Formal Methods in Computer Science

UE 185.A93 WS 2019/20

Exercise Sheet 1 solutions

Wolfgang Dvořák

Technische Universität Wien

WS 2019

Outline

Recall the problem

The graph $G = (V, E)$

- V a set of vertices,
- $E \subseteq V \times V$ a set of directed edges

A 4-coloring of a graph $G = (V, E)$ is a function $C : V \mapsto \{r, g, b, o\}$, where r,g,b,o stand for the colors red, green, blue, and orange.

A 4-coloring is called a valid coloring iff it satisfies all of the following:

- No two adjacent vertices are both colored red.
- A vertex b is colored blue or green iff there is a vertex a that is colored red and disables b.
- If a vertex a disables a red colored vertex then a is colored green.
- If a vertex a is colored orange then there is another vertex b that disables a and is not colored green.

Introduce Variables

Variables

For each vertex $v \in A$ we introduce the following four variables

- x_v^R ... v is colored red
- x_v^G ... v is colored green
- x_v^B ... v is colored blue
- x_v^O ... v is colored orange

Alternative:

- Use two variables x_v, y_v for each vertex $v \in A$

Color of v	x_v	y_v
red	true	false
green	false	true
blue	true	true
orange	false	false

We can encode the different constraints on valid colorings separately.

Ensure that each model is a coloring

Each vertex is colored either red, green, or blue:

- At least one of the colors

$$\varphi_1 = \bigwedge_{v \in A} (x_v^R \vee x_v^G \vee x_v^B \vee x_v^O)$$

- At most one of the colors:

$$\begin{aligned}\varphi_2 = \bigwedge_{v \in A} \big((\neg x_v^R \vee \neg x_v^G) \vee (\neg x_v^R \vee \neg x_v^B) \vee (\neg x_v^R \vee \neg x_v^O) \vee \\ (\neg x_v^G \vee \neg x_v^B) \vee (\neg x_v^G \vee \neg x_v^O) \vee (\neg x_v^B \vee \neg x_v^O)\big) \\ = \bigwedge_{v \in A} \bigwedge_{\{C_1, C_2\} \in \{R,G,B,O\}, C_1 < C_2} (\neg x_v^{C_1} \vee \neg x_v^{C_2})\end{aligned}$$

Encode conditions for valid colorings

- No two adjacent vertices are both colored red.

$$\varphi_3 = \bigwedge_{(v,u)\in E} \neg(x_v^R \wedge x_u^R) \equiv \bigwedge_{(v,u)\in E} (\neg x_v^R \vee \neg x_u^R)$$

Encode conditions for valid colorings

- A vertex b is colored blue or green iff there is a vertex a that is colored red and disables b.
  - A vertex b is colored blue or green if there is a vertex a that is colored red and disables b.

$$\varphi_{4a} = \bigwedge_{(u,v)\in E} (x_u^R \to (x_v^B \vee x_v^G)) \equiv \bigwedge_{(u,v)\in E} (\neg x_u^R \vee x_v^B \vee x_v^G)$$

  - A vertex b is colored blue or green only if there is a vertex a that is colored red and disables b.

$$\varphi_{4b} = \bigwedge_{v\in V} ((x_v^B \vee x_v^G) \to \bigvee_{(u,v)\in E} x_u^R)$$

$$\equiv \bigwedge_{v\in V} (\neg x_v^B \vee \bigvee_{(u,v)\in E} x_u^R) \wedge \bigwedge_{v\in V} (\neg x_v^G \vee \bigvee_{(u,v)\in E} x_u^R)$$

  By exploiting φ_1 and φ_2 we can also use the following:

$$\varphi'_{4b} = \bigwedge_{v\in V} (x_v^R \vee x_v^O \vee \bigvee_{(u,v)\in E} x_u^R)$$

- $\varphi_4 = \varphi_{4a} \wedge \varphi_{4b}$

Encode conditions for valid colorings

- If a vertex a disables a red colored vertex then a is colored green.

