, there is a subset $C \subseteq T$ of cardinality k such that all the subjects in S can be taught by, at least, a teacher in C. Implement a procedure that, given T, S, $S(t)$ (for each $t \in T$), and k, outputs an SMT-LIB script that checks whether this is possible.

4. Recall the `sqrt` example shown in the slides. Its correctness relies on the following verification conditions:

   - Loop invariant holds on entry.
     $\forall r. \forall x. r = 0 \Rightarrow r^2 \leq x$
   - Invariant is preserved by loop body.
     $\forall r. \forall x. \forall r'. r^2 \leq x \wedge (r+1)^2 \leq x \wedge r' = r + 1 \Rightarrow r'^2 \leq x$
   - Postcondition follows when exiting the loop.
     $\forall r. \forall x. r^2 \leq x \wedge \neg((r+1)^2 \leq x) \Rightarrow r^2 \leq x < (r+1)^2$

   Verify these conditions with Z3. Which logic have you used?[1]

5. Given the following function:

```
function swap(x: array, n: int, m: int) {
    var e: int;
    e := x[n];
    x[n] := x[m];
    x[m] := e;
}
```

   Assume that we call `swap`(x, n, m). If we denote by x' the state of the input array x after the function finishes, the following postcondition should hold:

$$(x'[n] = x[m]) \wedge (x'[m] = x[n]) \wedge (\forall i. i \neq n \wedge i \neq m \Rightarrow x'[i] = x[i])$$

   Which first-order formula do you need to prove in order to ensure that the postcondition holds after the execution of `swap`?. Use Z3 to check the validity of this formula.

6. Prove the following properties on sets by translating them into the EPL (Effectively Propositional Logic) fragment:

   (a) $A \cap B \neq \emptyset \wedge A \subseteq D \wedge B \subseteq C \Rightarrow C \cap D \neq \emptyset$

   (b) $B \subseteq A \Rightarrow B \cup A \subseteq A$

   (c) $A \cap B = \emptyset \Rightarrow A - B = A$

[1] Check `http://smtlib.cs.uiowa.edu/logics.shtml` for the list of available logics.