SMT: A Beginner's Tutorial

Clark Barrett[1], Cesare Tinelli[2], Haniel Barbosa[3], Aina Niemetz[1], Mathias Preiner[1] and Andrew Reynolds[2], Yoni Zohar[4]

[1] Stanford University, USA
[2] The University of Iowa, USA
[2] UFMG, Brazil
[2] Bar-Ilan University, Israel

FM Tutorial, September 10, 2024

About

About Me

- ▷ Yoni Zohar
- ▷ Faculty member at BIU CS Dept.
- ▷ Working on SMT, cvc5
- ▷ Also: logic and proof theory

About The Tutorial

- ▷ Compatible with cvc5 and z3
- ▷ Installations
- ▷ Background
- ▷ Examples and exercises
  - I solve an example
  - You solve an exercise
  - I also solve the exercise
- ▷ Summary

About You

- ▷ Students / Industry
- ▷ Research
- ▷ Interests
- ▷ Prior knowledge on SMT

Using an SMT Solver

Choose one option from each interface

Python Interface

▷ Terminal

```
python3 -m venv smt-tutorial
source smt-tutorial/bin/activate
python3 -m pip install cvc5-gpl
python3
from cvc5.pythonic import *
```

▷ Online

- `https://colab.research.google.com/`
- `!pip install cvc5-gpl`
- `from cvc5.pythonic import *`

Text Interface

▷ Terminal – download, unzip, run `./<...>/bin/cvc5`

- cvc5-Linux-arm64-static-gpl.zip
- cvc5-Linux-x86_64-static-gpl.zip
- cvc5-macOS-arm64-static-gpl.zip
- cvc5-macOS-x86_64-static-gpl.zip
- cvc5-Win64-x86_64-static.zip
  (incomplete, better to use WSL)

▷ Online

- `https://cvc5.github.io/app/`

These slides: `https://tinyurl.com/fm24-tutorial-smt`

Background

Entscheidungsproblem – "decision problem"

- ▷ Input: set of assumptions, conclusion
- ▷ Output: does the conclusion follow from the assumptions?
- ▷ Decidable? [Hilbert]
- ▷ Undecidable [Church, Turing]
- ▷ Option 1: give up
- ▷ Option 2: automated reasoning



Which Consequences?

- ▷ Typically:
  - Assumptions: axioms
  - Conclusion: particular formula
- ▷ Example:
  - Assumptions: Peano Arithmetic
  - Conclusion: $xy + xz = x(y+z)$
- ▷ More examples:
  - Assumptions: axioms of regular expressions
  - Conclusion: aaa belongs to L(a*)
  - ...
- ▷ Observation:
  - Assumptions are relatively constant
  - Conclusion changes
  - Axioms can be hard-coded

Satisfiability Modulo Theories (SMT)

Satisfiability Modulo Theories (SMT)

Satisfiability Modulo Theories (SMT)

- From consequences to satisfiability
  - $A_1 \wedge \ldots \wedge A_n \rightarrow B$ iff $A_1 \wedge \ldots \wedge An \wedge \neg B$ is UNSAT

Satisfiability Modulo Theories (SMT)

- From consequences to satisfiability
  - $A_1 \wedge \ldots \wedge A_n \rightarrow B$ iff $A_1 \wedge \ldots \wedge An \wedge \neg B$ is UNSAT
- Classic satisfiability problem (SAT / 3SAT)
  - All variables are Boolean
  - Operators are either $\wedge$, $\vee$, or $\neg$

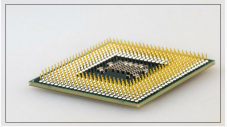
Satisfiability Modulo Theories (SMT)

- From consequences to satisfiability
  - $A_1 \wedge \ldots \wedge A_n \rightarrow B$ iff $A_1 \wedge \ldots \wedge An \wedge \neg B$ is UNSAT
- Classic satisfiability problem (SAT / 3SAT)
  - All variables are Boolean
  - Operators are either $\wedge$, $\vee$, or $\neg$
- SMT relaxes both restrictions
  - Arbitrary Boolean structure
  - Variables can be integers, reals, bit-vectors, arrays, strings, and more and more
  - Wide range of supported operators
- Prominent Solvers:
  - cvc5 (Stanford, U. Iowa), z3 (Microsoft)
  - Bitwuzla (Stanford), Yices (SRI)
  - Mathsat (FBK), Vampire (U. Manchester)

  ...

