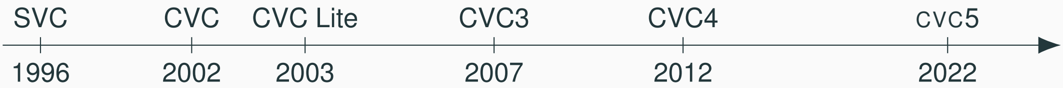
A Versatile and Industrial-Strength SMT Solver

Haniel Barbosa,[1] Clark Barrett,[2] Martin Brain,[3] Gereon Kremer,[2] Hanna Lachnitt,[2] Makai Mann,[2] Abdalrhman Mohamed,[4] Mudathir Mohamed,[4] Aina Niemetz,[2] **Andres Nötzli,**[2] Alex Ozdemir,[2] Mathias Preiner,[2] Andrew Reynolds,[4] Ying Sheng,[2] Cesare Tinelli,[4] Yoni Zohar[5]

[1] Universidade Federal de Minas Gerais, [2] Stanford University, [3] City, University of London, [4] The University of Iowa, [5] Bar-Ilan University

History



Introduction

- Support for *all* standard SMT-LIB and additional non-standard theories
- Beyond SMT solving
  - Proof generation
  - Syntax-Guided Synthesis (SyGuS)
  - Interpolation
  - Abduction

A Versatile and Industrial-Strength SMT Solver

- Extensively used in industry
- Comprehensive, stable API and documentation
- Permissive license

Introduction

- Support for *all* standard SMT-LIB and additional non-standard theories
- Beyond SMT solving
  - Proof generation
  - Syntax-Guided Synthesis (SyGuS)
  - Interpolation
  - Abduction

A Versatile and Industrial-Strength SMT Solver

- Extensively used in industry
- Comprehensive, stable API and documentation
- Permissive license

Introduction

- Support for *all* standard SMT-LIB and additional non-standard theories
- Beyond SMT solving
  - Proof generation
  - Syntax-Guided Synthesis (SyGuS)
  - Interpolation
  - Abduction

A Versatile and Industrial-Strength SMT Solver

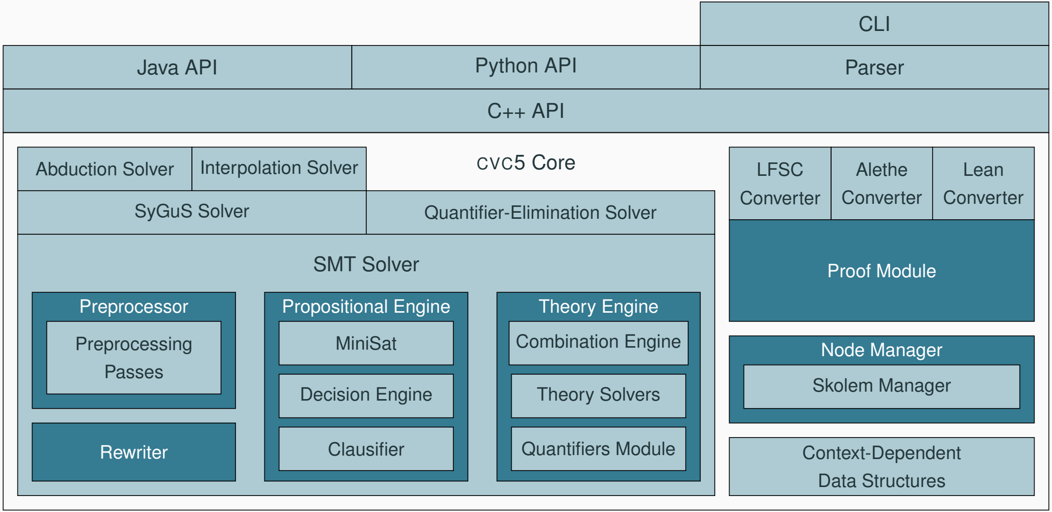
- Extensively used in industry
- Comprehensive, stable API and documentation
- Permissive license

Agenda

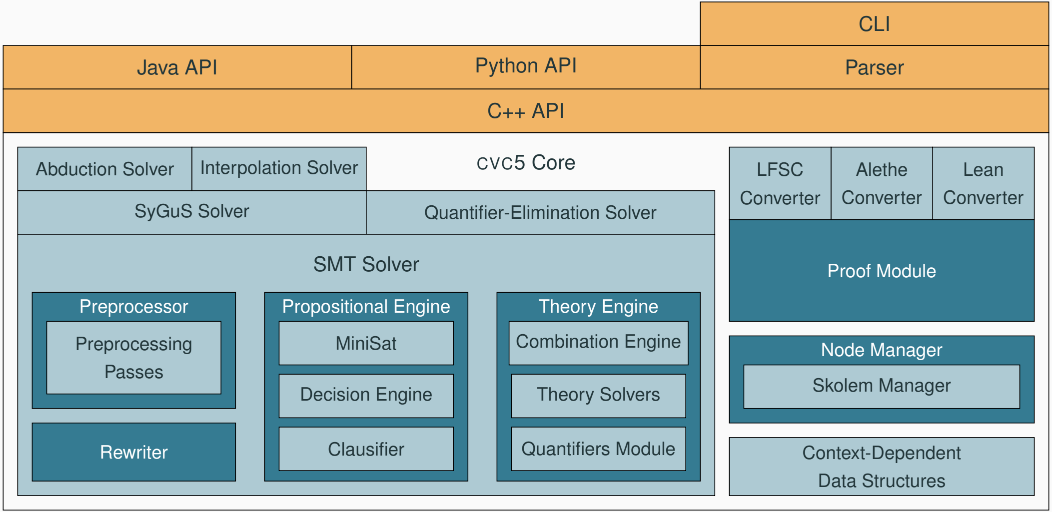
- Architecture
- Feature Highlights
  - New API
  - Proofs
  - SyGuS
  - Interpolation and Abduction
- Evaluation

