UNIVERSITÀ
DI TRENTO

Formal Method Mod. 1 (Automated Reasoning)

Laboratory 4

Giuseppe Spallitta
giuseppe.spallitta@unitn.it

Università degli studi di Trento

March 31, 2021

UNIVERSITÀ DEGLI STUDI
DI TRENTO

Outline

UNIVERSITÀ DEGLI STUDI DI TRENTO

MathSAT

- MathSAT 5 is an efficient Satisfiability modulo theories (SMT) solver jointly developed by FBK and University of Trento.
- MathSAT supports a wide range of theories (including e.g. equality and uninterpreted functions, linear arithmetic, bit-vectors, and arrays).
- More information can be found here:
  `https://mathsat.fbk.eu/`

UNIVERSITÀ DEGLI STUDI DI TRENTO

MathSAT input format: SMT-LIB

- If we want to use MathSAT, we need to know the **input** format and the **output** provided by the tool.
- The input format accepted by the tool is called **SMT-LIB format**.
- SMT-LIB is an international initiative aimed at facilitating research and development in SMT, the language is only one of several goal achieved.

General structure of a SMT-LIB file

UNIVERSITÀ DEGLI STUDI DI TRENTO

A typical SMT-LIB file is characterized by the following sections:

- The *Option* section
- The *Declaration* section
- The *Assertion* section
- The *Action* section

SMT-LIB file: option

- The header of the file can contain some commands to enable some additional functionalities, such as:
  - Generation of models
  - Extraction of UNSAT cores
  - Extraction of interpolants
  - Set background logic for more efficient computations
- While solving the exercises we will highlight the most popular options and their effects.

Warning

Each command must be written inside brackets!

SMT-LIB file: declaration

UNIVERSITÀ DEGLI STUDI DI TRENTO

- In this section we must declare each variable/function necessary to describe the problem.
- The declaration of variables can be done in the following way:

```
(declare-const <name> <type>)
```

- Types supported by SMT-LIB are:
  - `Bool`
  - `Int`
  - `Real`
  - (_ BitVec <size>)
  - (Array <type> <type>)
- The declaration of functions can be done in the following way:

```
(declare-fun <name> ([input types]) <type>)
```

SMT-LIB file: assertion

- Once defined the variables, it is necessary to determine the constraints that rules the satisfiability of the problem in the form of assertions.
- The declaration of assertions can be done in the following way:

```
(assert <condition>)
```

- Conditions can be basic (i.e. x = 5) or nested (x=2 or x=5).

Warning

In SMT-LIB operators always use a prefix notation!

SMT-LIB assertion: propositional logic

Of course Boolean operators are available to use:

- NEGATION is represented as *(not <var>)*
- OR is represented as *(or <var1> <var2>)*
- AND is represented as *(and <var1> <var2>)*
- IF is represented as *(=> <var1> <var2>)*.
- XOR can be represented as *(xor <var1> <var2>)*
- EQUALITY is represented as *(= <var1> <var2>)*

Warning

The `and` and `or` operators are not only binary operators and can be used with multiple arguments.

SMT-LIB assertion: arithmetic

UNIVERSITÀ DEGLI STUDI DI TRENTO

The SMT-LIB format standardizes syntax for arithmetic over integers and over reals.

- ADDITION is represented as +
- SUBTRACTION is represented as -
- MULTIPLICATION is represented as *
- DIVISION is represented by / (Real) and`div` (Int)
- REMAINDER (only using Int) is represented as `mod`
- Relations among variables (i.e. greater (or equal) than, lower (or equal) than) are represented respectively by > (>=) and < (<=)

Warning

The * and + operators are not only binary operators and can be used with multiple arguments.