$$\varphi_5 = \bigwedge_{(u,v)\in E} (x_v^R \to x_u^G) \equiv \bigwedge_{(u,v)\in E} (\neg x_v^R \vee x_u^G)$$

- If a vertex a is colored orange then there is another vertex b that disables a and is not colored green.

$$\varphi_6 = \bigwedge_{v\in V} (x_v^O \to \bigvee_{(u,v)\in E, u\neq v} \neg x_u^G) \equiv \bigwedge_{v\in V} (\neg x_v^O \vee \bigvee_{(u,v)\in E, u\neq v} \neg x_u^G)$$

Reduction - Correctness

Theorem

- *The formula $\varphi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \wedge \varphi_5 \wedge \varphi_6$ is satisfiable iff G has a valid 4-Coloring.*
- *Moreover there is a one-to-one correspondence between models and 4-colorings.*

Correctness Argument (Proof Sketch):
$\Rightarrow$: Assume $G = (V, E)$ has a valid 4-Coloring C.
We define the truth assignment.

$$\alpha_C(x_v^c) = \begin{cases} \mathit{true} & \text{if } C(v) = c \\ \mathit{false} & \text{otherwise} \end{cases}$$

We show that α_C is a model of φ.

Reduction - Correctness

$\Rightarrow$

We sketch the proof of α_C is a model of φ .

- $\varphi_1 \wedge \varphi_2$: As $C(v)$ assigns exactly one color to each vertex, exactly one of the variables $x_v^R, x_v^G, x_v^B, x_v^O$ is true.
- φ_3: By definition if $(u, v) \in E$ not both u, v are colored red.
- φ_{4a}: For $(u, v) \in E$ we have that if u is colored red then by condition 2 v is either colored blur or green.
- φ_{4b}: If a vertex is colored blue or green then all of its predecessors are colored red by condition 2.
- φ_5: For $(u, v) \in E$ we have that if v is colored red then, by condition 3, u is colored green.
- φ_6: If a vertex is colored orange then one of its predecessors is colored different form green, by condition 4.

Thus α_C is a model of φ.

Coalition Forming - Correctness

$\Leftarrow$

Correctness Argument (Proof Sketch):
$\Leftarrow$: Assume α is a model of φ and consider

$$C_\alpha(v) = \begin{cases} R & \text{if } \alpha(x_v^R) = \mathit{true} \\ G & \text{if } \alpha(x_v^G) = \mathit{true} \\ B & \text{if } \alpha(x_v^B) = \mathit{true} \\ O & \text{if } \alpha(x_v^O) = \mathit{true} \end{cases}$$

We show that C_α is a valid 4-Coloring of G.

- C_α is well-defined (assigns exactly one color): by $\varphi_1 \wedge \varphi_2$
- Check each of the properties for C_α being a valid coloring.
  - No two adjacent vertices are colored both red: by φ_3
  - ...

  More formally: Proof by contradiction

Exercise 2

Task: Provide a reduction that takes a graph G and a set of S_r vertices as input and provides a propositional formula such that

- the models of the formula are in one-to-one correspondence with the valid colorings of G that color all vertices in S_r red, and
- one can easily obtain such a coloring from a model of the formula.

Solution: For graph $G = (V, E)$ and $S_r \subseteq V$ consider the formula:

$$\psi = \varphi \wedge \bigwedge_{v \in S_r} x_v^R$$

Theorem

The formula ψ is satisfiable iff G has a valid 4-Coloring that colors all vertices in S_r red.

Identifying Maximal Valid Coloring

- Testing maximality is a co-NP problem
  $\hookrightarrow$ can be efficiently reduced to a SAT-problem
- computing a $\subseteq$-maximal coloring is beyond NP/co-NP
  $\hookrightarrow$ cannot be efficiently reduced to a single SAT-problem

We can compute a maximal valid coloring as follows:

- Compute a valid coloring C
- While C changes do
  - Test whether C is maximal
  - If not update C to the larger set computed in the previous step.
- return C

Notice:
We need a procedure that returns a counter example if C is not maximal.

