Congruence Closure with Free Variables: A quantifier-instantiation framework for SMT

Haniel Barbosa

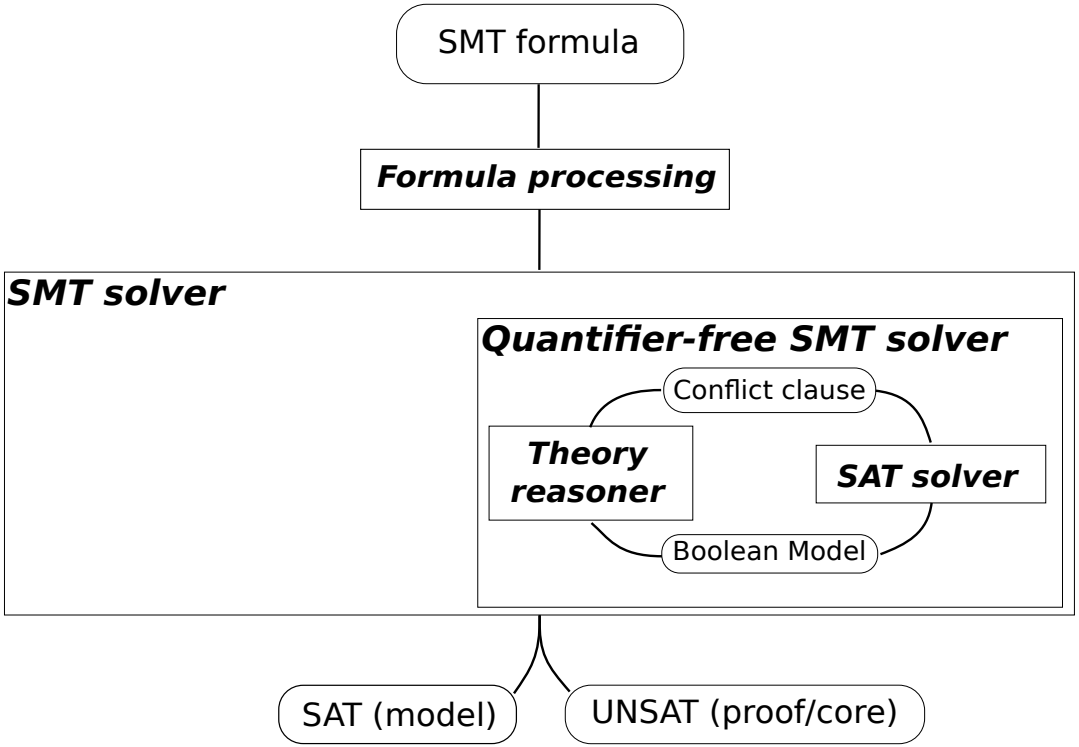
SRI Formal Topics Seminar

2020-07-16, The Internet

Overview

- Instantiation in SMT
- Congruence closure with free variables
  - E-ground (dis)unification
  - Casting instantiations techniques
  - Decision procedure
  - Implementation
  - Evaluation
- Issues and extensions
  - WIP towards HOSMT

