Challenges and Solutions for Higher-Order SMT Proofs

Chad E. Brown[1] **Mikoláš Janota**[1] Cezary Kaliszyk[2]

SMT 2022

[1] Czech Technical University in Prague
[2] University of Innsbruck

Motivation

- Proofs for SMT are a long-standing challenge

Motivation

- Proofs for SMT are a long-standing challenge
- Semantics is becoming more of a challenge (SMT3)

Motivation

"A significant enabler for the success of SMT has been the SMT-LIB standard input language, which is supported by most SMT solvers. So far, no standard proof format has emerged.

Motivation

"A significant enabler for the success of SMT has been the SMT-LIB standard input language, which is supported by most SMT solvers. So far, no standard proof format has emerged.

This is, however, no accident. Because of the ever increasing number of logical theories supported by SMT solvers, the variety of deductive systems used to describe the various solving algorithms, and the relatively young age of the SMT field, designing a single set of axioms and inference rules that would be a good target for all solvers does not appear to be practically feasible."

— Stump et al., Formal Methods in System Design 2013

Plan

SMT proofs

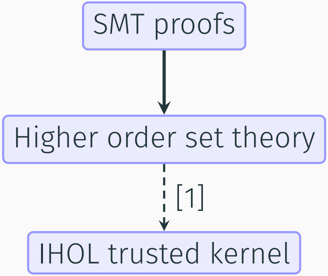
Plan

SMT proofs

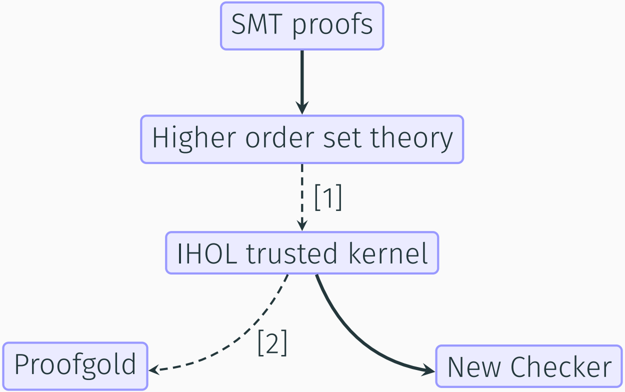
↓

Higher order set theory

Plan



Plan



SMT proofs
Higher order set theory
[1]
IHOL trusted kernel
[2]
Proofgold
New Checker

Flavors of Translation to Set Theory

0	$\{\}$

Flavors of Translation to Set Theory

0	$\{\}$
1	$\{0\}$

Flavors of Translation to Set Theory

0	$\{\}$
1	$\{0\}$
2	$\{0, 1\}$

Flavors of Translation to Set Theory

0	$\{\}$
1	$\{0\}$
2	$\{0, 1\}$
$\mathbb{B}$ (`Bool`)	2

Flavors of Translation to Set Theory

0	$\{\}$
1	$\{0\}$
2	$\{0, 1\}$
$\mathbb{B}$ (`Bool`)	2
$\mathbb{Z}$ (`Int`)	$\omega \cup \{-n \mid n \in \omega\}$

Flavors of Translation to Set Theory

0	$\{\}$
1	$\{0\}$
2	$\{0, 1\}$
$\mathbb{B}$ (`Bool`)	2
$\mathbb{Z}$ (`Int`)	$\omega \cup \{-n \mid n \in \omega\}$
$f : \mathbb{Z} \to \mathbb{B}$	$f \in \mathbb{B}^{\mathbb{Z}}$

Flavors of Translation to Set Theory

0	$\{\}$
1	$\{0\}$
2	$\{0, 1\}$
$\mathbb{B}$ (`Bool`)	2
$\mathbb{Z}$ (`Int`)	$\omega \cup \{-n \mid n \in \omega\}$
$f : \mathbb{Z} \to \mathbb{B}$	$f \in \mathbb{B}^{\mathbb{Z}}$
p is true	$0 \in p$

