Higher-Order SMT Solving

(Work in Progress)

Haniel Barbosa[1] Andrew Reynolds[1] Pascal Fontaine[2]
Daniel El Ouraoui[2] Cesare Tinelli[1]

University of Iowa, Iowa City, USA
haniel-barbosa,cesare-tinelli@uiowa.edu,andrew.j.reynolds@gmail.com

University of Lorraine, CNRS, Inria, and LORIA, Nancy, France
daniel.el-ouraoui,pascal.fontaine@inria.fr

21st July 2018

Contents

Contents

Why Higher-Order (HO)

Higher-Order logic

- Expressive
  - Mathematics
  - Verification conditions
- The language of proof assistants
  - Isabelle, Coq, Agda

Automation

- Hard to automatize
- Few provers to reason on it
  LEO-II, Leo-III, Satalax

Challenge

- New techniques for SMT
- Avoid automatic translation

Summary

Two procedures

cvc4 University of Stanford/Iowa
(`http://cvc4.cs.stanford.edu/web`)

veriT Université de Lorraine/UFRN
(`http://www.verit-solver.org`)

Features	Predicate calculus	λ-free	λ-calculus
function	✓	✓	✓
predicate	✓	✓	✓
functional arguments	✗	✓	✓
quantification on objects	✓	✓	✓
quantification on predicates	✗	✓	✓
quantification on functions	✗	✓	✓
partial applications	✗	✓	✓
anonymous functions	✗	✗	✓

Contents

First-Order to Higher-Order with CDCL(T)

- Ground

$$\neg(f\ a\ b \simeq b) \wedge g \simeq f\ a \wedge f\ a\,(f\ a\ b) \simeq g\ b \;\wedge\; \forall x y\, f\ x \simeq f\ y \Rightarrow x \simeq y$$

- Ground part described by the conjunctive sets of literals E
- Quantified part described by the sets of quantified formulas Q
- Check if $E \cup Q$ is consistent

First-Order to Higher-Order with CDCL(T)

- Ground

$$\neg(f\ a\ b \simeq b) \wedge g \simeq f\ a \wedge f\ a\,(f\ a\ b) \simeq g\ b \wedge \forall x y\, f\ x \simeq f\ y \Rightarrow x \simeq y$$

- Instantiation

- Ground part described by the conjunctive sets of literals E
- Quantified part described by the sets of quantified formulas Q
- Check if $E \cup Q$ is consistent

First-Order to Higher-Order with CDCL(T)

- Ground

$$\neg(f\ a\ b \simeq b) \wedge g \simeq f\ a \wedge f\ a\,(f\ a\ b) \simeq g\ b \wedge \forall xy\, f\ x \simeq f\ y \Rightarrow x \simeq y$$

- Instantiation

- Ground part described by the conjunctive sets of literals E
- Quantified part described by the sets of quantified formulas Q
- Check if $E \cup Q$ is consistent

Lift up SMT solver

Ground	Applicative encoding
	Suitable data-structure
Instantiation	E-matching extension

Contents

Applicative encoding

encoding

For all terms of the shape $(((f_{\tau_1 \to \ldots \to \tau_n \to \sigma}\ a_1) \ldots)\ a_n)) : \sigma$ given a unique symbol @ we have the translation App defined as following:

$$\mathrm{App}(((f\ a_1) \ldots)\ a_n)) = @(@(\ldots @(f, a_1), \ldots, a_n))$$

$f\ a\ b \simeq b \wedge f\ a\ (f\ a\ b) \simeq g\ b$

$@(@(f, a), b) \simeq b \wedge @(@(f, a),\ @(@(f, a), b)) \simeq @(g, b)$

where f, g become constant symbols

Applicative encoding

encoding

For all terms of the shape $(((f_{\tau_1 \to \ldots \to \tau_n \to \sigma}\ a_1) \ldots)\ a_n)) : \sigma$ given a unique symbol @ we have the translation App defined as following:

$$\mathrm{App}(((f\ a_1) \ldots)\ a_n)) = @(@(\ldots @(f, a_1), \ldots, a_n))$$

app translation

$f\ a\ b \simeq b \wedge f\ a\ (f\ a\ b) \simeq g\ b$

$@(@(f, a), b) \simeq b \wedge @(@(f, a),\ @(@(f, a), b)) \simeq @(g, b)$

where f, g become constant symbols

Lazy encoding

- Turn all partial applications into total
- Use first-order procedure on $\mathrm{App}(E)$
- Add remaining equalites between regular terms
  $E' = \mathrm{App}(E) \cup \{\mathrm{App}(f(a_1, \ldots, a_n)) \simeq f(a_1, \ldots, a_n), \ \ldots\}$
- Do it only for partial function symbols
- Check again E'

Example

$f\ a \simeq g \wedge f(a, a) \not\simeq g(a) \wedge g(a) \simeq h(a) \Rightarrow \{@(f, a) \simeq g,\ f(a, a) \not\simeq g(a),\ g(a) \simeq h(a)\} \subseteq E$

Lazy encoding

- Turn all partial applications into total
- Use first-order procedure on $\mathsf{App}(E)$
- Add remaining equalites between regular terms
  $E' = \mathsf{App}(E) \cup \{\mathsf{App}(f(a_1, \ldots, a_n)) \simeq f(a_1, \ldots, a_n), \ldots\}$
- Do it only for partial function symbols
- Check again E'

Example

$f\ a \simeq g \wedge f(a, a) \not\simeq g(a) \wedge g(a) \simeq h(a) \Rightarrow \{@(f, a) \simeq g,\ f(a, a) \not\simeq g(a),\ g(a) \simeq h(a)\} \subseteq E$

$$E \cup \{@(@(f, a), a) \simeq f(a, a),\ @(g, a) \simeq g(a)\} \Rightarrow @(@(f, a), a) \simeq @(g, a)$$

Extentionality

$$(\forall \bar{x}\, f(\bar{x}) \simeq g(\bar{x})) \leftrightarrow f \simeq g$$

- The "$\leftarrow$" direction is ensured by the functional congruence axiom:
  $$f \simeq g \rightarrow (\forall \bar{x}\, f(\bar{x}) \simeq g(\bar{x}))$$
- The "$\rightarrow$" direction is ensured by $f(\bar{k}) \not\simeq g(\bar{k})$ for some Skolem $\bar{k}$
- $f(\bar{k}) \not\simeq g(\bar{k}) \vee f \simeq g$ is added for each pair of functions of finite type

Model generation

For each satisfiable problem produce a first-order model M

$$f_1(0) \simeq f_1(1) \wedge f_1(1) \simeq f_2$$
$$f_2(0) \simeq f_2(1) \wedge f_2(1) \simeq 2$$

$f_1 : \mathsf{Int} \times \mathsf{Int} \rightarrow \mathsf{Int}$, and $f_2 : \mathsf{Int} \rightarrow \mathsf{Int}$

Model construction

$$M(f_1) = \lambda xy\, \mathsf{ite}(x \simeq 0, \lambda x\, \mathsf{ite}(x \simeq 1, 2, \_)(y), \mathsf{ite}(x \simeq 1, \lambda x\, \mathsf{ite}(x \simeq 1, 2, \_)(y), \_))$$

Polynomial construction

$$M(f_1) = \lambda xy\, \mathsf{ite}(x \simeq 0, M(f_2)(y), \mathsf{ite}(x \simeq 1, M(f_2)(y), \_))$$
$$M(f_2) = \lambda x\, \mathsf{ite}(x \simeq 1, 2, \_)$$

Trigger based instantiation

Triggers

A trigger T for a quantified formula $\forall \overline{x}_n.\psi$ is a set of non-ground terms $u_1, \ldots, u_n \in \mathbf{T}(\psi)$ such that: $\{\overline{x}\} \subseteq \mathsf{FV}(u_1) \cup \ldots \cup \mathsf{FV}(u_n)$.