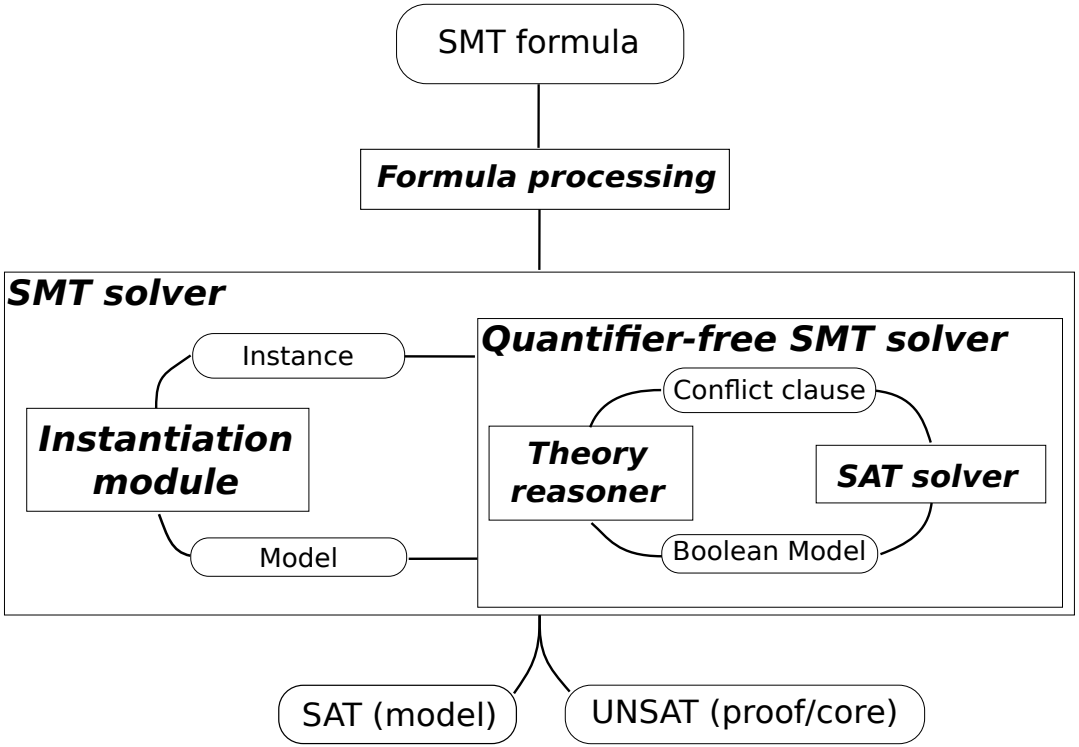
CDCL(T) architecture



Quantifier-free solver enumerates models E

- E is a set of ground literals $\{a \leq b,\ b \leq a + x,\ x \simeq 0,\ f(a) \not\simeq f(b)\}$

CDCL(T) architecture



Quantifier-free solver enumerates models $E \cup \mathcal{Q}$

- E is a set of ground literals $\{a \leq b,\ b \leq a + x,\ x \simeq 0,\ f(a) \not\simeq f(b)\}$
- $\mathcal{Q}$ is a set of quantified clauses $\{\forall xyz.\ f(x) \not\simeq f(z) \vee g(y) \simeq h(z)\}$

Instantiation module generates instances of $\mathcal{Q}$ $f(a) \not\simeq f(b) \vee g(a) \simeq h(b)$

Instantiation techniques

- ▷ Trigger-based [DNS05; MB07]
- ▷ Conflict-based [RTM14; BFR17]
- ▷ Model-based [GM09; RTG+13]

- ⊕ General: ∀+EUF+...
- ⊖ Finding instantiations is hard

- ▷ Enumerative [RBF18]
- ⊕ Easy to implement
- ⊕ Reliable last resort

- ▷ QE-based [Mon10; Bjø10; RDK+15; BJ15]
- ⊕ Decision procedures available
- ⊖ Pure fragments

Instantiation techniques

- ▷ Trigger-based [DNS05; MB07]
- ▷ Conflict-based [RTM14; BFR17]
- ▷ Model-based [GM09; RTG+13]

- ⊕ General: ∀+EUF+...
- ⊖ Finding instantiations is hard

- ▷ Enumerative [RBF18]
  - ⊕ Easy to implement
  - ⊕ Reliable last resort

- ▷ QE-based [Mon10; Bjø10; RDK+15; BJ15]
  - ⊕ Decision procedures available
  - ⊖ Pure fragments

CCFV is a unifying framework for trigger-, conflict-, and model-based instantiation

A bit of history

Dec. 2013: "Make veriT great on quantifiers. Probably try superposition."

After much suffering acting like a resolution prover...

"A resolution prover is like a prolific but not very well organized mathematician filling notebooks with trivial deductions, with no overall sense of where he is going. Once in a while he stumbles on something interesting." - David A. Plaisted [Pla15]

A bit of history

Dec. 2013: "Make veriT great on quantifiers. Probably try superposition."

Conflict-based instantiation [RTM14]

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

Finding conflicting instances requires deriving σ s.t. $E \models_T \neg\psi\sigma$

- ⊕ Goal-oriented instantiation technique
- ⊕ Efficient
- ⊖ Specialized solution
- ⊖ Incomplete matching

Let's look deeper into the problem (with $T = \mathrm{EUF}$)

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

Let's look deeper into the problem (with $T = \mathrm{EUF}$)

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

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

Let's look deeper into the problem (with $T = \mathrm{EUF}$)

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

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

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

Let's look deeper into the problem (with $T = \mathrm{EUF}$)

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

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

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

- Each literal in the right hand side delimits possible σ

Let's look deeper into the problem (with $T = \mathrm{EUF}$)

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

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

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

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

Let's look deeper into the problem (with $T = \mathrm{EUF}$)

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

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

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

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

Let's look deeper into the problem (with $T = \mathrm{EUF}$)

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

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

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

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

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

Let's look deeper into the problem (with $T = \mathrm{EUF}$)

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

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

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

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

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

or

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

Let's look deeper into the problem (with $T = \mathrm{EUF}$)

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

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

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

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

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

or

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

E-ground (dis)unification

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

- Solution space can be restricted into ground terms from $E \cup L$

- NP-complete
  - NP: solutions can checked in polynomial time
  - NP-hard: reduction of 3-SAT into the entailment

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

$$s_1\sigma \simeq t_1\sigma, \ldots, s_n\sigma \simeq t_n\sigma \models u\sigma \simeq v\sigma$$

