Extending SMT solvers to higher-order logic*

Haniel Barbosa
Andrew Reynolds
Cesare Tinelli

Daniel El Ouraoui

Clark Barrett

The University of Iowa

Inria informatics mathematics

Stanford

SMT 2019

2019–07–07, Lisbon, PT

*To be published in the proceedings of CADE 2019

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, MizAR, 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"

$$\mathsf{f}(\lambda x.\, \mathsf{g}(x) + \mathsf{h}(x)) \simeq \mathsf{f}(\lambda x.\, \mathsf{h}(x) + \mathsf{g}(x))$$

$$\downarrow \text{cong, ext}$$

$$(\forall x.\, \mathsf{g}(x) + \mathsf{h}(x) \simeq \mathsf{h}(x) + \mathsf{g}(x)) \Rightarrow \mathsf{f}(\lambda x.\, \mathsf{g}(x) + \mathsf{h}(x)) \simeq \mathsf{f}(\lambda x.\, \mathsf{h}(x) + \mathsf{g}(x))$$

$$\downarrow \neg, \text{CNF}$$

$$\mathsf{g}(\mathsf{sk}) + \mathsf{h}(\mathsf{sk}) \not\simeq \mathsf{h}(\mathsf{sk}) + \mathsf{g}(\mathsf{sk})$$
$$\mathsf{f}(\lambda x.\, \mathsf{g}(x) + \mathsf{h}(x)) \not\simeq \mathsf{f}(\lambda x.\, \mathsf{h}(x) + \mathsf{g}(x))$$

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	λfHOL	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

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

Fragments of interest

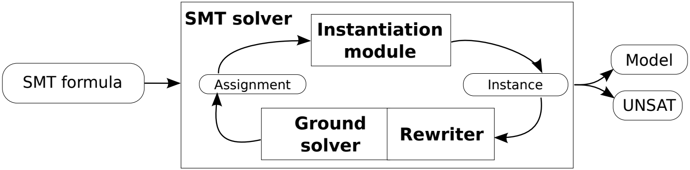
Features	FOL	λfHOL	HOL
function	✓	✓	✓
quantification on objects	✓	✓	✓
quantification on functions	✗	✓	✓
partial applications	✗	✓	✓
anonymous functions	✗	✗	✓

- Henkin semantics
  - Function interpretations restricted to terms expressible in formula's signature
- Extensionality

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

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

A CDCL($\mathcal{T}$) SMT solver



- ▷ Rewriter simplifies terms

  $x + 0 \to x$ $\quad$ $\mathsf{a} \not\simeq \mathsf{a} \to \bot$ $\quad$ $(\text{str.replace } x \ (\text{str.++ } x \ x) \ y) \to x$

- ▷ Ground solver enumerates assignments $\mathsf{E} \cup \mathsf{Q}$
  - E is a set of ground literals $\quad \{\mathsf{a} \leq \mathsf{b},\ \mathsf{b} \leq \mathsf{a} + x,\ x \simeq 0,\ \mathsf{f}(\mathsf{a}) \not\simeq \mathsf{f}(\mathsf{b})\}$
  - Q is a set of quantified clauses $\quad \{\forall xyz.\, \mathsf{f}(x) \not\simeq \mathsf{f}(z) \vee \mathsf{g}(y) \simeq \mathsf{h}(z)\}$

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

A pragmatic extension

- Preprocessing
  - Totalizing applications of theory symbols $$\frac{\varphi[1+]}{\varphi[\lambda x.\, 1 + x]}$$
  - λ-lifting $$\frac{\varphi[\lambda x.\, t]}{\varphi[\mathsf{f}(t)] \wedge \forall x.\, \mathsf{f}(x) \simeq t}$$
- Ground 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

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

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

Lazy applicative encoding

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

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

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

Lazy applicative encoding

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

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

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

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

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

2. Applications of f and g need to be encoded

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

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

   Note that h(a) is not encoded!

