Automating ITPs Using ATPs

Arjun Viswanathan

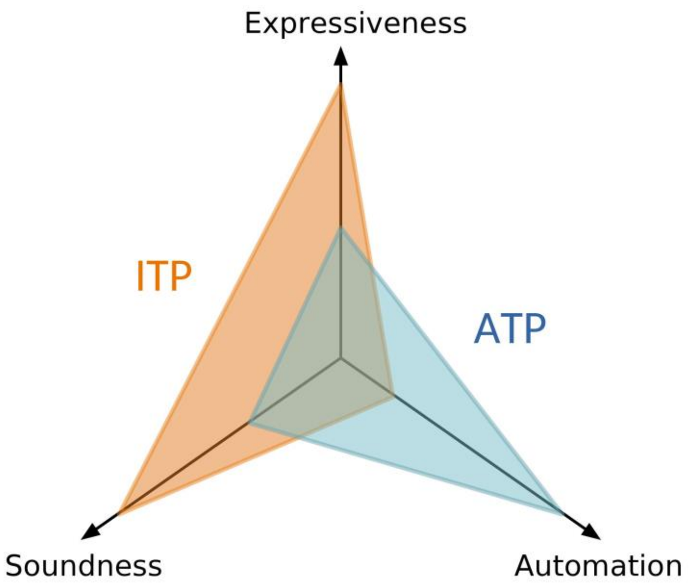
Advisor:

Cesare Tinelli

THE UNIVERSITY OF IOWA

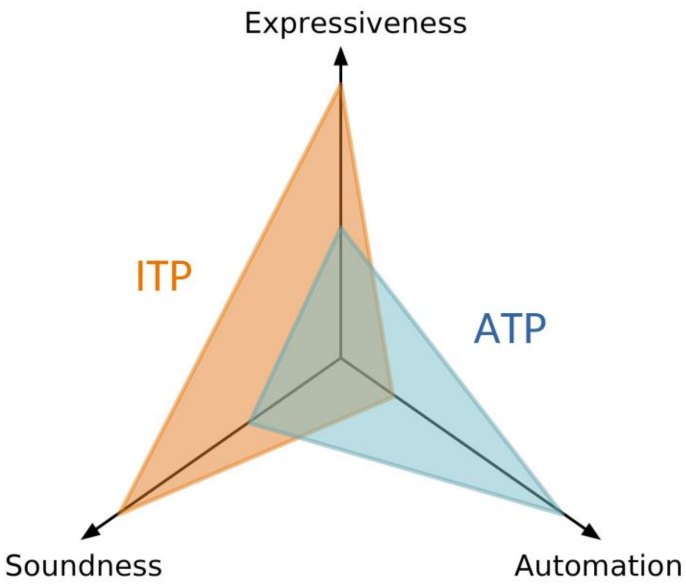
Introduction

- Theorem provers prove logic properties
- Used in hardware/software verification
- Theorem Provers:
  - Interactive (ITPs)
  - Automatic (ATPs)



ITPs	ATPs
Small proof kernel	Large code base
User interaction	Highly automated
Reliable proofs	Susceptible to bugs
Expressive logics (HOL)	Less expressive logic (FOL)
Coq, Isabelle/HOL, Agda, Lean	Superposition Provers: Vampire, E, SPASS; SMT Solvers: CVC4, Z3, VeriT

Motivation



- Combine ITPs and ATPS to:
  - automate proofs in ITPs
  - certify results of ATPs
- Autarkic approach – implement and prove correct ATP inside ITP
- Skeptical approach – ATP outputs checkable certificate
- Tools:
  - Hammers (e.g., Sledgehammer, CoqHammer)
  - Certified Checkers (e.g., SMTCoq)

Technical Preliminaries

Satisfiability

- Propositional/Boolean Satisfiability (SAT) : Satisfy F by mapping variables to True/False

$$\mathtt{p} \wedge \mathtt{q}$$
$$\mathtt{True} \wedge \mathtt{True} \quad ✓$$

- Satisfiability Modulo Theories (SMT) : Satisfy F by mapping variables to theory constants

$$(\mathrm{a} > 5) \wedge (\mathrm{a} < 0)$$
$$\mathtt{True} \wedge \mathtt{True} \quad ✗\,\mathrm{LIA}$$

Duality of Satisfiability and Validity

- F is valid iff ¬F is unsatisfiable

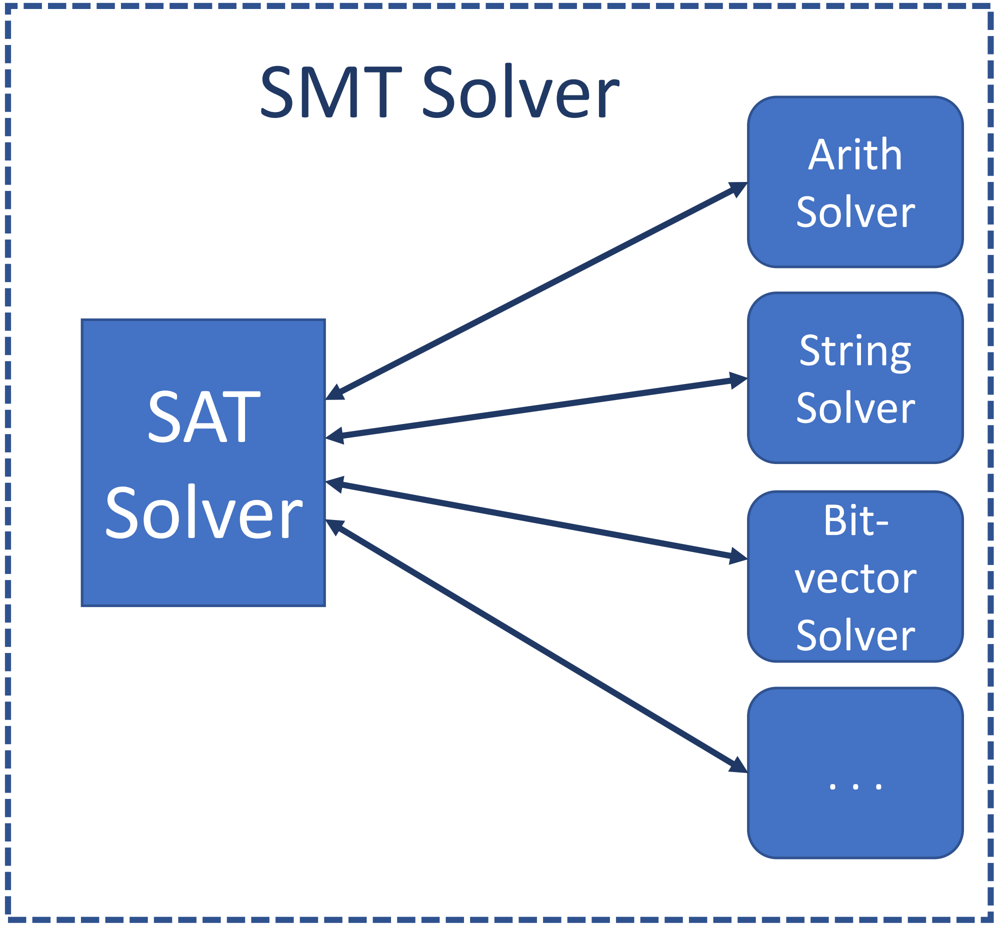
$$\mathbf{F} : \forall x_1, x_2, \ldots, x_n \ . \ H_1 \rightarrow H_2 \rightarrow \ldots \rightarrow H_m \rightarrow G$$

$$\neg\mathbf{F} : \neg(\forall x_1, x_2, \ldots, x_n \ . \ H_1 \rightarrow H_2 \rightarrow \ldots \rightarrow H_m \rightarrow G)$$
$$\neg\mathbf{F} : H_1 \wedge H_2 \wedge \ldots \wedge H_m \wedge \neg G$$