(Trusted) Kernel

- intuitionistic higher logic as the underlying trusted kernel

$$\frac{}{\Gamma \vdash \mathrm{Known}_s : s}\ s \in \mathcal{A} \qquad \frac{}{\Gamma \vdash u : s}\ u : s \in \Gamma \qquad \frac{\Gamma \vdash \mathcal{D} : s}{\Gamma \vdash \mathcal{D} : t}\ s{\approx}t$$

$$\frac{\Gamma, u : s \vdash \mathcal{D} : t}{\Gamma \vdash (\lambda u : s.\mathcal{D}) : s \to t} \qquad \frac{\Gamma \vdash \mathcal{D} : s \to t \qquad \Gamma \vdash \mathcal{E} : s}{\Gamma \vdash (\mathcal{D}\mathcal{E}) : t}$$

$$\frac{\Gamma \vdash \mathcal{D} : s \qquad x \in \mathcal{V}_\alpha \setminus \mathcal{F}\Gamma}{\Gamma \vdash (\lambda x.\mathcal{D}) : \forall x.s} \qquad \frac{\Gamma \vdash \mathcal{D} : \forall x.s \qquad x \in \mathcal{V}_\alpha, t \in \Lambda_\alpha}{\Gamma \vdash (\mathcal{D}t) : s_t^x}$$

$$\frac{f, g \in \mathcal{V}_{\alpha\beta}\ \text{distinct}, x \in \mathcal{V}_\alpha}{\Gamma \vdash \mathrm{Ext}_{\alpha,\beta} : (\forall fg.(\forall x.fx = gx) \to f = g)}$$

(Trusted) Kernel

- intuitionistic higher logic as the underlying trusted kernel
- each proof gives a proof term

$$\frac{}{\Gamma \vdash \mathrm{Known}_s : s}\ s \in \mathcal{A} \qquad \frac{}{\Gamma \vdash u : s}\ u : s \in \Gamma \qquad \frac{\Gamma \vdash \mathcal{D} : s}{\Gamma \vdash \mathcal{D} : t}\ s \approx t$$

$$\frac{\Gamma, u : s \vdash \mathcal{D} : t}{\Gamma \vdash (\lambda u : s.\mathcal{D}) : s \to t} \qquad \frac{\Gamma \vdash \mathcal{D} : s \to t \qquad \Gamma \vdash \mathcal{E} : s}{\Gamma \vdash (\mathcal{D}\mathcal{E}) : t}$$

$$\frac{\Gamma \vdash \mathcal{D} : s \qquad x \in \mathcal{V}_\alpha \setminus \mathcal{F}\Gamma}{\Gamma \vdash (\lambda x.\mathcal{D}) : \forall x.s} \qquad \frac{\Gamma \vdash \mathcal{D} : \forall x.s \qquad x \in \mathcal{V}_\alpha, t \in \Lambda_\alpha}{\Gamma \vdash (\mathcal{D}t) : s_t^x}$$

$$\frac{f, g \in \mathcal{V}_{\alpha\beta}\ \text{distinct}, x \in \mathcal{V}_\alpha}{\Gamma \vdash \mathrm{Ext}_{\alpha,\beta} : (\forall fg.(\forall x.fx = gx) \to f = g)}$$

Toy Example

```
(declare-fun p () Bool)
(assert p)
(assert (not p))
```

- (declare-fun p () Bool) $\ldots$ $p \in 2$

Toy Example

```
(declare-fun p () Bool)
(assert p)
(assert (not p))
```

- (declare-fun p () Bool) . . . $p \in 2$
- (assert p) . . . $0 \in p$

Toy Example

```
(declare-fun p () Bool)
(assert p)
(assert (not p))
```

- (declare-fun p () Bool) . . . $p \in 2$
- (assert p) . . . $0 \in p$
- (assert (not p)) . . . $0 \notin p$

