Showing a CakeML program is safe

First steps towards a semantic type system for CakeML

Hrutvik Kanabar\textsuperscript{1}

January 23, 2020

University of Kent

\textsuperscript{1}Supervised by Scott Owens.
Supported by the UK Research Institute in Verified Trustworthy Software Systems (VeTSS).
CakeML
Introduction to CakeML

- A verified implementation of ML
- Formally specified
- Implemented using HOL4
- Verified, bootstrappable compiler

Showing a CakeML program is safe – Hrutvik Kanabar
\[ \forall P \cdot \text{semantics}(P) \neq \text{Error} \implies \text{semantics}(P) = \text{semantics}_{x86}(\text{compile}(P)) \]
Guaranteeing Well-defined Semantics

semantics (≠) ≠ Error?
Guaranteeing Well-defined Semantics

\[ \Gamma \vdash e : ? + \text{ sound } (\_ \vdash \_ : \_ ) \]
Guaranteeing Well-defined Semantics

\[ \Gamma \vdash e : ? \quad + \quad \text{sound} \ (\_ \vdash \_ : \_ ) \]

- HOL $\xrightarrow{\text{translate}}$ CakeML + proof
Guaranteeing Well-defined Semantics

- $\Gamma \vdash e : \alpha$ and sound ($\_ \vdash \_ : \_ $)

- HOL $\xrightarrow{\text{translate}}$ CakeML + proof

- $\{ P \} \text{ cf } (e) \{ \lambda v . Q \}$
Guaranteeing Well-defined Semantics

- $\Gamma \vdash e : ? + \text{ sound } (\_ \vdash \_ : \_)$

Fast imperative code?

- HOL $\xrightarrow{\text{translate}}$ CakeML + proof

- $\{P\} \; cf \; (e) \; \{\lambda v . \; Q\}$
Guaranteeing Well-defined Semantics

- $\Gamma \vdash e : ? +$ sound ($\_ \vdash \_ : \_ $)

  Fast imperative code?

- $\text{HOL} \xrightarrow{\text{translate}} \text{CakeML + proof}$

  Non-termination?

- $\begin{align*}
  &\{ P \} \; cf \; (e) \; \{ \lambda v . \; Q \}
\end{align*}$
Guaranteeing Well-defined Semantics

- $\Gamma \vdash e : ?$ + sound ($\_ \vdash \_ : \_ )

Fast imperative code?

- HOL $\xrightarrow{\text{translate}}$ CakeML + proof

Non-termination?

- $\{P\} \ cf \ (e) \ \{\lambda v \ . \ Q\}$ Proof effort?

Showing a CakeML program is safe – Hrutvik Kanabar
Semantic typing
\( \Gamma \vdash e : \tau \)

\( e \) “looks like” it has type \( \tau \)
\[ \Gamma \vdash e : \tau \]

\( e \) “behaves like” it has type \( \tau \)
\[ \Gamma \models e : \tau \]

\( e \) is **safe to use** as if it has type \( \tau \)
Logical Relations

- Type-indexed family of predicates on terms
- Step-indexed (“fuelled”) for impredicativity
- Compositional:
  \[ R_{\tau_1 \rightarrow \tau_2}(e_1) \land R_{\tau_1}(e_2) \implies R_{\tau_2}(e_1 e_2). \]
- We use unary relations so far
The Story So Far... 

- System F with:
  \[ \exists \alpha \cdot \tau, \mu\alpha \cdot \tau \text{ (iso)}, \tau_1 \times \tau_2, \tau_1 + \tau_2, \text{crash} \]
  e.g. if \( i \leq a\text{.length} \) then \( a[i] \) else \text{crash}

- CakeML-like semantics, formalised in HOL4

- A model for our use cases!

Showing a CakeML program is safe – Hrutvik Kanabar
The Story So Far...

• System F with:
  \exists \alpha . \tau, \mu \alpha . \tau \text{ (iso), } \tau_1 \times \tau_2, \tau_1 + \tau_2, \text{ crash}
  e.g. if i \leq a.length then a[i] else crash

• CakeML-like semantics, formalised in HOL4

• A model for our use cases!

Next steps: ref \tau
Use cases
First prove *compatibility lemmas*...
First prove *compatibility lemmas*... 

\[ \vdash \ldots \vdash \implies \models \ldots \models \] 

... then compose **safe** and **unsafe** code, e.g. 

\[
\begin{align*}
\text{user} & \vdash \ldots \vdash \not\models \\
\text{lib} & \\
\end{align*}
\]

\[
\begin{align*}
\text{user} & \vdash \ldots \vdash \models \\
\text{lib} & \\
\end{align*}
\]
We can express invariants as semantic types, and so prove they are preserved.

Candle
A HOL kernel implemented in CakeML. LCF-style – relies on type abstraction for soundness!
Extracting Coq to CakeML

Current unverified extraction to OCaml:

Proposed verified extraction to CakeML:

Showing a CakeML program is safe – Hrutvik Kanabar
Current unverified extraction to OCaml:

\[
\text{Coq} \xrightarrow{\text{extract}} \text{OCaml} + \text{Obj.magic}
\]
Current unverified extraction to OCaml:

\[
\text{Coq} \xrightarrow{\text{extract}} \text{OCaml} + \text{Obj.magic}
\]

Proposed verified extraction to CakeML:

\[
\text{Coq} \xrightarrow{\text{extract}} \text{CakeML} + \text{Obj.magic}
\]

Showing a CakeML program is safe – Hrutvik Kanabar
Thanks for listening!