Conservative Definitions for Higher-order Logic with Overloading

Arve Gengelbach[1] Johannes Åman Pohjola[2] Tjark Weber[1]

[1] UPPSALA UNIVERSITET

[2] DATA 61

TCS Seminar, KTH

May 24, 2021

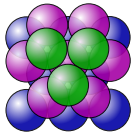
Interactive Theorem Proving



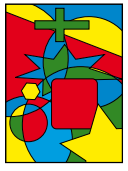
Interactive theorem proving allows to verify complex systems with respect to strong specifications.



Some Landmark Formalizations



Kepler conjecture



Four color theorem

- Odd order theorem
- ...

OS kernel (seL4)

ML compiler (CakeML)

- C compiler (CompCert)
- ...

Proof Assistants

Proof assistants are software tools that assist with the development of formal (machine-readable) models and proofs.

ACL2

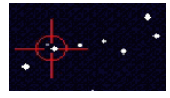
Agda

Coq

HOL4

Isabelle

Lean



Mizar

PVS

. . . and many others

Who Guards the Guardians?

Proof assistants are the guardians of truth. But who guards the guardians?

If you could prove False in your favorite proof assistant, you could prove anything.

Inconsistency Club

- Coq 8.4pl2 (Maxime Dénès and Daniel Scheple, 2013)

```
Hypothesis Heq : (False → False) = True.
Fixpoint contradiction (u : True) : False := contradiction
  ( match Heq in (_ = T) return T with | eq_refl => 
    fun f : False ⇒ match f with end end ).

Lemma foo : provable_prop_extensionality → False.
<four-line proof>
```

- Isabelle 2013-2 (Ondřej Kunčar, 2014)

```
consts c :: bool
typedef T = {True, c} by blast
defs c_bool_def : c :: bool ≡ ¬(∀(x :: T) y. x = y)

lemma aux: (∀(x :: T) y. x = y) ⟷ c
<one-line proof>

theorem False
<one-line proof>
```

(Examples courtesy of Andrei Popescu.)

Does (In)Consistency Matter?

Maybe not all *that* much:

- Honest users don't exploit inconsistencies in proofs.
- Bugs that allow to prove False are usually easily patched.

But:

- Automated provers know nothing about honesty.
- The trust story for proof assistants becomes complicated.

I am [...] part of the team that attempts to get a common criteria (CC EAL5) evaluation for PikeOS through, where the models and proofs were done with Isabelle. [...] I had a lengthy debate with Evaluators [...] which became aware [of a proof of False].

Burkhart Wolff

Threats to Consistency

- Implementation bugs in the user interface or tool layer
- Implementation bugs in the logical kernel
- Logical flaws

Threats to Consistency

- Implementation bugs in the user interface or tool layer
- Implementation bugs in the logical kernel
- Logical flaws
  - Axioms
  - Inference rules
  - Definitional mechanism

The Definitional Mechanism

Introducing New Symbols

- The user asserts axioms that describe properties of the new symbols.
  - Unchecked by the proof assistant
  - Very flexible
  - Users frequently (inadvertently) write inconsistent axioms
- The user states definitions of the new symbols.
  - Checked by the proof assistant
  - Cannot introduce inconsistencies (or can they?)
  - Less expressive than arbitrary axioms

The Definitional Mechanism

Users are strongly encouraged to work definitionally, rather than axiomatically. Popular proof assistants strive to make the definitional mechanism as expressive and convenient as possible.

Example

```
datatype 'a list = Nil | Cons 'a 'a list

fun append :: 'a list ⇒ 'a list ⇒ 'a list
where
  append Nil ys = ys
| append (Cons x xs) ys = Cons x (append xs ys)
```

Overloading

Isabelle/HOL implements higher-order logic with overloading. Users can declare (polymorphic) constants, and later define different instances.

Example

```
consts size :: 'a → nat

overloading size_prod ≡ size :: 'a × 'b → nat
            size_list ≡ size :: 'a list → nat
begin
  fun size_prod where size_prod (a, b) = size a + size b
  fun size_list where size_list xs = sum_list (map size xs)
end
```

Overloading enables Haskell-style type classes.

Checks for (Overloaded) Definitions

1. Definitions must be orthogonal, i.e., must not have a common instance.

Example (BAD)

$$\begin{array}{lllll} \text{overloading} & \text{size\_boolpair} & \equiv & \text{size} & :: \ \text{bool} \times 'a \to \text{nat} \\ & \text{size\_pairbool} & \equiv & \text{size} & :: \ 'a \times \text{bool} \to \text{nat} \end{array}$$

Checks for (Overloaded) Definitions

1. Definitions must be orthogonal, i.e., must not have a common instance.

Example (BAD)

overloading size_boolpair $\equiv$ size :: bool $\times$ 'a $\to$ nat
size_pairbool $\equiv$ size :: 'a $\times$ bool $\to$ nat

2. There must be no cyclic dependencies between symbols.

Example (BAD)

consts c :: bool
typedef T = {True, c} by blast
defs c_bool_def : c :: bool $\equiv \neg(\forall(x :: T)\ y.\ x = y)$

Here, $c_{bool} \rightsquigarrow T \rightsquigarrow c_{bool}$.