Architecture

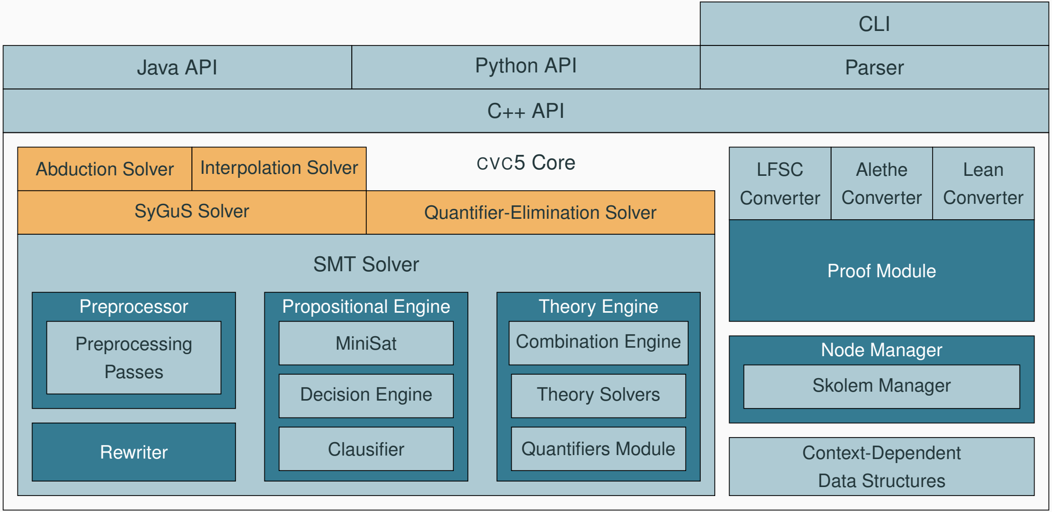
Workflow


CLI
Java API
Python API
Parser
C++ API
Abduction Solver
Interpolation Solver
cvc5 Core
LFSC Converter
Alethe Converter
Lean Converter
SyGuS Solver
Quantifier-Elimination Solver
SMT Solver
Proof Module
Preprocessor
Preprocessing Passes
Propositional Engine
MiniSat
Decision Engine
Clausifier
Theory Engine
Combination Engine
Theory Solvers
Quantifiers Module
Node Manager
Skolem Manager
Rewriter
Context-Dependent Data Structures

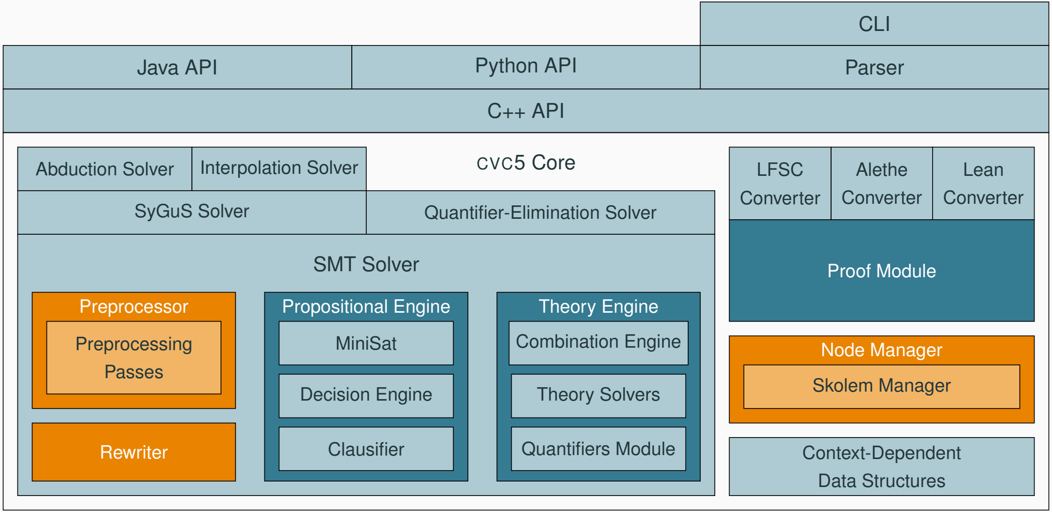
Workflow


CLI
Java API
Python API
Parser
C++ API
Abduction Solver
Interpolation Solver
cvc5 Core
LFSC Converter
Alethe Converter
Lean Converter
SyGuS Solver
Quantifier-Elimination Solver
SMT Solver
Proof Module
Preprocessor
Preprocessing Passes
Propositional Engine
MiniSat
Decision Engine
Clausifier
Theory Engine
Combination Engine
Theory Solvers
Quantifiers Module
Node Manager
Skolem Manager
Rewriter
Context-Dependent Data Structures

