

Extending SMT solvers to higher-order logic

Haniel Barbosa

Andrew Reynolds

Cesare Tinelli



Daniel El Ouraoui



Clark Barrett



CADE-27

2019-08-30, Natal, BR

Why higher-order logic?

Higher-Order logic

- ▷ Expressive
 - ▶ Mathematics
 - ▶ Verification conditions
- ▷ The language of proof assistants
 - ▶ Isabelle, Coq, Lean, ...

Automation

- ▷ Reducing the burden of proof on users

State of the art of HOL automation

- ▷ Higher-order provers Leo-III, Satalax, ...
 - ▶ Scalability issues on problems with large FO component
- ▷ Hammers HOLYHammer, MizAIR, Sledgehammer, ...
 - ▶ Issues with performance, soundness, or completeness

State of the art of HOL automation

- ▷ Higher-order provers Leo-III, Satalax, ...
 - ▶ Scalability issues on problems with large FO component
- ▷ Hammers HOLYHammer, MizAR, Sledgehammer, ...
 - ▶ Issues with performance, soundness, or completeness

“Timeouts into quick unsats”

$$f(\lambda x. g(x) + h(x)) \simeq f(\lambda x. h(x) + g(x))$$

↓ cong, ext

$$(\forall x. g(x) + h(x) \simeq h(x) + g(x)) \Rightarrow f(\lambda x. g(x) + h(x)) \simeq f(\lambda x. h(x) + g(x))$$

↓ ¬, CNF

$$\begin{aligned} g(sk) + h(sk) &\not\simeq h(sk) + g(sk) \\ f(\lambda x. g(x) + h(x)) &\not\simeq f(\lambda x. h(x) + g(x)) \end{aligned}$$

State of the art of HOL automation

- ▷ Higher-order provers Leo-III, Satalax, ...
 - ▶ Scalability issues on problems with large FO component
- ▷ Hammers HOLYHammer, MizAR, Sledgehammer, ...
 - ▶ Issues with performance, soundness, or completeness
- ▷ Extensions of first-order systems Zipperposition, Vampire, Ehoh
 - ▶ Graceful extension: only pay for what you use

Goal: **simplicity, practicality, and effectiveness**

An (initial) graceful extension of SMT solvers to HOL to handle:

- ▷ Partial applications
 - ▶ Term representation
 - ▶ Core algorithms

- ▷ Quantification on functional variables
 - ▶ Term indexing
 - ▶ Synthesizing λ -terms (e.g. via higher-order unification)

Goal: **simplicity, practicality, and effectiveness**

An (initial) graceful extension of SMT solvers to HOL to handle:

- ▷ Partial applications
 - ▶ Term representation
 - ▶ Core algorithms

- ▷ Quantification on functional variables
 - ▶ Term indexing
 - ▶ Synthesizing λ -terms (e.g. via higher-order unification)

Outline

- ▷ What we mean by higher-order logic
- ▷ Extending an SMT solver pragmatically
- ▷ Extending an SMT solver via redesign
- ▷ Evaluation

Fragments of interest

Features	FOL	λ FHOL	HOL
function	✓	✓	✓
quantification on objects	✓	✓	✓
quantification on functions	✗	✓	✓
partial applications	✗	✓	✓
anonymous functions	✗	✗	✓

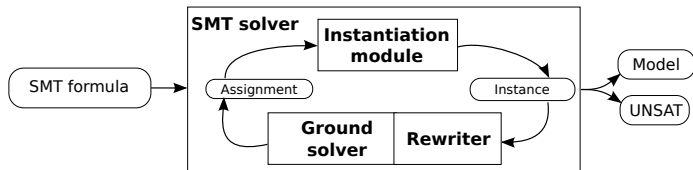
▷ Henkin semantics

- ▶ Function interpretations restricted to terms expressible in formula's signature

▷ Extensionality

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

A CDCL(\mathcal{T}) SMT solver



- ▷ Rewriter simplifies terms

$$x + 0 \rightarrow x \quad a \neq a \rightarrow \perp \quad (\text{str.replace } x \text{ (str.++ } x \text{) } y) \rightarrow x$$