Handling extensionality $\forall \bar{x}.\, \mathsf{f}(\bar{x}) \simeq \mathsf{g}(\bar{x}) \leftrightarrow \mathsf{f} \simeq \mathsf{g}$

- "$\leftarrow$" handled by lazy encoding and congruence

$$\dfrac{\dfrac{\dfrac{\mathsf{f} \simeq \mathsf{g}}{@(\mathsf{f},\, t_1) \simeq @(\mathsf{g},\, t_1)}\ \textsc{Cong}}{\cdots}\ \textsc{Cong}}{@(\ldots\ (@(\mathsf{f},\, t_1), \ldots),\, t_n) \simeq @(\ldots\ (@(\mathsf{g},\, t_1), \ldots),\, t_n)}\ \textsc{Cong}$$

- "$\rightarrow$" handled by

$$\dfrac{\mathsf{f} \not\simeq \mathsf{g}}{\mathsf{f}(\mathsf{sk}_1,\, \ldots,\, \mathsf{sk}_n) \not\simeq \mathsf{g}(\mathsf{sk}_1,\, \ldots,\, \mathsf{sk}_n)}\ \textsc{Extensionality}$$

Avoiding exponential model construction

Functions are interpreted as if-then-else:

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

Partial applications can lead to exponentially many cases!

$$\begin{array}{rl} & \mathsf{f}_1(\mathsf{a}) \simeq \mathsf{f}_1(\mathsf{b}) \wedge \mathsf{f}_1(\mathsf{b}) \simeq \mathsf{f}_2 \\ \wedge & \mathsf{f}_2(\mathsf{a}) \simeq \mathsf{f}_2(\mathsf{b}) \wedge \mathsf{f}_2(\mathsf{b}) \simeq \mathsf{f}_3 \\ \wedge & \mathsf{f}_3(\mathsf{a}) \simeq \mathsf{f}_3(\mathsf{b}) \wedge \mathsf{f}_3(\mathsf{b}) \simeq \mathsf{c} \end{array}$$

8 ite entries to model that $\mathsf{f}_1(x, y, z) \simeq \mathsf{c}$, for $x, y, z \in \{\mathsf{a}, \mathsf{b}\}$

Avoiding exponential model construction

Functions are interpreted as if-then-else:

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

Partial applications can lead to exponentially many cases!

$$\begin{array}{rl} & \mathsf{f}_1(\mathsf{a}) \simeq \mathsf{f}_1(\mathsf{b}) \land \mathsf{f}_1(\mathsf{b}) \simeq \mathsf{f}_2 \\ \land & \mathsf{f}_2(\mathsf{a}) \simeq \mathsf{f}_2(\mathsf{b}) \land \mathsf{f}_2(\mathsf{b}) \simeq \mathsf{f}_3 \\ \land & \mathsf{f}_3(\mathsf{a}) \simeq \mathsf{f}_3(\mathsf{b}) \land \mathsf{f}_3(\mathsf{b}) \simeq \mathsf{c} \end{array}$$

8 ite entries to model that $\mathsf{f}_1(x, y, z) \simeq \mathsf{c}$, for $x, y, z \in \{\mathsf{a}, \mathsf{b}\}$

Polynomial construction in the "depth" of functions chain

$$M(\mathsf{f}_1) = \lambda xyz.\, \mathsf{ite}(x \simeq \mathsf{a}, M(\mathsf{f}_2)(y, z), \mathsf{ite}(x \simeq \mathsf{b}, M(\mathsf{f}_2)(y, z), \_))$$

Avoiding exponential model construction

Functions are interpreted as if-then-else:

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

Partial applications can lead to exponentially many cases!

$$\begin{array}{rl} & \mathsf{f}_1(\mathsf{a}) \simeq \mathsf{f}_1(\mathsf{b}) \wedge \mathsf{f}_1(\mathsf{b}) \simeq \mathsf{f}_2 \\ \wedge & \mathsf{f}_2(\mathsf{a}) \simeq \mathsf{f}_2(\mathsf{b}) \wedge \mathsf{f}_2(\mathsf{b}) \simeq \mathsf{f}_3 \\ \wedge & \mathsf{f}_3(\mathsf{a}) \simeq \mathsf{f}_3(\mathsf{b}) \wedge \mathsf{f}_3(\mathsf{b}) \simeq \mathsf{c} \end{array}$$

8 ite entries to model that $\mathsf{f}_1(x, y, z) \simeq \mathsf{c}$, for $x, y, z \in \{\mathsf{a}, \mathsf{b}\}$

