SMT Proof Production and Integration with the Lean Theorem Prover

Haniel Barbosa, Universidade Federal de Minas Gerais

UFMG

Dagstuhl Seminar 23471

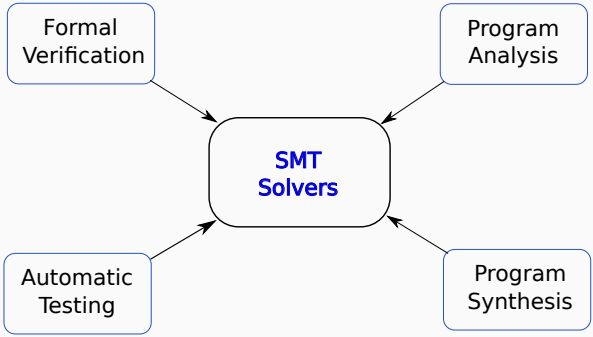
2023–11–20, Schloss Dagstuhl, DE

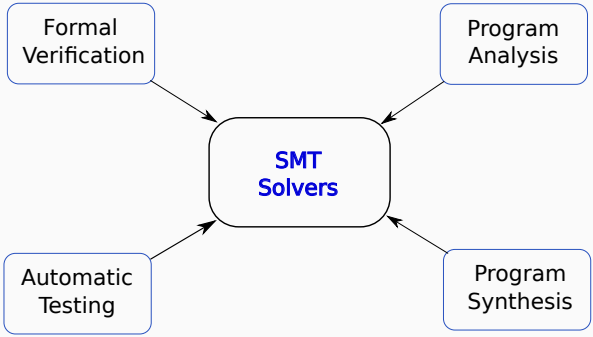
Agenda

- SMT solvers matter. Moreover we'd like to trust them.
- Producing SMT proofs
  - Challenges
  - Our approach
- Integration with the Lean theorem prover
  - How and challenges for proofs
  - Current and future work

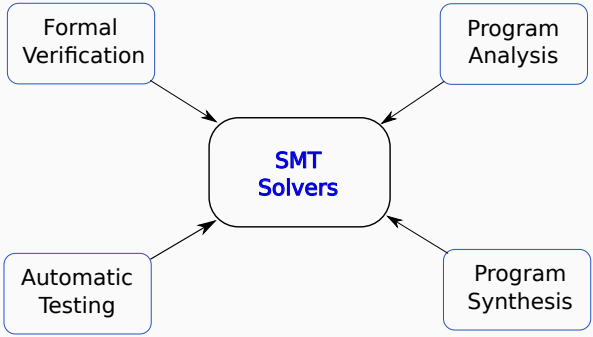
Joint work with:

- cvc5 proofs: Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar
- Alethe proof checking: Bruno Andreotti
- Isabelle/HOL integration: Hanna Lachnitt
- Lean integration: Tomaz Mascarenhas, Abdalrhman Mohamed, Harun Kahn, Wojciech Nawrocki


Formal Verification
Program Analysis
SMT Solvers
Automatic Testing
Program Synthesis



For example, SMT solvers called billions of times a day at AWS in a security critical setting...



For example, SMT solvers called billions of times a day at AWS in a security critical setting...

- Even a tiny fraction of wrong answers is bad

Bugs in SMT Solvers

- State-of-the-art solvers are large projects:
  - cvc5: 300k LoC (C++)
  - z3: 500k LoC (C++)

Bugs in SMT Solvers

- State-of-the-art solvers are large projects:
  - CVC5: 300k LoC (C++)
  - Z3: 500k LoC (C++)

- How do developers try to avoid bugs?
  - Code reviews
  - Testing on benchmark sets
  - Random input testing

Bugs in SMT Solvers

- State-of-the-art solvers are large projects:
  - cvc5: 300k LoC (C++)
  - z3: 500k LoC (C++)
- How do developers try to avoid bugs?
  - Code reviews
  - Testing on benchmark sets
  - Random input testing
- But bugs remain:
  - Every year SMT-COMP has disagreements between solvers
  - Fuzzing tools often find bugs in solvers

Can We Just Certify the Solvers?

- Large, complex code bases are too costly to certify
- A (simpler) certified system can be too slow [FBL18; Fle19]
- Certifying/qualifying a system freezes it, potentially blocking improvements
  - Working around adding new features slow and costly [BD18]

For your consideration: proofs!

- Proofs are a justification of the logical reasoning the solver has performed to find a solution
- A proof can be checked *independently*
  - Checkers have smaller trusted base
    - LFSC: 5.5k (C++) LoC checker + 2k (LFSC) LoC signatures
    - ALETHE: the CARCARA checker (and elaborator): 12k (Rust) LoC
  - Proof checking is generally more efficient than solving the problem
- Proofs can be reconstructed within skeptical proof assistants
  - Every logical inference verified by trusted kernel
  - Proof calculus can be embedded in the proof assistant (and proven correct)
  - Proof steps can be replayed within the proof assistant
  - Verified proof checkers can be extracted from proof assistant
- Confidence in results is decoupled from the solver's implementation

Applications of SMT Proofs

- Strong correctness guarantees
  - High-quality proofs can be used to facilitate automated compliance
- Integration with other systems
  - Automation in interactive theorem proving
  - External proof cecking can identify bugs in proof rules
- Valuable for debugging
- Formalization of proof rules improves code base
  - Uncovers existing issues
  - Forces modular and clean code design
  - Improves tool robustness
- A rich source of data for various purposes (e.g., interpolation, profiling, machine learning)