SMT-LIB assertion: Bit Vectors

UNIVERSITÀ DEGLI STUDI DI TRENTO

Numbers can be represented using a bit vector representation and require different operators

- ADDITION is represented as *bvadd <var1> <var2>*
- SUBTRACTION is represented as *bvsub <var1> <var2>*
- MULTIPLICATION is represented as *bvmul <var1> <var2>*
- DIVISION is represented *bvudiv <var1> <var2>*
- REMAINDER is represented as *bvurem <var1> <var2>*
- Relations among variables (i.e. greater (or equal) than, lower (or equal) than) are represented respectively by `bvugt` (`bvuge`) and `bvult` (`bvule`)

Warning

If you change the *u* into a *s* for the last two sets of operators, you obtain equivalent operations using signed vectors (thus changing the range of admitted values).

SMT-LIB assertion: Arrays

UNIVERSITÀ DEGLI STUDI DI TRENTO

- Arrays map an index type to an element type (similarly to Python `dict` type).
- To select the element associated to index *i* in array *a* the command to use is the following:

```
(select a i)
```

- To update the element associated to index *i* in array *a* with value *e* the command is the following:

```
(store a i em)
```

SMT-LIB file: action

- The bottom part of the file should describe the task the solver has to manage.
- First you should check satisfiability of the actual problem:

```
(check-sat)
```

- We can then ask for the model value of some of the constants (in this case x and z):

```
(get-value (x z))
```

