

Satisfiability and beyond:

Recent accomplishments and future trends in SMT

Cesare Tinelli

University of Belgrade, July 8, 2026



Satisfiability Modulo Theories

In the last 25 years, there has been a lot of R&D in automated reasoning focused on

- expressive enough **decidable fragments** of various formal logics
- incorporating **domain-specific** reasoning, e.g., on:
 - temporal reasoning
 - arithmetic reasoning
 - equality reasoning
 - reasoning about certain data structures (arrays, lists, finite sets, ...)
- combining specialized reasoners **modularly**

Two successful examples of this trend:

SAT: propositional formalization, Boolean reasoning

- + high degree of efficiency
- + sufficiently expressive (all NP-complete problems)
- often requiring involved encodings

Two successful examples of this trend:

SAT: propositional formalization, Boolean reasoning

- + high degree of efficiency
- + sufficiently expressive (all NP-complete problems)
- often requiring involved encodings

SMT: first-order formalization, Boolean + domain-specific reasoning

- + much improved expressiveness
- + greater scalability
- 0 some (but often acceptable) loss of efficiency
- much more complex implementation

Two successful examples of this trend:

SAT: propositional formalization, Boolean reasoning

- + high degree of efficiency
- + sufficiently expressive (all NP-complete problems)
- often requiring involved encodings

SMT: first-order formalization, Boolean + domain-specific reasoning

- + much improved expressiveness
- + greater scalability
- 0 some (but often acceptable) loss of efficiency
- much more complex implementation

The Basic SMT Problem

Determining the **satisfiability** of a first-order-logic formula **with respect to** (*modulo*) some combination *T* of **background theories**

The Basic SMT Problem

Determining the **satisfiability** of a first-order-logic formula **with respect to** (*modulo*) some combination *T* of **background theories**

Example

$$n > 3 * m + 1 \quad \wedge \quad (f(n) \leq head(l_1) \vee l_2 = f(n) :: l_1)$$

The Basic SMT Problem

Determining the **satisfiability** of a first-order-logic formula **with respect to** (*modulo*) some combination *T* of **background theories**

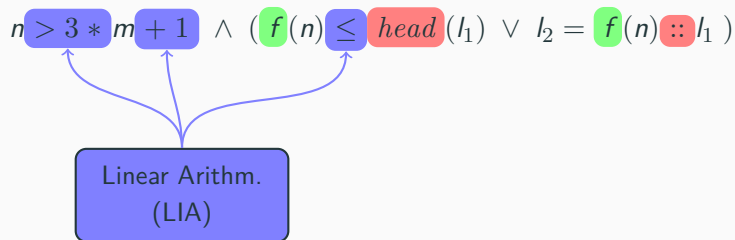
Example

$$n > 3 * m + 1 \wedge (f(n) \leq head(l_1) \vee l_2 = f(n) :: l_1)$$

The Basic SMT Problem

Determining the **satisfiability** of a first-order-logic formula **with respect to** (*modulo*) some combination **T** of **background theories**

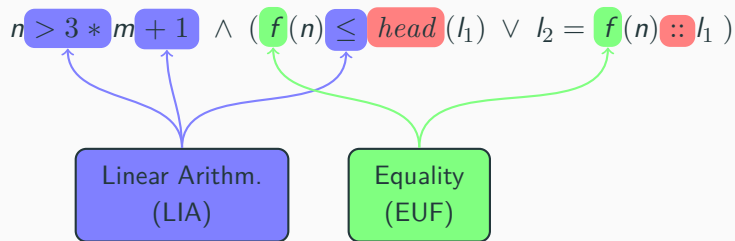
Example



The Basic SMT Problem

Determining the **satisfiability** of a first-order-logic formula **with respect to** (*modulo*) some combination **T** of **background theories**

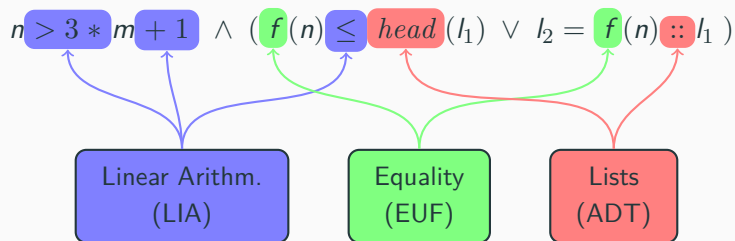
Example



The Basic SMT Problem

Determining the **satisfiability** of a first-order-logic formula **with respect to** (*modulo*) some combination **T** of **background theories**

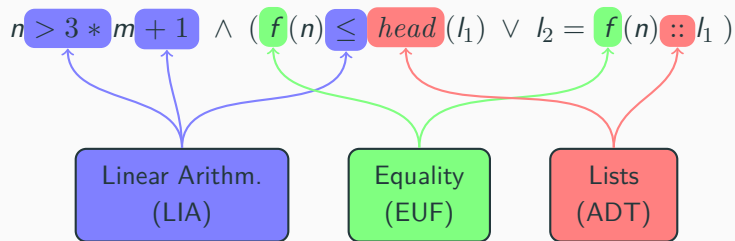
Example



The Basic SMT Problem

Determining the **satisfiability** of a first-order-logic formula **with respect to** (*modulo*) some combination **T** of **background theories**