Toy Example

```
(declare-fun p () Bool)
(assert p)
(assert (not p))
```

- (**declare-fun** p () **Bool**) $\ldots$ $p \in 2$
- (**assert** p) $\ldots$ $0 \in p$
- (**assert** (**not** p)) $\ldots$ $0 \notin p$
- Prove UNSAT: $\forall p \in 2.\, (0 \in p \to 0 \notin p \to \bot)$

Toy Example

```
(declare-fun p () Bool)
(assert p)
(assert (not p))
```

- **(declare-fun** p () **Bool)** ... $p \in 2$
- **(assert** p) ... $0 \in p$
- **(assert (not** p)) ... $0 \notin p$
- Prove UNSAT: $\forall p \in 2. (0 \in p \to 0 \notin p \to \bot)$
- Remarks: $\to$ is IHOL built-in, $\in$ is axiomatized

Toy Example

```
(declare-fun p () Bool)
(assert p)
(assert (not p))
```

- **(declare-fun** p () **Bool)** $\ldots$ $p \in 2$
- **(assert** p) $\ldots$ $0 \in p$
- **(assert (not** p)) $\ldots$ $0 \notin p$
- Prove UNSAT: $\forall p \in 2. (0 \in p \rightarrow 0 \notin p \rightarrow \bot)$
- Remarks: $\rightarrow$ is IHOL built-in, $\in$ is axiomatized
- Proof term: $\lambda p : \iota.\lambda u : p \in 2.\lambda v : 0 \in p.\lambda w : 0 \notin p.w\ v$

Schroeder-Bernstein for Arrays

- Schroeder-Bernstein:
  Injections α to β and β to α imply existence of a bijection

Schroeder-Bernstein for Arrays

- Schroeder-Bernstein:
  Injections α to β and β to α imply existence of a bijection
- In SMT2, for arrays:

Schroeder-Bernstein for Arrays

- Schroeder-Bernstein:
  Injections α to β and β to α imply existence of a bijection
- In SMT2, for arrays:
  - f injective array `Int` to `Int`

Schroeder-Bernstein for Arrays

- Schroeder-Bernstein:
  Injections α to β and β to α imply existence of a bijection
- In SMT2, for arrays:
  - f injective array `Int` to `Int`
  - g injective array `Int` to `Int`

Schroeder-Bernstein for Arrays

- Schroeder-Bernstein:
  Injections α to β and β to α imply existence of a bijection
- In SMT2, for arrays:
  - f injective array `Int` to `Int`
  - g injective array `Int` to `Int`
  - there is no array h bijective from `Int` to `Int`

Failure of Schroeder-Bernstein for Arrays

- Let A be injective array:

…	-2	-1	0	1	2	…
…	3	1	0	2	4	…

- Let A be injective array:

...	-2	-1	0	1	2	...
...	3	1	0	2	4	...

- Define universe U as finite modifications to A.

Failure of Schroeder-Bernstein for Arrays

- Let A be injective array:

...	-2	-1	0	1	2	...
...	3	1	0	2	4	...

- Define universe U as finite modifications to A.
- Arrays in U have a lower bound, no array in U is bijective.

Failure of Schroeder-Bernstein for Arrays

- Let A be injective array:

...	-2	-1	0	1	2	...
...	3	1	0	2	4	...

- Define universe U as finite modifications to A.
- Arrays in U have a lower bound, no array in U is bijective.
- Here existence of model

Failure of Schroeder-Bernstein for Arrays

- Let A be injective array:

...	-2	-1	0	1	2	...
...	3	1	0	2	4	...

- Define universe U as finite modifications to A.
- Arrays in U have a lower bound, no array in U is bijective.
- Here existence of model
    - by proving negation of the original.

Failure of Schroeder-Bernstein for Arrays

- Let A be injective array:

…	-2	-1	0	1	2	…
…	3	1	0	2	4	…

- Define universe U as finite modifications to A.
- Arrays in U have a lower bound, no array in U is bijective.
- Here existence of model
    - by proving negation of the original.
    - Not always possible — despite complete calculus.