Applications

- ▷ Hardware verification (e.g., Intel, Cadence)
  - Translate a circuit to a formula
  - Translate a specification of the circuit to a formula
  - Check for equivalence
- ▷ Software verification (e.g., Microsoft, Meta)
  - Similar idea to hardware
  - Different challenges
- ▷ Access control (e.g., AWS)
  - Model access control policies using regular expressions
  - Check that one policy is a refinement of another
- ▷ More
  - Planning, optimization, placement, general problem solving



Interfaces

Python

- ▷ Designed by Z3
- ▷ Meant to be simple and intuitive

```
from cvc5.pythonic import *
a, b = Ints('a b')
solve(a + 10 == 2 * b, b + 22 == 2 * a)
```

SMT-LIB

- ▷ Textual interface
- ▷ Also: a *standard*
- ▷ Solvers follow it (and it follows solvers)
- ▷ Used by both humans and computers

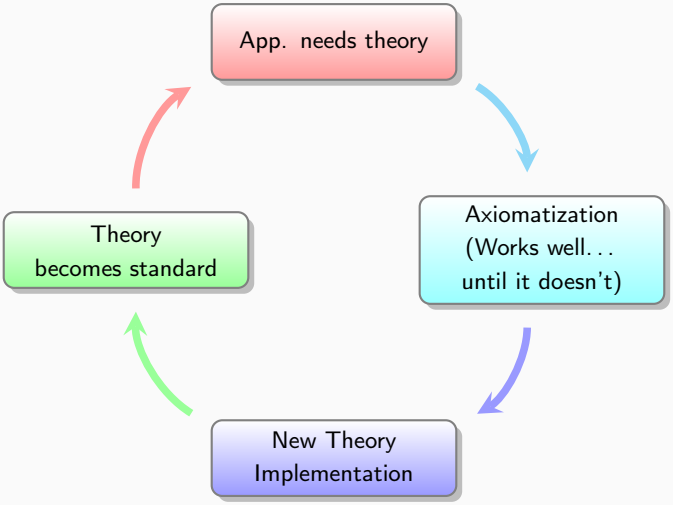
```
(set-logic QF_LIA)
(set-option :produce-models true)

(declare-const a Int)
(declare-const b Int)

(assert (= (+ a 10) (* 2 b)))
(assert (= (+ b 22) (* 2 a)))

(check-sat)
(get-model)
```

Other

- ▷ There are more interfaces
  - C, C++, OCaml, Rust, Java, ...
- ▷ Not in this tutorial

```
Term x = tm.mkConst(realSort, "x");
Term y = tm.mkConst(realSort, "y");
Term a = tm.mkConst(intSort, "a");
Term b = tm.mkConst(intSort, "b");
```

The SMT Cycle



This Tutorial

- ▷ Audience: beginners
- ▷ By example, hands-on
- ▷ Skills:
  - Encoding to SMT
  - Capabilities and limitations
  - Using SMT solvers
- ▷ Solvers: cvc5, Z3
- ▷ APIs: Python, textual
- ▷ Use either or both

Outline

Theories

Basic:

- ▷ Booleans
- ▷ Arithmetic

Advanced:

- ▷ Bit-vectors and Floating Points
- ▷ Strings
- ▷ Finite Fields

Containers:

- ▷ Uninterpreted Functions
- ▷ Arrays
- ▷ Datatypes
- ▷ Sequences and Finite Sets

Solver Interaction

- ▷ Combination of Theories
- ▷ Models
- ▷ Unsat Cores
- ▷ Proofs

Example
Find a way to partition Jerry, Kramer, George, Elaine, Newman, Susan into two groups, so that Jerry is not in the same group with neither Newman nor Kramer, Susan is not in the same group with neither George nor Elaine, and George must be in the same group as Jerry.