Polynomial construction in the "depth" of functions chain

$$M(\mathsf{f}_1) = \lambda xyz.\, \mathsf{ite}(x \simeq \mathsf{a}, M(\mathsf{f}_2)(y, z), \mathsf{ite}(x \simeq \mathsf{b}, M(\mathsf{f}_2)(y, z), \_))$$

$$M(\mathsf{f}_2) = \lambda xy.\; \mathsf{ite}(x \simeq \mathsf{a}, M(\mathsf{f}_3)(y), \mathsf{ite}(x \simeq \mathsf{b}, M(\mathsf{f}_3)(y), \_))$$

Avoiding exponential model construction

Functions are interpreted as if-then-else:

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

Partial applications can lead to exponentially many cases!

$$\begin{array}{ll} & \mathsf{f}_1(\mathsf{a}) \simeq \mathsf{f}_1(\mathsf{b}) \wedge \mathsf{f}_1(\mathsf{b}) \simeq \mathsf{f}_2 \\ \wedge & \mathsf{f}_2(\mathsf{a}) \simeq \mathsf{f}_2(\mathsf{b}) \wedge \mathsf{f}_2(\mathsf{b}) \simeq \mathsf{f}_3 \\ \wedge & \mathsf{f}_3(\mathsf{a}) \simeq \mathsf{f}_3(\mathsf{b}) \wedge \mathsf{f}_3(\mathsf{b}) \simeq \mathsf{c} \end{array}$$

8 ite entries to model that $\mathsf{f}_1(x, y, z) \simeq \mathsf{c}$, for $x, y, z \in \{\mathsf{a}, \mathsf{b}\}$

Polynomial construction in the "depth" of functions chain

$$M(\mathsf{f}_1) = \lambda xyz.\, \mathsf{ite}(x \simeq \mathsf{a}, M(\mathsf{f}_2)(y, z), \mathsf{ite}(x \simeq \mathsf{b}, M(\mathsf{f}_2)(y, z), \_))$$

$$M(\mathsf{f}_2) = \lambda xy.\ \ \mathsf{ite}(x \simeq \mathsf{a}, M(\mathsf{f}_3)(y), \mathsf{ite}(x \simeq \mathsf{b}, M(\mathsf{f}_3)(y), \_))$$

$$M(\mathsf{f}_3) = \lambda x.\quad \mathsf{ite}(x \simeq \mathsf{a}, \mathsf{c}, \mathsf{ite}(x \simeq \mathsf{b}, \mathsf{c}, \_))$$

Extending E-matching

- Since @ is overloaded, matching must account for types of arguments
  - $@(x, \mathsf{a})$ can't match $@(\mathsf{f}, \mathsf{a})$ if x and f of different types

- Indexing robust to mixed partial/total applications
  - In HOL applications with different heads can be equal

$$@(\mathsf{f}, \mathsf{a}) \simeq \mathsf{g} \text{ allows matching } \mathsf{g}(x) \text{ with } \mathsf{f}(\mathsf{a}, \mathsf{b})$$

- HO-E-matching left for future work

Using well-chosen axioms

- Store axiom

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

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

$$\mathsf{a} \not\simeq \mathsf{b} \wedge \forall F, G.\, F \simeq G \text{ is unsatisfiable}$$

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

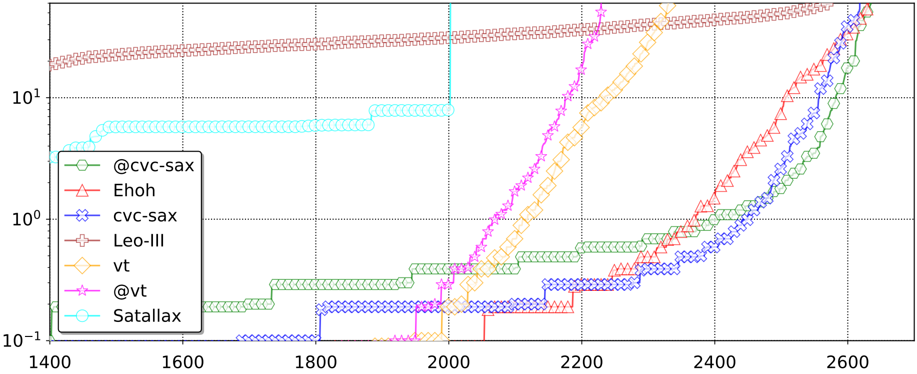
Redesigning the SMT solver

- 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
  - λfHO-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