Example



SMT formulas are formulas in **many-sorted FOL** with ***built-in* symbols**

Highly efficient tools for the SMT problem based on specialized logic engines

Usually obtained as a combination a SAT solver and several theory solvers

Highly efficient tools for the SMT problem based on specialized logic engines

Usually obtained as a combination a SAT solver and several theory solvers

as well as heuristic quantifier instantiation modules

Highly efficient tools for the SMT problem based on specialized logic engines

Usually obtained as a combination a SAT solver and several theory solvers

as well as heuristic quantifier instantiation modules

Building on the strengths of modern SAT solvers,
have revolutionized the way people solve problems in Computer Science and beyond:

Highly efficient tools for the SMT problem based on specialized logic engines

Usually obtained as a combination a SAT solver and several theory solvers
as well as heuristic quantifier instantiation modules

Building on the strengths of modern SAT solvers,
have revolutionized the way people solve problems in Computer Science and beyond:

Instead of building a special-purpose tool

- translate problem into a logical formula
- use an SMT solver as backend reasoner

Highly efficient tools for the SMT problem based on specialized logic engines

Usually obtained as a combination a SAT solver and several theory solvers
as well as heuristic quantifier instantiation modules

Building on the strengths of modern SAT solvers,
have revolutionized the way people solve problems in Computer Science and beyond:

Instead of building a special-purpose tool

- translate problem into a logical formula
- use an SMT solver as backend reasoner

Not only easier,
often better

Popular SMT Solvers (as of July 2025)

	Citations	Google Scholar Hits
Z3	12,044 ¹	34K
CVC 1–5	3,424 ²	4K
Yices 1–2	1,691 ³	3K
MathSat 1–5	1,220 ⁴	2K

¹[DMB08], ETAPS 2018 Test of Time Award to Z3 developers.

²CVC3, CVC4, cvc5 [BT07, BCD⁺11, ?], ETAPS 2026 Test-of-Time Tool Award to CVC4/cvc5

³Yices, Yices 2 [DdM06b, Dut14].

⁴MathSat 3, MathSat 4, MathSat 5 [BBC⁺05b, BCF⁺08, CGSS13].

Popular SMT Solvers (as of July 2025)

	Citations	Google Scholar Hits
Z3	12,044 ¹	34K
CVC 1–5	3,424 ²	4K
Yices 1–2	1,691 ³	3K
MathSat 1–5	1,220 ⁴	2K

Additional noteworthy solvers

Alt-Ergo, Bitwuzla, Boolector, Colibri, OpenSMT, SMTInterpol, SMT-RAT, STP, veriT

¹[DMB08], ETAPS 2018 Test of Time Award to Z3 developers.

²CVC3, CVC4, cvc5 [BT07, BCD⁺11, ?], ETAPS 2026 Test-of-Time Tool Award to CVC4/cvc5

³Yices, Yices 2 [DdM06b, Dut14].

⁴MathSat 3, MathSat 4, MathSat 5 [BBC⁺05b, BCF⁺08, CGSS13].

Applications of SMT

Some Applications of SMT

Model Checking

- Finite/infinite-state systems
- Hybrid systems
- Abstraction refinement
- State invariant generation
- Interpolation-based

Type Checking

- Dependent types
- Semantic subtyping/refinement types
- Type error localization

Program Analysis

- Symbolic execution

Program verification

Verification in separation logic

Termination analysis

Loop invariant generation

Procedure summaries

Race-condition analysis

Concurrency errors detection

Software Synthesis

Syntax-guided function synthesis

Automated program repair

Synthesis of reactive systems

Synthesis of self-stabilizing systems

Schedule synthesis for time-triggered networks

Security

- Automated exploit generation
- Protocol debugging and verification
- Analysis of access control policies
- Run-time monitoring

Compilers

- Compilation validation
- Arithmetic computation optimization

Networks

- Verification of reachability, fault-tolerance, router equivalence, and load balancing

- Scheduling overload for real-time systems
- Computing communication schedules
- Schedule synthesis

Software Engineering

- System model consistency
- Design analysis
- Test case generation
- Verification of ATL transformations
- Semantic search for code reuse
- Interactive (software) requirements prioritization
- Generating instances of meta-models
- Behavioral conformance of web services
- Verification of normative requirements

Even More Applications of SMT

Planning

Motion planning

Nonlinear PDDL planning

Machine Learning

Verification of deep NNs

Neurosymbolic Generative AI

Business

Verification of business rules

Spreadsheet debugging

Interactive Theorem Proving

Hammers for proof-assistants

Automated, proof-producing tactics

Background Theories

Some Background Theories used in SMT (and formulas valid in them)