Identifying Maximal Valid Colorings

Problem: Identifying valid colorings that color certain vertices red and the set of red colored vertices is maximal among all valid colorings.

- Given: Graph $G = (V, E)$ and a valid coloring C for G,
- Goal: a propositional formula such that the formula C is unsatisfiable if and only if C maximal among the valid colorings. This reduction should be such that one can easily obtain a coloring from a model of the formula, i.e., a counter-example C' for the minimality of C.

Maximal Valid Coloring

Idea:

- Start from φ s.t.
- add clauses ensuring all vertices colored red in C are colored red
- add a clause such that at least one new vertex is colored red

Additional clauses:

- If a vertex is red in C then it is also red in C'

$$\delta_1 = \bigwedge_{v:C(v)=R} x_v^R$$

- There is at least one additional red vertex in C'

$$\delta_2 = \bigvee_{v:C(v)\neq R} x_v^R$$

Maximal Valid Coloring - Correctness

Theorem

$\delta = \varphi \wedge \delta_1 \wedge \delta_2$ is unsatisfiable iff C is a maximal coloring in G.

Correctness Argument (Proof Sketch):
$\Rightarrow$: Assume $G = (V, E)$ has a valid 4-Coloring C' with $\{v \mid C(v) = R\} \subset \{v \mid C'(v) = R\}$.
We define the truth assignment.

$$\alpha_{C'}(x_v^c) = \begin{cases} \mathit{true} & \text{if } C'(v) = c \\ \mathit{false} & \text{otherwise} \end{cases}$$

We show that $\alpha_{C'}$ is a model of δ.

- It is a model of φ by the proof of Exercise 1
- It is a model of δ_1 by $\{v \mid C(v) = R\} \subseteq \{v \mid C'(v) = R\}$
- It is a model of δ_2 by $\{v \mid C(v) = R\} \not\supseteq \{v \mid C'(v) = R\}$

Maximal Valid Coloring - Correctness

Correctness Argument (Proof Sketch):
$\Leftarrow$: Assume α is a model of δ and consider

$$C'_\alpha(v) = \begin{cases} R & \text{if } \alpha(x_v^R) = \mathit{true} \\ G & \text{if } \alpha(x_v^G) = \mathit{true} \\ B & \text{if } \alpha(x_v^B) = \mathit{true} \\ O & \text{if } \alpha(x_v^O) = \mathit{true} \end{cases}$$

We show that C'_α is a valid coloring of G such that $\{v \mid C(v) = R\} \subset \{v \mid C'(v) = R\}$. As α is a model of φ we have that C'_α is a valid coloring of G.

- $\{v \mid C(v) = R\} \subseteq \{v \mid C'(v) = R\}$ by δ_1
- $\{v \mid C(v) = R\} \not\supseteq \{v \mid C'(v) = R\}$ by δ_2

More formally: Proof by contradiction

Code Snippets - Encoding of variables

To use the sat solver we have to encode our variables as integers.
We can assume $V = \{1, \ldots, n\}$:

Variable	Encoding
x_v^R	$4 \cdot (v-1) + 1$
x_v^G	$4 \cdot (v-1) + 2$
x_v^B	$4 \cdot (v-1) + 3$
x_v^O	$4 \cdot (v-1) + 4$

A lot of alternative ways to do that.

Pros of the above:

- Easy to compute (encoding and decoding)
- No need to store the mapping
- Starts with 1 and uses consecutive integers

Code Snippets - Encoding of variables

```
public int getVariable (Integer vertex, Color color) {
        int variable=-1;
        switch(color) {
                case RED: variable=1; break;
                case GREEN: variable=2; break;
                case BLUE: variable=3; break;
                case ORANGE: variable=4; break;
        }
        return 4*(vertex-1)+ variable;
}
```