Observation

- ▷ The *groups* can be just *true* and *false*.
- ▷ Text + Python solutions
- ▷ cvc5 + z3

Defininig Variables

- ▷ Python: python variables
- ▷ SMT-LIB: `declare-const` and `declare-fun`
  - In first-order logic variables can be converted to constants via *Skolemization*

Exercise

Show that $A \wedge (B \vee C)$ is equivalent to $(\neg A \vee \neg B) \Rightarrow (A \wedge C)$.

Hint: $\Rightarrow$ is Implies in the Python API and $=>$ in SMT-LIB.

Hint: X is equivalent to Y if $\neg(X \leftrightarrow Y)$ is UNSAT

Outline

Theories

Basic:

- ✓ Booleans
- Arithmetic

Advanced:

- Bit-vectors and Floating Points
- Strings
- Finite Fields

Containers:

- Uninterpreted Functions
- Arrays
- Datatypes
- Sequences and Finite Sets

Solver Interaction

- Combination of Theories
- Models
- Unsat Cores
- Proofs

Example

In 10 years, Alice will be twice as old as Bob is now, but in 22 years, Bob will be twice as old as Alice is now. How old are Alice and Bob?

Arithmetic

Example

In 10 years, Alice will be twice as old as Bob is now, but in 22 years, Bob will be twice as old as Alice is now. How old are Alice and Bob?

Exercise

Consider the following modification: The first assertion will stay the same, but for the second, let's assert that Bob will be twice as old as Alice in only 20 years. What output does the SMT solver give now?

Arithmetic

Example

In 10 years, Alice will be twice as old as Bob is now, but in 22 years, Bob will be twice as old as Alice is now. How old are Alice and Bob?

Exercise

Consider the following modification: The first assertion will stay the same, but for the second, let's assert that Bob will be twice as old as Alice in only 20 years. What output does the SMT solver give now?

Exercise

Go to the solution of the last exercise, but change the logic to `QF_LRA`, change the types of the variables from `Int` to `Real`, and append `.0` to each numeric constant. Now, what output does the solver give?

Arithmetic

Logics

- ▷ Logic = language + theory
- ▷ *ALL* logic is everything that the solver supports
- ▷ Specifying a more specific logic can improve performance and prevent modeling bugs

Which Arithmetic?

- ▷ Quantifier-free (QF)
- ▷ Linear / non-linear (L / N)
- ▷ Over reals or integers (R / I)
- ▷ "super linear" over integers: integer difference logic
- ▷ `QF_LRA`, `QF_NRA`, `QF_LIA`, `QF_NIA`, `QF_IDL`

Remark

- ▷ Non-linear over reals is exp^2, and over integers is undecidable
- ▷ Still, there are heuristics

Exercise

Are there real solutions for $x^2y + yz + 2xyz + 4xy + 8xz + 16 = 0$? Which logic did you specify?

Exercise

Are there real solutions for $x^2y + yz + 2xyz + 4xy + 8xz + 16 = 0$? Which logic did you specify?

Incremental Mode

- ▷ Interacting with the solver
- ▷ Requires turning on ‘–incremental‘
- ▷ Creates a stack of assertions

Exercise

Are there real solutions for $x^2y + yz + 2xyz + 4xy + 8xz + 16 = 0$? Which logic did you specify?

Incremental Mode

- ▷ Interacting with the solver
- ▷ Requires turning on '–incremental'
- ▷ Creates a stack of assertions

Example

Can you find a solution where all variables are positive? And one where all are negative?

“Easy" Arithmetic

Difference Logic

- ▷ All constraints look like $x - y \bowtie c$ or $x \bowtie c$, with $\bowtie \in \{=, <, >, \leq, \geq\}$, c is a constant.
- ▷ Polynomial complexity
- ▷ Name: `QF_IDL`
- ▷ Easy to solve, hard to encode

Example
Show that the maximum of x, y, z is greater than or equal to all of them.

ite and define-fun

- ▷ The `ite` operator is very useful (`If` in python)
- ▷ Not standard in Boolean logic, but heavily used in SMT.
- ▷ `define-fun` is a way to define macros (not needed in python)

Example

Suppose we have 3 jobs to complete on 2 machines. Job 1 requires machine 1 for 10 minutes and then machine 2 for 5 minutes. Job 2 requires machine 2 for 20 minutes and then machine 1 for 5 minutes. And Job 3 requires machine 1 for 5 minutes and then machine 2 for 5 minutes. Can all jobs be completed in 30 minutes?