SMT Solvers

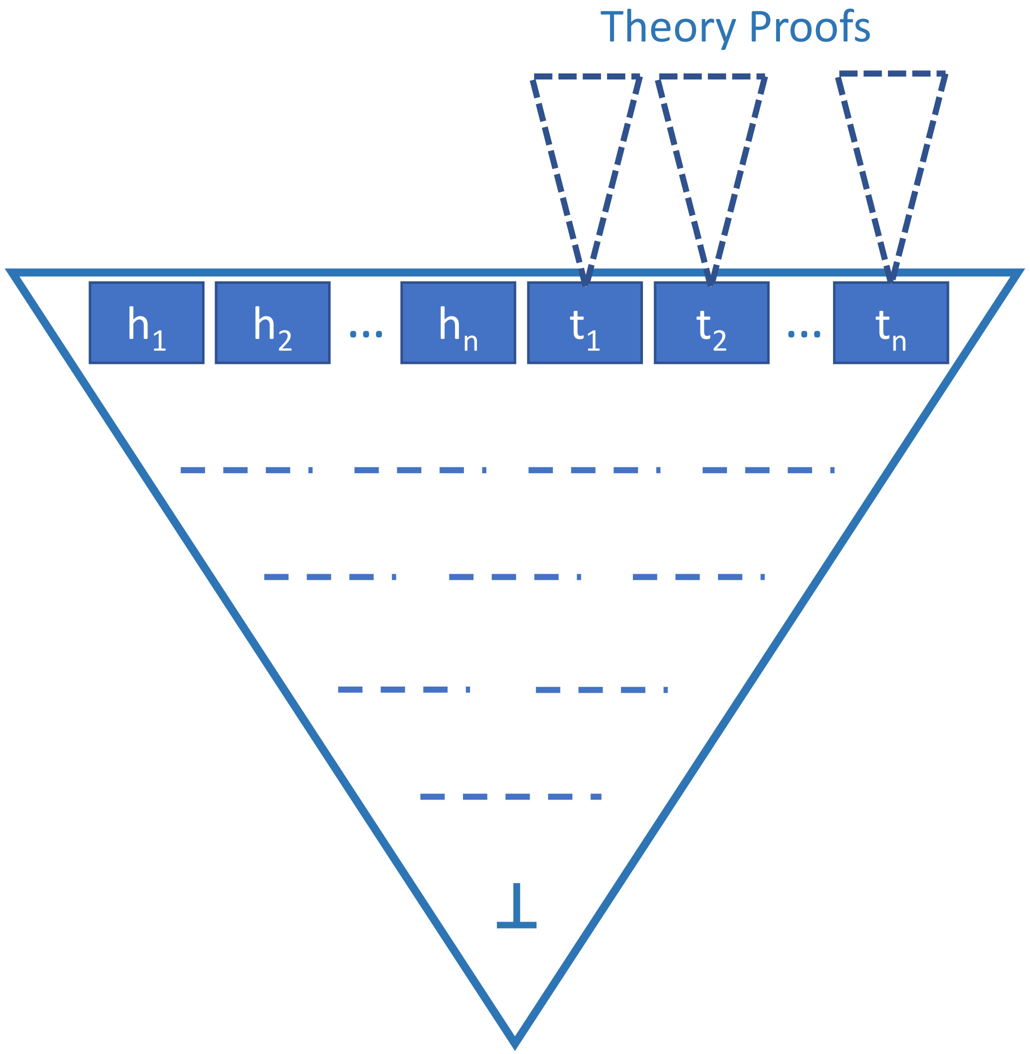
ATPs – SMT Solvers

- Is a given formula satisfiable or unsatisfiable?
- DPLL(T) architecture –
  abstract theory literals and use SAT solver
- SAT is NP-complete
- Theories and quantification may make solving undecidable



Proof Producing SMT Solvers

- Satisfiability – satisfying model
- Unsatisfiability – resolution proof tree
- Proof tree:
  - Input formulas – leaves
  - Theory lemmas – leaves
  - Empty clause – root
  - Node – rule applied to parents
  - Holes – unjustified simplifications



Proof Producing SMT Solvers

- Proof rules:
  - CNF Conversion
  - Resolution
  - Theory-specific
  - Quantifier
  - Rewrites

$$\texttt{CNF} : (x_1 \vee x_2 \vee \ldots) \wedge (y_1 \vee y_2 \vee \ldots) \wedge \ldots$$

$$\frac{x = y}{y = x}\ \texttt{symm}$$

$$\frac{P(c)}{\exists x.P(x)}\ \exists\texttt{intro}$$

$$x + 0 \mapsto x$$

$$\frac{\phi_1 \vee \ldots \vee \phi_n \vee \chi \quad \neg\chi \vee \psi_1 \vee \ldots \vee \psi_m}{\phi_1 \vee \ldots \vee \phi_n \vee \psi_1 \vee \ldots \vee \psi_m}\ \texttt{resolution} \qquad \frac{a \vee \neg b \quad b \vee c}{a \vee c}$$

$n, m \geq 0$

Interactive Theorem Provers

ITPs

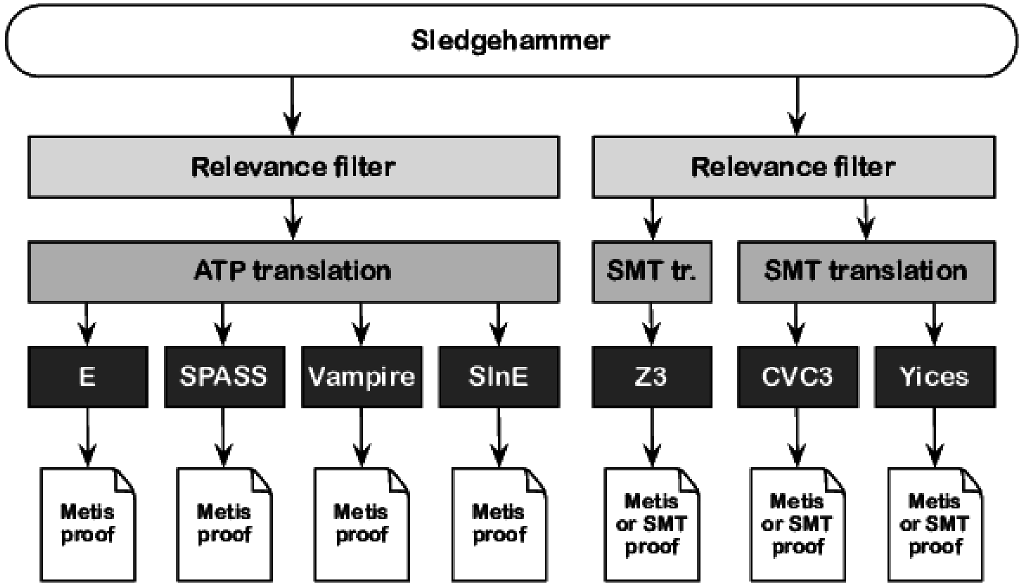
- Small, trustable proof kernel
- Human-machine collaboration
- *Tactics* based on proof rules

LCF-Based	Automath-Based
Theorem is an ADT	Theorem statements - types
ADT provides functions to create theorems using inference rules of logic	Proofs – programs inhabiting types
HOL Light, HOL4, Isabelle	Coq, Agda, Lean
Sledgehammer – Isabelle/HOL	SMTCoq - Coq

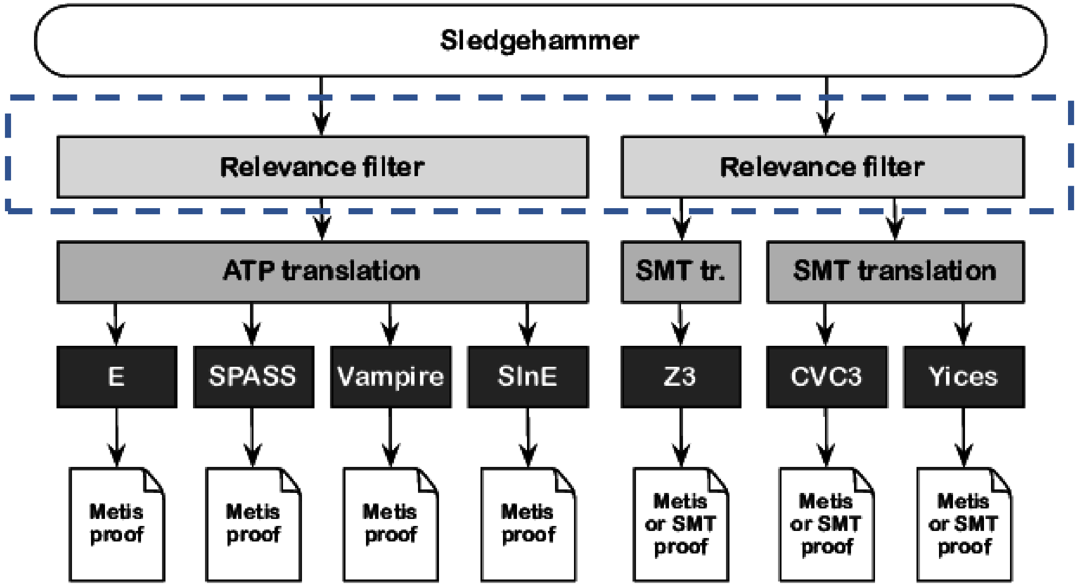
Sledgehammer

Sledgehammer

- Automate proving within ITP using ATP
  - Premise Selection
  - Translation
  - Proof Reconstruction
- Sledgehammer – Isabelle/HOL's hammer
- Metis – internal ATP
- We focus on Sledgehammer's SMT solver integration



