Congruence Closure with Free Variables

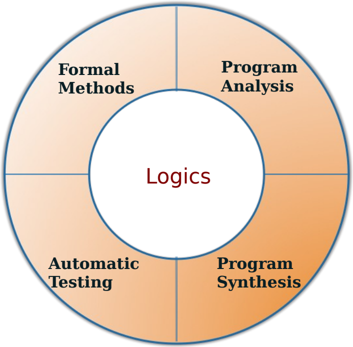
Haniel Barbosa[1] Pascal Fontaine[1] Andrew Reynolds[2]

[1]University of Lorraine, CNRS, Inria, LORIA, Nancy, France
[2]University of Iowa, Iowa City, U.S.A.

TACAS 2017

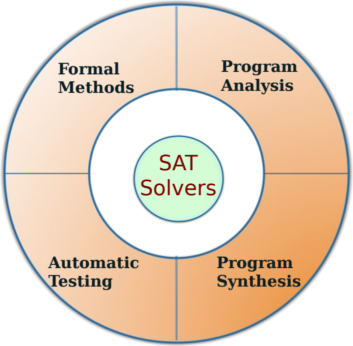
2017-04-28

SMT solvers are successfully used in a variety of applications, including many verification tools



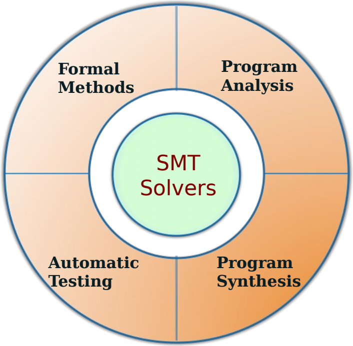
Picture credit: Vijay Ganesh

SMT solvers are successfully used in a variety of applications, including many verification tools



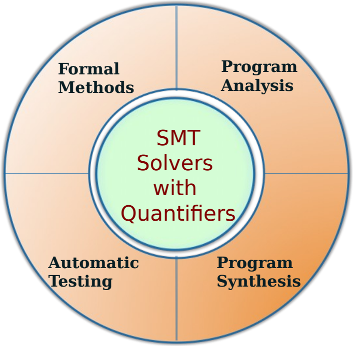
Picture credit: Vijay Ganesh

SMT solvers are successfully used in a variety of applications, including many verification tools



Picture credit: Vijay Ganesh

SMT solvers are successfully used in a variety of applications, including many verification tools



Picture credit: Vijay Ganesh

Quantifiers in SMT solvers

Quantifiers primarily handled with heuristic instantiation

Quantifiers in SMT solvers

Quantifiers primarily handled with heuristic instantiation

- ⊖ Too many instances swamp solver

Quantifiers in SMT solvers

Quantifiers primarily handled with heuristic instantiation

- ⊖ Too many instances swamp solver

Ex.: $\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)$

- ▶ Select patterns $\{f(x),\ h(y),\ f(z)\}$ or $\{f(x),\ h(y),\ g(z)\}$

Quantifiers in SMT solvers

Quantifiers primarily handled with heuristic instantiation

⊖ Too many instances swamp solver

Ex.: $\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)$

- Select patterns $\{f(x),\ h(y),\ f(z)\}$ or $\{f(x),\ h(y),\ g(z)\}$
- A ground model with 10^2 ground each applications for $f,\ g,\ h$ leads to up to 10^6 instantiations

Quantifiers in SMT solvers

Quantifiers primarily handled with heuristic instantiation

- ⊖ Too many instances swamp solver
- ⊖ Butterfly effect

Ex.: $\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)$

- ▶ Select patterns $\{f(x),\, h(y),\, f(z)\}$ or $\{f(x),\, h(y),\, g(z)\}$
- ▶ A ground model with 10^2 ground each applications for f, g, h leads to up to 10^6 instantiations