Casting instantiation techniques: Trigger-based

$$E \models (u_1 \simeq y_1 \wedge \cdots \wedge u_m \simeq y_m)\,\sigma$$

where $\{u_1, \ldots, u_m\}$ is a trigger for $\forall \bar{x}.\, \psi \in \mathcal{Q}$ and each $y_i\sigma \in \mathbf{T}(E)$

- Consider
  - $E = \{f(a) \simeq g(b),\ h(a) \simeq b,\ f(a) \simeq f(c)\}$
  - $\mathcal{Q} = \{\forall x.\, f(x) \not\simeq g(h(x))\}$, Trigger=$\{f(x)\}$
- Solving $E \models (f(x) \simeq y)\sigma$ yields
  - $\sigma_1 = \{y \mapsto f(a),\ x \mapsto a\}$
  - $\sigma_2 = \{y \mapsto f(c),\ x \mapsto c\}$
- The instantiation lemmas are:
  - $\forall x.\, f(x) \not\simeq g(h(x)) \rightarrow f(a) \not\simeq g(h(a))$
  - $\forall x.\, f(x) \not\simeq g(h(x)) \rightarrow f(c) \not\simeq g(h(c))$

Casting instantiation techniques: Conflict-based

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

- Consider
  - $E = \{f(a) \simeq g(b),\ h(a) \simeq b,\ f(a) \simeq f(c)\}$
  - $\mathcal{Q} = \{\forall x.\ f(x) \not\simeq g(h(x))\}$
- Solving $E \models (f(x) \simeq g(h(x)))\sigma$ yields
  - $\sigma = \{x \mapsto a\}$
- The instantiation lemma is:
  - $\forall x.\ f(x) \not\simeq g(h(x)) \rightarrow f(a) \not\simeq g(h(a))$

Casting instantiation techniques: Model-based

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

where E_{TOT} is a total extension of E s.t.:

- ground terms not in E necessary for evaluating Q are added
- all terms in $\mathbf{T}(E)$ not asserted equal are made disequal

▷ Consider

- $E = \{f(a) \simeq g(b),\ h(a) \simeq b\}$
- $\mathcal{Q} = \{\forall x.\, f(x) \not\simeq g(x),\ \forall xy.\, \psi\}$, $e = a$ as a default value, and

$$\begin{aligned} E_{\mathrm{TOT}} \quad &= E \cup \quad \{a \not\simeq b, a \not\simeq f(a), b \not\simeq f(a)\} \\ &\quad \cup \quad \{f(b) \simeq f(a), f(f(a)) \simeq f(a), g(a) \simeq a, g(f(a)) \simeq a\} \cup \{\ldots \end{aligned}$$

▷ Solving $\{\ldots, f(a) \simeq g(b), f(b) \simeq f(a), \ldots\} \models f(x) \simeq g(x)\sigma$ yields

- $\sigma = \{x \mapsto b\}$

▷ The lemma $\forall x.\, f(x) \not\simeq g(x) \rightarrow f(a) \not\simeq g(a)$ prevents the same E_{TOT}

How to solve the E-ground (dis)unification problem?

Entailment conditions:

- $E \models (x \simeq y)\sigma$
  - $x\sigma = y\sigma$ or
  - some t_1, t_2 s.t. $x\sigma \in [t_1]$, $y\sigma \in [t_2]$, and $[t_1] = [t_2]$

How to solve the E-ground (dis)unification problem?

Entailment conditions:

- $E \models (x \simeq y)\sigma$
  - $x\sigma = y\sigma$ or
  - some t_1, t_2 s.t. $x\sigma \in [t_1]$, $y\sigma \in [t_2]$, and $[t_1] = [t_2]$
- $E \models (x \simeq f(s_1, \ldots, s_n))\sigma$, x occurs in $f(s_1, \ldots, s_n)$,
  - some t_1, $t_2 \in \mathbf{T}(E)$ s.t. $x\sigma \in [t_1]$, $f(s_1, \ldots, s_n)\sigma \in [t_2]$, and $[t_1] = [t_2]$

How to solve the E-ground (dis)unification problem?

Entailment conditions:

- $E \models (x \simeq y)\sigma$
  - $x\sigma = y\sigma$ or
  - some t_1, t_2 s.t. $x\sigma \in [t_1]$, $y\sigma \in [t_2]$, and $[t_1] = [t_2]$
- $E \models (x \simeq f(s_1, \ldots, s_n))\sigma$, x occurs in $f(s_1, \ldots, s_n)$,
  - some t_1, $t_2 \in \mathbf{T}(E)$ s.t. $x\sigma \in [t_1]$, $f(s_1, \ldots, s_n)\sigma \in [t_2]$, and $[t_1] = [t_2]$
- $E \models (x \simeq f(s_1, \ldots, s_n))\sigma$, x does not occur in $f(s_1, \ldots, s_n)$ and
  - $x\sigma = f(s_1, \ldots, s_n)\sigma$ or
  - some t_1, t_2 s.t. $x\sigma \in [t_1]$, $f(s_1, \ldots, s_n)\sigma \in [t_2]$, and $[t_1] = [t_2]$

How to solve the E-ground (dis)unification problem?