Premise Selection

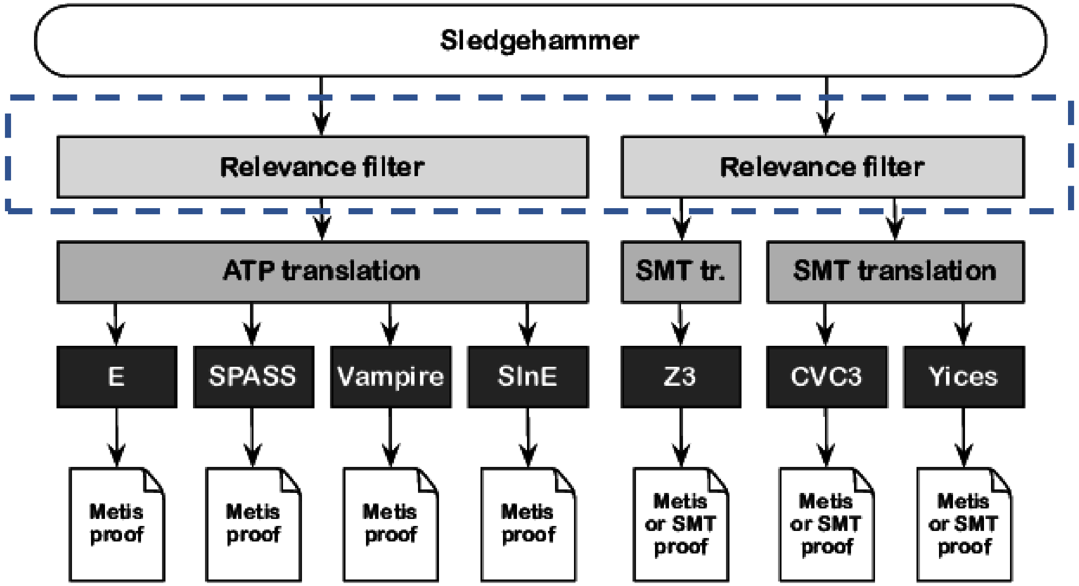


- Given ITP goal:

  $\texttt{F} : \forall x_1, x_2, ..., x_n \ . \ H_1 \to H_2 \to ... \to H_m \to G$

  G might depend on other facts
- ITPs have large libraries of proven facts
- Premise selection – filter out relevant facts to send with F
  - Delegate to user
  - Syntactic selection
  - Semantic selection
  - Hybrid

Premise Selection



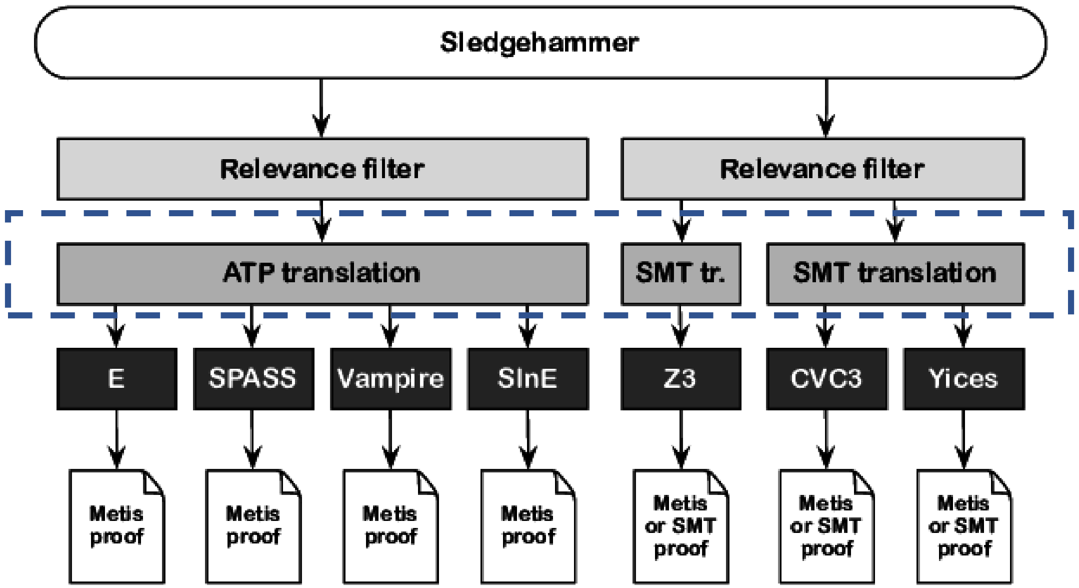
- Given ITP goal:

  $\mathtt{F} : \forall x_1, x_2, \ldots, x_n \ . \ H_1 \rightarrow H_2 \rightarrow \ldots \rightarrow H_m \rightarrow G$

  G might depend on other facts
- ITPs have large libraries of proven facts
- Premise selection – filter out relevant facts to send with G
  - Delegate to user
  - Syntactic selection – prioritize facts by common symbols with goal
  - Semantic selection
  - Hybrid

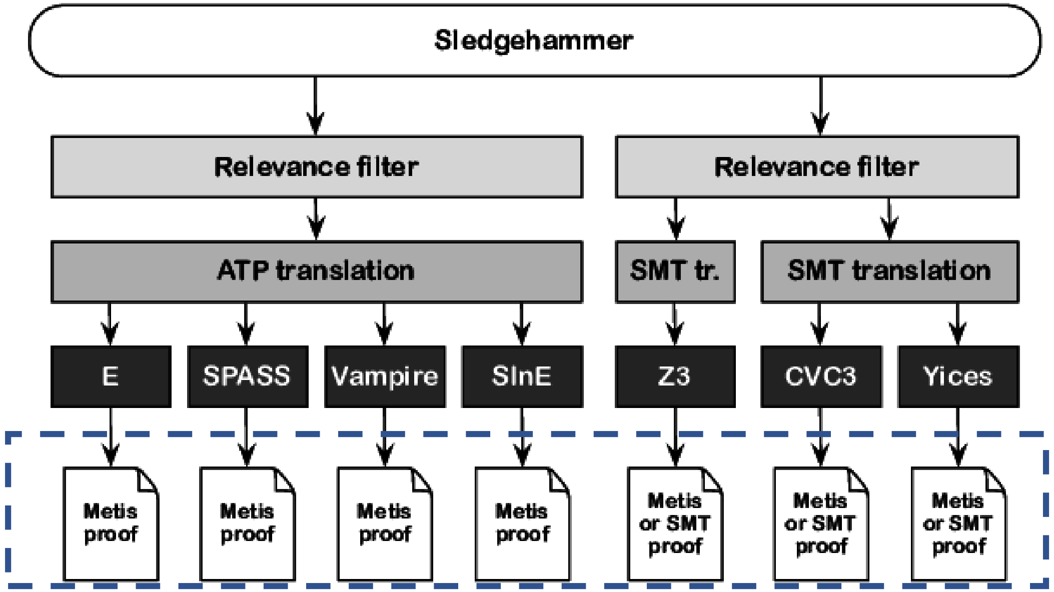
Translation

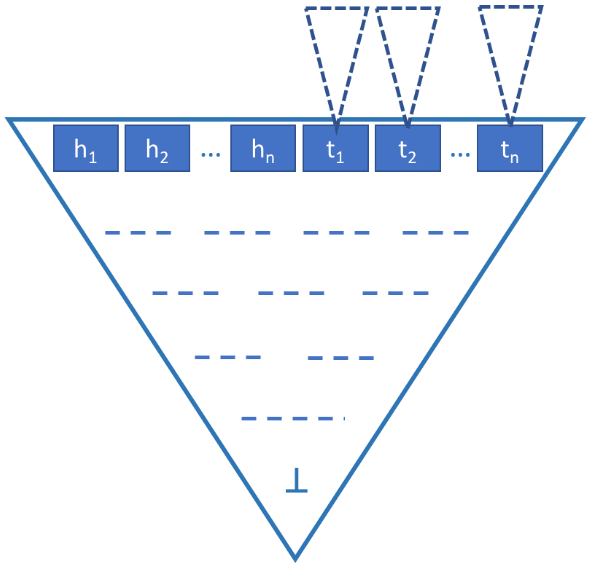
- ATP – typed/untyped FOL
- ITP – HOL
- FOL is a subset of HOL
- Translate non-FOL features of HOL such as anonymous functions, partial applications, etc.
- Map types in Isabelle to types in theories



Proof Reconstruction

- Sledgehammer with ATP
  - Trust as oracle – CVC4, Yices
  - Use as relevance filter – E, SPASS, Vampire
  - Reconstruct ATP proof – Z3
- Reconstruction
  - Inference-by-inference
  - Depth-first post-order
  - Reconstruction of a node implies a proven theorem
- Coarse-grained proofs necessitate proof search