Example

Suppose we have 3 jobs to complete on 2 machines. Job 1 requires machine 1 for 10 minutes and then machine 2 for 5 minutes. Job 2 requires machine 2 for 20 minutes and then machine 1 for 5 minutes. And Job 3 requires machine 1 for 5 minutes and then machine 2 for 5 minutes. Can all jobs be completed in 30 minutes?

Clarifications

- ▷ Each machine can work on at most 1 job at a time.
- ▷ The order of the jobs (1,2,3) does not matter.
- ▷ But in each job, the order in the description matters.

Arithmetic

Example

Suppose we have 3 jobs to complete on 2 machines. Job 1 requires machine 1 for 10 minutes and then machine 2 for 5 minutes. Job 2 requires machine 2 for 20 minutes and then machine 1 for 5 minutes. And Job 3 requires machine 1 for 5 minutes and then machine 2 for 5 minutes. Can all jobs be completed in 30 minutes?

Clarifications

- ▷ Each machine can work on at most 1 job at a time.
- ▷ The order of the jobs (1,2,3) does not matter.
- ▷ But in each job, the order in the description matters.

Idea

- ▷ Create a variable xij for when the *j*th step of job *i* starts.
- ▷ start times are non-negative
- ▷ the second step of each job starts after the first has ended
- ▷ machine times do not overlap.
- ▷ end times are less than 30

Exercise

What is the minimum amount of time that it will take to complete all of the jobs in the previous example? And what is it, if instead job 2 uses machine 2 for only 15 minutes?

Outline

Theories

Basic:

- ✓ Booleans
- ✓ Arithmetic

Advanced:

- Bit-vectors and Floating Points
- Strings
- Finite Fields

Containers:

- Uninterpreted Functions
- Arrays
- Datatypes
- Sequences and Finite Sets

Solver Interaction

- Combination of Theories
- Models
- Unsat Cores
- Proofs

Bit-vectors – Basic

Bit-vectors

- ▷ Models machine integers
- ▷ Supports common operators on them
- ▷ arithmetic overflows
- ▷ `(_ BitVec k)` or `BitVec("name", k)`

Example

Let x and y be 32-bit integers, with x a multiple of 2. Is it possible for the machine arithmetic product of x and y to be 1?

Bit-vectors – Basic

Bit-vectors

- ▷ Models machine integers
- ▷ Supports common operators on them
- ▷ arithmetic overflows
- ▷ `(_ BitVec k)` or `BitVec("name", k)`

Example

Let x and y be 32-bit integers, with x a multiple of 2. Is it possible for the machine arithmetic product of x and y to be 1?

Exercise

Find the machine multiplicative inverse of 5, using 32 bits.

Hint: `bvmul`, `(_ bv5 32)` in SMT-LIB. No hints for python.

Bit-vectors – Advanced

Example
Consider the following two implementations of the absolute value opeartor for 32-bit integers:

0. $abs_0(x) := x < 0 \; ? \; -x : x$
1. $abs_1(x) := (x \oplus (x \gg_a 31)) - (x \gg_a 31)$

Prove that they are equivalent.

Hint: `bvult`,`bvneg`,`bvashr`,`bvxor`, `>>`,`^`

Bit-vectors – Advanced

Example
Consider the following two implementations of the absolute value opeartor for 32-bit integers:

0. $abs_0(x) := x < 0\ ?\ -x : x$
1. $abs_1(x) := (x \oplus (x \gg_a 31)) - (x \gg_a 31)$

Prove that they are equivalent.

Hint: `bvult,bvneg,bvashr,bvxor, >>,^`

Exercise

Consider two more possible implementations of the same function:

2. $abs_2(x) := (x + (x \gg_a 31)) \oplus (x \gg_a 31)$
3. $abs_3(x) := x - ((x \ll 1)\ \&\ (x \gg_a 31))$

Prove that all four are equivalent to one another.

Hint: `bvshl,bvand,bvadd`

FP in SMT

- ▷ Follows IEEE 754-2019
- ▷ FP number = triple of bit-vectors
- ▷ Wide range of operators
  - take a rounding mode as input an additional input
  - rounding modes: round up, down, towards zero, etc.
- ▷ E.g., addition, multiplication, fused-multiplication-addition

Example

Given three single precision (Float32) floating-point numbers a, b, and c, show that the floating-point fused multiplication and addition of a, b, and c is different from first multiplying a and b and then adding c.