SMT solving

- A cooperation of propositional reasoning and theory-specific reasoning
- Employs a SAT solver to perform propositional reasoning
- Employs a combination of procedures for theory reasoning
  - Equality and uninterpreted functions (Congruence Closure)
  - Linear/non-linear integer/real arithmetic (Simplex, Linearization, CAD)
  - Bit-vectors, Floats (Bit-blasting)
  - Combination of theories (Nelson-Oppen)
  - ...
- Decidability depends on the theories being used
  - E.g. strings, non-linear integer arithmetic are incomplete
  - Problems involving axioms (user-defined theories) are at best semi-decidable

Boolean Satisfiability (SAT)

Propositional formulas in CNF:

$$\begin{array}{lll} C & ::= & p \mid \neg p \mid C \vee C \\ \varphi & ::= & C \mid \varphi \wedge \varphi \end{array}$$

Given a formula φ in propositional logic, finding an assignment $\mathcal{M}$ mapping every proposition φ to $\{\top, \bot\}$ such that $\mathcal{M}(\varphi) = \top$ (or $\mathcal{M} \models \varphi$).

Boolean Satisfiability (SAT)

Propositional formulas in CNF:

$$\begin{array}{lcl} C & ::= & p \mid \neg p \mid C \vee C \\ \varphi & ::= & C \mid \varphi \wedge \varphi \end{array}$$

Given a formula φ in propositional logic, finding an assignment $\mathcal{M}$ mapping every proposition φ to $\{\top, \bot\}$ such that $\mathcal{M}(\varphi) = \top$ (or $\mathcal{M} \models \varphi$).

Example

Is $\varphi = (p \vee \neg q) \wedge (\neg r \vee \neg p) \wedge q$ satisfiable?

Boolean Satisfiability (SAT)

Propositional formulas in CNF:

$$\begin{array}{lll} C & ::= & p \mid \neg p \mid C \vee C \\ \varphi & ::= & C \mid \varphi \wedge \varphi \end{array}$$

Given a formula φ in propositional logic, finding an assignment $\mathcal{M}$ mapping every proposition φ to $\{\top, \bot\}$ such that $\mathcal{M}(\varphi) = \top$ (or $\mathcal{M} \models \varphi$).

Example

Is $\varphi = (p \vee \neg q) \wedge (\neg r \vee \neg p) \wedge q$ satisfiable? **Yes**

$$\mathcal{M}(p) = \top, \mathcal{M}(q) = \top, \mathcal{M}(r) = \bot \quad \Rightarrow \quad \mathcal{M}(\varphi) = \top$$

Boolean Satisfiability (SAT)

Propositional formulas in CNF:

$$\begin{array}{lcl} C & ::= & p \mid \neg p \mid C \vee C \\ \varphi & ::= & C \mid \varphi \wedge \varphi \end{array}$$

Given a formula φ in propositional logic, finding an assignment $\mathcal{M}$ mapping every proposition φ to $\{\top, \bot\}$ such that $\mathcal{M}(\varphi) = \top$ (or $\mathcal{M} \models \varphi$).

Example

Is $\varphi = (p \vee \neg q) \wedge (\neg r \vee \neg p) \wedge q \wedge (r \vee \neg q)$ satisfiable?

Boolean Satisfiability (SAT)

Propositional formulas in CNF:

$$\begin{array}{lcl} C & ::= & p \mid \neg p \mid C \vee C \\ \varphi & ::= & C \mid \varphi \wedge \varphi \end{array}$$

Given a formula φ in propositional logic, finding an assignment $\mathcal{M}$ mapping every proposition φ to $\{\top, \bot\}$ such that $\mathcal{M}(\varphi) = \top$ (or $\mathcal{M} \models \varphi$).

Example

Is $\varphi = (p \vee \neg q) \wedge (\neg r \vee \neg p) \wedge q \wedge (r \vee \neg q)$ satisfiable? **No**

No combination of valuations for these propositions such that φ is $\top$.

Boolean Satisfiability (SAT)

Unsatisfiability proof of $(p \vee \neg q) \wedge (\neg r \vee \neg p) \wedge q \wedge (r \vee \neg q)$:

$$\dfrac{\dfrac{\dfrac{r \vee \neg q \qquad q}{r}\,\textsc{Res} \qquad \neg r \vee \neg p}{\neg p}\,\textsc{Res} \qquad \dfrac{p \vee \neg q \qquad q}{p}\,\textsc{Res}}{\bot}\,\textsc{Res}$$

Satisfiability Modulo Theories (SMT)

First-order formulas in CNF:

$$
\begin{array}{lcl}
t & ::= & x \mid f(t, \ldots, t) \\
\varphi & ::= & p(t, \ldots, t) \mid \neg\varphi \mid \varphi \vee \varphi \mid \forall x_1 \ldots x_n.\ \varphi
\end{array}
$$

Given a formula φ in FOL and background theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$, finding a model $\mathcal{M}$ giving an *interpretation* to all terms and predicates such that $\mathcal{M} \models_{\mathcal{T}_1,\ldots,\mathcal{T}_n} \varphi$

Example

Is φ satisfiable modulo *equality* and *arithmetic*?