Workflow

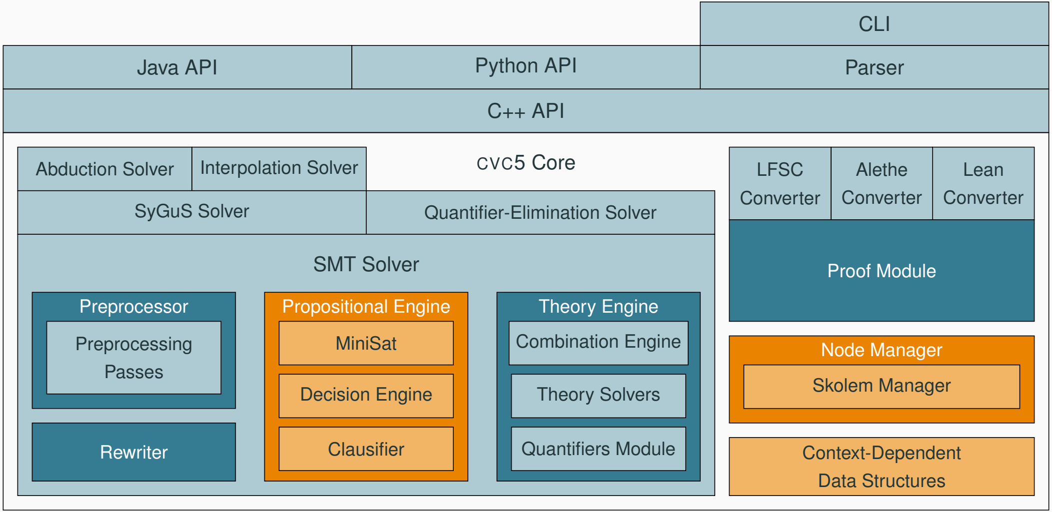

CLI
Java API
Python API
Parser
C++ API
Abduction Solver
Interpolation Solver
cvc5 Core
LFSC Converter
Alethe Converter
Lean Converter
SyGuS Solver
Quantifier-Elimination Solver
SMT Solver
Proof Module
Preprocessor
Preprocessing Passes
Propositional Engine
MiniSat
Decision Engine
Clausifier
Theory Engine
Combination Engine
Theory Solvers
Quantifiers Module
Node Manager
Skolem Manager
Rewriter
Context-Dependent Data Structures

Workflow

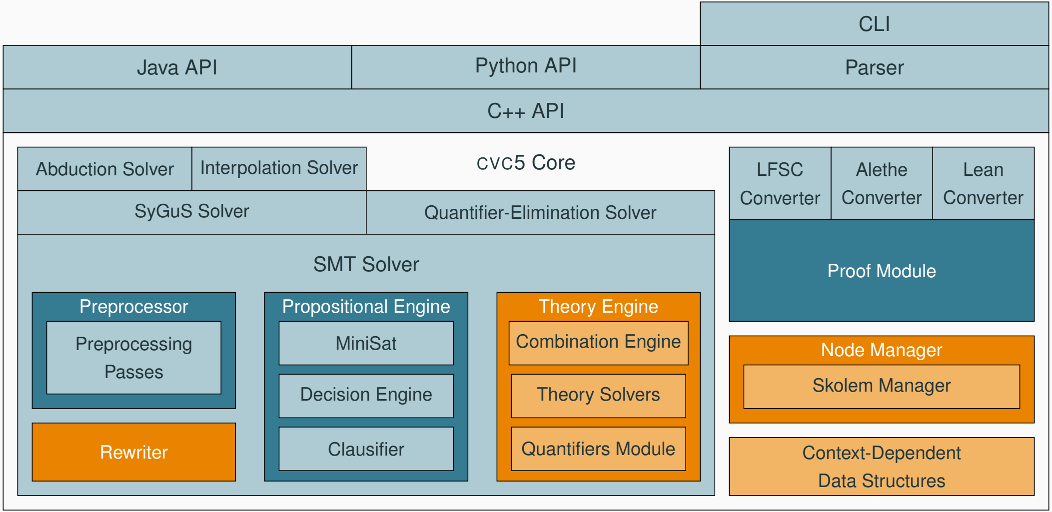


CLI
Java API
Python API
Parser
C++ API
Abduction Solver
Interpolation Solver
cvc5 Core
LFSC Converter
Alethe Converter
Lean Converter
SyGuS Solver
Quantifier-Elimination Solver
SMT Solver
Proof Module
Preprocessor
Preprocessing Passes
Propositional Engine
MiniSat
Decision Engine
Clausifier
Theory Engine
Combination Engine
Theory Solvers
Quantifiers Module
Node Manager
Skolem Manager
Rewriter
Context-Dependent Data Structures