Quantifiers in SMT solvers

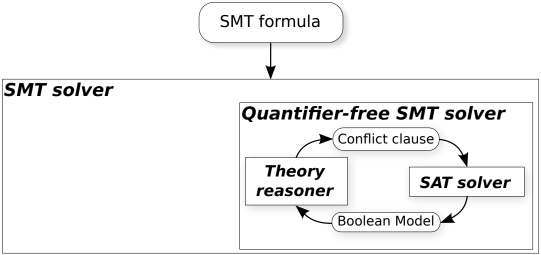
~~Quantifiers primarily handled with heuristic instantiation~~
Fast semantically guided instantiation techniques

- ⊖ ~~Too many instances swamp solver~~ **Fewer, necessary instances**
- ⊖ ~~Butterfly effect~~ **Reduce dependency on heuristics**

Ex.: $\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)$

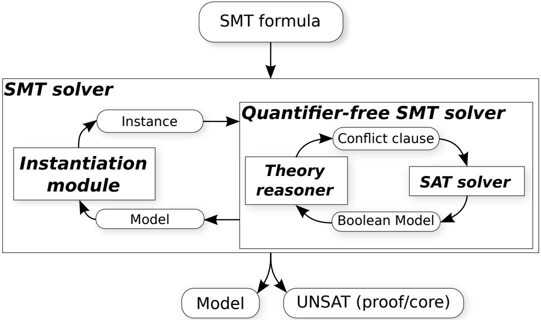
- ~~Select patterns $\{f(x), h(y), f(z)\}$ or $\{f(x), h(y), g(z)\}$~~
- ~~A ground model with 10^2 ground each applications for f, g, h leads to up to 10^6 instantiations~~
- **Derive instantiations that refute ground model**

Problem statement



- Quantifier-free solver enumerates models $E \cup \mathcal{Q}$
  - E is a conjunctive set of ground literals
  - $\mathcal{Q}$ is a conjunctive set of quantified clauses

Problem statement



- ▷ Quantifier-free solver enumerates models $E \cup \mathcal{Q}$
  - ▶ E is a conjunctive set of ground literals
  - ▶ $\mathcal{Q}$ is a conjunctive set of quantified clauses
- ▷ Instantiation module generates instances from $\mathcal{Q}$ and adds them to E