Satisfiability Modulo Theories (SMT)

First-order formulas in CNF:

$$\begin{array}{rcl} t & ::= & x \mid f(t, \ldots, t) \\ \varphi & ::= & p(t, \ldots, t) \mid \neg\varphi \mid \varphi \vee \varphi \mid \forall x_1 \ldots x_n.\ \varphi \end{array}$$

Given a formula φ in FOL and background theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$, finding a model $\mathcal{M}$ giving an *interpretation* to all terms and predicates such that $\mathcal{M} \models_{\mathcal{T}_1,\ldots,\mathcal{T}_n} \varphi$

Example

Is φ satisfiable modulo *equality* and *arithmetic*?

$$\varphi \quad = \quad (x_1 \geq 0) \wedge (x_1 < 1) \quad \wedge \quad (f(x_1) \not\approx f(0)) \quad \vee \quad x_2 + x_1 > x_2 + 1)$$

Satisfiability Modulo Theories (SMT)

First-order formulas in CNF:

$$\begin{array}{rcl} t & ::= & x \mid f(t, \ldots, t) \\ \varphi & ::= & p(t, \ldots, t) \mid \neg\varphi \mid \varphi \vee \varphi \mid \forall x_1 \ldots x_n.\ \varphi \end{array}$$

Given a formula φ in FOL and background theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$, finding a model $\mathcal{M}$ giving an *interpretation* to all terms and predicates such that $\mathcal{M} \models_{\mathcal{T}_1,\ldots,\mathcal{T}_n} \varphi$

Example

Is φ satisfiable modulo *equality* and *arithmetic*?

$$\varphi \quad = \quad \underbrace{(x_1 \geq 0) \wedge (x_1 < 1)}_{\text{LIA}} \quad \wedge \quad \underbrace{(f(x_1) \not\approx f(0)}_{\text{EUF}} \quad \vee \quad \underbrace{x_2 + x_1 > x_2 + 1)}_{\text{LIA}}$$

Satisfiability Modulo Theories (SMT)

First-order formulas in CNF:

$$\begin{array}{lcl} t & ::= & x \mid f(t, \ldots, t) \\ \varphi & ::= & p(t, \ldots, t) \mid \neg\varphi \mid \varphi \vee \varphi \mid \forall x_1 \ldots x_n.\ \varphi \end{array}$$

Given a formula φ in FOL and background theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$, finding a model $\mathcal{M}$ giving an *interpretation* to all terms and predicates such that $\mathcal{M} \models_{\mathcal{T}_1,\ldots,\mathcal{T}_n} \varphi$

Example

Is φ satisfiable modulo *equality* and *arithmetic*?

$$\varphi \quad = \quad \underbrace{(x_1 \geq 0) \wedge (x_1 < 1)}_{\text{LIA}} \quad \wedge \quad \underbrace{(f(x_1) \not\simeq f(0)}_{\text{EUF}} \quad \vee \quad \underbrace{x_2 + x_1 > x_2 + 1)}_{\text{LIA}}$$

$$\varphi \quad \models_{\text{LIA}} \quad x_1 \simeq 0$$

Satisfiability Modulo Theories (SMT)

First-order formulas in CNF:

$$\begin{array}{rcl} t & ::= & x \mid f(t, \ldots, t) \\ \varphi & ::= & p(t, \ldots, t) \mid \neg\varphi \mid \varphi \vee \varphi \mid \forall x_1 \ldots x_n.\ \varphi \end{array}$$

Given a formula φ in FOL and background theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$, finding a model $\mathcal{M}$ giving an *interpretation* to all terms and predicates such that $\mathcal{M} \models_{\mathcal{T}_1,\ldots,\mathcal{T}_n} \varphi$

Example

Is φ satisfiable modulo *equality* and *arithmetic*?

$$\varphi \quad = \quad \underbrace{(x_1 \geq 0) \wedge (x_1 < 1)}_{\text{LIA}} \quad \wedge \quad (\underbrace{f(x_1) \not\simeq f(0)}_{\text{EUF}} \quad \vee \quad \underbrace{x_2 + x_1 > x_2 + 1)}_{\text{LIA}}$$

$$\begin{array}{rcl} \varphi & \models_{\text{LIA}} & x_1 \simeq 0 \\ x_1 \simeq 0 & \models_{\text{EUF}} & f(x_1) \simeq f(0) \end{array}$$

Satisfiability Modulo Theories (SMT)

First-order formulas in CNF:

$$\begin{array}{rcl} t & ::= & x \mid f(t, \ldots, t) \\ \varphi & ::= & p(t, \ldots, t) \mid \neg\varphi \mid \varphi \vee \varphi \mid \forall x_1 \ldots x_n.\ \varphi \end{array}$$

Given a formula φ in FOL and background theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$, finding a model $\mathcal{M}$ giving an *interpretation* to all terms and predicates such that $\mathcal{M} \models_{\mathcal{T}_1,\ldots,\mathcal{T}_n} \varphi$

Example

Is φ satisfiable modulo *equality* and *arithmetic*?

$$\varphi \quad = \quad \underbrace{(x_1 \geq 0) \wedge (x_1 < 1)}_{\text{LIA}} \quad \wedge \quad (\underbrace{f(x_1) \not\simeq f(0)}_{\text{EUF}} \quad \vee \quad \underbrace{x_2 + x_1 > x_2 + 1}_{\text{LIA}})$$

$$\begin{array}{rcl} \varphi & \models_{\text{LIA}} & x_1 \simeq 0 \\ x_1 \simeq 0 & \models_{\text{EUF}} & f(x_1) \simeq f(0) \\ x_1 \simeq 0 & \models_{\text{LIA}} & x_2 + x_1 \not> x_2 + 1 \end{array}$$

Satisfiability Modulo Theories (SMT)

First-order formulas in CNF:

$$\begin{array}{rcl} t & ::= & x \mid f(t, \ldots, t) \\ \varphi & ::= & p(t, \ldots, t) \mid \neg\varphi \mid \varphi \vee \varphi \mid \forall x_1 \ldots x_n.\ \varphi \end{array}$$

Given a formula φ in FOL and background theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$, finding a model $\mathcal{M}$ giving an *interpretation* to all terms and predicates such that $\mathcal{M} \models_{\mathcal{T}_1,\ldots,\mathcal{T}_n} \varphi$

