

Verified Verification: Interval Methods in Proof Assistants

Pieter Collins¹

¹ Department of Advanced Computing Sciences
Maastricht University
Postbus 616
6200MD Maastricht
The Netherlands
`{pieter.collins@maastrichtuniversity.nl}`

Keywords: Proof Assistants, Constructive Analysis, Interval Arithmetic, Taylor Models, Dynamic Systems, Model Checking

Introduction

The aim of rigorous numerical methods such as those based on interval arithmetic is to give guaranteed bounds on the true value of the computed quantity. However, mistakes in the mathematical development of the method, or in the computational implementation, may mean that these guarantees are actually incorrect!

To obtain the best possible guarantees of correctness, we can develop our interval tools in a *proof assistant*, a software package which allows computational algorithms to be written and mathematical theorems about them stated and proved. The major proof assistants currently available are Rocq (formally Coq) [11], Lean [10] and Isabelle/HOL [9]. The proofs are checked by a small, trusted kernel. We call this process *verified verification* — a rigorous-numerics-based model-checking verification tool is itself verified by a logic-based proof assistant.

A motivating example for this work was a bug in the C++ source code of the ARIADNEtool [2] for reachability analysis and model-checking

of dynamic systems. This bug caused the error bounds of function approximations to be too low, and potentially invalid. This bug was not due to the algorithm itself, which had been formally verified [4], but in an incorrect efficiency optimization, illustrating the importance of verifying both code and implementation. As algorithms become more complicated, the potential to make mistakes is amplified.

Immler’s HOL-ODE-Numerics [7], which was used to verify [8] Tucker’s proof [12] of Smale’s 14th problem, is currently the largest body of formally-verified numerics for dynamical systems analysis.

Verified Rigorous Numerics

Our initial work on this project [4] focused on floating-point arithmetic and Taylor polynomial models in Rocq. We developed an abstraction of floating-point arithmetic using upward, downward and nearest rounded arithmetic, but without considering the machine epsilon, and used this to implement interval versions and the exponential function. We then implemented a data-type of Taylor polynomial models with floating-point coefficients, and proved correctness of arithmetical operations. The algorithms verified were those of ARIADNE, but the actual ARIADNE C++ code could not be verified.

The main results are that interval- and function-model based algorithms satisfy the *inclusion* property: If $\hat{X}$ is a computational data type, such as floating-point intervals, representing some mathematical type X , such as real numbers, then a function $f : X_1 \times X_2 \rightarrow Y$ is correctly modelled by a computational operation $\hat{f} : \hat{X}_1 \times \hat{X}_2 \rightarrow \hat{Y}$ if

$$\forall x_{1,2} \in X_{1,2}, \hat{x}_{1,2} \in \hat{X}_{1,2} \hat{x}_1 \ni x_1 \wedge \hat{x}_2 \ni x_2 \implies \hat{f}(\hat{x}_1, \hat{x}_2) \ni f(x_1, x_2).$$

In Rocq, this is described by:

Definition models {IX X : Type} : IX -> X -> Prop.

Proposition op_correct
 (iop : IX1 -> IX2 -> IY) (op : X1 -> X2 -> Y) :=
 forall (ix1 ix2 : IX) (i1 i2 : X),
 models mx1 x1 -> models mx2 x2 ->
 models (iop ix1 ix2) (op x1 x2).

In future work we will to incorporate genuine floating-point arithmetic such as with Rocq’s Flocq library [6]. Once this is in place, we will start developing full model-checking functionality, including differential equations and constraint satisfaction.

Verified Computable Analysis and Systems Theory

To support the rigorous numerics, we are also working on formalising the underlying theory of *computable analysis*, which shows what is possible from arbitrarily-accurate rigorous numerics. We are investigating an approach of finding logical axioms which make the constructive operations of Rocq coincide with the Turing-computable functions. Core data types include the compactified natural number type $\mathbb{N}_\infty$, the Cantor type $\{0, 1\}^\mathbb{N}$ and the Sierpinski type $\{\top, \bot\}$ as classifier for verifiable predicates. Results of [5] show that with the correct axioms, one can show that every constructible function $\mathbb{N} \rightarrow \{0, 1\}$ is continuous.

To support applications to systems theory, we are working on formalising compositional theory of input-output systems [3], with the ultimate goal of developing the theory for hybrid systems [1], which is sufficiently complicated that we are not sure all the results are correct. As part of this effort, we have work on timed-event systems and nondeterministic systems using *monads* as a model of nondeterminism. The formalisation effort uncovered a deep but subtle barrier to developing a compositional systems theory for stochastic systems.

Perspectives

We expect the use of proof assistants to become increasingly prevalent in mathematics and scientific computation. For safety-critical systems and use in complicated mathematical proofs, the benefits of having complete confidence in correctness is very appealing. Further, proof assistants synergise well with generative artificial intelligence, since they will catch machine-generated hallucinations and other mistakes, and encourage collaboration by giving a precisely-specified common frame of reference. The functional nature of these tools may mean that

efficiency is an issue, but this can probably be largely circumvented by designing efficient functional algorithms and exporting to compiled languages, such as Rocq to OCaml.

References

- [1] Davide Bresolin, Pieter Collins, Luca Geretti, Roberto Segala, Tiziano Villa, and Sanja Živanović Gonzalez. A computable and compositional semantics for hybrid systems. *Information and Computation*, 300:105189, 2024.
- [2] Pieter Collins, Luca Geretti, Alberto Casagrande, Davide Bresolin, Sanja Živanović Gonzalez, Ivan Zapreev, and Tiziano Villa. ARIADNE library for verification of cyber-physical systems, 2002–. <https://www.ariadne-cps.org/>.
- [3] Pieter Collins, Bastiaan Laarakker, and Sacha Sindorf. Verification of compositional frameworks in Coq. In *Proc. Mathematical Theory of Networks and Systems*, volume 58 of *IFAC-PapersOnLine*, pages 262–267, 2024.
- [4] Pieter Collins, Milad Niqui, and Nathalie Revol. A validated real function calculus. *Math. Comput. Sci.*, 5:437–467, 2011.
- [5] Martín Escardó. Constructive decidability of classical continuity. *Math. Struct. in Comp. Science*, 25:1578–1589, 2015.
- [6] Flocq (Floats for Coq). <https://flocq.gitlabpages.inria.fr/>.
- [7] Fabian Immler. HOL-ODE-Numerics, 2018–. <https://www.isa-afp.org/theories/hol-ode-numerics/>.
- [8] Fabian Immler. A verified ODE solver and the Lorenz attractor. *J. Autom. Reasoning*, 61:73–111, 2018.
- [9] Isabelle/HOL. <https://isabelle.in.tum.de/>.
- [10] The Lean language. <https://lean-lang.org/>.
- [11] The Rocq Development Team. The Rocq prover. <https://rocq-prover.org/>.
- [12] Warwick Tucker. A rigorous ODE solver and Smale’s 14th problem. *Found. Comp. Math.*, pages 53–117, 2002.