Uninterpreted Functions

$$x = y \Rightarrow f(x) = f(y)$$

Integer/Real Arithmetic

$$2x + y = 0 \wedge 2x - y = 4 \Rightarrow x = 1$$

Floating Point Arithmetic

$$x + 1 \neq NaN \wedge x < \infty \Rightarrow x + 1 > x$$

Fixed-size Bit Vectors

$$4 \cdot (x \gg 2) = x \& \sim 3$$

Strings and Regular Expressions

$$x = y \cdot z \wedge z \in ab^* \Rightarrow |x| > |y|$$

Arrays

$$i = j \wedge b = a[i := x] \Rightarrow b[j] = x$$

Algebraic Data Types

$$x \neq Leaf \Rightarrow \exists l, r : Tree(\alpha). \exists a : \alpha. x = Node(l, a, r)$$

Finite Sets

$$e_1 \in x \wedge e_2 \in x \setminus e_1 \Rightarrow \\ \exists y, z : Set(\alpha). |y| = |z| \wedge x = y \cup z \wedge y \neq \emptyset$$

Finite Relations

$$(x, y) \in r \wedge (y, z) \in r \Rightarrow (x, z) \in r \bowtie s$$

Equality and Uninterpreted Functions (EUF) [NO80, NO07]

Simplest first-order theory with equality, applications of uninterpreted functions, and variables of uninterpreted sorts

For all sorts σ, σ' and function symbols $f: \sigma \rightarrow \sigma'$

Reflexivity: $\forall x: \sigma. x = x$

Symmetry: $\forall x: \sigma. x = y \Rightarrow y = x$

Transitivity: $\forall x, y: \sigma. x = y \wedge y = z \Rightarrow x = z$

Congruence: $\forall \mathbf{x}, \mathbf{y}: \sigma. \mathbf{x} = \mathbf{y} \Rightarrow f(\mathbf{x}) = f(\mathbf{y})$

Example

$$h(h(h(a))) = b \wedge g(h(a), b) = a \wedge h(a) \neq a$$

Operates over sorts $\text{Array}(\sigma_i, \sigma_e)$, σ_i , σ_e and function symbols

$$_\[_] : \text{Array}(\sigma_i, \sigma_e) \times \sigma_i \rightarrow \sigma_e$$

$$\text{store} : \text{Array}(\sigma_i, \sigma_e) \times \sigma_i \times \sigma \rightarrow \text{Array}(\sigma_i, \sigma_e)$$

For any index sort σ_i and element sort σ_e

Read-Over-Write-1: $\forall a, i, e. \text{store}(a, i, e)[i] = e$

Read-Over-Write-2: $\forall a, i, j, e. i \neq j \Rightarrow \text{store}(a, i, e)[j] = a[j]$

Extensionality: $\forall a, b, i. a \neq b \Rightarrow \exists i. a[i] \neq b[i]$

Example

$$\text{store}(\text{store}(a, i, a[j]), j, a[i]) = \text{store}(\text{store}(a, j, a[i]), i, a[j])$$

Restricted fragments, over the reals or the integers, are interesting because they support efficient reasoning methods:

- Bounds: $x \bowtie k$ with $\bowtie \in \{<, >, \leq, \geq, =\}$ [BBC⁺05a]
- Difference constraints: $x - y \bowtie k$, with $\bowtie \in \{<, >, \leq, \geq, =\}$ [NO05, WIGG05, CM06]
- UTVPI: $\pm x \pm y \bowtie k$, with $\bowtie \in \{<, >, \leq, \geq, =\}$ [LM05]
- Linear arithmetic, e.g: $2x - 3y + 4z \leq 5$ [DdM06a]
- Non-linear arithmetic, e.g: $2xy + 4xz^2 - 5y \leq 10$ [BLNM⁺09, ZM10, JdM12]

Family of user-definable theories

Example

$\text{Color} \quad := \quad \text{red} \mid \text{green} \mid \text{blue}$

$\text{List}(\alpha) \quad := \quad [] \mid (\text{head} : \alpha) :: (\text{tail} : \text{List}(\alpha))$

Distinctiveness: $\forall h, t. [] \neq h :: t$

Exhaustiveness: $\forall l. l = [] \vee \exists h, t. l = h :: t$

Injectivity: $\forall h_1, h_2, t_1, t_2. h_1 :: t_1 = h_2 :: t_2 \Rightarrow h_1 = h_2 \wedge t_1 = t_2$

Selectors: $\forall h, t. \text{head}(h :: t) = h \wedge \text{tail}(h :: t) = t$

(Non-circularity: $\forall l, x_1, \dots, x_n. l \neq x_1 :: \dots :: x_n :: l)$

Other Theories Considered in SMT

- Parametric-size bit vectors
- Sequences [SNR⁺23]
- Finite multisets and tables [MRTB17, MRTB24]
- Finite fields [OKTB23, POP⁺25]
- Transcendental functions [GKC13, CGI⁺18]
- Ordinary differential equations [GKC13]
- Partial orders
- ...