- Lastly we end the file using:

```
(exit)
```

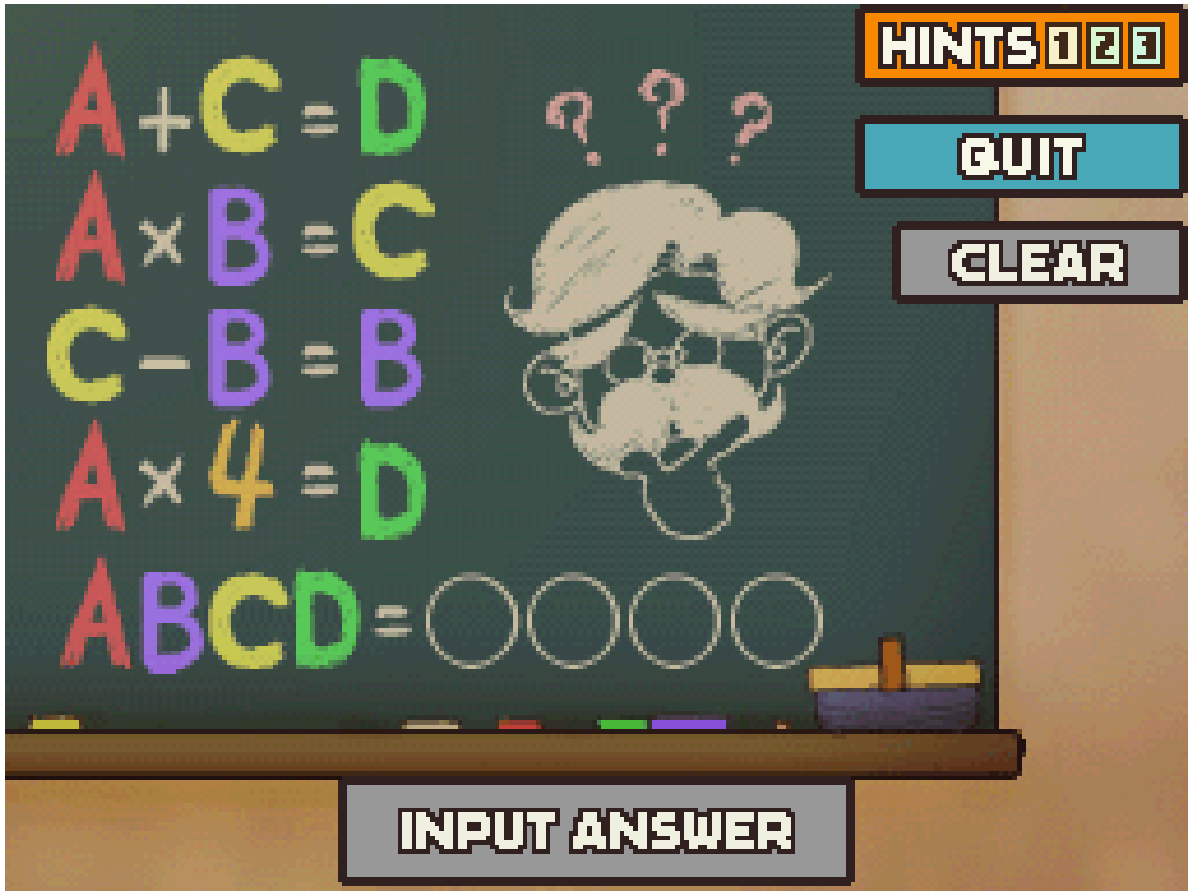
UNIVERSITÀ DEGLI STUDI
DI TRENTO

Outline

First encodings

Exercise 4.1: guess the code

A, B, C and D are single-digit numbers. The following equations can all be made with these numbers:



Encoding step-by-step

UNIVERSITÀ DEGLI STUDI DI TRENTO

The procedure to feed a problem into a SMT solver is identical to the one we adopted for SAT problems:

- Identify the variables that can describe the problem.
- Define the assertions to constraints the domains of each variables and check its satisfiability.

The only relevant difference is the expressive power of SMT-LIB with respect to DIMACS.

UNIVERSITÀ DEGLI STUDI
DI TRENTO

First encodings: variables

- Reading exercise 4.1, we requires 4 constants: A, B, C and D
- Since they are single digit numbers, we set them as *Int*.
- No additional functions are required for this exercise.

First encodings: assertions

- We must encode the 4 equations that are written on the blackboard, using the basic arithmetical operators.
- Moreover we must ensure that all the digits are different: we can use the command `distinct` to easily encode it. If you don't remember it during the exam don't worry, you can encode it by hand...

First encodings: output

- Once we add the final action, we can feed it to the SMT solver
  $\Rightarrow$ The solver returns SAT
- If we want to know the values of the variables, we have to add some options and some additional actions.

UNIVERSITÀ DEGLI STUDI
DI TRENTO

Outline

Solving geometric problems

UNIVERSITÀ DEGLI STUDI DI TRENTO

Exercise 3.2: intersecting lines

Given two points in the Euclidean space (i.e. A(1,3) and B(2,7)), let's define an encoding to determine the lines passing from both points and the value x where the line intersect the x-axis.

Solving geometric problems: variables

UNIVERSITÀ DEGLI STUDI DI TRENTO

- We can set 4 variables to store the coordinates of each point (xa, ya, xb, yb).
- We need also to define a function variable (we will call it f) with arity 1, so that we can have an analytical representation of the line.
- A line is represented by the formula:

$$f(x) = mx + q$$

Thus we need other two variables.

UNIVERSITÀ DEGLI STUDI DI TRENTO

Solving geometric problems: assertions (1)

- We start defining 4 assertions to set the value of the coordinates and one assertion to define the line equation.
- Then we can encode two assertions to calculate the values of m and q using the analytic formulae:

$$m = \frac{yb - ya}{xb - xa}$$

$$q = ya - m * xa$$

Solving geometric problems: assertions (2)

- Now an assertion to update the analytic function f using the calculated parameters is necessary.
- Lastly we intersect the generic line with the equation of the x-axis, which is:

$$y(x) = 0$$

We need to store the value of the horizontal intersection, so we can add a novel variable that will store the solution of this last assertion.

Solving geometric problems: results

UNIVERSITÀ DEGLI STUDI DI TRENTO

- Now we can feed the encoding into MathSAT, obtaining a valid solution.
- The problem can be easily adapted to different sets of points: if we change the coordinates, we will obtain a different line.
- You can also extend this code to generalize this exercises in the case you want to determine the intersection of two lines.

Unlocking phones

UNIVERSITÀ DEGLI STUDI DI TRENTO

Exercise 4.3: unlocking phones

You wants to unlock the mobile phone of your friend to see if they are dating someone. Sadly, there is a 2*2 grid pattern lock that stops you. You remember that the password requires all 4 pins to be connected; moreover there are no diagonal lines in the pattern. How many combinations you have to try in the worst case to unlock the phone?