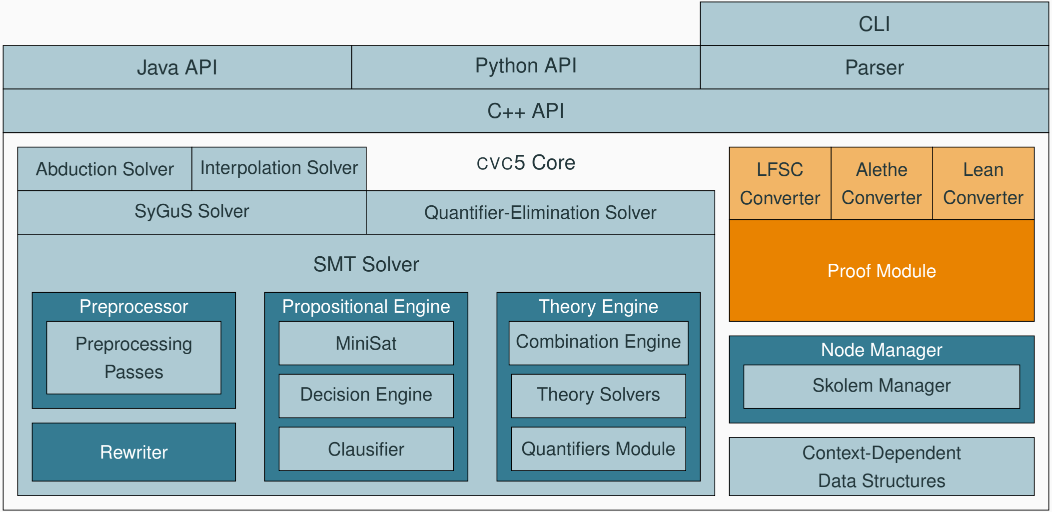
Workflow


CLI
Java API
Python API
Parser
C++ API
Abduction Solver
Interpolation Solver
cvc5 Core
LFSC Converter
Alethe Converter
Lean Converter
SyGuS Solver
Quantifier-Elimination Solver
SMT Solver
Proof Module
Preprocessor
Preprocessing Passes
Propositional Engine
MiniSat
Decision Engine
Clausifier
Theory Engine
Combination Engine
Theory Solvers
Quantifiers Module
Rewriter
Node Manager
Skolem Manager
Context-Dependent Data Structures

Workflow


CLI
Java API
Python API
Parser
C++ API
Abduction Solver
Interpolation Solver
cvc5 Core
LFSC Converter
Alethe Converter
Lean Converter
SyGuS Solver
Quantifier-Elimination Solver
SMT Solver
Proof Module
Preprocessor
Preprocessing Passes
Propositional Engine
MiniSat
Decision Engine
Clausifier
Theory Engine
Combination Engine
Theory Solvers
Quantifiers Module
Node Manager
Skolem Manager
Rewriter
Context-Dependent Data Structures

Workflow


CLI
Java API
Python API
Parser
C++ API
Abduction Solver
Interpolation Solver
cvc5 Core
LFSC Converter
Alethe Converter
Lean Converter
SyGuS Solver
Quantifier-Elimination Solver
SMT Solver
Proof Module
Preprocessor
Preprocessing Passes
Propositional Engine
MiniSat
Decision Engine
Clausifier
Theory Engine
Combination Engine
Theory Solvers
Quantifiers Module
Node Manager
Skolem Manager
Rewriter
Context-Dependent Data Structures

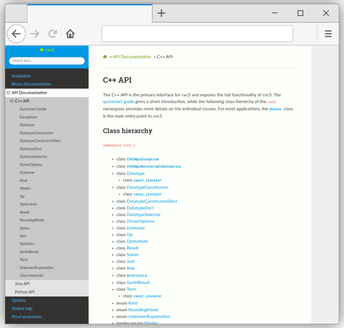
Theory Solvers

- Linear arithmetic [Kin14, KBD13, KBT14]
- **Non-linear arithmetic** [RTJB17]
- Arrays [JB13]
- **Bit-vectors**
- Datatypes [BST07, RB15, RVB+18]
- **Floating-point arithmetic** [BSS19]
- **Sets and relations** [BBRT17, MRTB17]
- Separation logic [RISK16]
- **Strings and sequences** [LRT+14, RWB+17, LTR+15, RNBT19, RNBT20]
- Uninterpreted functions (with support for finite cardinality constraints) [RTGK13]
- **Quantifiers** [RTdM14, BFR17, RTG+13, RBF18, RKK17, NPR+21a, NPR+21b, RK15, RBCT16, RDK+15]

Feature Highlights

Feature Highlights: API

- New C++ API
  - Lean, comprehensive, feature-complete
  - Parser module uses the same API
  - Comprehensive documentation
- Python bindings: 2 variants
  - Base bindings: Complete Cython-based bindings for the API
  - Pythonic bindings: High-level bindings, drop-in replacement for Z3py
- Java bindings
  - Complete JNI-based bindings for the API

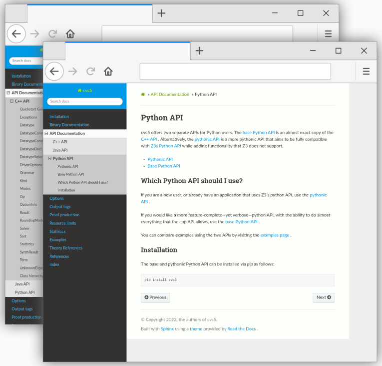


Demo

Solving a simple problem using the Pythonic API

Feature Highlights: API

- New C++ API
  - Lean, comprehensive, feature-complete
  - Parser module uses the same API
  - Comprehensive documentation
- Python bindings: 2 variants
  - Base bindings: Complete Cython-based bindings for the API
  - Pythonic bindings: High-level bindings, drop-in replacement for Z3py
- Java bindings
  - Complete JNI-based bindings for the API

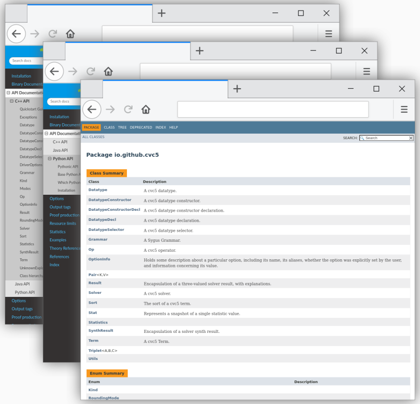


Demo

Solving a simple problem using the Pythonic API

Feature Highlights: API

- New C++ API
  - Lean, comprehensive, feature-complete
  - Parser module uses the same API
  - Comprehensive documentation
- Python bindings: 2 variants
  - Base bindings: Complete Cython-based bindings for the API
  - Pythonic bindings: High-level bindings, drop-in replacement for Z3py
- Java bindings
  - Complete JNI-based bindings for the API