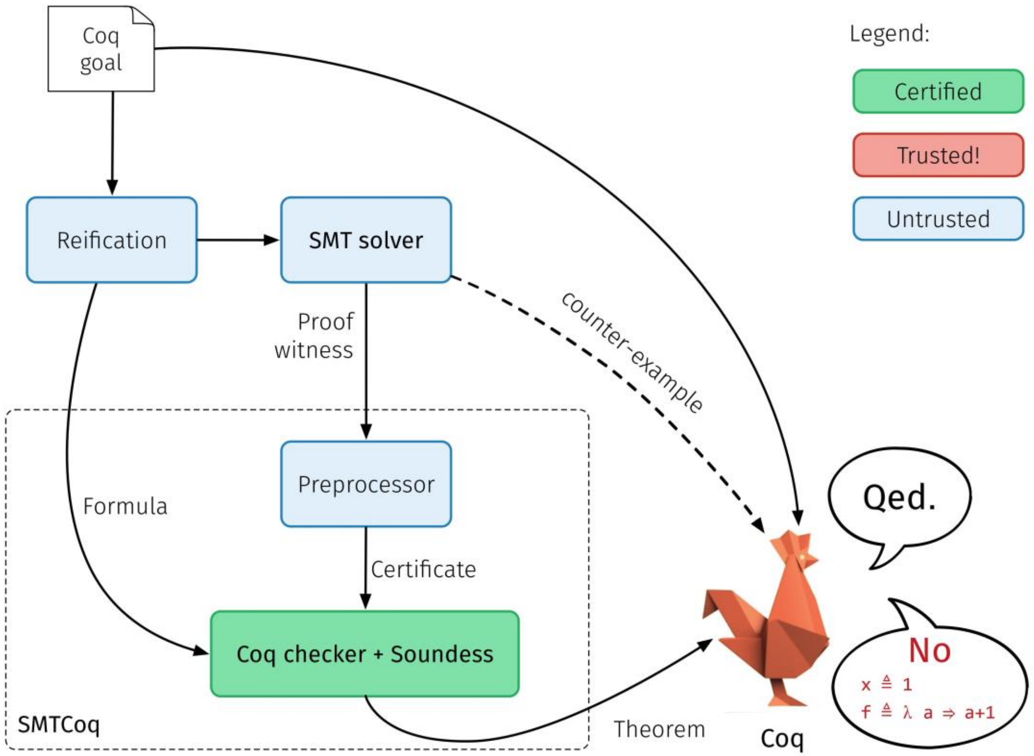




SMTCoq

SMTCoq

- Skeptical cooperation between Coq and SAT/SMT solvers
- Coq data structures represent SMT terms (*deep* embedding)
- Coq *Props* represent Coq theorems (*shallow* embedding)
- SMT proofs $\rightarrow$ Proof certificate
- Boolean decision procedure checks proof certificate by *computational reflection*



Computational Reflection

- Coq data structures represent SMT terms (*deep* embedding)
- Coq *Props* represent Coq theorems (*shallow* embedding)
- *Reflect* proofs from deep to shallow
- Boolean decision procedure checks
  deep terms $\leftrightarrow$ proof terms
- Correctness of decision procedure
- Reflection uses Coq's computational capabilities

Computational Reflection

- Reflection needs:
  - A Boolean decision procedure (*check*) that
    - takes a term *s : S* in the deep embedding,
    - a proof trace *t : T* from the ATP, and
    - checks that *t* justifies *s*
  - A proof of correctness of check (*reflection principle*)

    $$\texttt{check\_correct} : \forall (s : \mathrm{S})\ (t : \mathrm{T}), \texttt{check}\ s\ t = \texttt{true} \to \mathrm{P}\ s$$

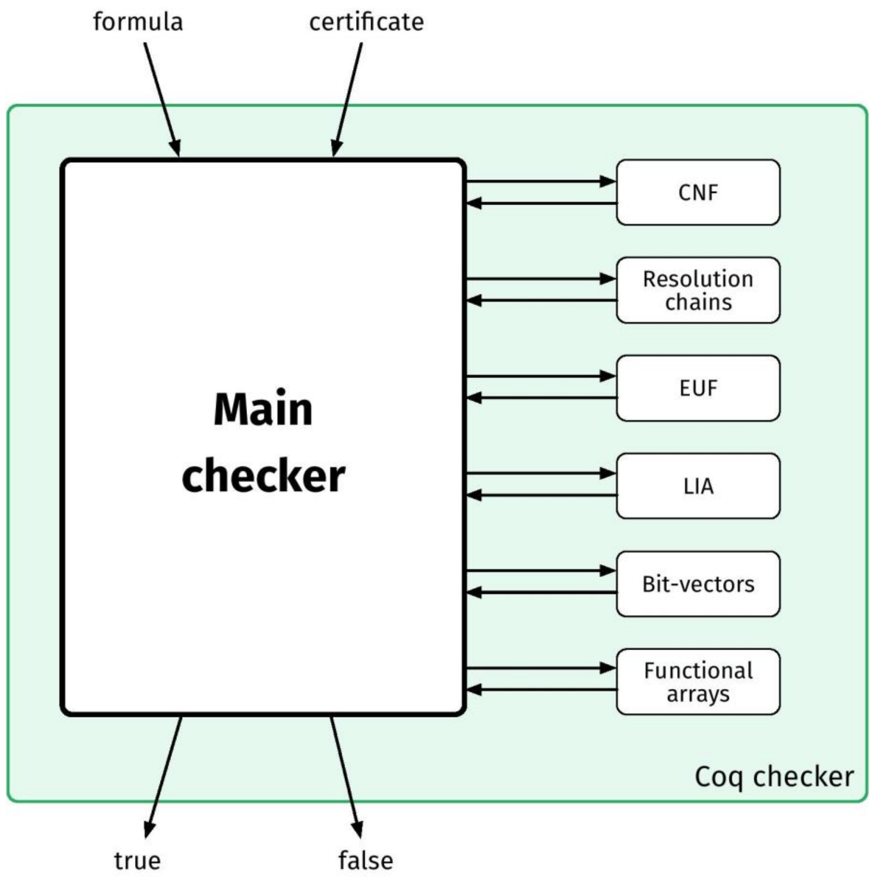
    p is a predicate on deep terms
- *check* is largely computational
- For a particular *s* and *t*, the proof of *P s* is:

  $$\texttt{check\_correct}\ s\ t\ (\texttt{refl\_equal}\ \_\ \texttt{true})) : \mathrm{P}\ s$$

- Reflection principle relates
  - computational behavior of *check*
  - propositional meaning

SMTCoq - Checker

- Divide:
  - Proofs $\longrightarrow$ steps
  - Main checker $\longrightarrow$ small checkers
- State : set of clauses
- Each step modifies state while maintaining unsatisfiability
- Main checker – is final state $\bot$?



Sledgehammer vs SMTCoq

Comparison

	Sledgehammer with Z3	SMTCoq

Comparison

	Sledgehammer with Z3	SMTCoq
Operation	• Trusts CVC3/4 and Yices as oracles • Uses superposition provers as relevance filters • Gets proof skeleton from Z3 and fills it using Metis	• Converts proofs to a certificate format • Uses reflection to reflect solver proofs

Comparison

	Sledgehammer with Z3	SMTCoq
Operation	• Trusts CVC3/4 and Yices as oracles • Uses superposition provers as relevance filters • Gets proof skeleton from Z3 and fills it using Metis	• Converts proofs to a certificate format • Uses computational reflection to reconstruct solver proofs
Logic	• Goals in FOL and a subset of HOL from Isabelle/HOL • Quantified FOL with EUF, LIA, BV	• Only FOL goals in Coq • Quantifier-free FOL with EUF, LIA, BV, Arrays

Comparison

	Sledgehammer with Z3	SMTCoq
Operation	• Trusts CVC3/4 and Yices as oracles • Uses superposition provers as relevance filters • Gets proof skeleton from Z3 and fills it using Metis	• Converts proofs to a certificate format • Uses computational reflection to reconstruct solver proofs
Logic	• Goals in FOL and a subset of HOL from Isabelle/HOL • Quantified FOL with EUF, LIA, BV	• Only FOL goals in Coq • Quantifier-free FOL with EUF, LIA, BV, Arrays
Extensibility	• Specific integration with Z3 • Trusts other solvers without proof reconstruction	• Integration with CVC4, VeriT, Zchaff, Glucose • Additional solvers can be added by adding a preprocessor

Comparison

	Sledgehammer with Z3	SMTCoq
Operation	• Trusts CVC3/4 and Yices as oracles • Uses superposition provers as relevance filters • Gets proof skeleton from Z3 and fills it using Metis	• Converts proofs to a certificate format • Uses computational reflection to reflect solver proofs
Logic	• Goals in FOL and a subset of HOL from Isabelle/HOL • Quantified FOL with EUF, LIA, BV	• Only FOL goals in Coq • Quantifier-free FOL with EUF, LIA, BV, Arrays
Extensibility	• Specific integration with Z3 • Trusts other solvers without proof reconstruction	• Integration with CVC4, VeriT, Zchaff, Glucose • Additional solvers can be added by adding a preprocessor
Premise Selection	• Uses various Sledgehammer premise selection techniques	• Doesn't consider facts outside a lemma