Example

Is φ satisfiable modulo *equality* and *arithmetic*?

$$\varphi \quad = \quad \underbrace{(x_1 \geq 0) \wedge (x_1 < 1)}_{\text{LIA}} \quad \wedge \quad \underbrace{(f(x_1) \not\simeq f(0)}_{\text{EUF}} \quad \vee \quad \underbrace{x_2 + x_1 > x_2 + 1)}_{\text{LIA}}$$

$$\begin{array}{rcl} \varphi & \models_{\text{LIA}} & x_1 \simeq 0 \\ x_1 \simeq 0 & \models_{\text{EUF}} & f(x_1) \simeq f(0) \\ x_1 \simeq 0 & \models_{\text{LIA}} & x_2 + x_1 \not> x_2 + 1 \end{array}$$

Therefore $\models_{\text{EUF} \cup \text{LIA}} \neg\varphi$

Satisfiability Modulo Theories (SMT)

Unsatisfiability proof of $(x_1 \geq 0) \wedge (x_1 < 1) \wedge (f(x_1) \not\simeq f(0) \vee x_2 + x_1 > x_2 + 1)$:

Let Π_1:

$$\frac{x_1 \geq 0 \qquad x_1 < 1}{x_1 \simeq 0}\ \text{LIA}$$

Then the final proof is:

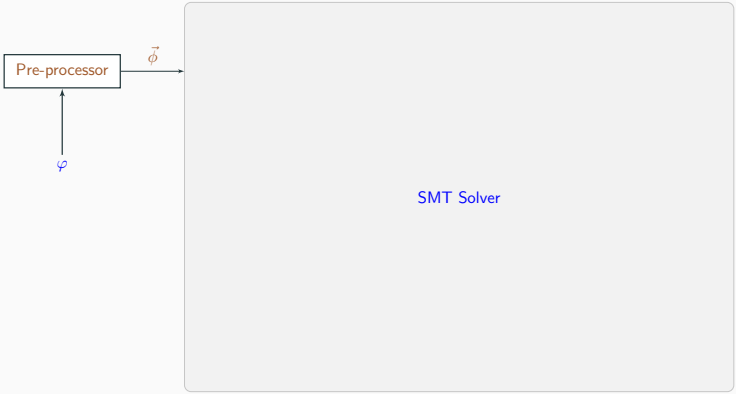
$$\dfrac{\dfrac{\dfrac{\Pi_1}{x_1 \simeq 0}}{f(x_1) \simeq f(0))}\ \text{EUF} \qquad f(x_1) \not\simeq f(0) \vee x_2 + x_1 > x_2 + 1}{x_2 + x_1 > x_2 + 1}\ \text{Res} \qquad \dfrac{\dfrac{\Pi_1}{x_1 \simeq 0}}{x_2 + x_1 \not> x_2 + 1}\ \text{LIA}}{\bot}\ \text{Res}$$

Proof module architecture for CDCL($\mathcal{T}$)

Proof module architecture for CDCL($\mathcal{T}$)

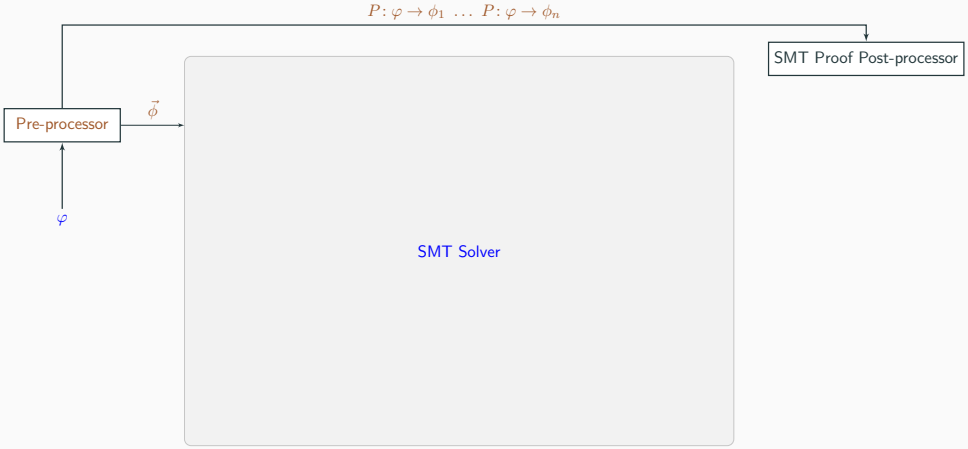
Proof module architecture for CDCL($\mathcal{T}$)

Proof module architecture for CDCL($\mathcal{T}$)



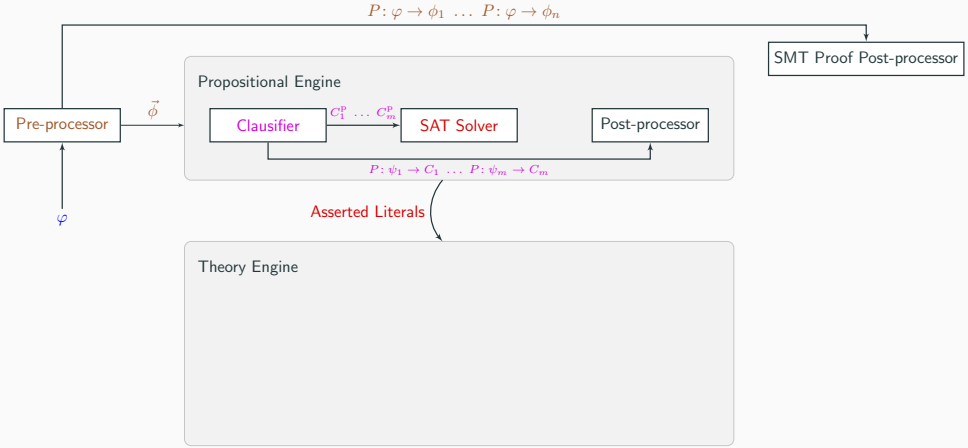
- Preprocessor simplifies formula globally:
  $x \simeq t \wedge F[x] \;\mapsto\; F[t] \qquad F[(ite\ P\ t_1\ t_2)] \;\mapsto\; F[t'] \wedge P \to t' \simeq t_1 \wedge \neg P \to t' \simeq t_2$