Heuristic instantiation

Pattern-matching of terms from $\mathcal{Q}$ into terms of E

No consistency check of $E \cup \mathcal{Q}$

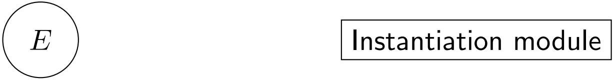
- ⊖ Fast, but too many instances

Heuristic instantiation

Pattern-matching of terms from $\mathcal{Q}$ into terms of E

No consistency check of $E \cup \mathcal{Q}$

- ⊖ Fast, but too many instances

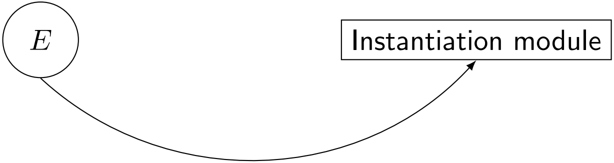


Heuristic instantiation

Pattern-matching of terms from $\mathcal{Q}$ into terms of E

No consistency check of $E \cup \mathcal{Q}$

- ⊖ Fast, but too many instances

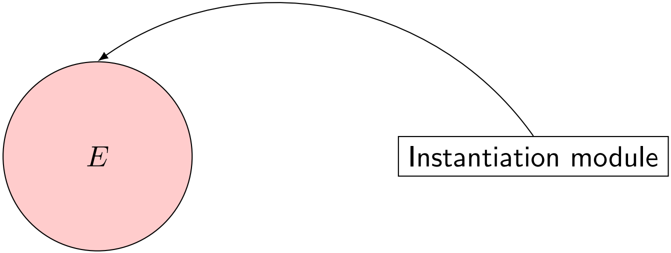


Heuristic instantiation

Pattern-matching of terms from $\mathcal{Q}$ into terms of E

No consistency check of $E \cup \mathcal{Q}$

- ⊖ Fast, but too many instances

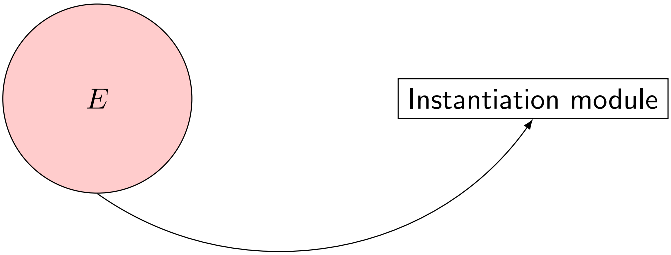


Heuristic instantiation

Pattern-matching of terms from $\mathcal{Q}$ into terms of E

No consistency check of $E \cup \mathcal{Q}$

- ⊖ Fast, but too many instances

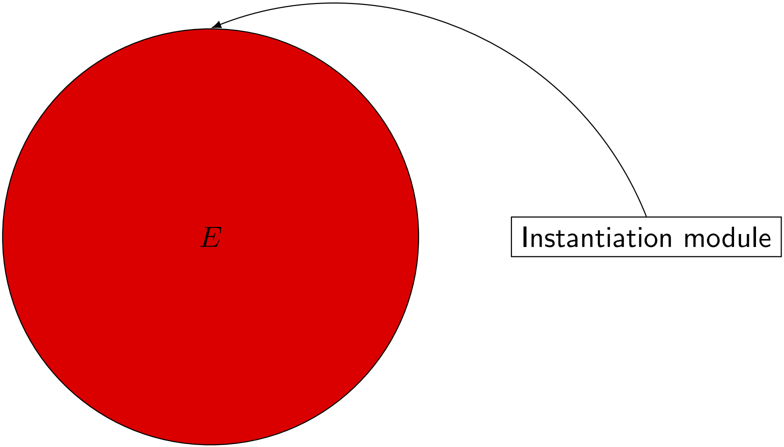


Heuristic instantiation

Pattern-matching of terms from $\mathcal{Q}$ into terms of E

No consistency check of $E \cup \mathcal{Q}$

- ⊖ Fast, but too many instances

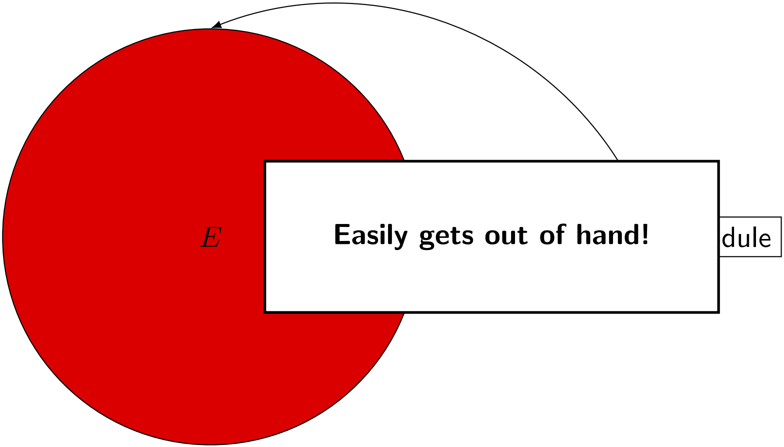


Heuristic instantiation

Pattern-matching of terms from $\mathcal{Q}$ into terms of E

No consistency check of $E \cup \mathcal{Q}$

- ⊖ Fast, but too many instances



Goal-oriented instantiation

Check consistency of $E \cup \mathcal{Q}$

- $\oplus$ Only instances refuting the current model are generated

Goal-oriented instantiation

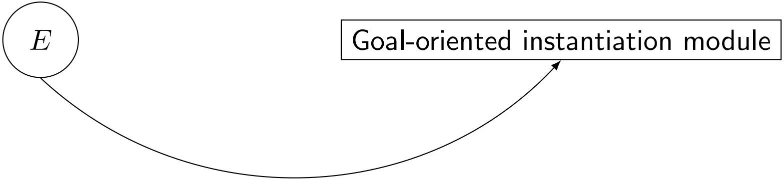
Check consistency of $E \cup \mathcal{Q}$