- ▷ Ground solver enumerates assignments $E \cup Q$

- ▶ E is a set of ground literals

$$\{a \leq b, b \leq a + x, x \simeq 0, f(a) \neq f(b)\}$$

- ▶ Q is a set of quantified clauses

$$\{\forall xyz. f(x) \neq f(z) \vee g(y) \simeq h(z)\}$$

- ▷ Instantiation module generates instances of Q

$$f(a) \neq f(b) \vee g(a) \simeq h(b)$$

A pragmatic extension

▷ Preprocessing

- ▶ Totalizing applications of theory symbols

$$\frac{\varphi[1+]}{\varphi[\lambda x. 1 + x]}$$

- ▶ λ -lifting
- $$\frac{\varphi[\lambda x. t]}{\varphi[f(t)] \wedge \forall x. f(x) \simeq t}$$

▷ Ground *equality and uninterpreted functions* (EUF) solver

- ▶ Lazy applicative encoding
- ▶ Extensionality lemmas
- ▶ Polynomial model construction for partial functions

▷ Instantiation module

- ▶ Extending E -matching
- ▶ Adding expressivity via axioms

Applicative encoding

- ▷ Every functional sort converted into an atomic sort
- ▷ Every n -ary function symbol converted into a constant
- ▷ Every function application converted into @ applications

$$\frac{\varphi[f(t_1, \dots, t_n)]}{\varphi[@(\dots (@(f, t_1), \dots), t_n)]}$$

$$\begin{array}{ccccc} f(a) \simeq g & \wedge & f(a, a) \not\simeq g(a) & \wedge & g(a) \simeq h(a) \\ \downarrow & & \downarrow & & \downarrow \\ @(f, a) \simeq g & \wedge & @(@(f, a), a) \not\simeq @(g, a) & \wedge & @(g, a) \simeq @(h, a) \end{array}$$

Lazy applicative encoding

- ▷ Encode partial applications eagerly
- ▷ Apply regular congruence closure
- ▷ Lazily encode relevant applications

1 $E = \{ @(f, a) \simeq g, f(a, a) \not\simeq g(a), g(a) \simeq h(a) \}$ is satisfiable

$$E \not\models f(a, a) \simeq g(a)$$

Lazy applicative encoding

- ▷ Encode partial applications eagerly
- ▷ Apply regular congruence closure
- ▷ Lazily encode relevant applications

1 $E = \{ @(f, a) \simeq g, f(a, a) \not\simeq g(a), g(a) \simeq h(a) \}$ is satisfiable

$$E \not\models f(a, a) \simeq g(a)$$

2 Applications of f and g need to be encoded

Lazy applicative encoding

- ▷ Encode partial applications eagerly
- ▷ Apply regular congruence closure
- ▷ Lazily encode relevant applications

1 $E = \{ @(f, a) \simeq g, f(a, a) \neq g(a), g(a) \simeq h(a) \}$ is satisfiable

$$E \not\models f(a, a) \simeq g(a)$$

2 Applications of f and g need to be encoded

3 $E' = E \cup \{ @(@(f, a), a) \simeq f(a, a), @(g, a) \simeq g(a) \}$ is **unsatisfiable**

$$E' \models f(a, a) \simeq g(a)$$

Note that $h(a)$ is not encoded!

- ▷ “ \leftarrow ” handled by lazy encoding and congruence

$$\frac{\frac{\frac{f \simeq g}{@(\mathbf{f}, t_1) \simeq @(\mathbf{g}, t_1)} \text{ CONG}}{\dots} \text{ CONG}}{@(\dots (@(\mathbf{f}, t_1), \dots), t_n) \simeq @(\dots (@(\mathbf{g}, t_1), \dots), t_n)} \text{ CONG}$$

- ▷ “ \rightarrow ” handled by

$$\frac{f \not\simeq g}{f(\mathbf{sk}_1, \dots, \mathbf{sk}_n) \not\simeq g(\mathbf{sk}_1, \dots, \mathbf{sk}_n)} \text{ EXTENSIONALITY}$$

