Language-Based Verification Will Change the World

Tim Sheard\textsuperscript{1}  Aaron Stump\textsuperscript{2}  Stephanie Weirich\textsuperscript{3}

\textsuperscript{1}Computer Science  
Portland State University

\textsuperscript{2}Computer Science  
The University of Iowa

\textsuperscript{3}Computer Science  
University of Pennsylvania

U.S. National Science Foundation grant: 0910510
Verification is Powerful!

- Formally verified compilers [Leroy '06].
- Operating systems [Klein et al. '09].
- Relational database management systems [Malecha et al. '10].

- Full power of higher-order logic, type theory.
- Very expressive logical languages.
- Sophisticated theorem-proving environments (Coq, Isabelle, etc.).
Verification is Powerful!

- Amazing *tour de force* examples:
  - Formally verified compilers [Leroy ’06].
  - Operating systems [Klein et al. ’09].
  - Relational database management systems [Malecha et al. ’10].
- Full power of higher-order logic, type theory.
  - Very expressive logical languages.
  - Sophisticated theorem-proving environments (COQ, ISABELLE, etc.).
Verification is Hard.
**Verification is Hard.**

**SEL4: Formal Verification of an OS Kernel [Klein et al ’09]**
- Verified that microkernel refines abstract spec of OS.
- Microkernel: 8700 lines C, 600 assembly.
- Proof: 200,000 lines ISABELLE.
- We estimate: 1 line proof \( \approx \) 10 lines of C.
- So equiv. to around **2 million lines** of C.
- Best paper SOSP 2009.
- True TDF verification.
Verification is Irrelevant.
Verification is Irrelevant.

- Amazing things possible.
- Needed for niche applications (e.g., safety-critical).
- But just too costly for mainstream.
But

What is Verification?

TDF verification, certainly.

But also type checking, static analysis:

▶ data structure invariants.
▶ path properties (e.g., adherence to library protocols).
▶ timing correctness (WCET).
▶ information-flow (security).

JAVA, C# programmers use verification every day!

This is the way forward.
But

What is Verification?

TDF verification, certainly. But also type checking, static analysis:
▶ data structure invariants.
▶ path properties (e.g., adherence to library protocols).
▶ timing correctness (WCET).
▶ information-flow (security).

AVA, C# programmers use verification every day!
This is the way forward.
But

What is Verification?

- TDF verification, certainly.
- But also type checking, static analysis:
  - data structure invariants.
  - path properties (e.g., adherence to library protocols).
  - timing correctness (WCET).
  - information-flow (security).
- JAVA, C# programmers use verification every day!
- This is the way forward.
Language-Based Verification

- Static typing provides light-weight verification.
- Scale up with more expressive types.
- **Dependent types:**
  - \([ \text{"Santa"} , \text{"Fe"} , \text{"NM"} ] : \text{list string} 3\]
  - \(\text{nil} : \text{list 'a 0}\)
  - \(\text{cons} : 'a \rightarrow \text{list 'a 'n} \rightarrow \text{list 'a ('n+1)}\)
  - \(\text{append} : \text{list 'a 'n} \rightarrow \text{list 'a 'm} \rightarrow \text{list 'a ('n+'m)}\)
- Continuum from very light properties to deep ones.
- Incremental verification, pay as you go.
- Familiar language, toolset.
- Catch errors very early in development.
Verification is Everywhere.

- Type checking is verification.
- More advanced typing on the way.
- Coming from Haskell, Scala to Java, C#.
- Dependent types next step in evolution (Coq, Agda, Trellys).
- A true change in the nature of programming.