- ⊕ Only instances refuting the current model are generated

Goal-oriented instantiation

Check consistency of $E \cup \mathcal{Q}$

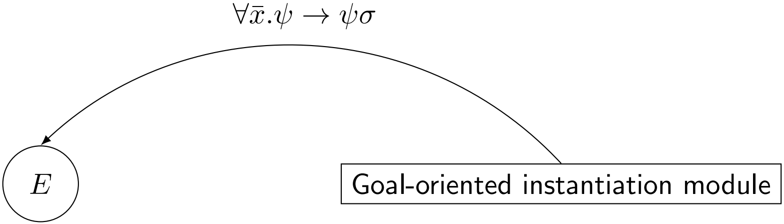
- $\oplus$ Only instances refuting the current model are generated



Goal-oriented instantiation

Check consistency of $E \cup \mathcal{Q}$

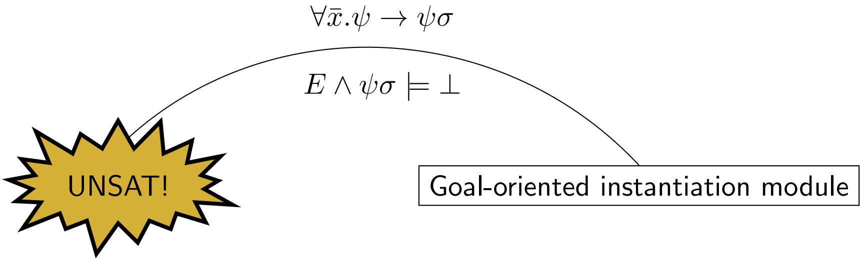
- $\oplus$ Only instances refuting the current model are generated



Goal-oriented instantiation

Check consistency of $E \cup \mathcal{Q}$

⊕ Only instances refuting the current model are generated



Previous work

Conflict-based instantiation [RTM14]

- ▷ Given a model $E \cup \mathcal{Q}$, for some $\forall \bar{x}.\ \psi \in \mathcal{Q}$ find σ s.t. $E \wedge \psi\sigma \models \bot$
- ▷ Add instance $\forall \bar{x}.\ \psi \rightarrow \psi\sigma$ to quantifier-free solver

Finding conflicting instances requires deriving σ s.t.

$$E \models \neg\psi\sigma$$

- ⊕ Goal-oriented
- ⊕ Efficient
- ⊖ Ad-hoc
- ⊖ Incomplete

Let's look deeper into the problem

$E \models \neg\psi\sigma$, for some $\forall\bar{x}.\ \psi \in \mathcal{Q}$

Let's look deeper into the problem

$$E \models \neg\psi\sigma, \text{ for some } \forall\bar{x}.\ \psi \in \mathcal{Q}$$

$$E = \{f(a) \simeq f(b),\, g(b) \not\simeq h(c)\},\ \mathcal{Q} = \{\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)\}$$

Let's look deeper into the problem

$$E \models \neg\psi\sigma, \text{ for some } \forall\bar{x}.\ \psi \in \mathcal{Q}$$

$$E = \{f(a) \simeq f(b),\, g(b) \not\simeq h(c)\},\ \mathcal{Q} = \{\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)\}$$

$$f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) \models (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma$$

Let's look deeper into the problem

$$E \models \neg\psi\sigma, \text{ for some } \forall\bar{x}.\ \psi \in \mathcal{Q}$$

$$E = \{f(a) \simeq f(b),\, g(b) \not\simeq h(c)\},\ \mathcal{Q} = \{\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)\}$$

$$f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) \models (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma$$

- Each literal in the right hand side delimits possible σ

Let's look deeper into the problem

$$E \models \neg\psi\sigma, \text{ for some } \forall\bar{x}.\ \psi \in \mathcal{Q}$$

$$E = \{f(a) \simeq f(b),\ g(b) \not\simeq h(c)\},\ \mathcal{Q} = \{\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)\}$$

$$f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) \models (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma$$

- ▷ Each literal in the right hand side delimits possible σ
  - ▶ $f(x) \simeq f(z)$: either $x \simeq z$ or $x \simeq a \wedge z \simeq b$ or $x \simeq b \wedge z \simeq a$

