UNIVERSITÀ DI TRENTO
Department of
Information Engineering and Computer Science

Automated Reasoning and Formal Verification

Laboratory 5

Gabriele Masina
gabriele.masina@unitn.it
https://github.com/masinag/arfv2025

Università di Trento

April 02, 2025

Outline

1. Introduction to OptiMathSAT
2. OMT Exercises
3. Automating SMT encoding
4. Homeworks

OptiMathSAT

- OptiMathSAT extends MathSAT 5 for Optimization Modulo Theories (OMT)
- Features:
  - **Optimize** objectives over **several theories** (linear and non-linear arithmetic over real and integers, bit-vectors, and floating-point)
  - **MaxSMT** with hard and soft constraints.
  - **Multiple objectives**, with several combination modes (boxed, lexicographic, pareto)
  - **Incremental** solving and optimization
- More at: `https://optimathsat.disi.unitn.it`

OptiMathSAT input format: Extended SMT-LIB

OptiMathSAT input: SMT-LIB format extended with optimization commands.

Command	**Description**
(minimize <term>)	Minimizes the given term.
(maximize <term>)	Maximizes the given term.
(minmax <term0> ... <termN>)	Minimize the maximum of the given terms.
(maxmin <term0> ... <termN>)	Maximize the minimum of the given terms.
(assert-soft <term> :weight w :id group)	MaxSMT: adds a soft constraint with the given weight.
(get-objectives)	Gets the value of the objective functions.

More at: https://optimathsat.disi.unitn.it/pages/smt2reference.html.

When calling OptiMathSAT from the terminal, remember to use the options:

```
-optimization=true -model_generation=true
```

Outline

Selling Apples

Exercise 5.1: Selling Apples

An orchard has 50 apple trees, each producing 800 apples per year.
Planting an additional tree reduces output per tree by 10 apples per year.

How many trees should be added to maximize total output?

Selling apples: variables

As always, we first define the variables to describe the problem:

- Goal: find the number n of additional trees to plant so to maximize the profit.
- n is of type `Int`.

Selling apples: properties

- Total number of trees planted:

$$trees = 50 + n.$$

- Number of apples produced by each tree:

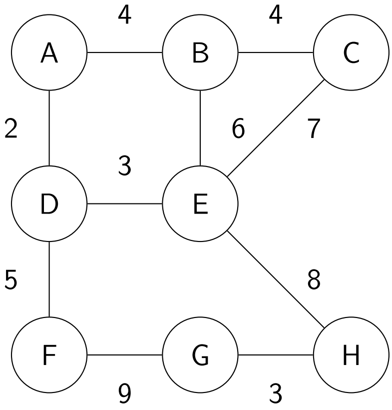
$$treeApples = 800 - 10n.$$

- Maximize total number of apples produced:

$$trees \cdot treeApples.$$

$\Longrightarrow$ Remember that you can maximize complex functions, not just variables.

Exercise 5.2: Shortest Path



Use OptiMathSAT to compute the shortest path between G and B.

Shortest path: variables

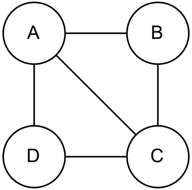
As always, we first define the variables to describe the problem:

- A variable for each node n, storing the value of the shortest path from G to n.
- Thus we must declare `Int` variables.

Shortest path: properties

- Set G's value to 0.
- Define constraints for other nodes based on edge scores.
- Minimize B's value (or any other point).
- We can also generalize to any two points instead of fixing G.

Exercise 5.3: Maximum Clique



Given an undirected graph, a **clique** is a subset of vertices such that every two distinct vertices in the clique are adjacent.

Find the maximum clique of this graph using OptiMathSAT.

Maximum Clique: Three Approaches

There are three approaches to solve the maximum clique problem:

1. OMT(LIA) with Pseudo-Boolean variables.
2. Partial MaxSMT with hard and soft clauses.

Maximum Clique — OMT(LIA)

Variables: A Pseudo-Boolean variable for each node, storing if it is in the clique.

Constraints: If two nodes are not adjacent, then we cannot choose both of them:

```
(assert (<= (+ xi xj) 1))
```

Objective: The goal is to maximize the sum of all variables, i.e. the size of the clique:

```
(maximize (+ x1 ... xn))
```

Result: The variables set to 1 represent the nodes in the maximum clique.

MaxSAT and MaxSMT

Given:

- A set of hard constraints that must be satisfied.
- A set of soft constraints that can be violated, each with a penalty if not satisfied.

Find an assignment of the variables that satisfies all hard constraints and minimizes/maximizes the penalty of the soft constraints.

Depending on the kind of constraints (Boolean or SMT), the problem is called MaxSAT or MaxSMT.

Maximum Clique — MaxSAT

Variables: A Boolean variable for each node, storing if it is in the clique.