Entailment conditions:

- $E \models (x \simeq y)\sigma$
  - $x\sigma = y\sigma$ or
  - some t_1, t_2 s.t. $x\sigma \in [t_1]$, $y\sigma \in [t_2]$, and $[t_1] = [t_2]$
- $E \models (x \simeq f(s_1, \ldots, s_n))\sigma$, x occurs in $f(s_1, \ldots, s_n)$,
  - some t_1, $t_2 \in \mathbf{T}(E)$ s.t. $x\sigma \in [t_1]$, $f(s_1, \ldots, s_n)\sigma \in [t_2]$, and $[t_1] = [t_2]$
- $E \models (x \simeq f(s_1, \ldots, s_n))\sigma$, x does not occur in $f(s_1, \ldots, s_n)$ and
  - $x\sigma = f(s_1, \ldots, s_n)\sigma$ or
  - some t_1, t_2 s.t. $x\sigma \in [t_1]$, $f(s_1, \ldots, s_n)\sigma \in [t_2]$, and $[t_1] = [t_2]$
- $E \models (f(u_1, \ldots, u_n) \simeq g(v_1, \ldots, v_n))\sigma$ and
  - $f = g$ and $E \models u_1\sigma \simeq v_1\sigma$, $\ldots$, $E \models u_n\sigma \simeq v_n\sigma$ or
  - some t_1, $t_2 \in \mathbf{T}(E)$ s.t. $[t_1] = [t_2]$, $f(u_1, \ldots, u_n)\sigma \in [t_1]$, and $g(v_1, \ldots, v_n)\sigma \in [t_2]$

How to solve the E-ground (dis)unification problem?

Entailment conditions:

- $E \models (x \simeq y)\sigma$
  - $x\sigma = y\sigma$ or
  - some t_1, t_2 s.t. $x\sigma \in [t_1]$, $y\sigma \in [t_2]$, and $[t_1] = [t_2]$
- $E \models (x \simeq f(s_1, \ldots, s_n))\sigma$, x occurs in $f(s_1, \ldots, s_n)$,
  - some t_1, $t_2 \in \mathbf{T}(E)$ s.t. $x\sigma \in [t_1]$, $f(s_1, \ldots, s_n)\sigma \in [t_2]$, and $[t_1] = [t_2]$
- $E \models (x \simeq f(s_1, \ldots, s_n))\sigma$, x does not occur in $f(s_1, \ldots, s_n)$ and
  - $x\sigma = f(s_1, \ldots, s_n)\sigma$ or
  - some t_1, t_2 s.t. $x\sigma \in [t_1]$, $f(s_1, \ldots, s_n)\sigma \in [t_2]$, and $[t_1] = [t_2]$
- $E \models (f(u_1, \ldots, u_n) \simeq g(v_1, \ldots, v_n))\sigma$ and
  - $f = g$ and $E \models u_1\sigma \simeq v_1\sigma$, $\ldots$, $E \models u_n\sigma \simeq v_n\sigma$ or
  - some t_1, $t_2 \in \mathbf{T}(E)$ s.t. $[t_1] = [t_2]$, $f(u_1, \ldots, u_n)\sigma \in [t_1]$, and $g(v_1, \ldots, v_n)\sigma \in [t_2]$
- $E \models (u \not\simeq v)\sigma$
  - some t_1, $t_2 \in \mathbf{T}(E)$ s.t. $E \models t_1 \not\simeq t_2$, $u\sigma \in [t_1]$, and $v\sigma \in [t_2]$

Congruence Closure with Free Variables

Congruence Closure with Free Variables (CCFV) is a sound, complete and terminating calculus for solving E-ground (dis)unification