Let's look deeper into the problem

$$E \models \neg\psi\sigma, \text{ for some } \forall\bar{x}.\ \psi \in \mathcal{Q}$$

$$E = \{f(a) \simeq f(b),\, g(b) \not\simeq h(c)\},\, \mathcal{Q} = \{\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)\}$$

$$f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) \models (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma$$

- Each literal in the right hand side delimits possible σ
  - $f(x) \simeq f(z)$: either $x \simeq z$ or $x \simeq a \wedge z \simeq b$ or $x \simeq b \wedge z \simeq a$
  - $h(y) \not\simeq g(z)$: $y \simeq c \wedge z \simeq b$

Let's look deeper into the problem

$$E \models \neg\psi\sigma, \text{ for some } \forall\bar{x}.\ \psi \in \mathcal{Q}$$

$$E = \{f(a) \simeq f(b),\, g(b) \not\simeq h(c)\},\ \mathcal{Q} = \{\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)\}$$

$$f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) \models (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma$$

- ▷ Each literal in the right hand side delimits possible σ
  - ▶ $f(x) \simeq f(z)$: either $\underline{x \simeq z}$ or $x \simeq a \wedge z \simeq b$ or $x \simeq b \wedge z \simeq a$
  - ▶ $h(y) \not\simeq g(z)$: $\underline{y \simeq c \wedge z \simeq b}$

$$\sigma = \{x \mapsto b,\, y \mapsto c,\, z \mapsto b\}$$

Let's look deeper into the problem

$$E \models \neg\psi\sigma, \text{ for some } \forall\bar{x}.\ \psi \in \mathcal{Q}$$

$$E = \{f(a) \simeq f(b),\, g(b) \not\simeq h(c)\},\ \mathcal{Q} = \{\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)\}$$

$$f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) \models (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma$$

- ▷ Each literal in the right hand side delimits possible σ
  - $f(x) \simeq f(z)$: either $x \simeq z$ or $\underline{x \simeq a \wedge z \simeq b}$ or $x \simeq b \wedge z \simeq a$
  - $h(y) \not\simeq g(z)$: $\underline{y \simeq c \wedge z \simeq b}$

$$\sigma = \{x \mapsto b,\, y \mapsto c,\, z \mapsto b\}$$

or

$$\sigma = \{x \mapsto a,\, y \mapsto c,\, z \mapsto b\}$$

Let's look deeper into the problem

$$E \models \neg\psi\sigma, \text{ for some } \forall\bar{x}.\ \psi \in \mathcal{Q}$$

$$E = \{f(a) \simeq f(b),\ g(b) \not\simeq h(c)\},\ \mathcal{Q} = \{\forall xyz.\ f(x) \simeq f(z) \rightarrow h(y) \simeq g(z)\}$$

$$f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) \models (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma$$

- ▷ Each literal in the right hand side delimits possible σ
  - $f(x) \simeq f(z)$: either $x \simeq z$ or $x \simeq a \wedge z \simeq b$ or $\underline{x \simeq b \wedge z \simeq a}$
  - $h(y) \not\simeq g(z)$: $\underline{y \simeq c \wedge z \simeq b}$

$$\sigma = \{x \mapsto b,\ y \mapsto c,\ z \mapsto b\}$$

or

$$\sigma = \{x \mapsto a,\ y \mapsto c,\ z \mapsto b\}$$

E-ground (dis)unification

Given conjunctive sets of equality literals E and L, with E ground, finding a substitution σ s.t. $E \models L\sigma$

E-ground (dis)unification

Given conjunctive sets of equality literals E and L, with E ground, finding a substitution σ s.t. $E \models L\sigma$

- $\triangleright$ Variant of classic (non-simultaneous) rigid E-unification

E-ground (dis)unification

Given conjunctive sets of equality literals E and L, with E ground, finding a substitution σ s.t. $E \models L\sigma$

- ▷ Variant of classic (non-simultaneous) rigid E-unification

- ▷ NP-complete

  NP: Solutions can be restricted to ground terms in $E \cup L$
  NP-hard: reduction of 3-SAT

Congruence Closure with Free Variables (CCFV)

CCFV is a sound, complete and terminating calculus for solving E-ground (dis)unification

Congruence Closure with Free Variables (CCFV)