Hard-Constraints: If two nodes are not adjacent, then we cannot choose both of them:

```
(assert (or (not xi) (not xj)))
```

Soft-Constraints: For each node, we add a soft clause that allows it to be in the clique:

```
(assert-soft (not xi) :weight 1 :id penalty)
```

Objective: The goal is to maximize penalty (the number of soft-constraints violated).

```
(maximize penalty)
```

Result: The variables set to `true` represent the nodes in the maximum clique.

Maximum Clique — MaxSMT

MaxSMT allows setting SMT constraints as soft constraints.
$\implies$ We can mix-up Pseudo-Boolean and assert-soft constraints.

Variables: A Pseudo-Boolean variable for each node, storing if it is in the clique.

Hard-Constraints: If two nodes are not adjacent, then we cannot choose both of them:

```
(assert (<= (+ xi xj) 1))
```

Soft-Constraints: For each node, we add a soft clause that allows it to be in the clique:

```
(assert-soft (= xi 0) :weight 1 :id penalty)
```

Objective: The goal is to maximize penalty (the number of soft-constraints violated).

```
(maximize penalty)
```

Result: The variables set to `true` represent the nodes in the maximum clique.

OMT vs MaxSMT: Which One to Choose?

OMT

- Allows to optimize arbitrary functions over the variables.
  $\Longrightarrow$ More general!

MaxSMT

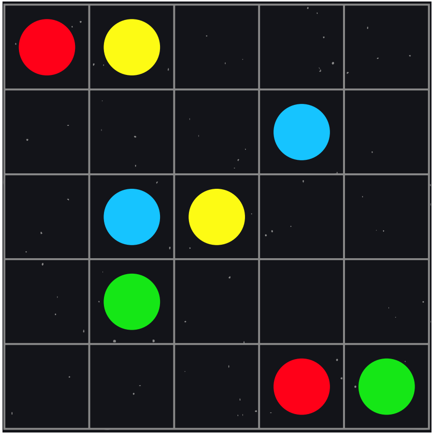
- The objective function could be expressed as a sum of pseudo-Boolean constraints: the optimization search has only a Boolean component.
- Generally, using MaxSMT is more efficient than OMT for this kind of problems.

Outline

1. Introduction to OptiMathSAT
2. OMT Exercises
3. Automating SMT encoding
4. Homeworks

Connect Dots

Exercise 5.4: Connect Dots



Use MathSAT to solve the puzzle shown in the figure. The rules are simple: you must connect dots with the same color with a single line and all cells must be used to generate a valid solution.

Connect dots: variables

As always, we first define the variables that efficiently describe the problem:

- For each cell of the grid a variable
- Creating an `Int`-to-Color mapping (thus setting `Int` as variable type) is enough for our goal.

Connect dots: properties (1)

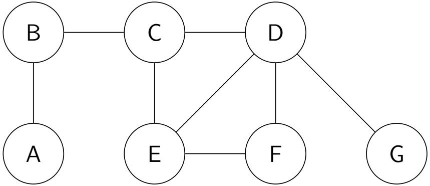
- For each cell we must ensure their value is in the range of the admitted color. Setting both lower and upper bound is enough.
- In each line connecting two dots, the following properties always hold:
  - The extremes have one neighbor cell with the same color.
  - The internal nodes of the lines two neighbor cells with the same color.

Connect dots: properties (2)

- Both conditions can be encoded using respectively *AtLeastOne* and *AtLeastTwo*.
- According to our reasoning we should use *ExactlyOne* and *ExactlyTwo*, but the *AtLeast* operators are sufficient to correctly constrain the problem in our case.

Outline

Homework 5.1: Minimum Vertex Cover



A **vertex-cover set** of an undirected graph is a subset of vertices such that **if an edge belongs to the graph**, then **at least one of the two nodes linked by this edge belongs to the vertex-cover** set.

Use OptiMathSAT to compute the vertex-cover of the **minimum** size of the graph in the figure.

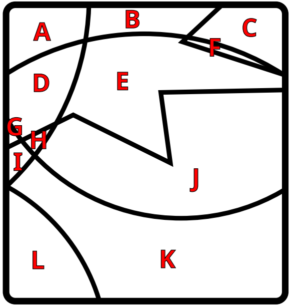
Try to encode it using both OMT and MaxSMT.

Homework 5.2: Nonlinear Programming

Find two non-negative numbers x and y so that:

- Their sum is 9.
- The product of one number and the square of the other number is a maximum in the range $(0, 200)$.

Homework 5.3: Minimum Graph Coloring



Modified from "Four Colour Map Example", licensed under CC BY-SA 3.0. Source: https://commons.wikimedia.org/wiki/File:Four_Colour_Map_Example.svg

Solve the color graph problem again with the following map of countries (so you must ensure adjacent countries do not have the same color).

Use OptiMathSAT to retrieve the minimum number of colors that satisfy the problem.