E-matching

Given a conjunctive set of equality literals E and terms u and t, with t ground, the E-matching problem is that of finding a substitution σ such that $E \models u\sigma \simeq t$.

$E = \{f(a) \simeq g(b),\ a \simeq g(b)\}$
$Q = \{\forall x\, f(g(x)) \not\simeq g(x)\}$

$f(a)$ E-matches $f(g(x))$ under $\{x \mapsto b\}$

E-matching

- E-matching relies on indexing term by head symbols for efficiency
- At Higher-Order level two applications can be equals with different head symbol $f \simeq g \wedge f\ a \simeq g\ b$
- Common term indexing
- First-order E-matching with applicative encoding and suitable indexing

E-matching

$$\varphi = q(k(0,1)) \land \neg p(k(0,0)) \land \forall (f : \mathsf{Int} \times \mathsf{Int} \to \mathsf{Int})\,(y, z : \mathsf{Int}).\ p(f(y,z)) \lor \neg q(f(1,y))$$

- Extend first-order *E*-matching to derive new lambda expressions
- From Huet's algorithm to higher-order matching
- Unsatisfiable with regular Henkin semantics

$$\{f \mapsto \lambda w_1 w_2.\ k(0, w_1),\ y \mapsto 0,\ z \mapsto 0\}$$

Evaluation

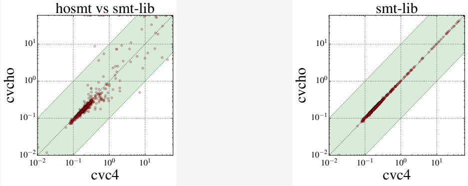


Figure: Time comparison of cvc4 configurations on “Judgement day” benchmarks.

	hosmt		*smt-lib*	
	#unsat	avg time (s)	#unsat	avg time (s)
cvc4-HO	648	1.08	662	1.02
cvc4	4	0.06	662	1.01

Table: cvc4 configurations on “Judgement day” benchmarks with 60s timeout.

Contents

Congruence closure

Theory of equality $\mathcal{T}_E$
$\Sigma_f = \{a, b, f, g, \ldots\}$
$\Sigma_p = \{=, p, q, \ldots\}$

$$\forall (x : \tau)\ x = x \qquad \text{(reflexivity)}$$

$$\forall (xy : \tau)\ x = y \Rightarrow y = x \qquad \text{(symmetry)}$$

$$\forall (xyz : \tau)\ (x = y \Rightarrow y = z) \Rightarrow x = z \qquad \text{(transitivity)}$$

HO congruence

$$x = y \Rightarrow f\ x = f\ y \qquad \text{(right cong)}$$

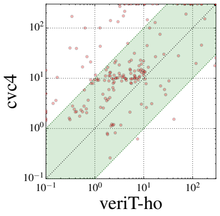
$$f = g \Rightarrow f\ x = g\ x \qquad \text{(left cong)}$$

Congruence closure

Deciding a conjunction of $\mathcal{T}_E$:
How can we check whether a set of $\mathcal{T}_E$ is satisfiable ?

- Union find algorithm
- Optimal time complexity: $\mathcal{O}(n \log n)$
- Graphs with connected component
- Not optimal time complexity: $\mathcal{O}(n^2)$

Evaluation



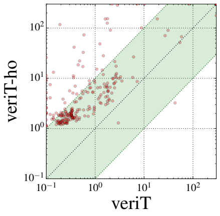


Figure: Time comparison of cvc4 veriT and veriT-Ho on QFUF benchmarks.

Contents

Conclusions and future directions

- No significant overhead
- HO ATPs such LEO-II, Leo-III, Satalax should be investigated
- Towards an effective and refutationally complete calculus
- Improving and extend veriT in the same fashion