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

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

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

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

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

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

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

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

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

$$\varnothing \,\Big|$$

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

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

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

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

$$\varnothing \Big|$$

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

$$y \simeq b \Big|$$

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

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

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

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

$$\varnothing \Big|$$

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

$$y \simeq b \Big|$$

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

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

$$f(x) \simeq f(c)$$

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

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

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

$$\varnothing \;\Big|$$

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

$$y \simeq b \;\Big|$$

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

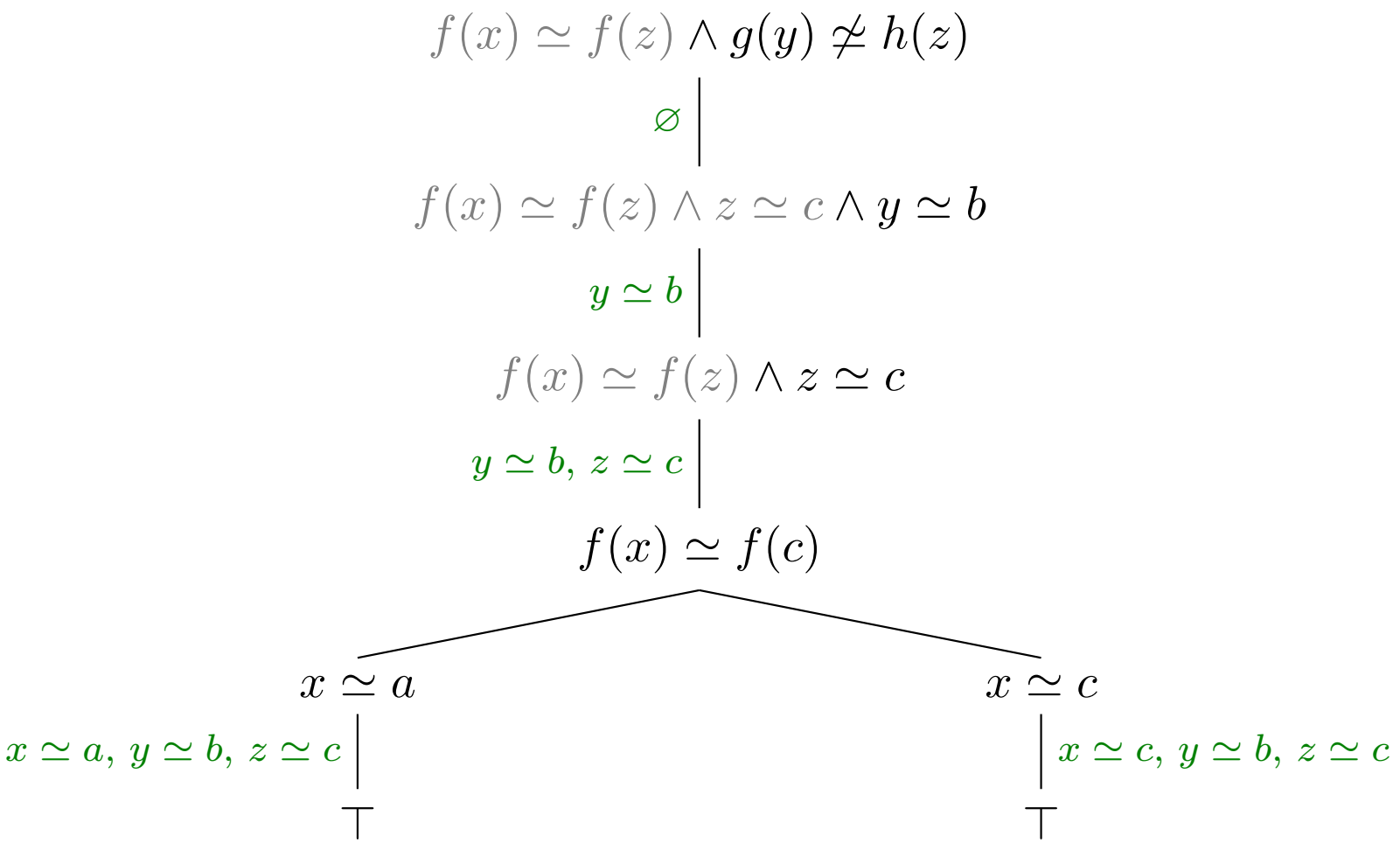
$$y \simeq b,\ z \simeq c \;\Big|$$

$$f(x) \simeq f(c)$$

$$x \simeq a \qquad\qquad x \simeq c$$

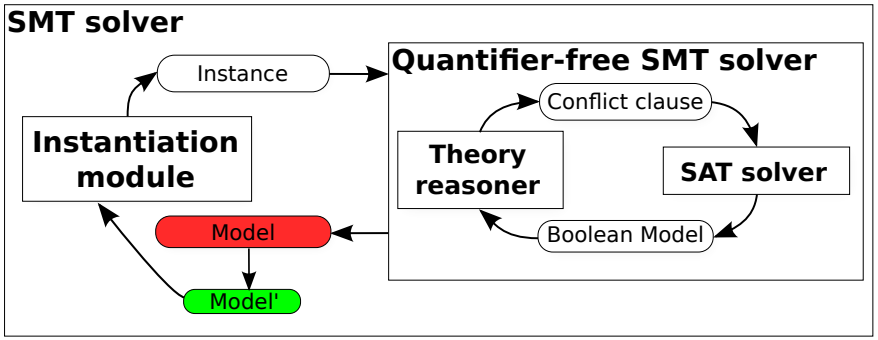
Finding solutions σ for $E \models L\sigma$

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



Implementation

- Model minimization
  - Relevancy
  - Prime implicant



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

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

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

    $E \models f(x)\sigma \simeq g(y)\sigma$ only if some $[t]$ contains some $f(t')$ and some $g(t'')$
    - Bitmasks for fast checking if symbol has applications in congruence class

