Conservative Definitions for Higher-order Logic with Overloading

Arve Gengelbach\textsuperscript{1}  Johannes Åman Pohjola\textsuperscript{2}  Tjark Weber\textsuperscript{1}

TCS Seminar, KTH
May 24, 2021
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.
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 .

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

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

**Example**

```isar
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)

<table>
<thead>
<tr>
<th>overloading</th>
<th>size_boolpair ≡ size :: bool × 'a → nat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>size_pairbool ≡ size :: 'a × bool → nat</td>
</tr>
</tbody>
</table>
Checks for (Overloaded) Definitions

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

Example (BAD)

\[
\text{overloading size_boolpair} \equiv \text{size} :: \text{bool} \times 'a \rightarrow \text{nat} \\
\text{size_pairbool} \equiv \text{size} :: 'a \times \text{bool} \rightarrow \text{nat}
\]

2. There must be no cyclic dependencies between symbols.

Example (BAD)

\[
\text{consts} \ c :: \text{bool} \\
\text{typedef} \ T = \{\text{True}, \ c\} \ \text{by} \ \text{blast} \\
\text{defs} \ c_{\text{bool_def}} :: c :: \text{bool} \equiv \neg(\forall (x :: T) \ y. \ x = y)
\]

Here, \(c_{\text{bool}} \hookrightarrow T \hookrightarrow c_{\text{bool}}\).
Checks for (Overloaded) Definitions

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

Example (BAD)

overloading  size_boolpair ≡ size :: bool × 'a → nat
size_pairbool ≡ size :: 'a × bool → 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 ≡ ¬(∀(x :: T) y. x = y)

Here, c_bool ↪ T ↪ c_bool.

Are these two checks sufficient to guarantee consistency (and stronger properties, e.g., conservativity)?
Results
Model-theoretic Conservativity

**Theorem**

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**

Let $T, T'$ be definitional theories with $T \subseteq T'$. Let $U$ be the set of symbols defined in $T' \setminus T$.

Every model $M$ of $T$ can be extended to a model $M'$ of $T'$ such that $M$ and $M'$ 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

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

$F_U$ 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

$$F_U := \text{Symb} \setminus \{x \mid \exists u \in U, \rho. \ x \xrightarrow{\downarrow \star} \rho(u)\}$$

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

**Example**

Consider the theory $\{c_\alpha \equiv d_\alpha, d_{\text{bool}} \equiv \text{True}\}$ and $U = \{d_{\text{bool}}\}$:

- $d_{\text{bool}} \notin F_U$ because $d_{\text{bool}} \xrightarrow{\downarrow \star} d_{\text{bool}}$
- $c_{\text{bool}} \notin F_U$ because $c_{\text{bool}} \xrightarrow{\downarrow} d_{\text{bool}}$ because $c_\alpha \xrightarrow{\downarrow} d_\alpha$
- $c_{\text{bool} \rightarrow \text{bool}} \in F_U$ because $c_{\text{bool} \rightarrow \text{bool}} \xrightarrow{\downarrow \star} d_{\text{bool}}$
Proof-theoretic Conservativity

**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 $\varphi$ 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

**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 $\varphi$ 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

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

(and since $\emptyset \not\vdash \text{False}$, it follows that $T' \not\vdash \text{False}$).
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.

Future work:
▶ A formally verified algorithm to check orthogonality of definitions and acyclicity of the dependency relation

[LPAR’20] [LFMTP’20]
Thank you!

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


Application deadline: May 28