Project Outline

- Formally verify type safety of System FC using the Coq Proof Assistant
- Type safety guarantees the absence of security holes like buffer overruns
- Verification of System FC has a direct correspondence to verification of Haskell

Haskell

- High-level, statically-typed, functional language
- Used because of “type safety” guarantees
- System FC, the underlying language of Haskell, is not proven to be type safe

GHC Core & System FC

- GHC Core is the implementation of System FC within GHC
- Haskell code is syntactic sugar for GHC Core programs

The Coq Proof Assistant

- Interactive, automated theorem prover
- Proofs are mechanically verified by the Coq interpreter

Glasgow Haskell Compiler (GHC) Stages

Haskell → GHC Core → C-- → LLVM → X86

GHC Core

\[ \text{id} :: \forall a \rightarrow a \]
\[ \text{id} \ x = x \]
\[ y :: \text{Int} \]
\[ y = \text{id} \ 1 \]

System FC

\[ \text{id} := \Delta T . \lambda x : T . x \]
\[ \emptyset \vdash \text{id} : \forall T , T \rightarrow T \]
\[ y := \text{id} \ [\text{Int}] \ 1 \]
\[ \emptyset \vdash y : \text{Int} \]

Progress

Theorem progress : forall t T,
\[ \text{empty} \vdash t \ \text{in} \ T \rightarrow \]
\[ \text{value} \ t \ \not\exists \ t' , t \Longrightarrow t' . \]

Preservation

Theorem preservation : forall Gamma t t' T,
\[ \Gamma \vdash t \ \text{in} \ T \rightarrow \]
\[ t \Longrightarrow t' \rightarrow \]
\[ \Gamma \vdash t' \ \text{in} \ T . \]

Type Safety

- **Progress**: any well-typed expression is either a value, or can continue to be evaluated
- **Preservation**: the type of an expression is preserved after evaluation
- **Soundness**: any well-typed term evaluates to a value

Future Work

- Verify translation to GHC Core
- Use formalization to verify compiler optimizations and extensions to GHC Core