Proof module architecture for CDCL($\mathcal{T}$)



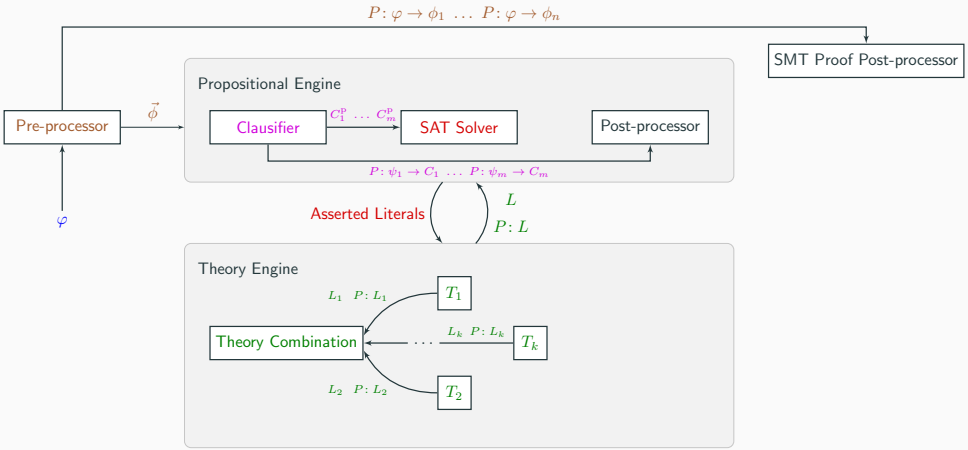
- Preprocessor simplifies formula globally:
  $x \simeq t \wedge F[x] \ \mapsto \ F[t] \qquad F[(ite\ P\ t_1\ t_2)] \ \mapsto \ F[t'] \wedge P \to t' \simeq t_1 \wedge \neg P \to t' \simeq t_2$

Proof module architecture for CDCL($\mathcal{T}$)



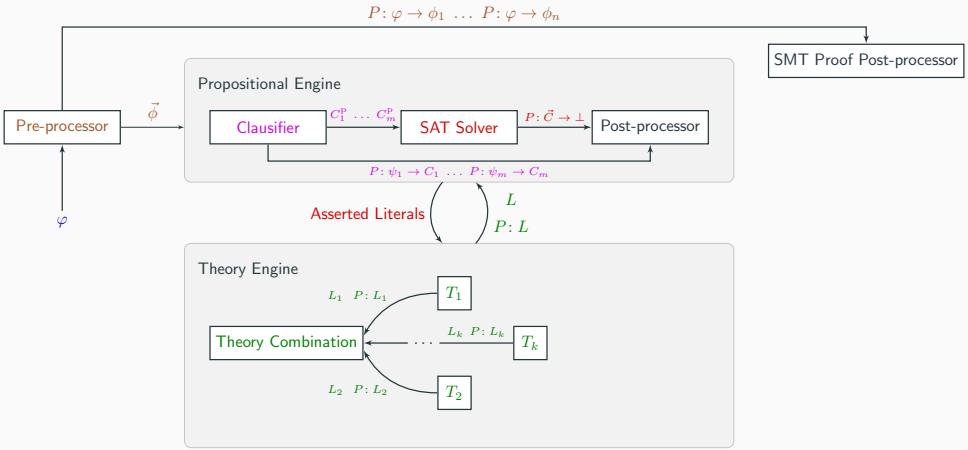
- Clausifier converts to Conjunctive Normal Form (CNF)
  SAT solver asserts literals that must hold based on Boolean abstraction

Proof module architecture for CDCL($\mathcal{T}$)



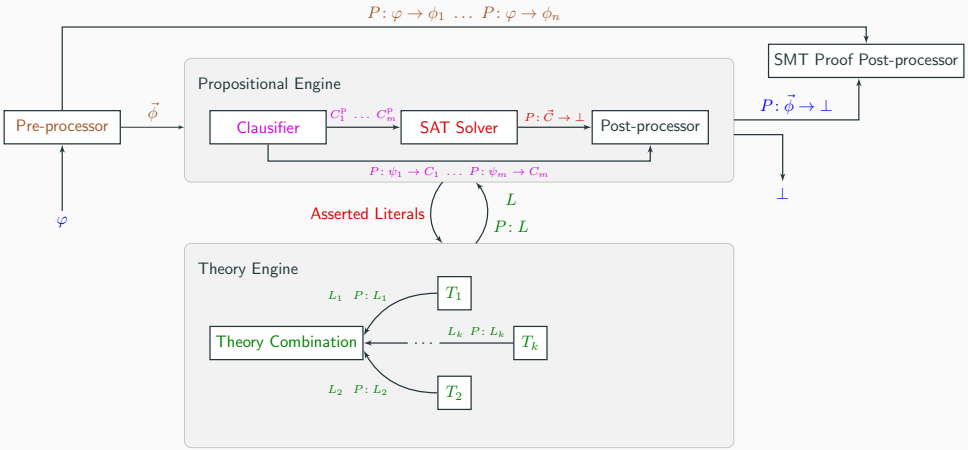
- Theory solvers check consistency in the theory