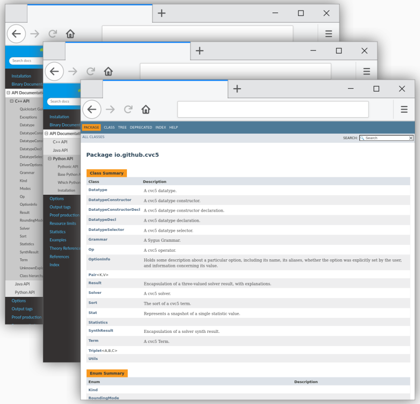


Demo

Solving a simple problem using the Pythonic API

Feature Highlights: API

- New C++ API
  - Lean, comprehensive, feature-complete
  - Parser module uses the same API
  - Comprehensive documentation
- Python bindings: 2 variants
  - Base bindings: Complete Cython-based bindings for the API
  - Pythonic bindings: High-level bindings, drop-in replacement for Z3py
- Java bindings
  - Complete JNI-based bindings for the API

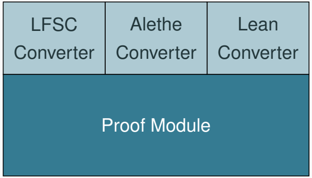


Demo

Solving a simple problem using the Pythonic API

Feature Highlights: Proofs

- New module for producing proofs for unsatisfiable inputs
  - Enables independent checking of answers
  - Automating proofs in interactive theorem provers
- Goals
  - Low overhead
  - Detailed, efficiently checkable proofs
  - Support all performance-critical components
  - Output in different proof formats

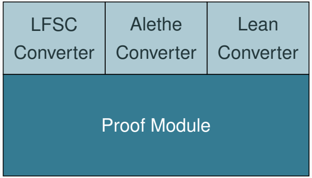


Demo

Generating a proof for a simple problem.

Feature Highlights: Proofs

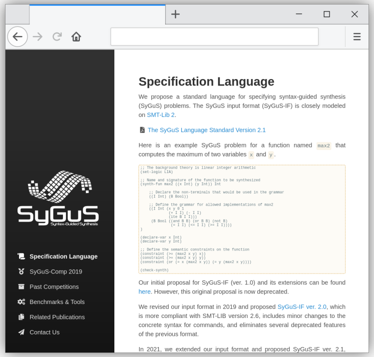
- New module for producing proofs for unsatisfiable inputs
  - Enables independent checking of answers
  - Automating proofs in interactive theorem provers
- Goals
  - Low overhead
  - Detailed, efficiently checkable proofs
  - Support all performance-critical components
  - Output in different proof formats



Demo

Generating a proof for a simple problem.

Feature Highlights: Syntax-Guided Synthesis (SyGuS)



Specification

$$\exists f. \forall x. P(f, x)$$

There exists a function f for which property P holds for all x in some theory T.

Syntax

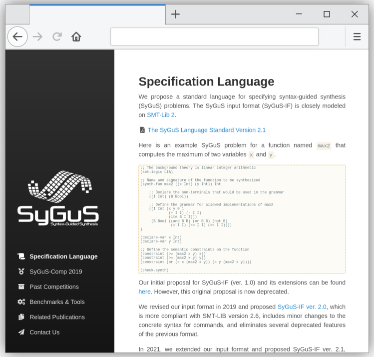
$$A := A + A \mid -A \mid x \mid y \mid 0 \mid 1 \mid \mathrm{ite}(B, A, A)$$

$$B := B \wedge B \mid \neg B \mid A = A \mid A \geqslant A \mid \bot$$

Demo

Flash Fill-style synthesis.

Feature Highlights: Syntax-Guided Synthesis (SyGuS)



Specification

$$\exists f.\forall x.P(f, x)$$

There exists a function f for which property P holds for all x in some theory T.

Syntax

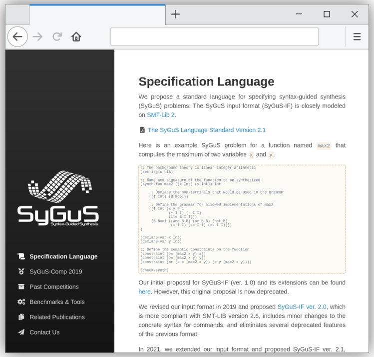
$$A := A + A \mid -A \mid x \mid y \mid 0 \mid 1 \mid \mathrm{ite}(B, A, A)$$

$$B := B \wedge B \mid \neg B \mid A = A \mid A \geqslant A \mid \bot$$

Demo

Flash Fill-style synthesis.

Feature Highlights: Syntax-Guided Synthesis (SyGuS)



SyGuS
Syntax-Guided Synthesis

- Specification Language
- SyGuS-Comp 2019
- Past Competitions
- Benchmarks & Tools
- Related Publications
- Contact Us

Specification Language

We propose a standard language for specifying syntax-guided synthesis (SyGuS) problems. The SyGuS input format (SyGuS-IF) is closely modeled on SMT-Lib 2.