Code Snippets - Decoding of variables

```
public void getColoringFromModel(Coloring coloring, int[] model){

  for(int modelEntry: model) {
    if (modelEntry > 0) {

      if ((modelEntry % 4) == 1)
          coloring.setColor(modelEntry/4+1, Color.RED);
      else if ((modelEntry % 4) == 2)
          coloring.setColor(modelEntry/4+1, Color.GREEN);
      else if ((modelEntry % 4) == 3)
          coloring.setColor(modelEntry/4+1, Color.BLUE);
      else if ((modelEntry % 4) == 0)

      coloring.setColor(modelEntry/4, Color.ORANGE);
    }
  }
}
```

Code Snippets - Building the CNF

```
public IVec<IVecInt> fourColoring(
            DefaultDirectedGraph<Integer, DefaultEdge> graph) {

  IVec<IVecInt> cnf = new Vec<IVecInt>();
  IVecInt clause, clause1, clause2, clause3;
  int i,j;

  [...]

  return cnf;

}
```

Code Snippets - Building the CNF

```
[...]
for(int vertex : graph.vertexSet()) {

  // Each vertex has at least one color
  clause = new VecInt();
      clause.push(getVariable(vertex,Color.RED));
      clause.push(getVariable(vertex,Color.GREEN));
      clause.push(getVariable(vertex,Color.BLUE));
      clause.push(getVariable(vertex,Color.ORANGE));
  cnf.push(clause);

  // Each vertex has at most one color
  Color colors[] = Color.values();
  for(i=0; i<colors.length; i++) {
      for(j=i+1; j<colors.length; j++) {
        clause = new VecInt();
          clause.push(-getVariable(vertex,colors[i]));
          clause.push(-getVariable(vertex,colors[j]));
        cnf.push(clause);
      }
  }
}
[...]
```

Code Snippets - Building the CNF

```
[...]
for(int vertex : graph.vertexSet()) {

  for (DefaultEdge edge : graph.incomingEdgesOf(vertex)) {
    int s = graph.getEdgeSource(edge);
    int t = graph.getEdgeTarget(edge);

    // No two adjacent vertices are both colored red
    cnf.push(new VecInt(new int[]{-getVariable(s,Color.RED),
                                  -getVariable(t,Color.RED)}));

    // If a is colored RED then its successors are colored GREEN
       or BLUE
    cnf.push(new VecInt(new int[]{-getVariable(s,Color.RED),
      getVariable(t,Color.GREEN), getVariable(t,Color.BLUE)}));

    // If a vertex $a$ disables a red colored vertex then $a$ is
       colored green.
    cnf.push(new VecInt(new int[]{getVariable(s,Color.GREEN),
                                  -getVariable(t,Color.RED)}));
   }
}
[...]
```

Code Snippets - Building the CNF

```
[...]
for(int vertex : graph.vertexSet()) {
  // If a vertex is colored GREEN then one of its successors
     is colored RED
  clause1 = new VecInt();
  clause1.push(-getVariable(vertex,Color.GREEN));
  //If a vertex a is colored orange then there is another vertex
    b that disables a and is not colored green.
  clause3 = new VecInt();
  clause3.push(-getVariable(vertex,Color.ORANGE));

  for (DefaultEdge edge : graph.incomingEdgesOf(vertex)) {
      int s = graph.getEdgeSource(edge);
      clause1.push(getVariable(s,Color.RED));

      if(s!=vertex){
         clause3.push(-getVariable(s,Color.GREEN));
      }
   }
  cnf.push(clause1);
  cnf.push(clause3);
}
[...]
```

Code Snippets - Building the CNF

```
public IVec<IVecInt> maxColoring( DefaultDirectedGraph<Integer,
                                  DefaultEdge> graph, Coloring coloring)
{
  IVec<IVecInt> cnf = fourColoring(graph);
  IVecInt clause1, clause2;

  clause2 = new VecInt();

  for(int vertex: graph.vertexSet()) {
    if(coloring.getColor(vertex)==Color.RED) {
      clause1 = new VecInt();
      clause1.push(getVariable(vertex,Color.RED));
      cnf.push(clause1);
    } else {
      clause2.push(getVariable(vertex,Color.RED));
    }
  }
  cnf.push(clause2);

  return cnf;
}
```

Questions?

Feedback?