Floating-Points Arithmetic

FP in SMT

- ▷ Follows IEEE 754-2019
- ▷ FP number = triple of bit-vectors
- ▷ Wide range of operators
  - take a rounding mode as input an additional input
  - rounding modes: round up, down, towards zero, etc.
- ▷ E.g., addition, multiplication, fused-multiplication-addition

Example

Given three single precision (Float32) floating-point numbers a, b, and c, show that the floating-point fused multiplication and addition of a, b, and c is different from first multiplying a and b and then adding c.

Exercise

Modify the solution to show that floating-point addition is not associative, i.e., $a + (b + c) \neq (a + b) + c$.

Outline

Theories

Basic:

- ✓ Booleans
- ✓ Arithmetic

Advanced:

- ✓ Bit-vectors and Floating Points
- Strings
- Finite Fields

Containers:

- Uninterpreted Functions
- Arrays
- Datatypes
- Sequences and Finite Sets

Solver Interaction

- Combination of Theories
- Models
- Unsat Cores
- Proofs

Strings

SMT Strings

- ▷ Represent common programming languages Unicode strings
- ▷ Supports a wide range of operators
  - concatenation, length, substring, regular expressions, etc

Example
Given two strings, `x1` and `x2`, each consisting of no more than two characters, is it possible to build the string `"abbaabb"` using only 3 string concatenations (where each concatenation may use any previous result including `x1` and `x2`)?

Idea

- ▷ Many options: x1x1x1, x1x1x2, ((x1x2)(x1x2))(x1x2), ...
- ▷ Additional variables for intermmediate concatenations
- ▷ str.len, str.++, Length, Concat,...
- ▷ Brute force (Or) vs. circuit (`ite`)

Exercise

Use SMT to determine how many concatenations are needed to get `"abbaabb"` if `x1` and `x2` are both restricted to have a length of 1.

Outline

Theories

Basic:

- ▷ ✓ Booleans
- ▷ ✓ Arithmetic

Advanced:

- ▷ ✓ Bit-vectors and Floating Points
- ▷ ✓ Strings
- ▷ Finite Fields

Containers:

- ▷ Uninterpreted Functions
- ▷ Arrays
- ▷ Datatypes
- ▷ Sequences and Finite Sets

Solver Interaction

- ▷ Combination of Theories
- ▷ Models
- ▷ Unsat Cores
- ▷ Proofs

Finite Fields

Finite Fields

- ▷ Shows up in crypto and blockchains
- ▷ Currently is not supported by z3
- ▷ Fields (associativity, distributivity, etc.)
- ▷ Order = number of elements
- ▷ Supports prime orders only
- ▷ isomorphic to integers modulo p

Example
In a finite field of order 13, find two elements such that their sum and product are both equal to the multiplicative identity in the field.

Finite Fields

Finite Fields

- ▷ Shows up in crypto and blockchains
- ▷ Currently is not supported by z3
- ▷ Fields (associativity, distributivity, etc.)
- ▷ Order = number of elements
- ▷ Supports prime orders only
- ▷ isomorphic to integers modulo p

Example

In a finite field of order 13, find two elements such that their sum and product are both equal to the multiplicative identity in the field.

Exercise

In a finite field of order 13, find an element such that if you square it twice you get the multiplicative identity.

Outline

Theories

Basic:

- ▷ ✓ Booleans
- ▷ ✓ Arithmetic

Advanced:

- ▷ ✓ Bit-vectors and Floating Points
- ▷ ✓ Strings
- ▷ ✓ Finite Fields

Containers:

- ▷ Uninterpreted Functions
- ▷ Arrays
- ▷ Datatypes
- ▷ Sequences and Finite Sets

Solver Interaction

- ▷ Combination of Theories
- ▷ Models
- ▷ Unsat Cores
- ▷ Proofs

Uninterpreted Functions

Example
Let f be a unary function from U to U, for some set U. Check that, whatever the meaning of f, if $f(f(f(x))) = x$ and $f(f(f(f(f(x))))) = x$, then $f(x) = x$.