Failure of Schroeder-Bernstein for Arrays

- Let A be injective array:

...	-2	-1	0	1	2	...
...	3	1	0	2	4	...

- Define universe U as finite modifications to A.
- Arrays in U have a lower bound, no array in U is bijective.
- Here existence of model
    - by proving negation of the original.
    - Not always possible — despite complete calculus.
- Are we happy about this result?

Alternatives: Directly to Calculus of Inductive Constructions

- Map SMT types to CIC types

Alternatives: Directly to Calculus of Inductive Constructions

- Map SMT types to CIC types
- Use e.g. Coq as checker

Alternatives: Directly to Calculus of Inductive Constructions

- Map SMT types to CIC types
- Use e.g. Coq as checker
- Example issue: `Bool` vs. `Prop`
  $p = (\forall i.\ i < 0)$
  (**declare-fun** `p` () **Bool**) (**assert** (= `p` (**forall** ((i **Int**)) (< i 0))))

Alternatives: Directly to Calculus of Inductive Constructions

- Map SMT types to CIC types
- Use e.g. Coq as checker
- Example issue: `Bool` vs. `Prop`
  $p = (\forall i.\ i < 0)$
  (**declare-fun** p () **Bool**) (**assert** (= p (**forall** ((i **Int**)) (< i 0))))
- Example issue: Type-checking of parametric bitvectors
  Type-checks? $\texttt{bv}_1[n]\ \texttt{++}\ \texttt{bv}_2[m] = \texttt{bv}_2[m]\ \texttt{++}\ \texttt{bv}_1[n]$

Alternatives: Go to Set Theory but Use CIC as Kernel

- Possible

Alternatives: Go to Set Theory but Use CIC as Kernel

- Possible
- No clear advantage

Alternatives: Go to Set Theory but Use CIC as Kernel

- Possible
- No clear advantage
- Heavier kernel

Current Infrastructure

- Megalodon: proof assistant for IHOL

Current Infrastructure

- Megalodon: proof assistant for IHOL
- Proofgold: blockchain with proofs
  (distributed mathematics)

Current Infrastructure

- Megalodon: proof assistant for IHOL
- Proofgold: blockchain with proofs
  (distributed mathematics)
- Proofgold checker: part of Proofgold

Current Infrastructure

- Megalodon: proof assistant for IHOL
- Proofgold: blockchain with proofs
  (distributed mathematics)
- Proofgold checker: part of Proofgold
- New checker: can be used independently

Conclusion

- Question:
  What is a good proof-theoretical framework for (new) SMT?

Conclusion

- Question:
  What is a good proof-theoretical framework for (new) SMT?
- Proposal:
  Higher order set theory
  axiomatized in intuitionistic higher order logic,
  obtaining small trusted kernel

Conclusion

- Question:
  What is a good proof-theoretical framework for (new) SMT?
- Proposal:
  Higher order set theory
  axiomatized in intuitionistic higher order logic,
  obtaining small trusted kernel
- Natural translation of SMT concepts to sets.

Conclusion

- Question:
  What is a good proof-theoretical framework for (new) SMT?
- Proposal:
  Higher order set theory
  axiomatized in intuitionistic higher order logic,
  obtaining small trusted kernel
- Natural translation of SMT concepts to sets.
- Feasibility on concrete examples and available checkers.

Chad E. Brown and Karol Pąk.
A tale of two set theories.
In Cezary Kaliszyk, Edwin C. Brady, Andrea Kohlhase, and Claudio Sacerdoti Coen, editors, *Intelligent Computer Mathematics - 12th International Conference, CICM 2019, Prague, Czech Republic, July 8-12, 2019, Proceedings*, volume 11617 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2019.

Bill White.
Qeditas: A formal library as a bitcoin spin-off, 2016.