Proof module architecture for CDCL($\mathcal{T}$)



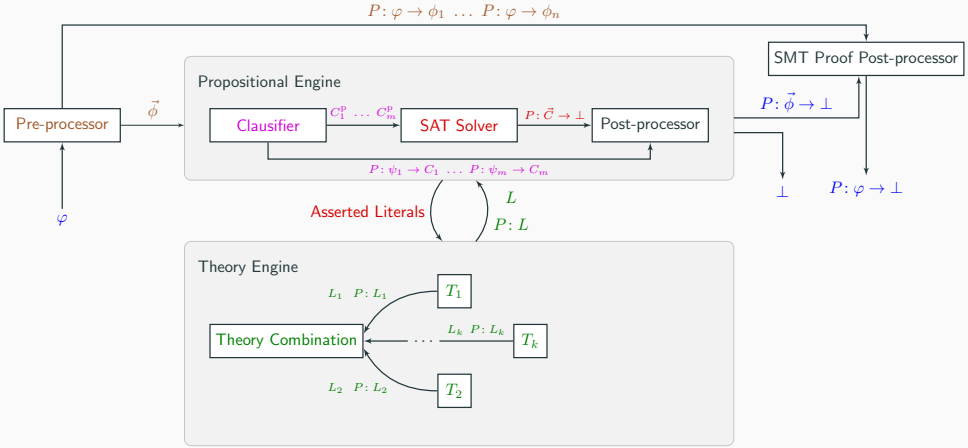
- Theory solvers check consistency in the theory

Proof module architecture for CDCL($\mathcal{T}$)



- Theory solvers check consistency in the theory

Proof module architecture for CDCL($\mathcal{T}$)



- Theory solvers check consistency in the theory

Demo!

Consider the following unsatisfiable SMT problem:

$$a \simeq b \wedge c \simeq d \wedge (p_1 \wedge \top) \wedge ((\neg p_1) \vee (p_2 \wedge p_3)) \wedge (\neg p_3 \vee (f(a,\, c) \not\simeq f(b,\, d)))$$

which in SMT-LIB is:

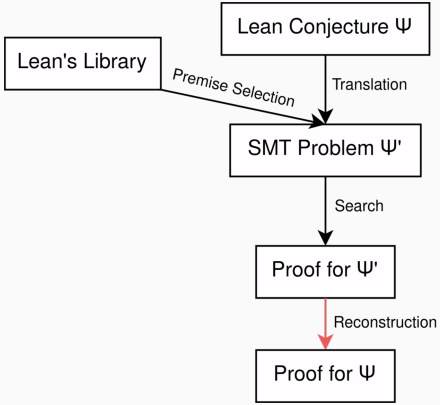
```
(set-logic QF_UF)
(declare-sort U 0)
(declare-const p1 Bool)
(declare-const p2 Bool)
(declare-const p3 Bool)
(declare-const a U)
(declare-const b U)
(declare-const c U)
(declare-const d U)
(declare-fun f (U U) U)

(assert (= a b))
(assert (= c d))
(assert (and p1 true))
(assert (or (not p1) (and p2 p3)))
(assert (or (not p3) (not (= (f a c) (f b d)))))
(check-sat)
```

Lean

- Lean is both a functional programming language and a proof assistant
- Its features include algebraic datatypes, pattern matching, polymorphism, typeclasses and a robust macro system
- Growing community, Mathlib, high enthusiasm from mathematicians
- It can benefit from a hammer

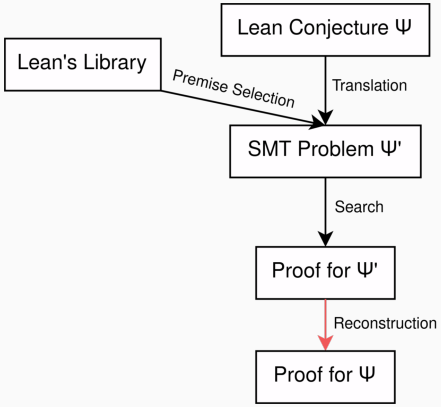
General workflow for a proof-reconstruction based-hammer

- Translation Module
- Premise Selection Module
- Proof Reconstruction Module
  - Certified vs Certifying
- Successful proof-reconstruction-based hammers in other proof assistants:
  - SMTCoq in Coq
  - Sledgehammer (smt) in Isabelle/HOL



General workflow for a proof-reconstruction based-hammer

- Translation Module
- Premise Selection Module
- **Proof Reconstruction Module**
  - Certified vs **Certifying**
- Successful proof-reconstruction-based hammers in other proof assistants:
  - SMTCoq in Coq
  - Sledgehammer (smt) in Isabelle/HOL



The certifying approach

- Lean has a rich framework for implementing your own tactics based on *metaprogramming*
  - The hammer itself and in particular the proof reconstruction are implemented tactics
- Tactics inspect the goal and the set of hypothesis via the internal representation used by Lean's compiler
- Based on this, they generate a proof that will then be checked by the kernel (every time)
- Proofs do not rely on normalization of terms, so they can potentially be faster
- It can be significantly easier to implement a tactic for a rule than to prove its corresponding theorem
  - the tactic operates on instances of the rule, while the theorem must consider the most general case
- Easier to update the tactic than to reprove the theorem

Proof reconstruction for SMT proofs in Lean

- We have build a library of tactics (and of underlying theorems) for the internal proof calculus of the cvc5 SMT solver
- This library covers:
  - Booleans, Linear Arithmetic and Equality and Uninterpreted Functions
  - Quantifiers (instantiation and Skolemization)
  - Bit-vectors (bit-blasting)
- Demo!