- Mapping from congruence class to classes it's disequal to

Implementation

- Selection strategies

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

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

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

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

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

Implementation

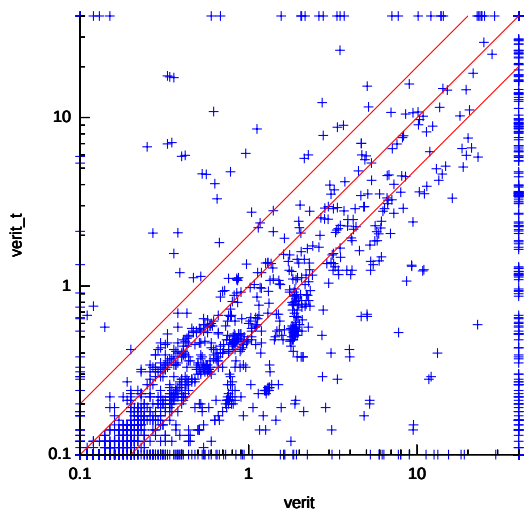
A breadth-first implementation of CCFV:

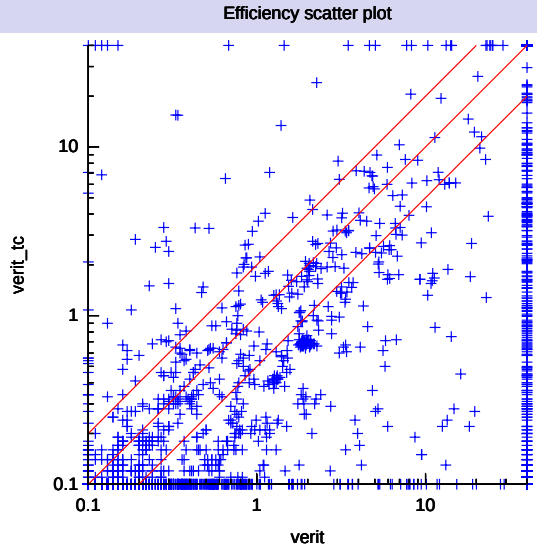
- ▷ Explores sets of solutions at a time

$$E \models \underbrace{\underbrace{\ell_1 \downarrow \mathfrak{S}_1}_{\text{individual solutions for each literal}} \sqcap \ldots \sqcap \underbrace{\ell_n \downarrow \mathfrak{S}_n}_{\text{individual solutions for each literal}}}_{\mathfrak{S}} \wedge \ldots \wedge \underbrace{\ell_1 \downarrow \mathfrak{S}_1 \sqcap \ldots \sqcap \ell_n \downarrow \mathfrak{S}_n}_{\text{combination of compatible solutions}}$$

- ⊕ Heavy use of memoization
- ⊖ Bottleneck in merging solution sets

Evaluation

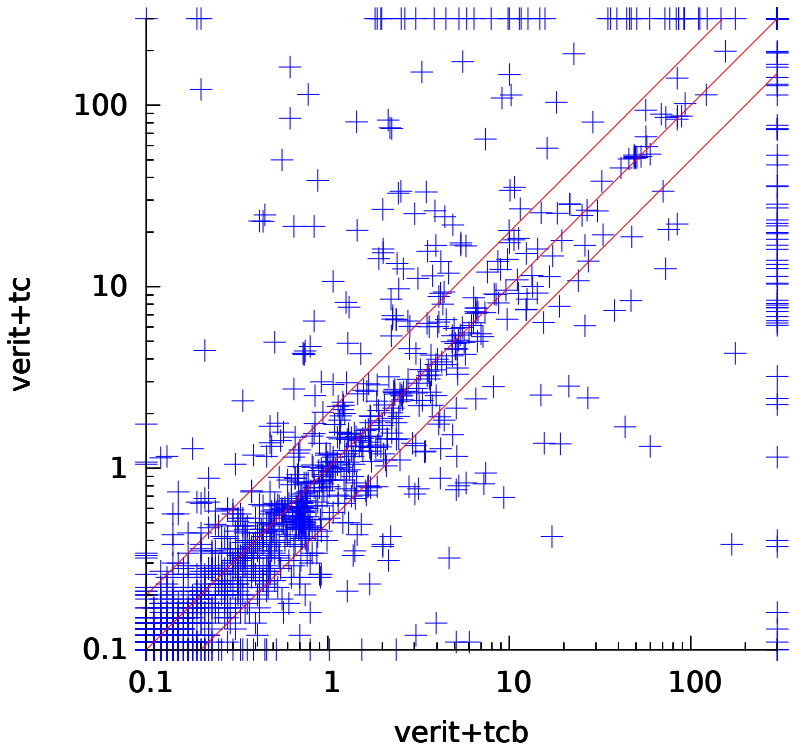




Logic	Class	Z3	CVC4	verit+tc	verit+tcb	verit+t	verit
UF	grasshopper	418	411	430	**435**	418	413
	sledgehammer	1249	**1438**	1277	1278	1134	1066
UFLIA	boogie	**852**	844	706	690	660	661
	sexpr	**26**	12	7	7	5	5
	grasshopper	341	322	326	**361**	340	335
	sledgehammer	1581	**1944**	1790	1799	1620	1569
	simplify	**831**	766	803	801	735	690
	simplify2	**2337**	2330	2307	2303	2291	2177
	Total	7635	**8067**	7676	7678	7203	6858

- ▷ experiments in the "UF", "UFLIA", "UFLRA" and "UFIDL" categories of SMT-LIB, which have $10\,495$ benchmarks annotated as *unsatisfiable*, with 30s timeout. Circa 2017.