Avoiding exponential model construction

Functions are interpreted as if-then-else:

$$M(f) = \lambda x. \text{ite}(x \simeq t_1, s_1, \dots \text{ite}(x \simeq t_{n-1}, s_{n-1}, s_n) \dots)$$

Partial applications can lead to exponentially many cases!

$$\begin{aligned} & f_1(a) \simeq f_1(b) \wedge f_1(b) \simeq f_2 \\ \wedge & f_2(a) \simeq f_2(b) \wedge f_2(b) \simeq f_3 \\ \wedge & f_3(a) \simeq f_3(b) \wedge f_3(b) \simeq c \end{aligned}$$

8 ite entries to model that $f_1(x, y, z) \simeq c$, for $x, y, z \in \{a, b\}$

Avoiding exponential model construction

Functions are interpreted as if-then-else:

$$M(f) = \lambda x. \text{ite}(x \simeq t_1, s_1, \dots \text{ite}(x \simeq t_{n-1}, s_{n-1}, s_n) \dots)$$

Partial applications can lead to exponentially many cases!

$$\begin{aligned} & f_1(a) \simeq f_1(b) \wedge f_1(b) \simeq f_2 \\ \wedge & f_2(a) \simeq f_2(b) \wedge f_2(b) \simeq f_3 \\ \wedge & f_3(a) \simeq f_3(b) \wedge f_3(b) \simeq c \end{aligned}$$

8 ite entries to model that $f_1(x, y, z) \simeq c$, for $x, y, z \in \{a, b\}$

Polynomial construction in the “depth” of functions chain

$$M(f_1) = \lambda xyz. \text{ite}(x \simeq a, M(f_2)(y, z), \text{ite}(x \simeq b, M(f_2)(y, z), -))$$

Avoiding exponential model construction

Functions are interpreted as if-then-else:

$$M(f) = \lambda x. \text{ite}(x \simeq t_1, s_1, \dots \text{ite}(x \simeq t_{n-1}, s_{n-1}, s_n) \dots)$$

Partial applications can lead to exponentially many cases!

$$\begin{aligned} & f_1(a) \simeq f_1(b) \wedge f_1(b) \simeq f_2 \\ \wedge & f_2(a) \simeq f_2(b) \wedge f_2(b) \simeq f_3 \\ \wedge & f_3(a) \simeq f_3(b) \wedge f_3(b) \simeq c \end{aligned}$$

8 ite entries to model that $f_1(x, y, z) \simeq c$, for $x, y, z \in \{a, b\}$

Polynomial construction in the “depth” of functions chain

$$M(f_1) = \lambda xyz. \text{ite}(x \simeq a, M(f_2)(y, z), \text{ite}(x \simeq b, M(f_2)(y, z), -))$$

$$M(f_2) = \lambda xy. \text{ite}(x \simeq a, M(f_3)(y), \text{ite}(x \simeq b, M(f_3)(y), -))$$

Avoiding exponential model construction

Functions are interpreted as if-then-else:

$$M(f) = \lambda x. \text{ite}(x \simeq t_1, s_1, \dots \text{ite}(x \simeq t_{n-1}, s_{n-1}, s_n) \dots)$$

Partial applications can lead to exponentially many cases!

$$\begin{aligned} & f_1(a) \simeq f_1(b) \wedge f_1(b) \simeq f_2 \\ \wedge & f_2(a) \simeq f_2(b) \wedge f_2(b) \simeq f_3 \\ \wedge & f_3(a) \simeq f_3(b) \wedge f_3(b) \simeq c \end{aligned}$$