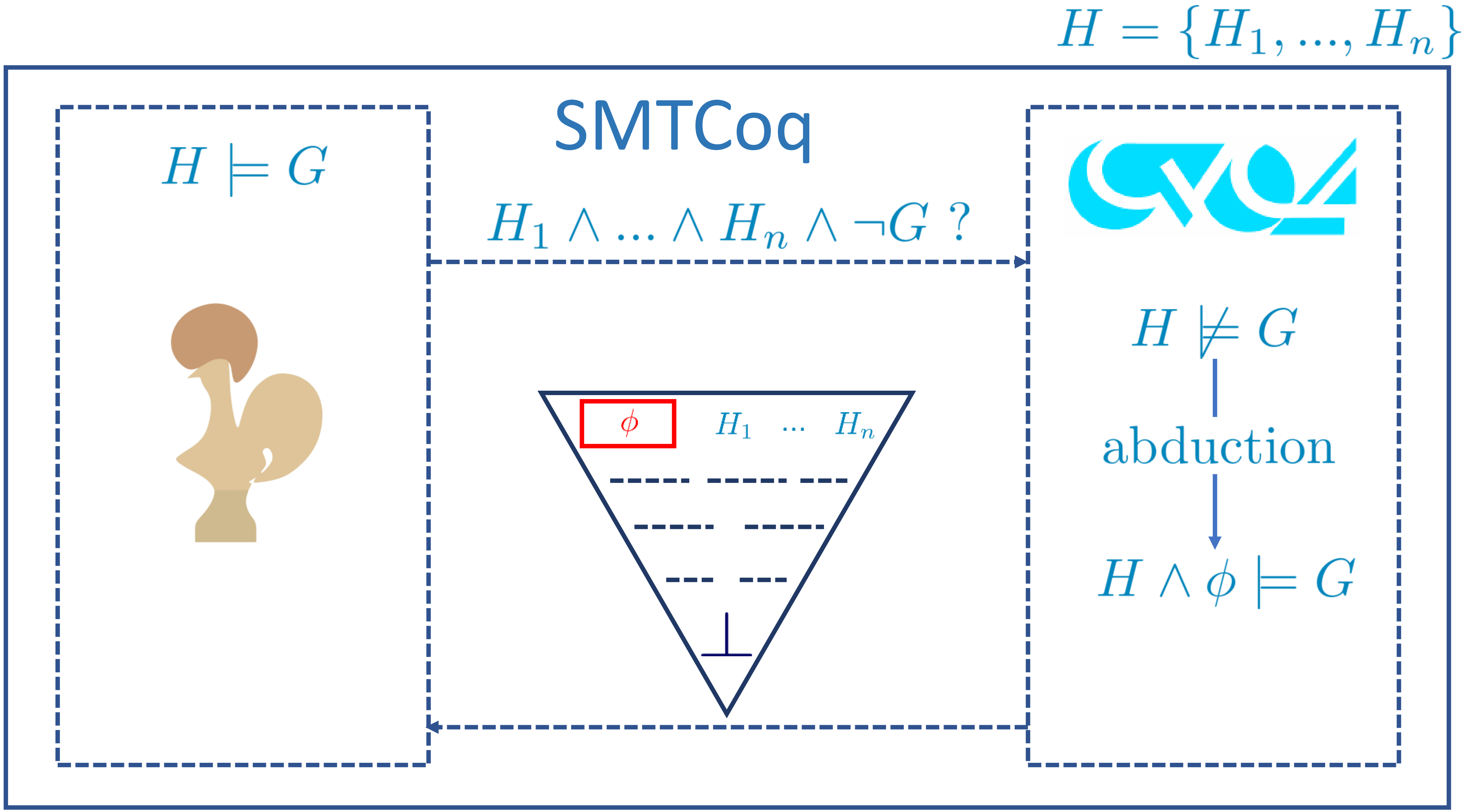
Future Work - Abduction

- Given a set of axioms A, goal G, the abduct (if it exists) is a formula ϕ s.t.

$$A \wedge \phi \models G$$

- Find formula ϕ that is consistent with the axioms and when added to them, allows the goal to be proven

Future Work - Abduction



Questions?

Questions?

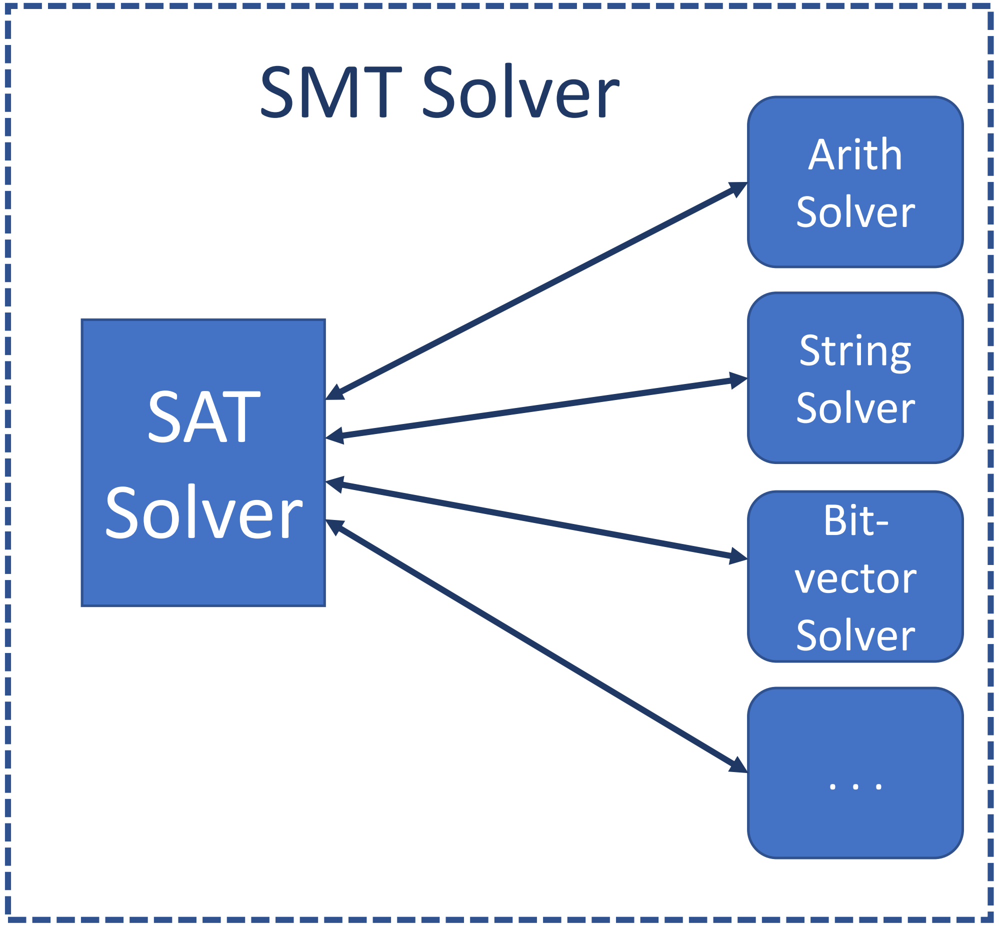
Questions?

DPLL(T) Architecture

$$\mathtt{F} : (\mathtt{a} > 5) \wedge (\mathtt{a} < 0)$$

ATPs – SMT Solvers

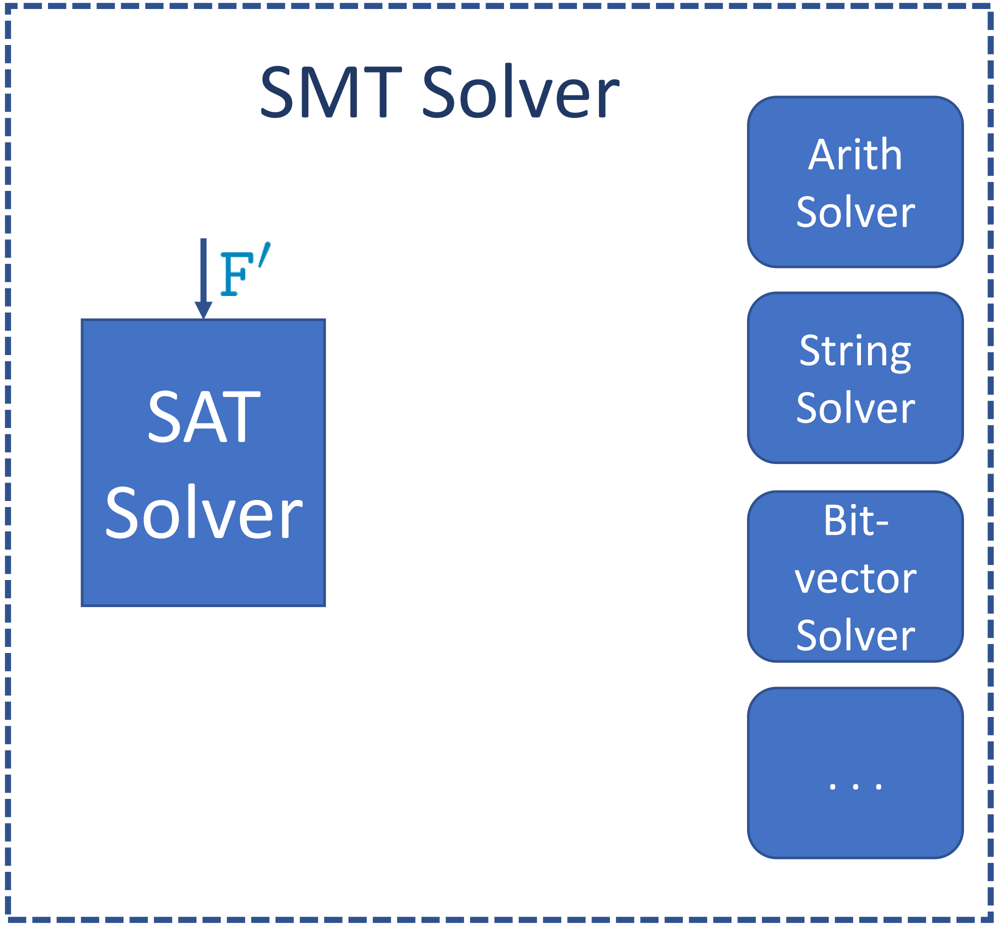
- Is a given formula satisfiable or unsatisfiable?
- DPLL(T) architecture –
  abstract theory literals and use SAT solver