CCFV is a sound, complete and terminating calculus for solving E-ground (dis)unification

- $\oplus$ Goal-oriented
- $\oplus$ **(More)** Efficient

Congruence Closure with Free Variables (CCFV)

CCFV is a sound, complete and terminating calculus for solving E-ground (dis)unification

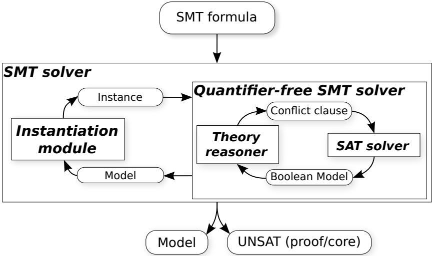
- ⊕ Goal-oriented
- ⊕ **(More)** Efficient
- ⊖ ~~Ad-hoc~~ **Versatile framework, recasting many instantiation techniques as a CCFV problem**
- ⊖ ~~Incomplete~~ **Finds all conflicting instances of a quantified formula**

Existing techniques as special cases

- ▷ Conflict-based instantiation [RTM14]
  - ⊕ CCFV provides formal guarantees and more clear extensions
- ▷ E-matching based heuristic instantiation [DNS05; MB07]
  - ⊕ CCFV allows to easily discard instances already entailed by E
- ▷ Model-based instantiation [GM09; RTG+13]
  - ⊕ No need for a secondary ground SMT solver
  - ⊕ No need to guess solutions

Towards a theory solver for instantiation

- Model generation
- **Conflict set generation**
- **Propagation**
- Incrementality



Congruence Closure with Free Variables

Finding solutions σ for $E \models L\sigma$

- ▷ Search for solutions as a series of AND-OR constraints depending on the entailment of conditions of literals in L

Finding solutions σ for $E \models L\sigma$

- ▷ Search for solutions as a series of AND-OR constraints depending on the entailment of conditions of literals in L

- ▷ Congruence closure as a core element
  - ▶ All terms inferred equal are kept in the same class
  - ▶ Constraints to be entailed are normalized according to partial solutions

Finding solutions σ for $E \models L\sigma$

- Search for solutions as a series of AND-OR constraints depending on the entailment of conditions of literals in L

- Congruence closure as a core element
  - All terms inferred equal are kept in the same class
  - Constraints to be entailed are normalized according to partial solutions

- Different possibilities for building solutions are handled with branching and backtracking

$$
\begin{array}{rcl}
E & \models & L\sigma \\
f(a) \simeq f(b) \land g(b) \not\simeq h(c) & \models & (f(x) \simeq f(z) \land h(y) \not\simeq g(z))\,\sigma
\end{array}
$$

$$\begin{array}{rcl} E & \models & L\sigma \\ f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) & \models & (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma \end{array}$$

$$f(x) \simeq f(z) \wedge h(y) \not\simeq g(z)$$

$$E \models L\sigma$$

$$f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) \models (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma$$

$$f(x) \simeq f(z) \wedge h(y) \not\simeq g(z)$$

$$\varnothing \;\Big|$$

$$y \simeq c \wedge z \simeq b \wedge f(x) \simeq f(z)$$

$$E \models L\sigma$$

$$f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) \models (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma$$

$$f(x) \simeq f(z) \wedge h(y) \not\simeq g(z)$$

$$\varnothing$$

$$y \simeq c \wedge z \simeq b \wedge f(x) \simeq f(z)$$

$$y \simeq c$$

$$z \simeq b \wedge f(x) \simeq f(z)$$

$$
\begin{array}{rcl}
E & \models & L\sigma \\
f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) & \models & (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma
\end{array}
$$

$$f(x) \simeq f(z) \wedge h(y) \not\simeq g(z)$$

$$\emptyset$$

$$y \simeq c \wedge z \simeq b \wedge f(x) \simeq f(z)$$

$$y \simeq c$$

$$z \simeq b \wedge f(x) \simeq f(z)$$

$$y \simeq c,\; z \simeq b$$

$$f(x) \simeq f(b)$$

$$E \models L\sigma$$

$$f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) \models (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma$$