Depth-first vs Breadth-first CCFV



The depth-first CCFV outperforms its breadth-first counterpart by a small margin.

Both perform well and are viable approaches

* experiments in the "UF", "UFLIA", "UFLRA" and "UFIDL" categories of SMT-LIB, which have $10\,495$ benchmarks annotated as *unsatisfiable*, with 100s timeout.

Limitations

- Ground congruence closure not closed to disequality entailment

  E.g. $g(f(a), h(b)) \not\simeq g(f(b), h(a)) \in E$ should lead to adding $a \not\simeq b$ to E

- No learning when backtracking

- Hard to check entailment for theories other than EUF

$$f(1) \simeq 5 \overset{?}{\models}_{\text{LIA}} (f(x+y) \leq x+2y)\sigma$$

Limitations

- Ground congruence closure not closed to disequality entailment

  E.g. $g(f(a), h(b)) \not\simeq g(f(b), h(a)) \in E$ should lead to adding $a \not\simeq b$ to E

- No learning when backtracking

- Hard to check entailment for theories other than EUF

$$f(1) \simeq 5 \overset{?}{\models}_{\text{LIA}} (f(x+y) \leq x+2y)\sigma$$

  Yes, for $\sigma = \{x \mapsto -3, y \mapsto 4\}$

An extension for HOL

- Build on initial extension of SMT solvers to HOL [BRO+19]
- Lifting CCFV allows directly lifting compatible instantiation techniques
- Issues
  - Currying complicates indexing
    - Terms now have the form $@(@(f, a), b)$
  - Higher-order unification (i.e. lambdas) complicates entailment checking
    - Equalities between functions
    - Unification with different arguments
    - Undecidability

Lifting CCFV to λfHOL via SAT encoding [ST20]

- Simpler fragment as sandbox
- Encode entailment checks to SAT
  - Easier to grasp
  - Free learning
  - Allows optimizations from "global reasoning"
- Build substitutions from SAT models
- Still very much in progress

Congruence Closure with Free Variables: A quantifier-instantiation framework for SMT

Haniel Barbosa

SRI Formal Topics Seminar

2020-07-16, The Internet

CCFV calculus

$$\frac{E_\sigma \Vdash_E x \simeq s \wedge C}{E_\sigma \cup \{x \simeq s\} \Vdash_E \underline{rep}(\{x \simeq s\}, C)}\ \textsc{Assign} \qquad \text{if } x \notin \mathsf{FV}(s)$$

$$\frac{E_\sigma \Vdash_E x \simeq f(\bar{u}) \wedge C}{E_\sigma \Vdash_E \bigvee_{[t] \in E^{\mathrm{CC}},\ f(\bar{t}) \in [t]} (x \simeq t \wedge u_1 \simeq t_1 \wedge \cdots \wedge u_n \simeq t_n \wedge C)}\ \textsc{Uvar} \quad \text{if } x \in \mathsf{FV}(f(\bar{u}))$$

$$\frac{E_\sigma \Vdash_E f(\bar{u}) \simeq f(\bar{s}) \wedge C}{E_\sigma \Vdash_E (u_1 \simeq s_1 \wedge \cdots \wedge u_n \simeq s_n \wedge C) \vee \bigvee_{[t] \in E^{\mathrm{CC}},\, f(\bar{t}) \in [t],\, f(\bar{t'}) \in [t]} \left(\begin{array}{l} u_1 \simeq t_1 \wedge \cdots \wedge u_n \simeq t_n \wedge \\ s_1 \simeq t'_1 \wedge \cdots \wedge s_n \simeq t'_n \wedge C \end{array} \right)}\ \textsc{Ucomp}$$

$$\frac{E_\sigma \Vdash_E f(\bar{u}) \simeq g(\bar{s}_m) \wedge C}{E_\sigma \Vdash_E \bigvee_{\substack{[t] \in E^{\mathrm{CC}}, \\ f(\bar{t}) \in [t],\, g(\bar{t'}_m) \in [t]}} \left(\begin{array}{l} u_1 \simeq t_1 \wedge \cdots \wedge u_n \simeq t_n \wedge \\ s_1 \simeq t'_1 \wedge \cdots \wedge s_m \simeq t'_m \wedge C \end{array} \right)}\ \textsc{Ugen} \quad \text{if } f \neq g$$