Checks for (Overloaded) Definitions

1. Definitions must be orthogonal, i.e., must not have a common instance.

Example (BAD)

$$\begin{array}{lllll} \text{overloading} & \text{size\_boolpair} & \equiv & \text{size} & :: \ \text{bool} \times 'a \to \text{nat} \\ & \text{size\_pairbool} & \equiv & \text{size} & :: \ 'a \times \text{bool} \to \text{nat} \end{array}$$

2. There must be no cyclic dependencies between symbols.

Example (BAD)

$$\begin{array}{l} \text{consts} \ \ c :: \text{bool} \\ \text{typedef} \ \ T = \{\text{True}, c\} \ \text{by blast} \\ \text{defs} \ \ \text{c\_bool\_def}: \ c :: \text{bool} \equiv \neg(\forall (x :: T)\ y.\ x = y) \end{array}$$

Here, $c_{\text{bool}} \rightsquigarrow T \rightsquigarrow c_{\text{bool}}$.

Are these two checks sufficient to guarantee consistency (and stronger properties, e.g., conservativity)?

Results

Model-theoretic Conservativity

Theorem [LSFA'17]

Let T, T' be definitional theories with $T \subseteq T'$. Let U be the set of symbols defined in $T' \setminus T$.
Every model $\mathcal{M}$ of T can be extended to a model $\mathcal{M}'$ of T' such that $\mathcal{M}$ and $\mathcal{M}'$ agree on the interpretation of all symbols in F_U.

Model-theoretic Conservativity

Theorem [LSFA'17]

Let T, T' be definitional theories with $T \subseteq T'$. Let U be the set of symbols defined in $T' \setminus T$.
Every model $\mathcal{M}$ of T can be extended to a model $\mathcal{M}'$ of T' such that $\mathcal{M}$ and $\mathcal{M}'$ agree on the interpretation of all symbols in F_U.

Corollary (Kunčar and Popescu, ITP 2015)

Every definitional theory has a model.

Corollary

Every definitional theory is consistent.

The Independent Fragment

Definition

Let U be a set of symbols. The *U-independent fragment* is

$$F_U := \mathsf{Symb} \setminus \{x \mid \exists u \in U, \rho.\ x \rightsquigarrow^{\downarrow *} \rho(u)\}$$

F_U contains all symbols that do not depend on an instance of a symbol in U.

The Independent Fragment

Definition

Let U be a set of symbols. The *U-independent fragment* is

$$F_U := \mathsf{Symb} \setminus \{x \mid \exists u \in U, \rho.\ x \rightsquigarrow^{\downarrow *} \rho(u)\}$$

F_U contains all symbols that do not depend on an instance of a symbol in U.

Example

Consider the theory $\{\mathsf{c}_\alpha \equiv \mathsf{d}_\alpha,\ \mathsf{d}_{\mathsf{bool}} \equiv \mathrm{True}\}$ and $U = \{\mathsf{d}_{\mathsf{bool}}\}$:

- $\mathsf{d}_{\mathsf{bool}} \notin F_U$ because $\mathsf{d}_{\mathsf{bool}} \rightsquigarrow^{\downarrow *} \mathsf{d}_{\mathsf{bool}}$
- $\mathsf{c}_{\mathsf{bool}} \notin F_U$ because $\mathsf{c}_{\mathsf{bool}} \rightsquigarrow^{\downarrow} \mathsf{d}_{\mathsf{bool}}$ because $\mathsf{c}_\alpha \rightsquigarrow \mathsf{d}_\alpha$
- $\mathsf{c}_{\mathsf{bool}\to\mathsf{bool}} \in F_U$ because $\mathsf{c}_{\mathsf{bool}\to\mathsf{bool}} \not\rightsquigarrow^{\downarrow *} \mathsf{d}_{\mathsf{bool}}$

Proof-theoretic Conservativity

[ICTAC'20]

Theorem

Let T, T' be definitional theories with $T \subseteq T'$. Let U be the set of symbols defined in $T' \setminus T$.
For any formula φ whose symbols are from F_U, we have

$$T \vdash \varphi \iff T' \vdash \varphi$$

Definitions are not required to prove statements that do not depend on the newly defined symbols.

Proof-theoretic Conservativity

[ICTAC'20]

Theorem

Let T, T' be definitional theories with $T \subseteq T'$. Let U be the set of symbols defined in $T' \setminus T$.
For any formula φ whose symbols are from F_U, we have

$$T \vdash \varphi \iff T' \vdash \varphi$$

Definitions are not required to prove statements that do not depend on the newly defined symbols.

Example (Consistency)

Consider any definitional theory T'. Then

$$\emptyset \vdash \mathrm{False} \iff T' \vdash \mathrm{False}$$

(and since $\emptyset \nvdash \mathrm{False}$, it follows that $T' \nvdash \mathrm{False}$).

Conclusion

- For HOL with overloading, extensions by definitions are model- and proof-theoretically conservative.
- We have generalized the model-theoretic conservativity result to constant *specifications*, and mechanized it in HOL4.

[LPAR'20] [LFMTP'20]

Future work:

- A formally verified *algorithm* to check orthogonality of definitions and acyclicity of the dependency relation

Thank you!

We are recruiting a 2-year post-doc to apply formal methods to cybersecurity:

`https://uu.varbi.com/se/what:job/jobID:398570/`

Application deadline: May 28