8 ite entries to model that $f_1(x, y, z) \simeq c$, for $x, y, z \in \{a, b\}$

Polynomial construction in the “depth” of functions chain

$$M(f_1) = \lambda xyz. \text{ite}(x \simeq a, M(f_2)(y, z), \text{ite}(x \simeq b, M(f_2)(y, z), -))$$

$$M(f_2) = \lambda xy. \text{ite}(x \simeq a, M(f_3)(y), \text{ite}(x \simeq b, M(f_3)(y), -))$$

$$M(f_3) = \lambda x. \text{ite}(x \simeq a, c, \text{ite}(x \simeq b, c, -))$$

Extending E -matching

- ▷ Since $@$ is overloaded, matching must account for types of arguments
 - ▶ $@(x, a)$ can't match $@(f, a)$ if x and f of different types
- ▷ Indexing robust to mixed partial/total applications
 - ▶ In HOL applications with different heads can be equal
 $@(f, a) \simeq g$ allows matching $g(x)$ with $f(a, b)$
- ▷ HO- E -matching left for future work

Using well-chosen axioms

- ▷ Store axiom

$$\forall F. \forall x, y. \exists G. \forall z. G(z) \simeq \text{ite}(z \simeq x, y, F(z))$$

- ▷ Instances from the larger set of functions representable in the signature

$a \neq b \wedge \forall F, G. F \simeq G$ is unsatisfiable

- ▷ Requires $F \mapsto (\lambda w. a)$, $G \mapsto (\lambda w. b)$
- ▷ E -matching can't derive this instantiation

Redesigning the SMT solver

- ▷ Simpler and more flexible congruence closure
 - ▶ Graph representation rather than UNION-FIND
 - ▶ Quadratic instead of $\mathcal{O}(n \log n)$

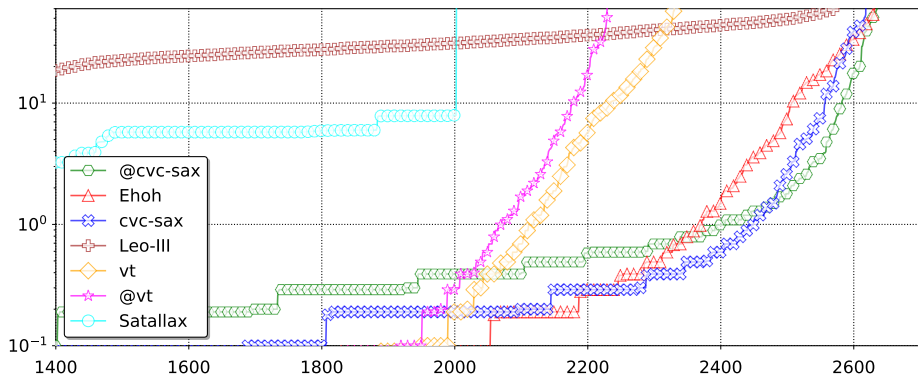
- ▷ Ground solver uses two term representations
 - ▶ Curried for EUF
 - ▶ Regular for the rest

- ▷ Theory combination and instantiation operate via interface

Evaluation

- ▷ Pragmatic CVC4 and redesigned VERiT
- ▷ Benchmarks
 - ▶ Monomorphic TPTP-THF
 - ▶ Benchmarks from Sledghammer, with 32, 512 and 1024 axioms
- ▷ Compared against
 - ▶ Full encoding-based versions of CVC4 and VERiT
 - ▶ HO-provers Leo-III and Satallax
 - ▶ λ HO-prover Ehoh

Evaluation



- ▷ Solved problems among 5,543 benchmarks supported by all solvers
- ▷ 60s timeout

Evaluation

- ▶ Extended CVC4 complementary to its encoding-based counterpart
- ▶ Both versions of CVC4 on par with Ehoh
- ▶ Extended VERiT clearly ahead of its encoding-based counterpart
- ▶ Leo-III and Satallax much ahead on THF, but fail to scale on Sledghammer problems
- ▶ FO-performance of the extensions is not compromised

Conclusions

- ▷ Successful extensions of SMT solvers to HOL
- ▷ On par with encoding-based approach

Future work

- ▷ Tackle HO-unification
 - ▶ Will allow extending conflict-based instantiation
- ▷ Implement dedicated simplifications