ATPs – SMT Solvers

$$\mathtt{F} : (\mathtt{a} > 5) \wedge (\mathtt{a} < 0)$$
$$\mathtt{F'} : \mathtt{p} \wedge \mathtt{q}$$

- Is a given formula satisfiable or unsatisfiable?
- DPLL(T) architecture – abstract theory literals and use SAT solver



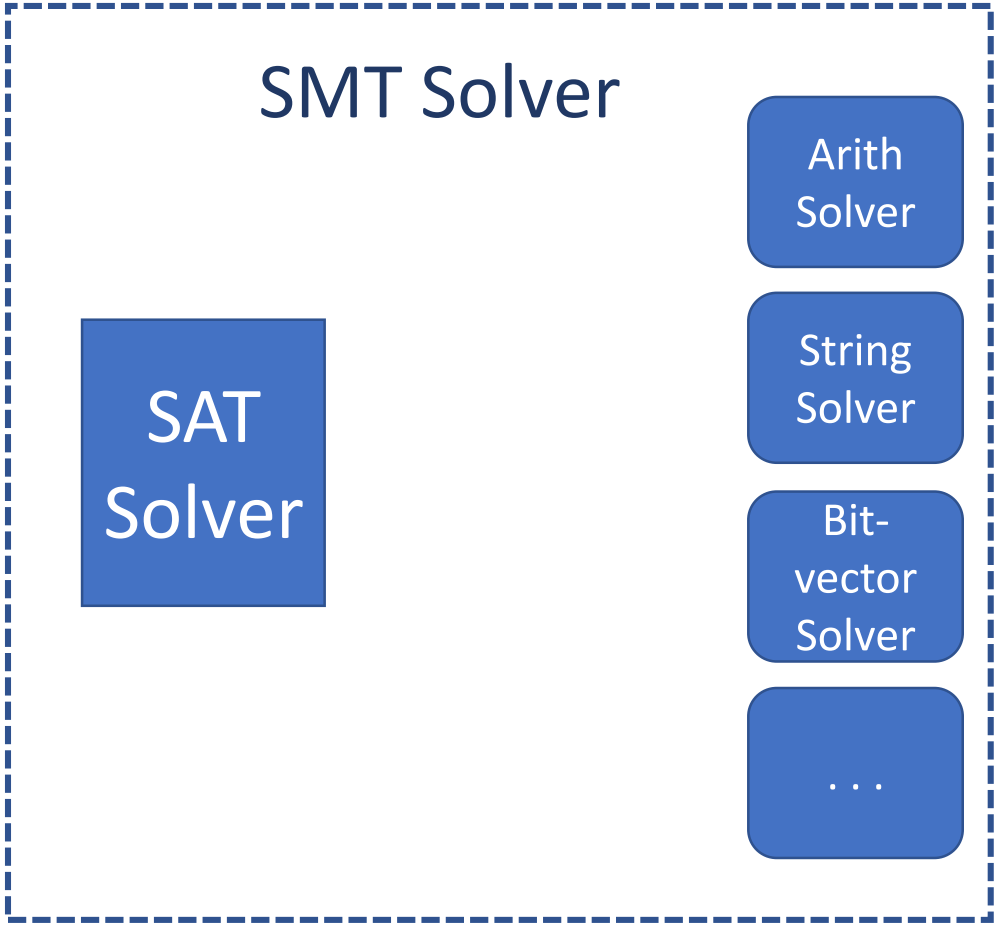
ATPs – SMT Solvers

- Is a given formula satisfiable or unsatisfiable?
- DPLL(T) architecture – abstract theory literals and use SAT solver

$$\mathtt{F} : (\mathtt{a} > 5) \wedge (\mathtt{a} < 0)$$

$$\mathtt{F}' : \mathtt{p} \wedge \mathtt{q}$$

$$\mathtt{M} : \{\mathtt{p} \mapsto \mathtt{T},\ \mathtt{q} \mapsto \mathtt{T}\}$$



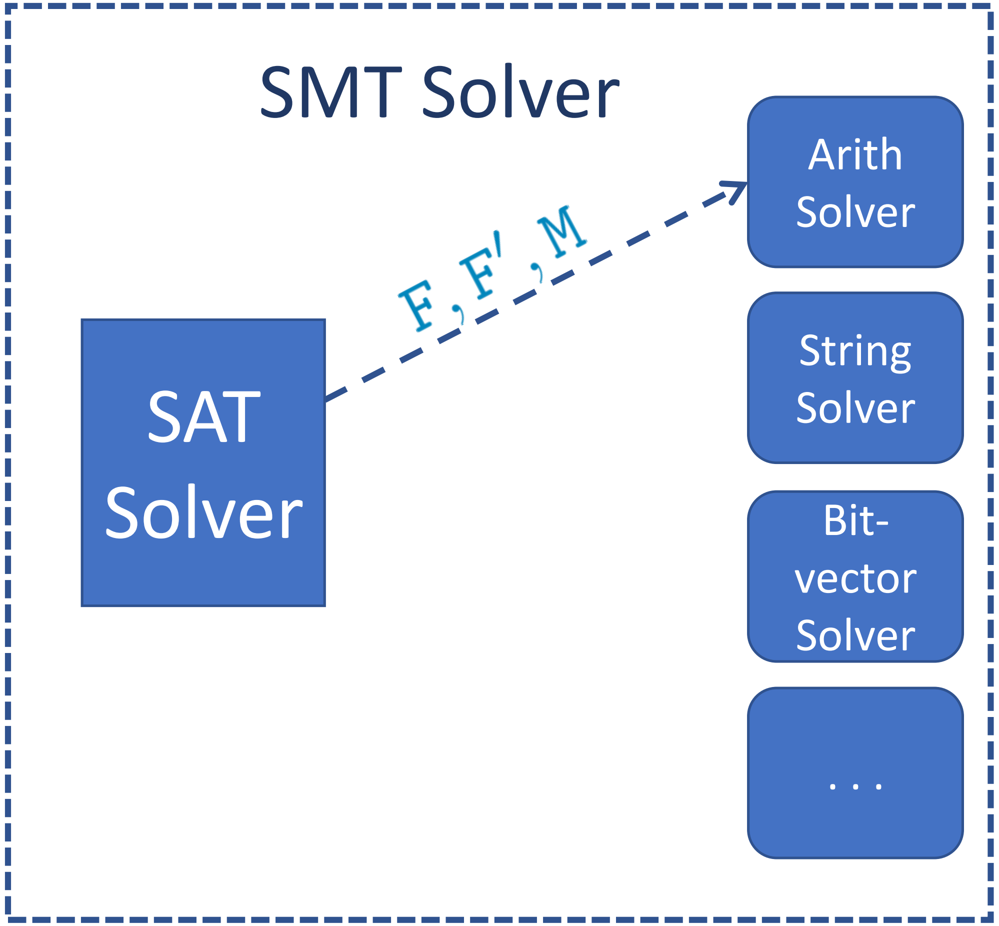
ATPs – SMT Solvers

- Is a given formula satisfiable or unsatisfiable?
- DPLL(T) architecture – abstract theory literals and use SAT solver