Example

Suppose we know that x is either equal to y or z, depending on the value of the Boolean b. Suppose we further know that w is equal to one of y or z. Does it follow that $x = w$?

Comments

- ▷ Sorts are like types in PL
- ▷ Uninterpreted sorts are used when predefined sorts cannot model the problem
- ▷ Constraints with UFs can be satisfied with any interpretation of the UFs

Uninterpreted Functions

Example
Let f be a unary function from U to U, for some set U. Check that, whatever the meaning of f, if $f(f(f(x))) = x$ and $f(f(f(f(f(x))))) = x$, then $f(x) = x$.

Example

Suppose we know that x is either equal to y or z, depending on the value of the Boolean b. Suppose we further know that w is equal to one of y or z. Does it follow that $x = w$?

Comments

- ▷ Sorts are like types in PL
- ▷ Uninterpreted sorts are used when predefined sorts cannot model the problem
- ▷ Constraints with UFs can be satisfied with any interpretation of the UFs

Exercise

Make the first example satisfiable and the second unsatisfiable. How does the model look?

Quantifiers

Example
If all humans are mortal, and Socrates is a human, then must Socrates be mortal?

Quantifiers

Example
If all humans are mortal, and Socrates is a human, then must Socrates be mortal?

Exercise

If all humans are mortal, and Socrates is mortal, then must Socrates be human?

Quantifiers

Example
If all humans are mortal, and Socrates is a human, then must Socrates be mortal?

Exercise

If all humans are mortal, and Socrates is mortal, then must Socrates be human?

Quantifier in SMT

- ▷ Many SMT solvers support quantifiers with all theories. But...
- ▷ UF with quantifiers is undecidable
- ▷ Many heuristics
- ▷ (Example: finite model finding)

Quantifiers

Example
If all humans are mortal, and Socrates is a human, then must Socrates be mortal?

Exercise

If all humans are mortal, and Socrates is mortal, then must Socrates be human?

Quantifier in SMT

- ▷ Many SMT solvers support quantifiers with all theories. But...
- ▷ UF with quantifiers is undecidable
- ▷ Many heuristics
- ▷ (Example: finite model finding)

Exercise

Is there a unary function f over some set S such that the cardinality of its image is ≤ 2?

Outline

Theories

Basic:

- ▷ ✓ Booleans
- ▷ ✓ Arithmetic

Advanced:

- ▷ ✓ Bit-vectors and Floating Points
- ▷ ✓ Strings
- ▷ ✓ Finite Fields

Containers:

- ▷ ✓ Uninterpreted Functions
- ▷ Arrays
- ▷ Datatypes
- ▷ Sequences and Finite Sets

Solver Interaction

- ▷ Combination of Theories
- ▷ Models
- ▷ Unsat Cores
- ▷ Proofs

Arrays

The Theory of Arrays

- ▷ Three sorts: index, element, arrays
- ▷ two operators: read, write
- ▷ Actually models maps

Example
Consider the following python function:

```
def swap(a,i,j):
  tmp = a[i]
  a[i] = a[j]
  a[j] = tmp
```

show that if `a[i]` and `a[j]` are equal, then so are `a` and `swap(a,i,j)`.

Arrays

The Theory of Arrays

- ▷ Three sorts: index, element, arrays
- ▷ two operators: read, write
- ▷ Actually models maps

Example
Consider the following python function:

```
def swap(a,i,j):
  tmp = a[i]
  a[i] = a[j]
  a[j] = tmp
```

show that if `a[i]` and `a[j]` are equal, then so are `a` and `swap(a,i,j)`.

Exercise

Another property of `swap` that we can prove is that if `a[i]` and `a[j]` are distinct, then `swap` would change `a`. Modify the solution to prove this property.

Outline

Theories

Basic:

- ✓ Booleans
- ✓ Arithmetic

Advanced:

- ✓ Bit-vectors and Floating Points
- ✓ Strings
- ✓ Finite Fields

Containers:

- ✓ Uninterpreted Functions
- ✓ Arrays
- Datatypes
- Sequences and Finite Sets

Solver Interaction

- Combination of Theories
- Models
- Unsat Cores
- Proofs

Datatypes

Theory of Datatypes

- ▷ Models lists and trees
- ▷ Constructors, selectors, testers
- ▷ Useful for common data structures
- ▷ What is $\mathit{first}(\mathit{nil})$?