CCFV calculus

$$\frac{E_\sigma \Vdash_E x \not\simeq y \wedge C}{E_\sigma \Vdash_E \bigvee_{[t],\,[t']\in E^{\mathrm{CC}},\, E\models t\not\simeq t'} (x \simeq t \wedge y \simeq t' \wedge C)}\ \textsc{Dvar}$$

$$\frac{E_\sigma \Vdash_E x \not\simeq f(\bar{s}) \wedge C}{E_\sigma \Vdash_E \bigvee_{\substack{[t],\,[t']\in E^{\mathrm{CC}},\\ E\models t\not\simeq t',\, f(\bar{t'})\in[t']}} (x \simeq t \wedge s_1 \simeq t'_1 \wedge \cdots \wedge s_n \simeq t'_n \wedge C)}\ \textsc{Dfapp}$$

$$\frac{E_\sigma \Vdash_E f(\bar{u}) \not\simeq g(\bar{s}_m) \wedge C}{E_\sigma \Vdash_E \bigvee_{\substack{[t],\,[t']\in E^{\mathrm{CC}},\, E\models t\not\simeq t',\\ f(\bar{t})\in[t],\, g(\bar{t'}_m)\in[t']}} \left(\begin{array}{l} u_1 \simeq t_1 \wedge \cdots \wedge u_n \simeq t_n \ \wedge \\ s_1 \simeq t'_1 \wedge \cdots \wedge s_m \simeq t'_m \wedge C \end{array}\right)}\ \textsc{Dgen}$$

$$\frac{E_\sigma \Vdash_E C_1 \vee C_2}{E_\sigma \Vdash_E C_1 \qquad E_\sigma \Vdash_E C_2}\ \textsc{Split} \qquad \frac{E_\sigma \Vdash_E \ell \wedge C}{E_\sigma \Vdash_E C}\ \textsc{Yield} \quad \text{if } E \models \ell$$

$$\frac{E_\sigma \Vdash_E \ell \wedge C}{E_\sigma \Vdash_E \bot}\ \textsc{Fail} \quad \text{if } \ell \text{ is ground and } E \not\models \ell$$

References

Haniel Barbosa, Pascal Fontaine, and Andrew Reynolds. "Congruence Closure with Free Variables". In: Tools and Algorithms for Construction and Analysis of Systems (TACAS), Part II. Ed. by Axel Legay and Tiziana Margaria. Vol. 10206. Lecture Notes in Computer Science. 2017, pp. 214–230.

Nikolaj Bjørner and Mikolas Janota. "Playing with Quantified Satisfaction". In: Logic for Programming, Artificial Intelligence, and Reasoning (LPAR). Ed. by Ansgar Fehnker, Annabelle McIver, Geoff Sutcliffe, et al. Vol. 35. EPiC Series in Computing. EasyChair, 2015, pp. 15–27.

Nikolaj Bjørner. "Linear Quantifier Elimination as an Abstract Decision Procedure". In: International Joint Conference on Automated Reasoning (IJCAR). Ed. by Jürgen Giesl and Reiner Hähnle. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 316–330.

Haniel Barbosa, Andrew Reynolds, Daniel El Ouraoui, et al. "Extending SMT Solvers to Higher-Order Logic". In: Proc. Conference on Automated Deduction (CADE). Ed. by Pascal Fontaine. Vol. 11716. Lecture Notes in Computer Science. Springer, 2019, pp. 35–54.

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.

David Monniaux. "Quantifier Elimination by Lazy Model Enumeration". In: Computer Aided Verification (CAV). Ed. by Tayssir Touili, Byron Cook, and Paul B. Jackson. Vol. 6174. Lecture Notes in Computer Science. Springer, 2010, pp. 585–599.

David A. Plaisted. "History and Prospects for First-Order Automated Deduction". In: Automated Deduction - CADE-25 - 25th International Conference on Automated Ded Ed. by Amy P. Felty and Aart Middeldorp. Vol. 9195. Lecture Notes in Computer Science. Springer, 2015, pp. 3–28.

References

Andrew Reynolds, Haniel Barbosa, and Pascal Fontaine. "Revisiting Enumerative Instantiation". In: Tools and Algorithms for Construction and Analysis of Systems (TACAS), Part II. Ed. by Dirk Beyer and Marieke Huisman. Vol. 10806. Lecture Notes in Computer Science. Springer, 2018, pp. 112–131.

Andrew Reynolds, Morgan Deters, Viktor Kuncak, et al. "Counterexample-Guided Quantifier Instantiation for Synthesis in SMT". In: Computer Aided Verification (CAV). Ed. by Daniel Kroening and Corina S. Pasareanu. Vol. 9207. Lecture Notes in Computer Science. Springer, 2015, pp. 198–216.

Andrew Reynolds, Cesare Tinelli, Amit Goel, et al. "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.

References

Daniel El Ouraoui Haniel Barbosa Sophie Tourret Pascal Fontaine. "Higher-Order SMT Solving (work in progress)". SMT 2020. 2020.