The SyGuS Language Standard Version 2.1

Here is an example SyGuS problem for a function named `max2` that computes the maximum of two variables `x` and `y`.

```
;; The background theory is linear integer arithmetic
(set-logic LIA)

;; Name and signature of the function to be synthesized
(synth-fun max2 ((x Int) (y Int)) Int

    ;; Declare the non-terminals that would be used in the grammar
    ((I Int) (B Bool))

    ;; Define the grammar for allowed implementations of max2
    ((I Int (x y 0 1
             (+ I I) (- I I)
             (ite B I I)))
     (B Bool ((and B B) (or B B) (not B)
              (= I I) (<= I I) (>= I I))))
)

(declare-var x Int)
(declare-var y Int)

;; Define the semantic constraints on the function
(constraint (>= (max2 x y) x))
(constraint (>= (max2 x y) y))
(constraint (or (= x (max2 x y)) (= y (max2 x y))))

(check-synth)
```

Our initial proposal for SyGuS-IF (ver. 1.0) and its extensions can be found here. However, this original proposal is now deprecated.

We revised our input format in 2019 and proposed SyGuS-IF ver. 2.0, which is more compliant with SMT-LIB version 2.6, includes minor changes to the concrete syntax for commands, and eliminates several deprecated features of the previous format.

In 2021, we extended our input format and proposed SyGuS-IF ver. 2.1,

Specification

$$\exists f.\forall x.P(f,x)$$

There exists a function f for which property P holds for all x in some theory T.

Syntax

$$A := A + A \mid -A \mid x \mid y \mid 0 \mid 1 \mid \mathrm{ite}(B, A, A)$$

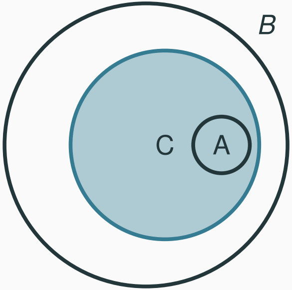
$$B := B \wedge B \mid \neg B \mid A = A \mid A \geqslant A \mid \bot$$

Demo

Flash Fill-style synthesis.

Feature Highlights: Interpolation/Abduction

Interpolation



Find a term C such that $A \models C$ and $C \models B$. Free symbols in C are from set of shared symbols between A and B.

Abduction

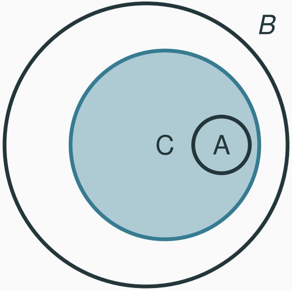
Find a term C such that $A \wedge C$ is satisfiable and $A \wedge C \models B$.

Demo

Fixing a floating-point rewrite using abduction.

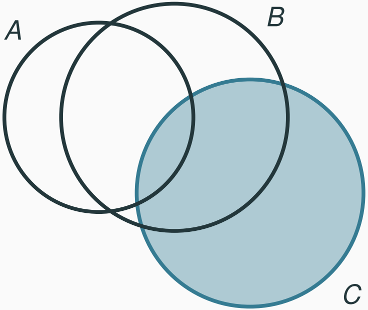
Feature Highlights: Interpolation/Abduction

Interpolation



Find a term C such that $A \models C$ and $C \models B$. Free symbols in C are from set of shared symbols between A and B.

Abduction



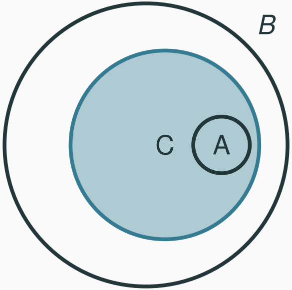
Find a term C such that $A \wedge C$ is satisfiable and $A \wedge C \models B$.

Demo

Fixing a floating-point rewrite using abduction.

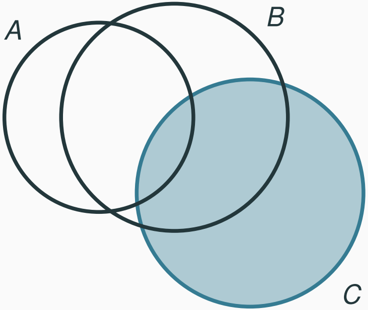
Feature Highlights: Interpolation/Abduction

Interpolation



Find a term C such that $A \models C$ and $C \models B$. Free symbols in C are from set of shared symbols between A and B.

Abduction



Find a term C such that $A \wedge C$ is satisfiable and $A \wedge C \models B$.

Demo

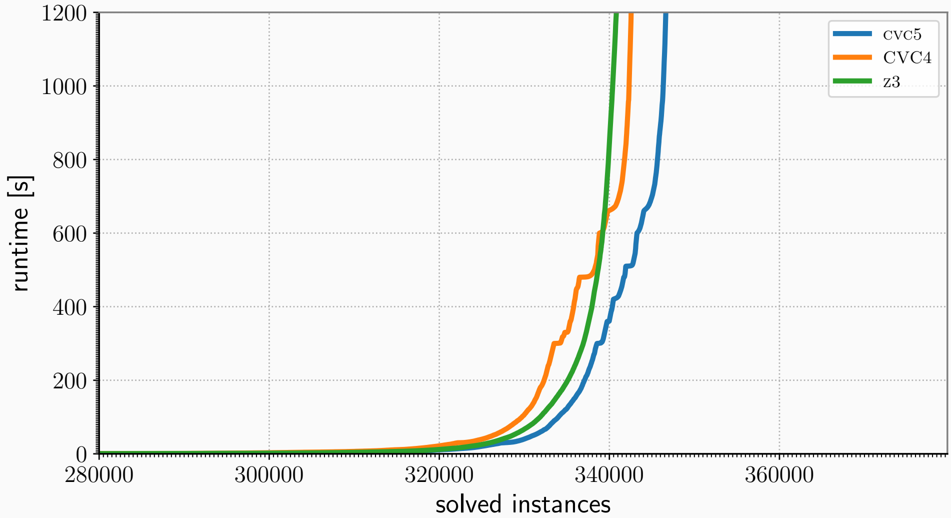
Fixing a floating-point rewrite using abduction.