Example

Model a binary tree containing integer data. Find trees x and y such that (i) the left subtree of x is the same as the right subtree of y and (ii) the data stored in (the root of) x is greater than 100.

Datatypes

Theory of Datatypes

- ▷ Models lists and trees
- ▷ Constructors, selectors, testers
- ▷ Useful for common data structures
- ▷ What is $first(nil)$?

Example

Model a binary tree containing integer data. Find trees x and y such that (i) the left subtree of x is the same as the right subtree of y and (ii) the data stored in (the root of) x is greater than 100.

Exercise

Show that a tree cannot be equal to its own left subtree.

Outline

Theories

Basic:

- ✓ Booleans
- ✓ Arithmetic

Advanced:

- ✓ Bit-vectors and Floating Points
- ✓ Strings
- ✓ Finite Fields

Containers:

- ✓ Uninterpreted Functions
- ✓ Arrays
- ✓ Datatypes
- Sequences and Finite Sets

Solver Interaction

- Combination of Theories
- Models
- Unsat Cores
- Proofs

Sequences

Sequences

- ▷ Combines strings and arrays
- ▷ Models std::vector in cpp, List in Java, lists in python, etc.
- ▷ Relatively new, not officially in the standard yet

Example

Let `x` be a sequence of integers. Find a value for `x` such that the first and last elements sum to 9, and if we concatenate `x` with itself, then `(3,4,5)` appears as a subsequence.

Sequences

Sequences

- ▷ Combines strings and arrays
- ▷ Models std::vector in cpp, List in Java, lists in python, etc.
- ▷ Relatively new, not officially in the standard yet

Example

Let `x` be a sequence of integers. Find a value for `x` such that the first and last elements sum to 9, and if we concatenate `x` with itself, then `(3,4,5)` appears as a subsequence.

Exercise

Show that it's not possible to have sequences `x`, `y`, and `z` such that `x` is a proper prefix of `y`, `y` is a proper prefix of `z`, and `z` is a proper prefix of `x`.

Hint: seq.prefixof, PrefixOf

Finite Sets

A Theory of Sets

- ▷ Finite sets
- ▷ Models std::set in cpp, Set in Java, set in python
- ▷ Supports many operators: cardinality, union, intersection, etc.
- ▷ Relatively new, not officially in the standard yet

Example
Verify that union distributes over intersection.

Finite Sets

A Theory of Sets

- ▷ Finite sets
- ▷ Models std::set in cpp, Set in Java, set in python
- ▷ Supports many operators: cardinality, union, intersection, etc.
- ▷ Relatively new, not officially in the standard yet

Example
Verify that union distributes over intersection.

Exercise

Does set difference distribute over intersection? If not, find a counterexample.

Outline

Theories

Basic:

- ✓ Booleans
- ✓ Arithmetic

Advanced:

- ✓ Bit-vectors and Floating Points
- ✓ Strings
- ✓ Finite Fields

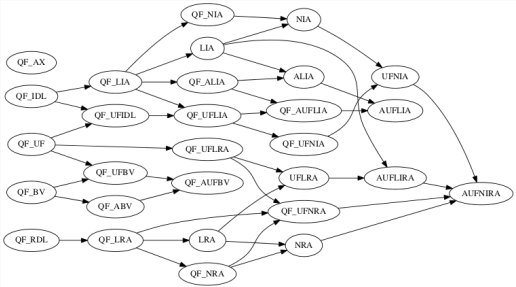
Containers:

- ✓ Uninterpreted Functions
- ✓ Arrays
- ✓ Datatypes
- ✓ Sequences and Finite Sets

Solver Interaction

- Combination of Theories
- Models
- Unsat Cores
- Proofs

Combination of Theories



Logics

- ▷ You can always use (`set-logic ALL`)
- ▷ Using specific names can be more efficient
- ▷ Rules
  - `QF_` or not;
  - `A` for arrays;
  - `UF` for UFS;
  - `BV` for bit-vectors;
  - `FP` for floating points;
  - `DT` for datatypes;
  - `S` for strings and sequences; one of the arithmetic names (e.g., `LIA`);
  - `FF` for finite fields;
  - `FS` for finite sets