$$f(x) \simeq f(z) \wedge h(y) \not\simeq g(z)$$

$\varnothing$

$$y \simeq c \wedge z \simeq b \wedge f(x) \simeq f(z)$$

$y \simeq c$

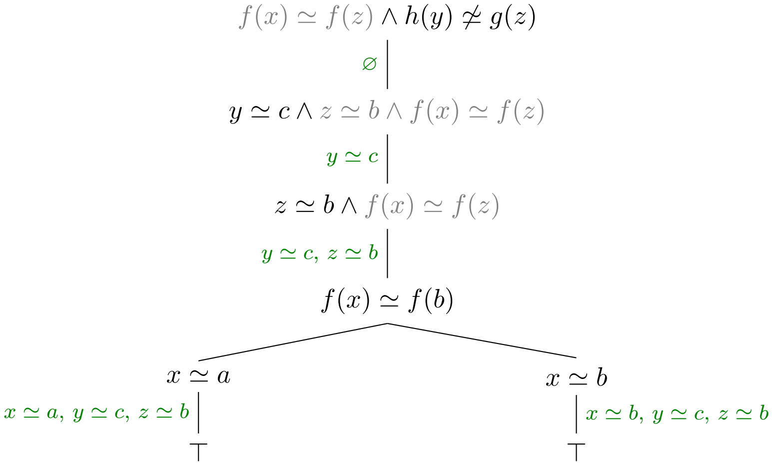
$$z \simeq b \wedge f(x) \simeq f(z)$$

$y \simeq c,\ z \simeq b$

$$f(x) \simeq f(b)$$

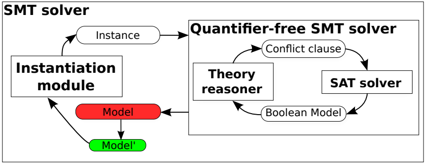
$x \simeq a$ | $x \simeq b$

$$
\begin{array}{rcl}
E & \models & L\sigma \\
f(a) \simeq f(b) \wedge g(b) \not\simeq h(c) & \models & (f(x) \simeq f(z) \wedge h(y) \not\simeq g(z))\,\sigma
\end{array}
$$



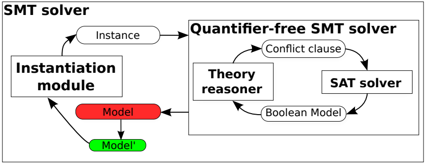
Implementation

- Model minimisation



Implementation

- ▷ Model minimisation

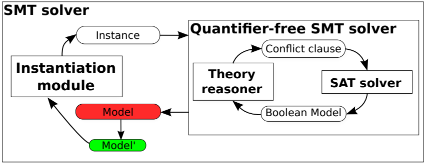


- ▷ Top symbol indexing of E-graph from ground congruence closure

$$E \models f(x)\sigma \simeq t \text{ only if } [t] \text{ contains some } f(t')$$

Implementation

▷ Model minimisation



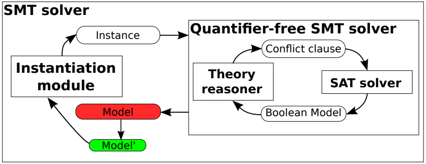
▷ Top symbol indexing of E-graph from ground congruence closure

$$E \models f(x)\sigma \simeq t \text{ only if } [t] \text{ contains some } f(t')$$

$$f \to \begin{cases} f([t_1], \ldots, [t_n]) \\ \ldots \\ f([t'_1], \ldots, [t'_n]) \end{cases}$$

Implementation

▷ Model minimisation



▷ Top symbol indexing of E-graph from ground congruence closure

$$E \models f(x)\sigma \simeq t \text{ only if } [t] \text{ contains some } f(t')$$

$$f \rightarrow \begin{cases} f([t_1], \ldots, [t_n]) \\ \ldots \\ f([t'_1], \ldots, [t'_n]) \end{cases}$$

