This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

SAND2022-9448C

Cornell University

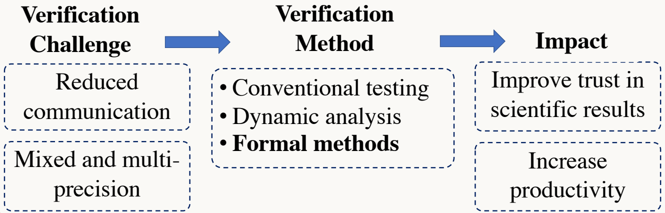
A Verification Toolchain for Numerical Programs

Ariel E. Kellison (with Andrew W. Appel) *

Sandia National Laboratories

1. Motivation

As core numerical algorithms evolve to fully exploit high-performance computers, verifying the accuracy and correctness of their implementations becomes more challenging.



Verification Challenge → **Verification Method** → **Impact**

Verification Challenge:
- Reduced communication
- Mixed and multi-precision

Verification Method:
- Conventional testing
- Dynamic analysis
- **Formal methods**

Impact:
- Improve trust in scientific results
- Increase productivity

2. Formal Methods

Formal verification tools use mathematical logic to *exhaustively check* and *formally prove* properties such as program correctness.

Formally verifying *numerical* software requires reasoning about program properties at least three *layers*:

1. **Language-level properties** such as data-structures, function pointers, data abstraction, and design patterns.
2. **Floating-point properties** such as those described by the IEEE-754 specification of floating-point arithmetic.
3. **Domain specific properties** such as abstractions for derivatives and matrices.

3. Proposed Framework

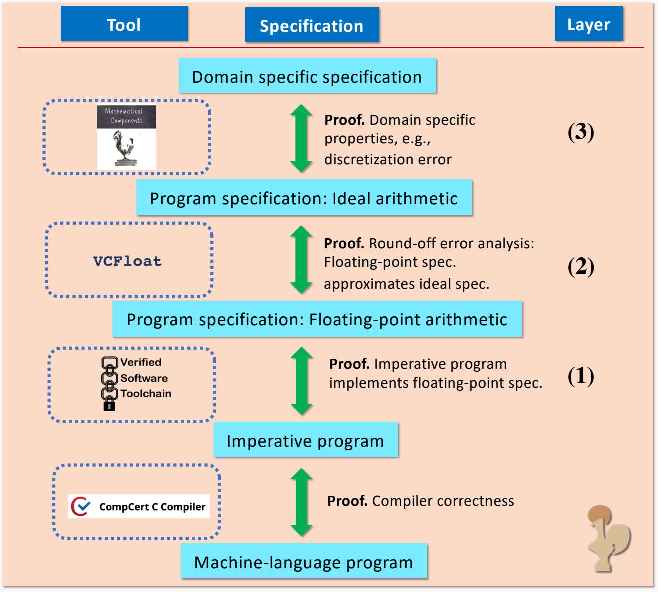
- Develop specifications that encapsulate expected program behavior at each *layer*: **(1)** language-level behavior, **(2)** floating-point behavior, and **(3)** domain specific behavior.
- Formalize specifications and correctness proofs at each layer in a *proof-assistant* ( ) to reduce logical gaps at specification interfaces.
- Compose correctness proofs from each layer.

Guarantees program correctness from the compiled program to domain specific properties.

- Develop embedded tools in the proof-assistant as necessary.

☺ Unlike ***software testing,*** framework not limited by a test set.

☺ Can guarantee ***reproducibility*** and correctness.



Tool | **Specification** | **Layer**

Domain specific specification

Mathematical Components

Proof. Domain specific properties, e.g., discretization error (3)

Program specification: Ideal arithmetic

`VCFloat`

Proof. Round-off error analysis: Floating-point spec. approximates ideal spec. (2)

Program specification: Floating-point arithmetic

Verified Software Toolchain

Proof. Imperative program implements floating-point spec. (1)

Imperative program

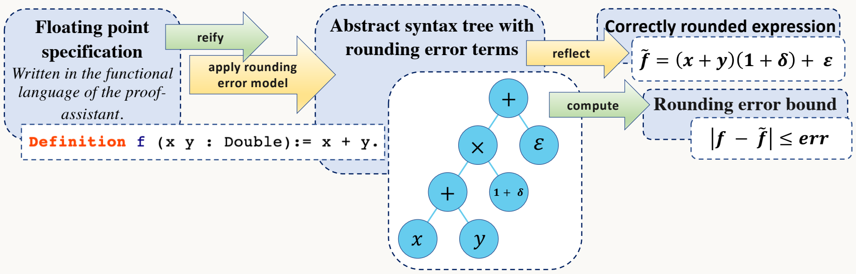
CompCert C Compiler

Proof. Compiler correctness

Machine-language program

4. A Formal Verification Tool for Floating-Point Properties (`VCFloat`)

`VCFloat`: Process floating-point specifications to automatically generate provably correct rounding error bounds [1, 2].



Floating point specification

Written in the functional language of the proof-assistant.

```
Definition f (x y : Double):= x + y.
```

Abstract syntax tree with rounding error terms

Correctly rounded expression

$\tilde{f} = (x+y)(1+\delta) + \varepsilon$

Rounding error bound

$|f - \tilde{f}| \leq err$

5. Ongoing and Future Work

- Framework applied to verifying a symplectic solver for an ODE [3].
  - Challenges: proof automation & extensibility; rounding error bounds on conserved quantities & symplectic error.
- Future Applications: pipelined CG, multiple precision iterative solvers, double-double & quad-double arithmetic packages.

1. "A Unified Coq Framework for Verifying C Programs with Floating-Point Computations," by Tahina Ramananandro et al., in *CPP'16*.
2. "VCFloat2: Floating-point Error Analysis in Coq," by Andrew W. Appel and Ariel E. Kellison, draft, April 2022.
3. "Verified Numerical Methods for Ordinary Differential Equations," by Ariel E. Kellison and Andrew W. Appel, to appear in *NSV'22*.

* In collaboration with
David Bindel, Geoffrey Hulette, and Heidi Thornquist.

DOE CSGF

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.