Towards an SMT-based hammer for Lean

```
import Smt

variable {G : Type} [Nonempty G] (op : G -> G -> G) (inv : G -> G) (e : G)

axiom assoc : ∀ a b c, op a (op b c) = op (op a b) c
axiom inverse : ∀ a, op (inv a) a = e
axiom ident : ∀ a, op e a = a

theorem inverse' : ∀ a, op a (inv a) = e := by
  smt [assoc op, inverse op inv e, ident op e]

theorem identity' : ∀ a, op a e = a := by
  smt [assoc op, inverse op inv e, ident op e, inverse' op inv e]

theorem unique_identity : ∀ e', (∀ z, op e' z = z) -> e' = e := by
  smt [assoc op, inverse op inv e, ident op e]
```

- Example from SMTCoq's website

Non-linear Arithmetic

Two main strategies implemented by state-of-the-art solvers

- Incremental linearization (cvc5, MathSAT)
  - Fast but incomplete
  - An abstraction-refinement loop between the linear solver and a validator to its candidate solutions
  - When a candidate is invalid, the abstraction is refined via a lemma
  - Heavily dependent on the lemma schemas used
- Variations of cylindrical algebraic decomposition (cvc5, z3, yices, SMT-RAT)
  - Complete but costly

Incremental linearization

Consider the satisfiability of $x \cdot y > 0 \wedge x > 1 \wedge y < 0$.

Incremental linearization

Consider the satisfiability of $x \cdot y > 0 \wedge x > 1 \wedge y < 0$.

- $x \cdot y$ is abstracted as a variable and the constraints are given to the linear solver

Incremental linearization

Consider the satisfiability of $x \cdot y > 0 \wedge x > 1 \wedge y < 0$.

- $x \cdot y$ is abstracted as a variable and the constraints are given to the linear solver
- A model is $x \mapsto 2$, $y \mapsto -1$, and $x \cdot y \mapsto 1$.

Incremental linearization

Consider the satisfiability of $x \cdot y > 0 \land x > 1 \land y < 0$.

- $x \cdot y$ is abstracted as a variable and the constraints are given to the linear solver
- A model is $x \mapsto 2$, $y \mapsto -1$, and $x \cdot y \mapsto 1$.
- Since it is not correct, a lemma is required to refine the abstraction.
- For example, $x > 0 \land y < 0 \to x \cdot y < 0$.

Proofs for incremental linearization

- Each lemma schema requires a proof rule.
- The multiplication property of the signal of its result depending on the signal of the arguments:

$$\frac{- \mid f_1 \dots f_k, m}{(f_1 \wedge \dots \wedge f_k) \rightarrow m \diamond 0}$$

- The solver must produce proofs with this rule when generating such a lemma
- Similarly, proof checkers must be able to understand and verify them

Proofs for incremental linearization

- Each lemma schema requires a proof rule.
- The multiplication property of the signal of its result depending on the signal of the arguments:

$$\frac{- \mid f_1 \dots f_k, m}{(f_1 \wedge \dots \wedge f_k) \rightarrow m \diamond 0}$$

- The solver must produce proofs with this rule when generating such a lemma
- Similarly, proof checkers must be able to understand and verify them
- Demo!

Proofs for incremental linearization

- We are currently working on making cvc5 fully proof-producing when using incremental linearization
- In parallel we are augmenting our work-in-progress integration between cvc5 and Lean to support them
- Proving the correctness of the previous rule and defining a reconstruction procedure for it requires 250 lines of Lean code
  - Not counting the dependencies of Lean's mathematical library, Mathlib.
- Multiple lemma schemas employed by cvc5 also for solving problems involving transcendental functions (such as trigonometric functions).

More current/future work

- Proof production and reconstruction for finite fields
  - Reasoning with Gröbner basis for solving problems with finite fields [OKTB23]
  - Polynomial calculus for reasoning with machine arithmetic [KBK20]

- Proofs from SMT solvers for CAD-based methods are a big challenge
  - Preliminar work in tracing information necessary for fine-grained proofs
  - Substantial work in mechanizing CAD theory in proof assistants
    - Specially in Isabelle/HOL and Coq
  - Necessary to determine how SMT solvers can best represent proofs to enable scalable proof checking