$$\mathtt{F} : (\mathtt{a} > 5) \wedge (\mathtt{a} < 0)$$

$$\mathtt{F}' : \mathtt{p} \wedge \mathtt{q}$$

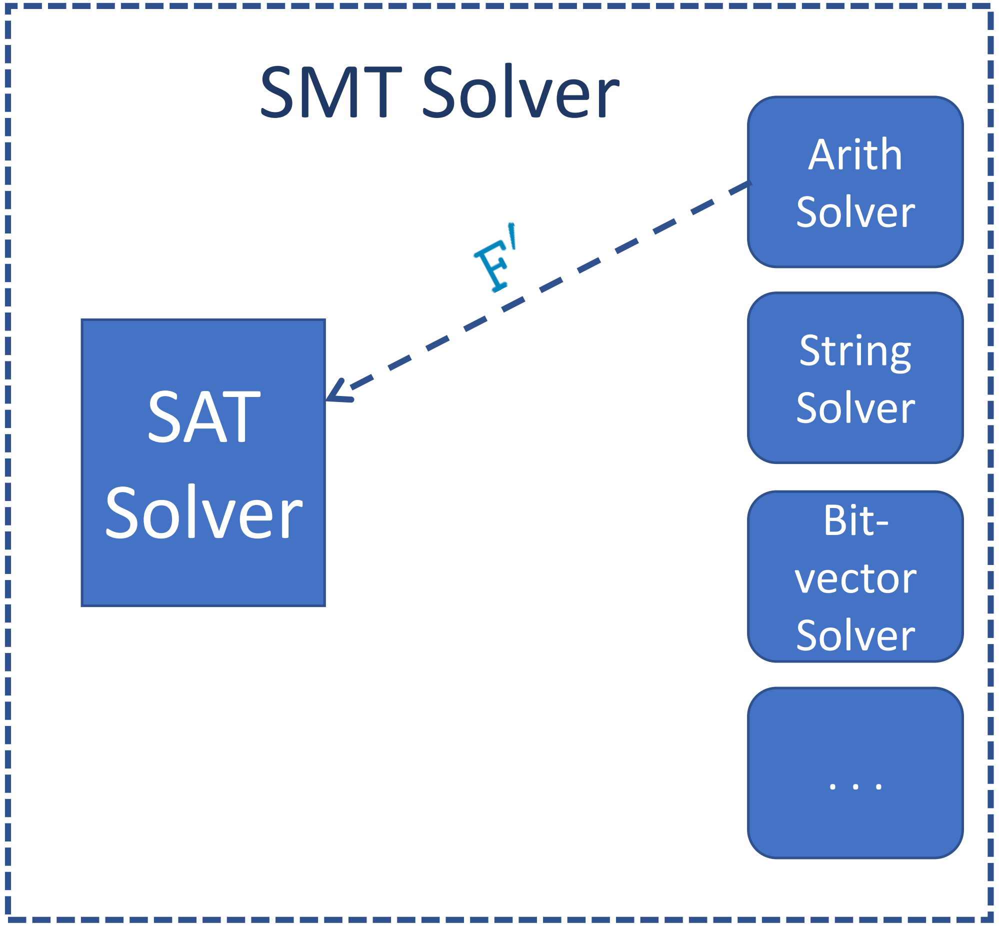
$$\mathtt{M} : \{\mathtt{p} \mapsto \mathtt{T},\ \mathtt{q} \mapsto \mathtt{T}\}$$



ATPs – SMT Solvers

- Is a given formula satisfiable or unsatisfiable?
- DPLL(T) architecture – abstract theory literals and use SAT solver

$$\mathtt{F} : (\mathtt{a} > 5) \wedge (\mathtt{a} < 0)$$
$$\mathtt{F}' : \mathtt{p} \wedge \mathtt{q} \wedge \neg(\mathtt{p} \wedge \mathtt{q})$$
$$\mathtt{M} : \{\mathtt{p} \mapsto \mathtt{T},\ \mathtt{q} \mapsto \mathtt{T}\}$$

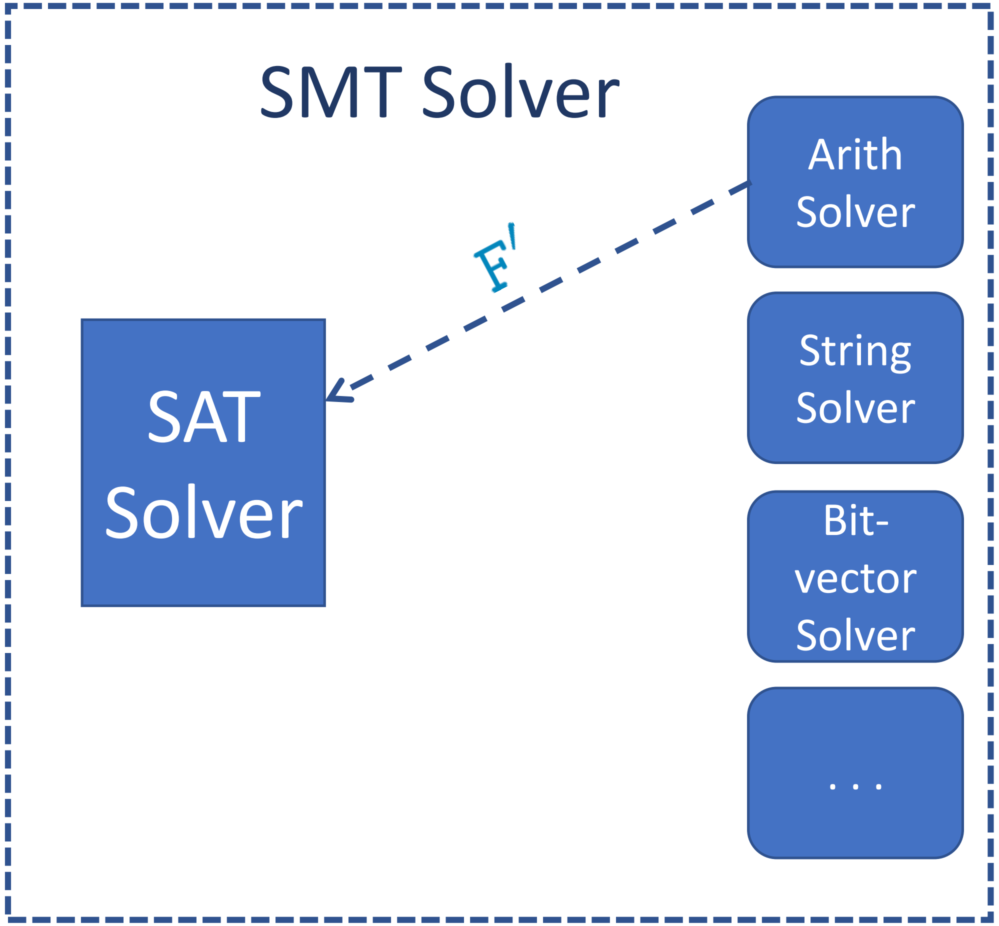


ATPs – SMT Solvers

$$\mathtt{F} : (\mathtt{a} > 5) \wedge (\mathtt{a} < 0)$$

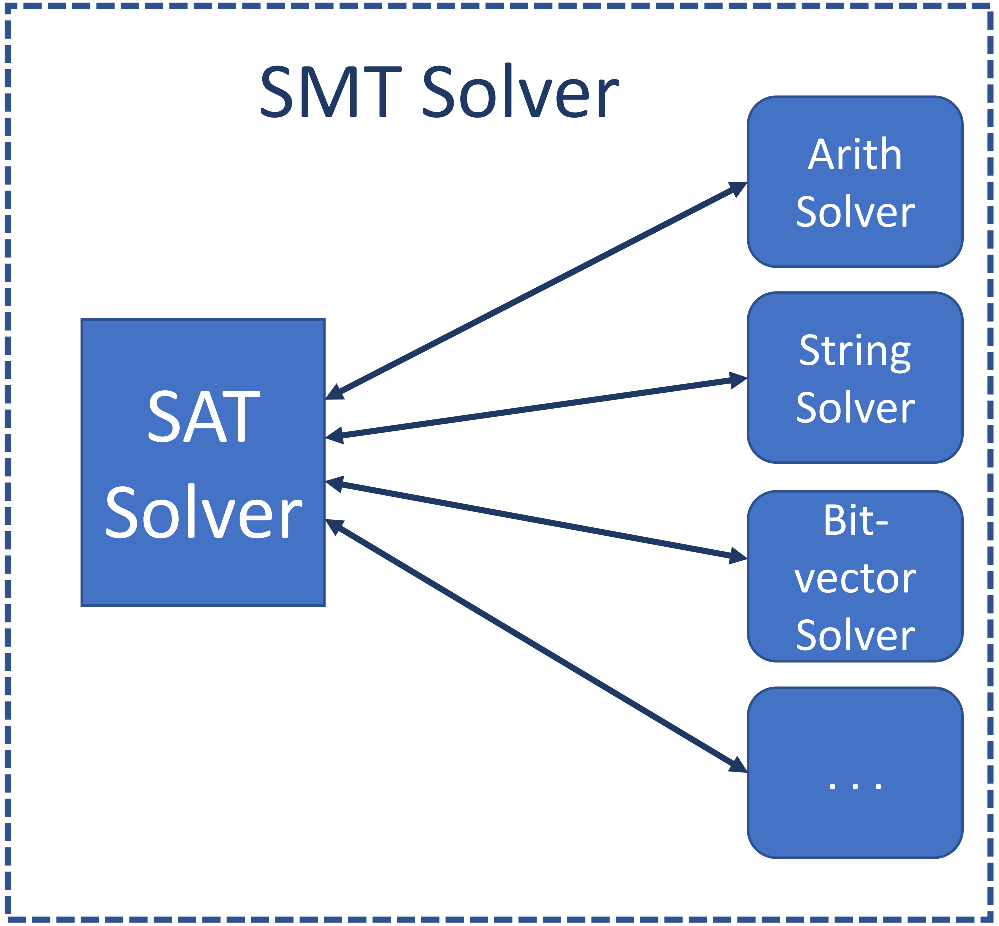
$$\mathtt{F}' : \mathtt{p} \wedge \mathtt{q} \wedge \neg(\mathtt{p} \wedge \mathtt{q})$$

- Is a given formula satisfiable or unsatisfiable?
- DPLL(T) architecture – abstract theory literals and use SAT solver



ATPs – SMT Solvers

- Is a given formula satisfiable or unsatisfiable?
- DPLL(T) architecture – abstract theory literals and use SAT solver
- SAT is NP-complete
- Theories and quantification may make solving undecidable