Functionality

Legend

v value — i.e., distinguished variable-free term

$\varphi[\mathbf{x}]$ formula with free vars from $\mathbf{x} = (x_1, \dots, x_n)$

$\varphi[\mathbf{x} \mapsto \mathbf{v}]$ formula obtained by replacing free occurrences of variables from $\mathbf{x}$ in φ with corresponding values from $\mathbf{v} = (v_1, \dots, v_n)$

$\mathbf{x} = \mathbf{v}$ $x_1 = v_1 \wedge \dots \wedge x_n = v_n$

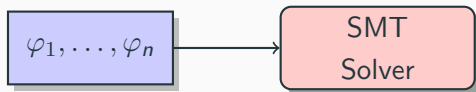
$\mathbf{z} \subseteq \mathbf{x}$ every element of $\mathbf{z}$ occurs in $\mathbf{x}$

$M \models \varphi$ model M satisfies formula φ

$\varphi \models_T \psi$ formula φ entails formula ψ in theory T

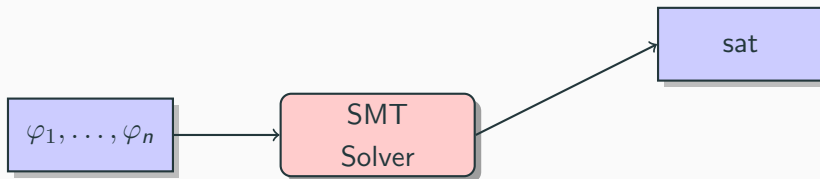
SMT Solver Basic Functionality

Background theory T



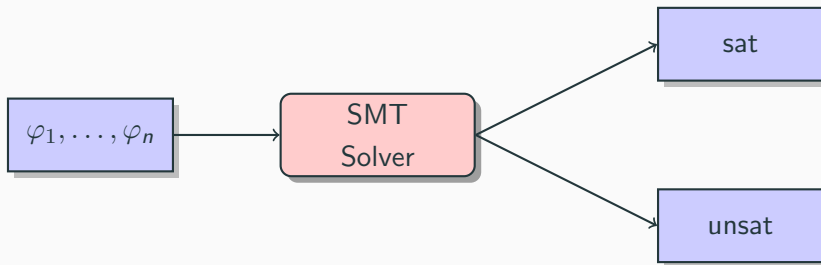
SMT Solver Basic Functionality

Background theory T



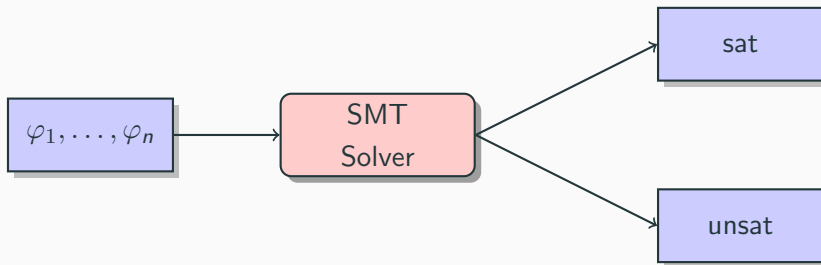
SMT Solver Basic Functionality

Background theory T



SMT Solver Basic Functionality

Background theory T

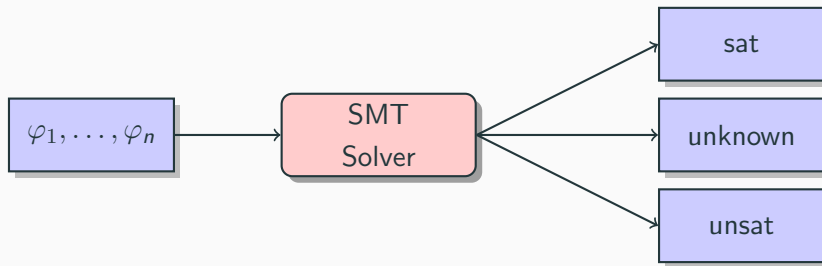


sat/unsat: there is **a/no** model M of T such that

$$M \models \varphi_1 \wedge \dots \wedge \varphi_n$$

SMT Solver Basic Functionality

Background theory T



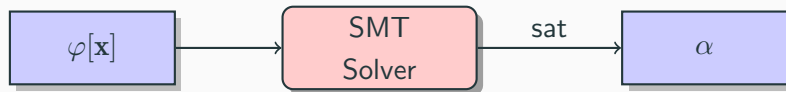
sat/unsat: there is **a/no** model M of T such that

$$M \models \varphi_1 \wedge \dots \wedge \varphi_n$$

unknown: inconclusive — because of resource limits or incompleteness

SMT Solver Output: Satisfying Assignments

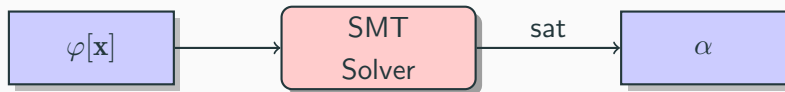
Background theory T



α is a *satisfying assignment* for $\mathbf{x} = (x_1, \dots, x_n)$:

SMT Solver Output: Satisfying Assignments

Background theory T



α is a *satisfying assignment* for $\mathbf{x} = (x_1, \dots, x_n)$:

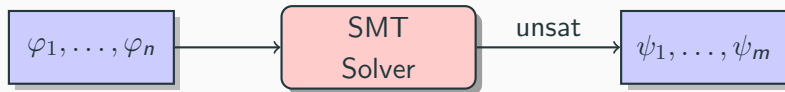
1. $\alpha = \{x_1 \mapsto v_1, \dots, x_n \mapsto v_n\}$ for some *values* $\mathbf{v} = (v_1, \dots, v_n)$
2. $M \models \varphi[\mathbf{x} \mapsto \mathbf{v}]$ for some model M of T

Note.