- Bitsets for fast checking if a symbol has applications in a congruence class

Implementation

- Selection strategies

$$E \models f(x, y) \simeq h(z) \wedge x \simeq t \wedge C$$

Implementation

- Selection strategies

$$E \models f(x, y) \simeq h(z) \wedge \underline{x \simeq t} \wedge C$$

Implementation

- Selection strategies

$$E \models f(x,y) \simeq h(z) \wedge \underline{x \simeq t} \wedge C$$

- Eagerly checking whether constraints can be discarded
  - After assigning x to t, the remaining problem is normalized

$$E \models f(t,y) \simeq h(z) \wedge C$$

Implementation

- ▷ Selection strategies

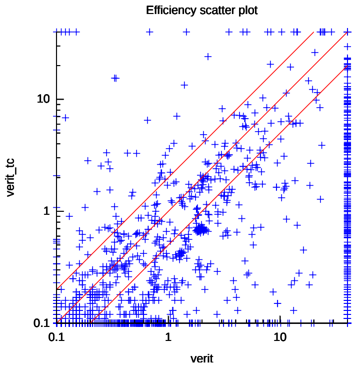
$$E \models f(x, y) \simeq h(z) \wedge \underline{x \simeq t} \wedge C$$

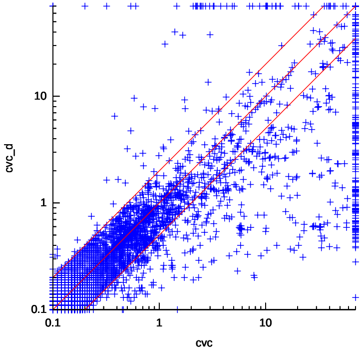
- ▷ Eagerly checking whether constraints can be discarded
  - ▶ After assigning x to t, the remaining problem is normalized

$$E \models f(t, y) \simeq h(z) \wedge C$$

  - ▶ $E \models f(t, y)\sigma \simeq h(z)\sigma$ only if there is some $f(t', t'')$ s.t.

$$E \models t \simeq t'$$





veriT: + 800 out of $1\,785$ unsolved problems

CVC4: + 200 out of 745 unsolved problems

* experiments in the "UF", "UFLIA", "UFLRA" and "UFIDL" categories of SMT-LIB, which have $10\,495$ benchmarks annotated as unsatisfiable, with 30s timeout.

Conclusions and future work

- ▷ A unifying framework for quantified formulas with equality and uninterpreted functions
- ▷ Lifting congruence closure to accommodate free variables
- ▷ Efficient implementations in the SMT solvers CVC4 and veriT

Conclusions and future work

- A unifying framework for quantified formulas with equality and uninterpreted functions
- Lifting congruence closure to accommodate free variables
- Efficient implementations in the SMT solvers CVC4 and veriT

Extensions

- Finding conflicting instances across multiple quantified formulas

$$E \models \neg\psi_1\sigma \vee \cdots \vee \neg\psi_n\sigma, \quad \forall \bar{x}.\ \psi \in \mathcal{Q}$$

- Incrementality
- Learning-based search for solutions
- Beyond theory of equality
- Handle variables in E

Thank you

References

David Detlefs, Greg Nelson, and James B. Saxe. "Simplify: A Theorem Prover for Program Checking". In: J. ACM 52.3 (2005), pp. 365–473.

Yeting Ge and Leonardo de Moura. "Complete Instantiation for Quantified Formulas in Satisfiabiliby Modulo Theories". In: Computer Aided Verification (CAV). Ed. by Ahmed Bouajjani and Oded Maler. Vol. 5643. Lecture Notes in Computer Science. Springer, 2009, pp. 306–320.

Leonardo de Moura and Nikolaj Bjørner. "Efficient E-Matching for SMT Solvers". In: Proc. Conference on Automated Deduction (CADE). Ed. by Frank Pfenning. Vol. 4603. Lecture Notes in Computer Science. Springer, 2007, pp. 183–198.

Andrew Reynolds, Cesare Tinelli, Amit Goel, Sava Krsti, Morgan Deters, and Clark Barrett. "Quantifier Instantiation Techniques for Finite Model Finding in SMT". In: Proc. Conference on Automated Deduction (CADE). Ed. by Maria Paola Bonacina. Vol. 7898. Lecture Notes in Computer Science. Springer, 2013, pp. 377–391.

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). IEEE, 2014, pp. 195–202.