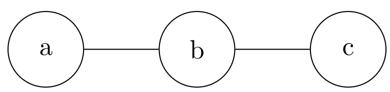
Graph Coloring with SAT

October 24, 2022

Problem Description: A graph (V, E) consists of the set of nodes V and the set of edges E where the elements of E are pairs of nodes. We call two nodes $v, w \in V$ connected if $(v, w) \in E$. Given n colors, the task is to assign each node a color such that connected nodes have different colors.

Example: The following graph (V, E) with $V = \{a, b, c\}$ and $E = \{(a, b), (b, c)\}$ has three nodes: a, b, c and two edges (a, b) and (b, c). In this example, a and b are connected and b and c are connected. The graph can be visualized as follows:



Obviously, if we have only one color, then it is not possible to assign this color in such a way that connected nodes have a different color. If we have two colors, e.g., color 1 and color 2, then we find two solutions: (1) nodes a and c have color 1 and node b has color 2 and (2) nodes a and c have color 2 and node b has color 1.

Encoding of the n-coloring problem in propositional logic: In the following we construct a SAT encoding of an n-coloring problem for a given graph. The obtained formula is satisfiable if and only if the considered graph has one or more n-coloring. The models of the formula represent such colorings.

- variables: for each node of the graph we introduce n propositional variables where n is the number of colors we have. The variable x_i shall be true if node x has color i and otherwise false. In our example above, we have the variables $a_1, a_2, b_1, b_2, c_1, c_2$.
- constraints: Next, we have to construct a formula that relates these variables with the following constraints.
  1. Every node has a color: First we have to ensure that every node is assigned a color. For instance, in our example a_1 and a_2 cannot both be false. Hence, we have to add the following clauses to our encoding:

$$\bigwedge_{v \in V} (\bigvee_{1 \leq j \leq n} v_j)$$

For our example, we get $(a_1 \vee a_2) \wedge (b_1 \vee b_2) \wedge (c_1 \vee c_2)$ which is encoded in `limboole` as follows:

```
(a1 | a2) & (b1 | b2) & (c1 | c2)
```

This formula has only models in which each node has at least one color. If we call `limboole`, we get the following model:

```
./limboole -s gc.boole
% SATISFIABLE formula (satisfying assignment follows)
a1 = 1
a2 = 1
b1 = 1
b2 = 1
c1 = 1
c2 = 1
```

Obviously, we have to say that a single node can have at most one color.

2. Every node has at most one color: This we can say with the following constraint: $(\neg v_i \vee \neg v_j)$ with $v \in V, 1 \leq i < j \leq n$. For our example, we get $(\neg a_1 \vee \neg a_2) \wedge (\neg b_1 \vee \neg b_2) \wedge (\neg c_1 \vee \neg c_2)$. Note that this is different than the constraints above. If we had a 3-coloring problem, we would write $(a_1 \vee a_2 \vee a_3)$ for ensuring that node a has one of the colors $1, 2, 3$. For the ensuring that node a has at most one color, we would get $(\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3)$. For the 2-coloring problem of our example, we get the following encoding for `limboole`:

```
(!a1 | !a2) & (!b1 | !b2) & (!c1 | !c2)
```

Now the solver finds the following satisfying assignment:

```
./limboole -s gc
% SATISFIABLE formula (satisfying assignment follows)
a1 = 1
a2 = 0
b1 = 1
b2 = 0
c1 = 1
c2 = 0
```

In this result, all nodes have the same color (also the connected ones). Hence, we have to add some more constraints.

- Connected nodes have different colors: This is expressed with the following constraints: $(\neg v_i \vee \neg w_i)$ with $(v, w) \in E, 1 \leq i \leq n$. These clauses say that if v and w are connected, they have to have different colors. In our example, we get $(\neg a_1 \vee \neg b_1) \wedge (\neg a_2 \vee \neg b_2) \wedge (\neg c_1 \vee \neg b_1) \wedge (\neg c_2 \vee \neg b_2)$ In `limboole` syntax the constraints are encoded as follows:

```
(!a1 | !b1) & (!a2 | !b2) &
(!c1 | !b1) & (!c2 | !b2)
```

Now the full encoding looks as follows:

```
(a1 | a2) & (b1 | b2) & (c1 | c2) &
(!a1 | !a2) & (!b1 | !b2) & (!c1 | !c2) &
(!a1 | !b1) & (!a2 | !b2) &
(!c1 | !b1) & (!c2 | !b2)
```

and `limboole` gives us the following solution:

```
./limboole -s gc.boole
% SATISFIABLE formula (satisfying assignment follows)
a1 = 1
a2 = 0
b1 = 0
b2 = 1
c1 = 1
c2 = 0
```

Here nodes a and c have color 1 and node b has color 2. In order to force a to have color 2, we append the unit clause a_2 to the formula by a conjunction and get

```
(a1 | a2) & (b1 | b2) & (c1 | c2) &
(!a1 | !a2) & (!b1 | !b2) & (!c1 | !c2) &
(!a1 | !b1) & (!a2 | !b2) &
(!c1 | !b1) & (!c2 | !b2)
&
a2
```

Then `limboole` gives us the other solution:

```
./limboole -s gc.boole
% SATISFIABLE formula (satisfying assignment follows)
a1 = 0
a2 = 1
b1 = 1
b2 = 0
c1 = 0
c2 = 1
```

For asking if there is a coloring where a and b have color 2, we give the following formula to `limboole`.

```
(a1 | a2) & (b1 | b2) & (c1 | c2) &
(!a1 | !a2) & (!b1 | !b2) & (!c1 | !c2) &
(!a1 | !b1) & (!a2 | !b2) &
```

```
(!c1 | !b1) & (!c2 | !b2)
&
a2
&
b2
```

As expected, the result is unsatisfiable.