$\mathbf{x}$ may consist of first- and second-order variables
(aka, *uninterpreted constants and function symbols*)

SMT Solver Output: **Unsat Cores**

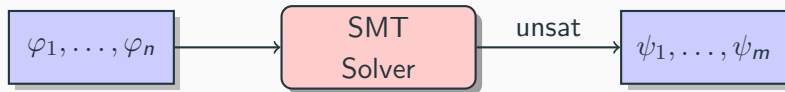
Background theory T



$\psi_1, \dots, \psi_m$ is a **unsat core** of $\{\varphi_1, \dots, \varphi_n\}$:

SMT Solver Output: **Unsat Cores**

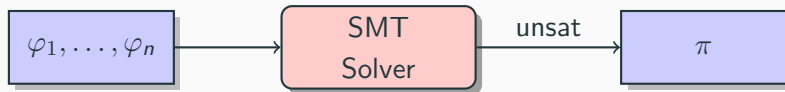
Background theory T



$\psi_1, \dots, \psi_m$ is a **unsat core** of $\{\varphi_1, \dots, \varphi_n\}$:

1. $\{\psi_1, \dots, \psi_m\} \subseteq \{\varphi_1, \dots, \varphi_n\}$
2. $\{\psi_1, \dots, \psi_m\}$ is unsat in T
3. $\{\psi_1, \dots, \psi_m\}$ is minimal (or smallish)

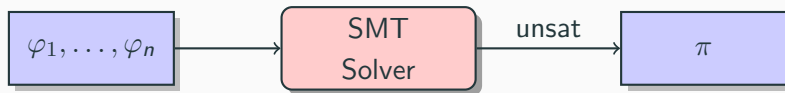
Background theory T



π is a checkable *proof certificate* for $\{\varphi_1, \dots, \varphi_n\}$, i.e. π :

SMT Solver Output: Proofs

Background theory T

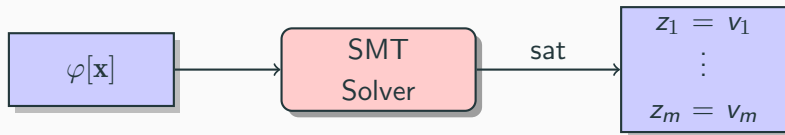


π is a checkable *proof certificate* for $\{\varphi_1, \dots, \varphi_n\}$, i.e. π :

1. is a proof term in some formal proof system
2. expresses a *refutation* of $\{\varphi_1, \dots, \varphi_n\}$
3. can be *efficiently checked* by an *external* proof checker

Further SMT Solver Output: *Backbones*

Background theory T

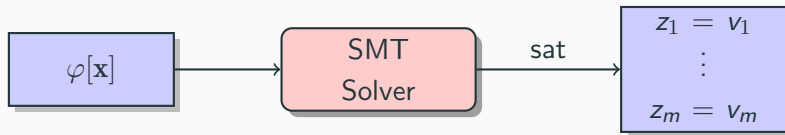


$\mathbf{z} = \mathbf{v}$ is a *backbone* for φ :

1. $\mathbf{z} \subseteq \mathbf{x}$
2. $\varphi \models_T \mathbf{z} = \mathbf{v}$
3. $\mathbf{z}$ is maximal (or largish)

Further SMT Solver Output: Sat Cores

Background theory T

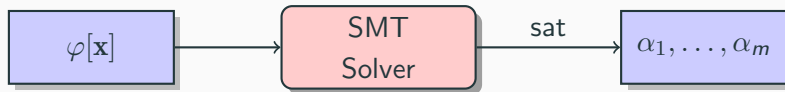


$\mathbf{z} = \mathbf{v}$ is a *sat core* for φ :

1. $\mathbf{z} \subseteq \mathbf{x}$
2. $\mathbf{y} = \mathbf{x} \setminus \mathbf{z}$
3. $\forall \mathbf{y} (\varphi \wedge \mathbf{z} = \mathbf{v})$ is satisfiable in T (i.e., $\mathbf{z} = \mathbf{v}$ is a *partial satisfying* assignment for φ)
4. $\mathbf{z}$ is minimal (or smallish)

Further SMT Solver Output: All Satisfying Assignments

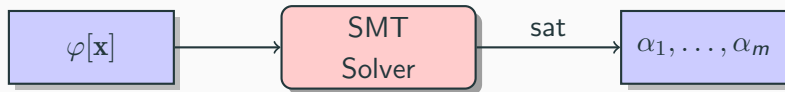
Background theory T



$\alpha_1, \dots, \alpha_m$ comprise all the satisfying assignments for $\mathbf{x}$:

Further SMT Solver Output: All Satisfying Assignments

Background theory T

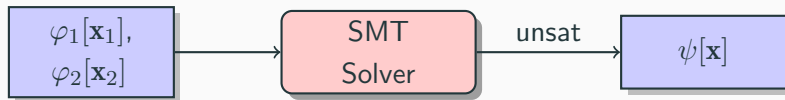


$\alpha_1, \dots, \alpha_m$ comprise **all** the satisfying assignments for $\mathbf{x}$:

1. every α_i is a **partial** satisfying assignment for φ
2. every full satisfying assignment for φ is a completion of some α_i

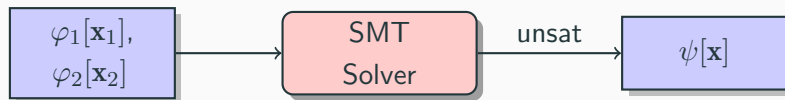
Extended Functionality: *Interpolation*

Background theory T



ψ is a logical *interpolant* of φ_1 and φ_2 :

Background theory T

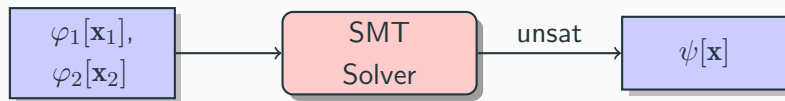


ψ is a logical *interpolant* of φ_1 and φ_2 :

1. $\varphi_1 \models_T \psi$ and $\psi \models_T \neg\varphi_2$
2. $\mathbf{x} = \mathbf{x}_1 \cap \mathbf{x}_2$

Extended Functionality: Interpolation

Background theory T



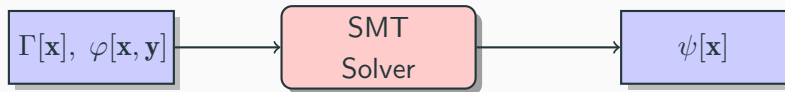
ψ is a logical *interpolant* of φ_1 and φ_2 :

1. $\varphi_1 \models_T \psi$ and $\psi \models_T \neg\varphi_2$
2. $\mathbf{x} = \mathbf{x}_1 \cap \mathbf{x}_2$

Ideally, ψ is *quantifier-free* whenever φ_1 and φ_2 are

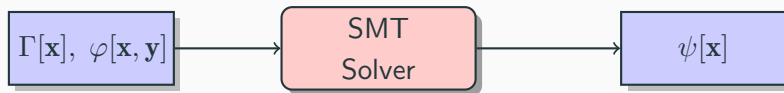
Extended Functionality: Quantifier Elimination

Background theory T



Extended Functionality: Quantifier Elimination

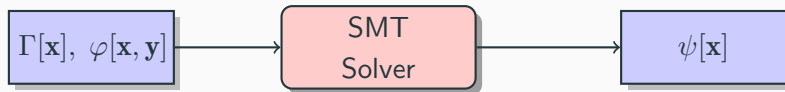
Background theory T



ψ is a *projection* of φ over $\mathbf{y}$ with respect to Γ :

Extended Functionality: Quantifier Elimination

Background theory $\mathcal{T}$

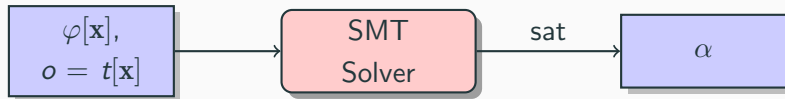


ψ is a *projection* of φ over $\mathbf{y}$ with respect to Γ :

1. $\Gamma \models_{\mathcal{T}} \psi \Leftrightarrow \exists \mathbf{y} \varphi$

Extended Functionality: Optimization

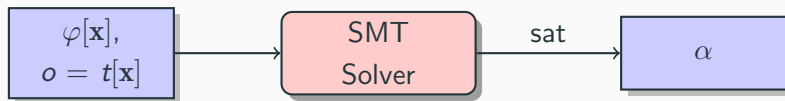
Background theory T



α is a *an optimal assignment* for φ :

Extended Functionality: Optimization

Background theory T

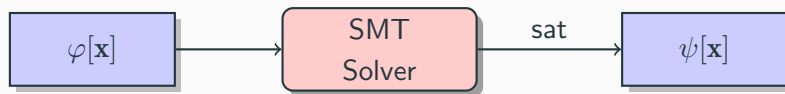


α is a *an optimal assignment* for φ :

1. $\alpha = \{x_1 \mapsto v_1, \dots, x_n \mapsto v_n\}$ for some *values* $v_1, \dots, v_n$
2. $M \models \varphi[\mathbf{x} \mapsto \mathbf{v}]$ for some model M of T
3. α minimizes/maximizes *objective* o

Extended Functionality: Prime Implicate Synthesis

Background theory T

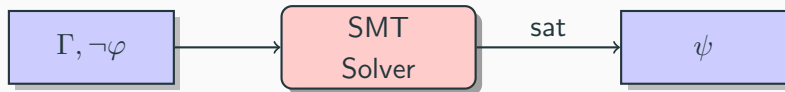


ψ is a *prime implicate* of φ :

1. ψ is a disjunction of literals
2. $\varphi \models_T \psi$
3. there is no disjunction of literals $\psi' \notin \{\varphi, \psi\}$ s.t. $\varphi \models_T \psi'$ and $\psi' \models_T \psi$

Extended Functionality: Abduction

Background theory T



ψ is an *abduction hypothesis* for φ wrt Γ :

1. Γ, ψ is satisfiable in T
2. $\Gamma, \psi \models_T \varphi$
3. ψ is *maximal*, e.g., with respect to $\models_T$
(if ψ' satisfies 1 and 2 and $\psi \models_T \psi'$ then $\psi' \models_T \psi$)

- Quantified formulas

- Quantified formulas
- User-defined polymorphic types

- Quantified formulas
- User-defined polymorphic types
- Higher-order logic (lambdas, folds, maps, etc.)