Evaluation

Evaluation: Setup

- Comparison with CVC4 1.8 and Z3
- Benchmark set: 379,750 non-incremental SMT-LIB benchmarks
  - All logics (quantified and quantifier-free)
  - Excluding 1,173 misclassified benchmarks
- Timeout: 1,200 seconds (like SMT-COMP)

Evaluation: Results



Future Work

- Optimization solver
  - Computing satisfying assignments that optimize objectives
- New theories/extensions of theories
  - Support for higher-order map/fold combinators
- Parallel SMT solving
  - Integrated support for running multiple configurations in parallel/sequence
- Performance tuning
  - Complete replacement of ANTLR parser
  - Lifting local search approach for bit-vectors to floating-point arithmetic

Future Work

- Optimization solver
  - Computing satisfying assignments that optimize objectives
- New theories/extensions of theories
  - Support for higher-order map/fold combinators
- Parallel SMT solving
  - Integrated support for running multiple configurations in parallel/sequence
- Performance tuning
  - Complete replacement of ANTLR parser
  - Lifting local search approach for bit-vectors to floating-point arithmetic

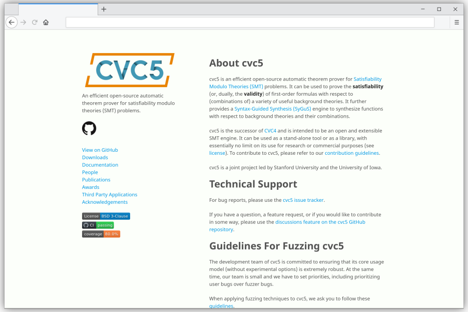
Future Work

- Optimization solver
  - Computing satisfying assignments that optimize objectives
- New theories/extensions of theories
  - Support for higher-order map/fold combinators
- Parallel SMT solving
  - Integrated support for running multiple configurations in parallel/sequence
- Performance tuning
  - Complete replacement of ANTLR parser
  - Lifting local search approach for bit-vectors to floating-point arithmetic

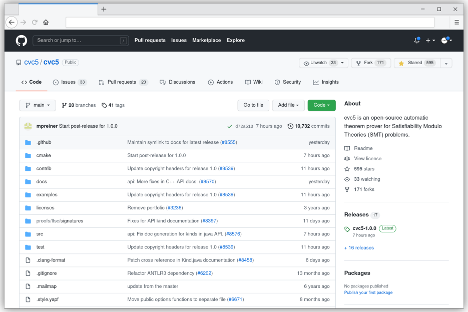
Future Work

- Optimization solver
  - Computing satisfying assignments that optimize objectives
- New theories/extensions of theories
  - Support for higher-order map/fold combinators
- Parallel SMT solving
  - Integrated support for running multiple configurations in parallel/sequence
- Performance tuning
  - Complete replacement of ANTLR parser
  - Lifting local search approach for bit-vectors to floating-point arithmetic

More Information



https://cvc5.github.io/



https://github.com/cvc5/cvc5/

Results

Division	cvc5	CVC4	Z3
Arith (7104)	6593	6498	**6844**
Bitvec (6045)	**5741**	5690	5664
Equality (12159)	6677	**6681**	4688
Equality+LinearArith (55948)	49395	48487	**49503**
Equality+MachineArith (4712)	**2065**	1832	1804
Equality+NonLinearArith (17260)	**11088**	10906	9341
FPArith (3170)	**2625**	2113	2593
QF Bitvec (42450)	**41569**	41448	40582
QF Equality (16254)	**16124**	16121	16115
QF Equality+Bitvec (16518)	16274	**16333**	16318
QF Equality+LinearArith (3924)	3778	3782	**3822**
QF Equality+NonLinearArith (673)	598	610	**616**
QF FPArith (76084)	**75998**	75965	75816
QF LinearIntArith (9765)	8619	**8778**	8464
QF LinearRealArith (2008)	1849	**1881**	1864
QF NonLinearIntArith (24261)	17525	16860	**18357**
QF NonLinearRealArith (11552)	**10889**	9207	10354
QF Strings (69863)	69231	**69367**	68074
Total (379750)	**346638**	342559	340819

Kshitij Bansal, Clark W. Barrett, Andrew Reynolds, and Cesare Tinelli.
A new decision procedure for finite sets and cardinality constraints in SMT.
CoRR, abs/1702.06259, 2017.

Haniel Barbosa, Pascal Fontaine, and Andrew Reynolds.
Congruence closure with free variables.
In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II*, volume 10206 of *Lecture Notes in Computer Science*, pages 214–230, 2017.

Martin Brain, Florian Schanda, and Youcheng Sun.
Building better bit-blasting for floating-point problems.
In *TACAS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I*, volume 11427 of *LNCS*, pages 79–98. Springer, 2019.

Clark Barrett, Igor Shikanian, and Cesare Tinelli.

An abstract decision procedure for a theory of inductive data types.
JSAT, 3(1-2):21–46, 2007.

Dejan Jovanovic and Clark W. Barrett.
Being careful about theory combination.
Formal Methods Syst. Des., 42(1):67–90, 2013.

Tim King, Clark W. Barrett, and Bruno Dutertre.
Simplex with sum of infeasibilities for SMT.
In *Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013*, pages 189–196. IEEE, 2013.

Tim King, Clark W. Barrett, and Cesare Tinelli.
Leveraging linear and mixed integer programming for SMT.
In *Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014*, pages 139–146. IEEE, 2014.