Outline

Theories

Basic:

- ✓ Booleans
- ✓ Arithmetic

Advanced:

- ✓ Bit-vectors and Floating Points
- ✓ Strings
- ✓ Finite Fields

Containers:

- ✓ Uninterpreted Functions
- ✓ Arrays
- ✓ Datatypes
- ✓ Sequences and Finite Sets

Solver Interaction

- ✓ Combination of Theories
- Models
- Unsat Cores
- Proofs

Models

Example

Find 3 integers x, y, z such that the sum of the first two is equal to the sum of the last two.

Use both `get-model` and `get-value` (model(), model()[]).

Models

Example

Find 3 integers x, y, z such that the sum of the first two is equal to the sum of the last two.

Use both `get-model` and `get-value` (model(), model()[]).

Exercise

Find the sum of said 3 integers, using `get-value`.

Models

Example
Find 3 integers x, y, z such that the sum of the first two is equal to the sum of the last two.
Use both `get-model` and `get-value` (model(), model()[]).

Exercise

Find the sum of said 3 integers, using `get-value`.

Exercise

Present a function different from the identity function, over some domain.

Models

- ▷ We have already seen this
- ▷ `get-value` can evaluate complex terms
  - It is actually a calculator with variables
- ▷ Notice uninterpreted sorts and functions

Outline

Theories

Basic:

- ✓ Booleans
- ✓ Arithmetic

Advanced:

- ✓ Bit-vectors and Floating Points
- ✓ Strings
- ✓ Finite Fields

Containers:

- ✓ Uninterpreted Functions
- ✓ Arrays
- ✓ Datatypes
- ✓ Sequences and Finite Sets

Solver Interaction

- ✓ Combination of Theories
- ✓ Models
- Unsat Cores
- Proofs

Unsat Cores

1. Not all assumptions are needed!
2. We can get the ones that were needed this time
3. Not necessarilly minimal
4. Run-dependent concept
5. `--dump-unsat-cores --print-cores-full`

Example
Is there a positive integer that is greater than two and is equal to its own square but smaller than its own cube? Are all assumptions regarding said integer needed? Use SMT-LIB.

Outline

Theories

Basic:

- ▷ ✓ Booleans
- ▷ ✓ Arithmetic

Advanced:

- ▷ ✓ Bit-vectors and Floating Points
- ▷ ✓ Strings
- ▷ ✓ Finite Fields

Containers:

- ▷ ✓ Uninterpreted Functions
- ▷ ✓ Arrays
- ▷ ✓ Datatypes
- ▷ ✓ Sequences and Finite Sets

Solver Interaction

- ▷ ✓ Combination of Theories
- ▷ ✓ Models
- ▷ ✓ Unsat Cores
- ▷ Proofs

Proofs

Example
Using SMT-LIB, *Prove* that if all humans are mortal, and Socrates is a human, then must Socrates be mortal?

Proofs

- ▷ New and ongoing research project
- ▷ Already strong capabilities
- ▷ Formats: `cpc`, `alethe`, `dot`,...
- ▷ `dump-proofs`

Outline

Theories

Basic:

- ✓ Booleans
- ✓ Arithmetic

Advanced:

- ✓ Bit-vectors and Floating Points
- ✓ Strings
- ✓ Finite Fields

Containers:

- ✓ Uninterpreted Functions
- ✓ Arrays
- ✓ Datatypes
- ✓ Sequences and Finite Sets

Solver Interaction

- ✓ Combination of Theories
- ✓ Models
- ✓ Unsat Cores
- ✓ Proofs

Summary

SMT

- ▷ Very powerful tool in many contexts
- ▷ Accessible via various APIs
- ▷ Usable on manual benchmarks and auto-generated ones

This Tutorial

- ▷ Basic introduction
- ▷ Covered all main theories and some new ones
- ▷ Same for features

Go Beyond

- ▷ SMT-LIB: definitions, examples, links, benchmarks.
- ▷ Papers and book-chapters
- ▷ Solver papers
- ▷ This full tutorial with everything solved