SMT Proof Rules

Proof Producing SMT Solvers

- Proof rules:
  - **CNF Conversion**
- CNF transformation algorithm by Tseitin
- Transformed formula is linear in size
- Procedure:
  - introduce new variable for each sub-term
  - unfold equivalences

$$\mathtt{CNF} : (x_1 \vee x_2 \vee \ldots) \wedge (y_1 \vee y_2 \vee \ldots) \wedge \ldots$$

$$\mathtt{F} : ((p \vee q) \wedge r) \rightarrow \neg s$$

$$x_1 \leftrightarrow \neg s$$
$$x_2 \leftrightarrow p \vee q$$
$$x_3 \leftrightarrow x_2 \wedge r$$
$$x_4 \leftrightarrow x_3 \rightarrow x_1$$

$$\begin{aligned} x_1 \leftrightarrow \neg s &\equiv (x_1 \rightarrow \neg s) \wedge (\neg s \rightarrow x_1) \\ &\equiv (\neg x_1 \vee \neg s) \wedge (\neg\neg s \vee x_1) \\ &\equiv (\neg x_1 \vee \neg s) \wedge (s \vee x_1) \end{aligned}$$

Proof Producing SMT Solvers

- Proof rules:
  - CNF Conversion
  - **Resolution**

$$\frac{a \vee \neg b \quad b \vee c}{a \vee c}$$

$$\frac{\phi_1 \vee \ldots \vee \phi_n \vee \chi \quad \neg\chi \vee \psi_1 \vee \ldots \vee \psi_m}{\phi_1 \vee \ldots \vee \phi_n \vee \psi_1 \vee \ldots \vee \psi_m}\ resolution$$

$n, m \geq 0$

Proof Producing SMT Solvers

- Proof rules:
  - CNF Conversion
  - Resolution
  - **Theory-specific**

$$\frac{x = y}{y = x}\ \texttt{symm}$$

$$\frac{a = b \quad b = c}{a = c}\ \texttt{trans}$$

$$\frac{i \neq j}{\mathrm{read}(a[i] := b, j) = \mathrm{read}(a, j)}\ \texttt{row}$$

Proof Producing SMT Solvers

- Proof rules:
  - CNF Conversion
  - Resolution
  - Theory-specific
  - **Quantifier**
- Existential ($\exists$) and universal ($\forall$) quantifiers
- Quantifier introduction rules
- Instantiation rule to eliminate $\forall$
- Skolemization to eliminate $\exists$

$$\frac{P(c)}{\exists x.P(x)}\ \exists\mathtt{intro}$$

$$\frac{\forall x.P(x)}{P(c/x)}\ \mathtt{inst}_{\forall}$$

Proof Producing SMT Solvers

- Proof rules:
  - CNF Conversion
  - Resolution
  - Theory-specific
  - Quantifier Elimination
  - **Rewrites**
- Some rewrites are proven
- Some are proof holes

$$p \land \neg p \mapsto \texttt{false}$$

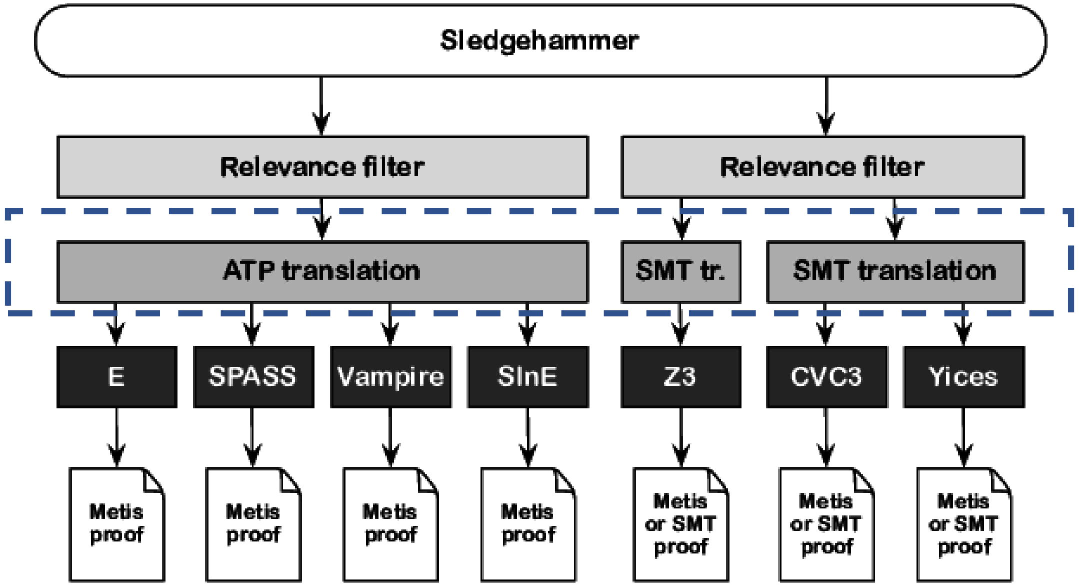
$$x \lor y \mapsto y \lor x$$

$$x + 0 \mapsto x$$

HOL -> FOL Translation

Translation

- ATP – typed/untyped FOL
- ITP – HOL
- FOL is a subset of HOL
- Non-FOL features of HOL:
  - Type variables – monomorphization
  - Anonymous functions – named function + quantified constraint
  - Partial applications – explicit application
  - Compound types – new FO type
  - Types - sorts



$$\alpha \to \beta \mapsto \texttt{Bool} \to \texttt{Bool}$$

$$t[\lambda x.\ u] \mapsto t[c] \wedge (\forall x.\ c\ x = u)$$

$$f\ x\ y \mapsto \texttt{app}\ (f\ x)\ y$$

$$\texttt{list bool} \mapsto \kappa$$

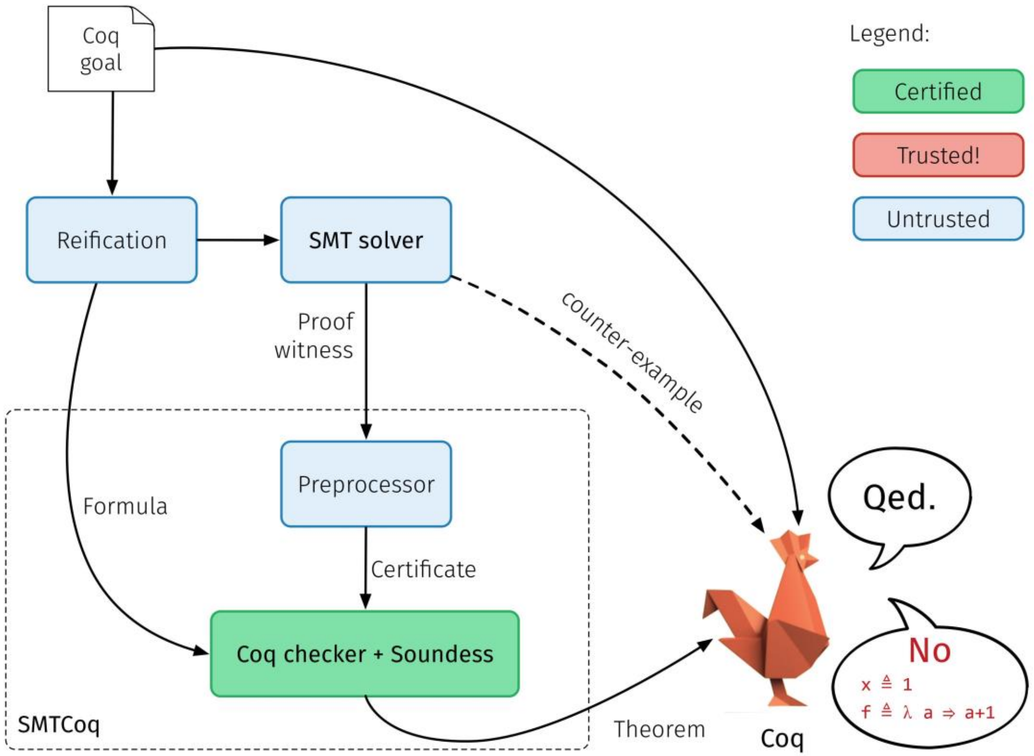
Reflection

Computational Reflection

- *Reflect* terms in the ITP's language (*shallow* embedding) to terms in a datatype written in the ITP's language (*deep* embedding)
- Deep embedding represents terms from ATP
- Use ATP methods over deep terms
  - Pattern matching is allowed over deep but not shallow terms
- Prove correctness result for the transformation :
  theorems in deep embedding give theorems in shallow embedding

SMTCoq

- Boolean decision procedure in Coq checks SMT proofs
- Reflection in SMTCoq
  - Deep embedding – datatypes $\leftrightarrow$ FOL terms
  - Shallow embedding – Coq terms $\leftrightarrow$ FOL terms
  - Interpretation – deep $\rightarrow$ shallow terms
  - Reification – shallow $\rightarrow$ deep terms
  - Ssreflect: Bool $\rightarrow$ Prop



Duality of Satisfiability and Validity

- F is valid iff ¬F is unsatisfiable

$$F : (\neg p \wedge q) \vee \neg q \vee p$$

VALID

$$\neg F : \neg(\neg p \wedge q) \wedge \neg\neg q \wedge \neg p$$

$$\neg F : (p \vee \neg q) \wedge \neg q \wedge \neg p$$

UNSAT