Tim King.

***Effective Algorithms for the Satisfiability of Quantifier-Free Formulas Over Linear Real and Integer Arithmetic*.**
PhD thesis, New York University, 2014.

Tianyi Liang, Andrew Reynolds, Cesare Tinelli, Clark W. Barrett, and Morgan Deters.
A DPLL(T) theory solver for a theory of strings and regular expressions.
In *CAV*, volume 8559 of *Lecture Notes in Computer Science*, pages 646–662. Springer, 2014.

Tianyi Liang, Nestan Tsiskaridze, Andrew Reynolds, Cesare Tinelli, and Clark W. Barrett.
A decision procedure for regular membership and length constraints over unbounded strings.
In *FroCos*, volume 9322 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 2015.

Baoluo Meng, Andrew Reynolds, Cesare Tinelli, and Clark W. Barrett.
Relational constraint solving in SMT.

In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 148–165. Springer, 2017.

Aina Niemetz, Mathias Preiner, Andrew Reynolds, Clark W. Barrett, and Cesare Tinelli.
On solving quantified bit-vector constraints using invertibility conditions.
Formal Methods Syst. Des., 57(1):87–115, 2021.

Aina Niemetz, Mathias Preiner, Andrew Reynolds, Clark W. Barrett, and Cesare Tinelli.
Syntax-guided quantifier instantiation.
In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings,*

Part II, volume 12652 of *Lecture Notes in Computer Science*, pages 145–163. Springer, 2021.

Andrew Reynolds and Jasmin Christian Blanchette.
A decision procedure for (co)datatypes in SMT solvers.
In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 197–213. Springer, 2015.

Andrew Reynolds, Jasmin Christian Blanchette, Simon Cruanes, and Cesare Tinelli.
Model finding for recursive functions in SMT.
In Nicola Olivetti and Ashish Tiwari, editors, *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, volume 9706 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2016.

Andrew Reynolds, Haniel Barbosa, and Pascal Fontaine.

Revisiting enumerative instantiation.
In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II*, volume 10806 of *Lecture Notes in Computer Science*, pages 112–131. Springer, 2018.

Andrew Reynolds, Morgan Deters, Viktor Kuncak, Cesare Tinelli, and Clark W. Barrett.
Counterexample-guided quantifier instantiation for synthesis in SMT.
In Daniel Kroening and Corina S. Pasareanu, editors, *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II*, volume 9207 of *Lecture Notes in Computer Science*, pages 198–216. Springer, 2015.

Andrew Reynolds, Radu Iosif, Cristina Serban, and Tim King.
A decision procedure for separation logic in SMT.

In Cyrille Artho, Axel Legay, and Doron Peled, editors, *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings*, volume 9938 of *Lecture Notes in Computer Science*, pages 244–261, 2016.

Andrew Reynolds and Viktor Kuncak.
Induction for SMT solvers.
In Deepak D'Souza, Akash Lal, and Kim Guldstrand Larsen, editors, *Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015. Proceedings*, volume 8931 of *Lecture Notes in Computer Science*, pages 80–98. Springer, 2015.

Andrew Reynolds, Tim King, and Viktor Kuncak.
Solving quantified linear arithmetic by counterexample-guided instantiation.
Formal Methods Syst. Des., 51(3):500–532, 2017.

Andrew Reynolds, Andres Nötzli, Clark W. Barrett, and Cesare Tinelli.
High-level abstractions for simplifying extended string constraints in SMT.

In *CAV (2)*, volume 11562 of *Lecture Notes in Computer Science*, pages 23–42. Springer, 2019.

Andrew Reynolds, Andres Nötzli, Clark W. Barrett, and Cesare Tinelli.
A decision procedure for string to code point conversion.
In *IJCAR (1)*, volume 12166 of *Lecture Notes in Computer Science*, pages 218–237. Springer, 2020.

Andrew Reynolds, Cesare Tinelli, and Leonardo Mendonça de Moura.
Finding conflicting instances of quantified formulas in SMT.
In *Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014*, pages 195–202. IEEE, 2014.

Andrew Reynolds, Cesare Tinelli, Amit Goel, Sava Krstic, Morgan Deters, and Clark W. Barrett.
Quantifier instantiation techniques for finite model finding in SMT.
In Maria Paola Bonacina, editor, *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013.*

Proceedings, volume 7898 of *Lecture Notes in Computer Science*, pages 377–391. Springer, 2013.

Andrew Reynolds, Cesare Tinelli, Amit Goel, and Sava Krstic.
Finite model finding in SMT.
In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 640–655. Springer, 2013.

Andrew Reynolds, Cesare Tinelli, Dejan Jovanovic, and Clark W. Barrett.
Designing theory solvers with extensions.
In Clare Dixon and Marcelo Finger, editors, *Frontiers of Combining Systems - 11th International Symposium, FroCoS 2017, Brasília, Brazil, September 27-29, 2017, Proceedings*, volume 10483 of *Lecture Notes in Computer Science*, pages 22–40. Springer, 2017.

Andrew Reynolds, Arjun Viswanathan, Haniel Barbosa, Cesare Tinelli, and Clark W. Barrett.
Datatypes with shared selectors.
In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, volume 10900 of *Lecture Notes in Computer Science*, pages 591–608. Springer, 2018.

Andrew Reynolds, Maverick Woo, Clark W. Barrett, David Brumley, Tianyi Liang, and Cesare Tinelli.
Scaling up DPLL(T) string solvers using context-dependent simplification.
In *CAV (2)*, volume 10427 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2017.