- Meaningful diagnostics (esp. for unknown answers)

- Meaningful diagnostics (esp. for unknown answers)
- Leveraging concurrent/parallel architectures

- Meaningful diagnostics (esp. for unknown answers)
- Leveraging concurrent/parallel architectures
- Stability

- Meaningful diagnostics (esp. for unknown answers)
- Leveraging concurrent/parallel architectures
- Stability
- Modularity

- Meaningful diagnostics (esp. for unknown answers)
- Leveraging concurrent/parallel architectures
- Stability
- Modularity
- Correctness!

- Meaningful diagnostics (esp. for unknown answers)
- Leveraging concurrent/parallel architectures
- Stability
- Modularity
- Correctness!
- ...

References

Cited Works

 M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Schulz, and R. Sebastiani.

An incremental and layered procedure for the satisfiability of linear arithmetic logic.

In *Tools and Algorithms for the Construction and Analysis of Systems, 11th Int. Conf., (TACAS)*, volume 3440 of *Lecture Notes in Computer Science*, pages 317–333, 2005.

 Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Tommi A. Junttila, Peter van Rossum, Stephan Schulz, and Roberto Sebastiani.

The MathSAT 3 system.

In Robert Nieuwenhuis, editor, *Proceedings of the 20th International Conference on Automated Deduction (CADE-20)*, volume 3632 of *Lecture Notes in Computer Science*, pages 315–321. Springer, 2005.



Clark Barrett, Christopher Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds, and Cesare Tinelli.

CVC4.

In G. Gopalakrishnan and S. Qadeer, editors, *23rd International Conference on Computer Aided Verification (CAV'11), Snowbird, Utah*, volume 6806 of *Lecture Notes in Computer Science*, pages 171–177. Springer, 2011.



Roberto Bruttomesso, Alessandro Cimatti, Anders Franzén, Alberto Griggio, and Roberto Sebastiani.

The MathSAT 4 SMT solver.

In Aarti Gupta and Sharad Malik, editors, *Proceedings of the 20th international conference on Computer Aided Verification (CAV'08)*, volume 5123 of *Lecture Notes in Computer Science*, pages 299–303. Springer, 2008.



C. Borralleras, S. Lucas, R. Navarro-Marset, E. Rodríguez-Carbonell, and A. Rubio.

Solving Non-linear Polynomial Arithmetic via SAT Modulo Linear Arithmetic.

In R. A. Schmidt, editor, *22nd International Conference on Automated Deduction , CADE-22*, volume 5663 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2009.



M. Bofill, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio.

A Write-Based Solver for SAT Modulo the Theory of Arrays.

In *Formal Methods in Computer-Aided Design, FMCAD*, pages 1–8, 2008.



Clark Barrett, Igor Shikanian, and Cesare Tinelli.

An abstract decision procedure for satisfiability in the theory of recursive data types.

Electronic Notes in Theoretical Computer Science, 174(8):23–37, 2007.



Clark Barrett and Cesare Tinelli.

CVC3.

In W. Damm and H. Hermanns, editors, *Proceedings of the 19th International Conference on Computer Aided Verification (CAV'07), Berlin, Germany*, Lecture Notes in Computer Science. Springer, 2007.



Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani.

Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions.

ACM Trans. Comput. Log., 19(3):19:1–19:52, 2018.



A. Cimatti, A. Griggio, B. Schaafsma, and R. Sebastiani.

The MathSAT5 SMT solver.

In Nir Piterman and Scott A. Smolka, editors, *Proceedings of TACAS*, volume 7795 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2013.



S. Cotton and O. Maler.

Fast and Flexible Difference Constraint Propagation for DPLL(T).

In A. Biere and C. P. Gomes, editors, *9th International Conference on Theory and Applications of Satisfiability Testing, SAT'06*, volume 4121 of *Lecture Notes in Computer Science*, pages 170–183. Springer, 2006.



Bruno Dutertre and Leonardo de Moura.

A Fast Linear-Arithmetic Solver for DPLL(T).

In T. Ball and R. B. Jones, editors, *18th International Conference on Computer Aided Verification, CAV'06*, volume 4144 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2006.



Bruno Dutertre and Leonardo de Moura.

The YICES SMT solver.

Technical report, SRI International, 2006.



Leonardo De Moura and Nikolaj Bjørner.

Z3: an efficient smt solver.

In *Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems, TACAS'08/ETAPS'08*, pages 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.



L. de Moura and N. Bjørner.

Generalized, efficient array decision procedures.