SMT Proof Production and Integration with the Lean Theorem Prover

Haniel Barbosa, Universidade Federal de Minas Gerais

UFMG

Dagstuhl Seminar 23471

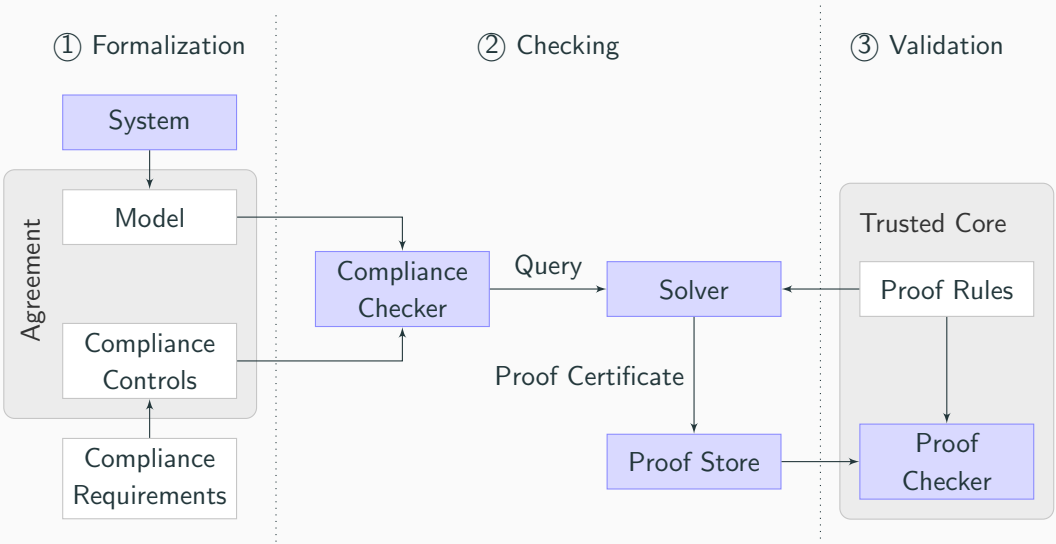
2023–11–20, Schloss Dagstuhl, DE

Having fun with π in Lean

$$\frac{- \mid l, u}{\pi \geq l \wedge \pi \leq u}$$

- cvc5's π lower bound: 3.1415926530
  cvc5's π upper bound: 3.1415926539
- Current tighter bounds on π in Lean's `mathlib`:
  theorem Real.pi_gt_3141592 : $3.141592 < \pi$
  theorem Real.pi_lt_3141593 : $\pi < 3.141593$
- These theorems are proved via a tactic based on Leibniz's series for π, which allows proving its value is within a certain threshold. The tactic is slow.
- Lean theory for bounds on π

Applications of SMT Proofs: Compliance



Anecdotes

- Internal proof checker is highly valuable for development
- Error localization for proofs is important
- Formalization of proof rules uncovers existing issues
- Performance issues
  - In a few cases, proof checker indicated it could prove something stronger
- Soundness issues
  - Cannot write proper proof checker if the reasoning of the solver is wrong
- Proofs are also valuable for debugging
  - Soundness bug reported, proofs used to easily isolate the incorrect rewrite
- Combination of approaches for proof generation

How some proofs look like

$$\frac{A \vee \ell \qquad B \vee \overline{\ell}}{A \vee B}$$

$$\frac{\varphi_1 \wedge \cdots \wedge \varphi_n}{\varphi_i}$$

$$\neg(a \simeq b) \vee f(a) \simeq f(b)$$
$$\neg(y > 1) \vee \neg(x < 1) \vee y > x$$

$$\neg(\varphi_1 \wedge \cdots \wedge \varphi_n) \vee \varphi_i$$

A particular challenge has been String solving

- Preprocessing
- Clausification
- SAT solving
- UF theory solver
- Linear Arithmetic solver
- Theory combination
- Quantifier instantiation
- Rewriting
  - Including complex string methods [RNBT19]
- Strings theory solver
  - Core calculus [LRT+14]
  - Extended function reductions [RWB+17]
  - Regular expression unfolding

References

[BD18] Lilian Burdy and David Déharbe. "Teaching an Old Dog New Tricks - The Drudges of the Interactive Prover in Atelier B". In:
Abstract State Machines, Alloy, B, TLA, VDM, and Z - 6th International Conference, ABZ 2018, Southampton, UK, June 5-8, 2
Ed. by Michael J. Butler, Alexander Raschke, Thai Son Hoang, et al. Vol. 10817. Lecture Notes in Computer Science. Springer, 2018, pp. 415–419.

[FBL18] Mathias Fleury, Jasmin Christian Blanchette, and Peter Lammich. "A verified SAT solver with watched literals using imperative HOL". In:
Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA,
Ed. by June Andronick and Amy P. Felty. ACM, 2018, pp. 158–171.

[Fle19] Mathias Fleury. "Optimizing a Verified SAT Solver". In:
NASA Formal Methods - 11th International Symposium, NFM 2019, Houston, TX, USA, May 7-9, 2019, Proceedings.
Ed. by Julia M. Badger and Kristin Yvonne Rozier. Vol. 11460. Lecture Notes in Computer Science. Springer, 2019, pp. 148–165.

References ii

[KBK20] Daniela Kaufmann, Armin Biere, and Manuel Kauers. "Incremental column-wise verification of arithmetic circuits using computer algebra". In: Formal Methods Syst. Des. 56.1 (2020), pp. 22–54.

[LRT+14] Tianyi Liang, Andrew Reynolds, Cesare Tinelli, et al. "A DPLL(T) Theory Solver for a Theory of Strings and Regular Expressions". In: Computer Aided Verification (CAV). Ed. by Armin Biere and Roderick Bloem. Vol. 8559. Lecture Notes in Computer Science. Springer, 2014, pp. 646–662.

[OKTB23] Alex Ozdemir, Gereon Kremer, Cesare Tinelli, et al. "Satisfiability Modulo Finite Fields". In: IACR Cryptol. ePrint Arch. (2023), p. 91.

[RNBT19] Andrew Reynolds, Andres Nötzli, Clark W. Barrett, et al. "High-Level Abstractions for Simplifying Extended String Constraints in SMT". In: Computer Aided Verification (CAV), Part II. Ed. by Isil Dillig and Serdar Tasiran. Vol. 11562. Lecture Notes in Computer Science. Springer, 2019, pp. 23–42.

[RWB+17] Andrew Reynolds, Maverick Woo, Clark Barrett, et al. "Scaling Up DPLL(T) String Solvers Using Context-Dependent Simplification". In: Computer Aided Verification (CAV). Ed. by Rupak Majumdar and Viktor Kuncak. Vol. 10427. Lecture Notes in Computer Science. Springer, 2017, pp. 453–474.