UNIVERSITÀ DEGLI STUDI
DI TRENTO

Unlocking phones: variables

- This exercise can be modeled as a SAT problem, so we can reason in the same way as the first laboratories.
- In particular we need 16 variables, labeled x_{ij}, where i is the cell in the grid and j is the order in the sequence.

Unlocking phones: assertions

UNIVERSITÀ DEGLI STUDI DI TRENTO

- For each cell in the grid, exactly one temporal position in the sequence is correct.
- For each temporal position in the sequence, exactly one cell in grid must be chosen.
- If a cell in the grid is chosen, we must ensure that the next one is not the diagonal one.

Unlocking phones: results

- If we simply run the *(check-sat)* command we will see that the problem is SAT (thus at least one password exists), but we are interested in knowing the total number of solutions admitted...
- The *(check-allsat)* command returns all possible solution given a set of Boolean variables (if no set is passed as arguments, all te defined Boolean variables are considered). Thanks to it, we can see how many solutions can be generated.

Proof of theories

Exercise 4.4: checking formulae

Prove that the following combination of Linear Arithmetic over Integers and Uninterpreted Function is unsatisfiable:

$$(1 \leq x_1) \wedge (x_2 \leq 3) \wedge (x_3 \leq x_2) \wedge (x_1 \leq 3 - x_3) \wedge (x_3 = 1) \wedge g(x_1) \wedge \neg g(2) \wedge f(x_2) \neq f(x_1) \wedge f(x_2) \neq f(3) \wedge \neg g(x_2 - x_1)$$

Once we prove it, can we find a minimal set of conditions that makes it unsatisfiable?

Proof of theories: variables

UNIVERSITÀ DEGLI STUDI DI TRENTO

First we must define the variables and the functions:

- **Variables**: x_1, x_2, x_3
- **Functions**: f and g, with integers as input and Bool as output.

UNIVERSITÀ DEGLI STUDI DI TRENTO

Proof of theories: assertions

- The conversion into the SMT-LIB format is trivial: each clause of the formula can be independently written using basic operators.
- We can quickly check the unsatisfiability of the formula, but now we must discover the UNSAT core to answer the main question.

UNIVERSITÀ DEGLI STUDI DI TRENTO

Proof of theories: extracting cores

- To extract the core we need to add a label for each assertions using a specific sintax, the command *(! <formula> :named <label>)*.
- Now we can use the *(get-unsat-core)* to obtain the information, assuming the input returns UNSAT.
- What happens if we add the constraint $x_1 >= x_2$?

UNIVERSITÀ DEGLI STUDI
DI TRENTO

Outline

Homework

UNIVERSITÀ DEGLI STUDI DI TRENTO

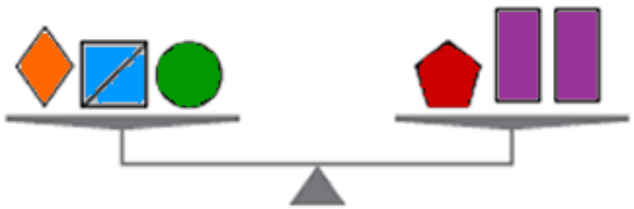
Homework 4.1: math olympics

Find the number of positive integers with three not necessarily distinct digits, abc, with a $\neq 0$ and c $\neq 0$ such that both abc and cba are multiples of 4.

Homework

Homework 4.2: balance puzzle

Solve it using an SMT solver (use some temporary variables to store the possible solutions...)



a. 

b. 

c. 

d. 

Homework

UNIVERSITÀ DEGLI STUDI DI TRENTO

Homework 4.3: checking the correctness of the slides

Apply the extraction of UNSAT cores using MathSAT to see if we can obtain the same answer of the example shown in slide 95 (handouts) of this presentation: `http://disi.unitn.it/~rseba/DIDATTICA/fm2021/SLIDES/03-smt_handouts.pdf`