In *9th International Conference on Formal Methods in Computer-Aided Design, FMCAD 2009*, pages 45–52. IEEE, 2009.



Bruno Dutertre.

Yices 2.2.

In Armin Biere and Roderick Bloem, editors, *Proceedings of the 26th International Conference on Computer Aided Verification (CAV 2014)*, volume 8559 of *Lecture Notes in Computer Science*, pages 737–744. Springer, 2014.



Sicun Gao, Soonho Kong, and Edmund M Clarke.

Satisfiability modulo ODEs.

In *Formal Methods in Computer-Aided Design (FMCAD)*, 2013, pages 105–112. IEEE, 2013.



Dejan Jovanović and Leonardo de Moura.

Solving Non-linear Arithmetic.

In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *6th International Joint Conference on Automated Reasoning (IJCAR '12)*, volume 7364 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012.



Shuvendu K. Lahiri and Madanlal Musuvathi.

An Efficient Decision Procedure for UTVPI Constraints.

In B. Gramlich, editor, *5th International Workshop on Frontiers of Combining Systems, FroCos'05*, volume 3717 of *Lecture Notes in Computer Science*, pages 168–183. Springer, 2005.



John McCarthy.

Towards a mathematical science of computation.

In *Program Verification*, pages 35–56. Springer, 1993.



Baoluo Meng, Andrew Reynolds, Cesare Tinelli, and Clark Barrett.

Relational constraint solving in SMT.

In Leonardo de Moura, editor, *Proceedings of the 26th International Conference on Automated Deduction*, volume 10395 of *Lecture Notes in Computer Science*, pages 148–165. Springer, 2017.



Mudathir Mohamed, Andrew Reynolds, Cesare Tinelli, and Clark W. Barrett.

Verifying SQL queries using theories of tables and relations.

In Nikolaj S. Bjørner, Marijn Heule, and Andrei Voronkov, editors, *LPAR 2024: Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning, Port Louis, Mauritius, May 26-31, 2024*, volume 100 of *EPiC Series in Computing*, pages 445–463. EasyChair, 2024.



Greg Nelson and Derek C. Oppen.

Fast decision procedures based on congruence closure.

Journal of the ACM, 27(2):356–364, 1980.



Robert Nieuwenhuis and Albert Oliveras.

DPLL(T) with Exhaustive Theory Propagation and its Application to Difference Logic.

In Kousha Etessami and Sriram K. Rajamani, editors, *Proceedings of the 17th International Conference on Computer Aided Verification, CAV'05 (Edinburgh, Scotland)*, volume 3576 of *Lecture Notes in Computer Science*, pages 321–334. Springer, July 2005.



Robert Nieuwenhuis and Albert Oliveras.

Fast congruence closure and extensions.

Information and Computation, 205(4):557–580, 2007.



Alex Ozdemir, Gereon Kremer, Cesare Tinelli, and Clark W. Barrett.

Satisfiability modulo finite fields.

In Constantin Enea and Akash Lal, editors, *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II*, volume 13965 of *Lecture Notes in Computer Science*, pages 163–186. Springer, 2023.



Elizaveta Pertseva, Alex Ozdemir, Shankara Pailoor, Alp Bassa, Sorawee Porncharoenwase, Isil Dillig, and Clark W. Barrett.

Integer reasoning modulo different constants in SMT.

In Ruzica Piskac and Zvonimir Rakamaric, editors, *Computer Aided Verification - 37th International Conference, CAV 2025, Zagreb, Croatia, July 23-25, 2025, Proceedings, Part I*, volume 15931 of *Lecture Notes in Computer Science*, pages 339–362. Springer, 2025.



Andrew Reynolds and Jasmin Christian Blanchette.

A decision procedure for (co)datatypes in SMT solvers.

Journal of Automated Reasoning, 58(3):341–362, 2016.



A. Stump, C. W. Barrett, D. L. Dill, and J. R. Levitt.

A Decision Procedure for an Extensional Theory of Arrays.

In *16th Annual IEEE Symposium on Logic in Computer Science, LICS'01*, pages 29–37.
IEEE Computer Society, 2001.



Ying Sheng, Andres Nötzli, Andrew Reynolds, Yoni Zohar, David L. Dill, Wolfgang Grieskamp, Junkil Park, Shaz Qadeer, Clark W. Barrett, and Cesare Tinelli.

Reasoning about vectors: Satisfiability modulo a theory of sequences.

J. Autom. Reason., 67(3):32, 2023.



C. Wang, F. Ivancic, M. K. Ganai, and A. Gupta.

Deciding Separation Logic Formulae by SAT and Incremental Negative Cycle Elimination.

In G. Sutcliffe and A. Voronkov, editors, *12th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'05*, volume 3835 of *Lecture Notes in Computer Science*, pages 322–336. Springer, 2005.



Harald Zankl and Aart Middeldorp.

Satisfiability of Non-linear (Ir)rational Arithmetic.

In Edmund M. Clarke and Andrei Voronkov, editors, *16th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'10*, volume 6355 of *Lecture Notes in Computer Science*, pages 481–500. Springer, 2010.