From Coq proofs to certified exact real computation in AERN*

Michal Konečný[1], Sewon Park[2], and Holger Thies[3]

[1]Aston University, UK [2]KAIST, S. Korea [3]Kyoto University, Japan

Software packages for exact real computation like AERN [Kon21] or iRRAM [Mül00] provide high-level abstract data-types for exact manipulation of real numbers. While these types allow to write elegant programs without thinking about rounding errors, subtleties in the semantics of real number computation such as multivaluedness still pose difficulties when trying to write correct code, especially in larger programs. Therefore strong correctness guarantees like formal correctness proofs for programs or their specific parts are highly desirable.

In this work, we use the Coq proof assistant to provide a backend for verifying exact real number computation algorithms and to extract efficient certified programs from the correctness proofs. In contrast to previous work [KST20, STT21], we do not directly formulate a model of computable analysis but instead introduce a new axiomatization of constructive real numbers, formalizing the reals in a conceptually similar way as some mature implementations of exact real computation. Our theory is formally defined on top of a simple type theory inspired by the one used in Coq and we prove its soundness with respect to the realizability interpretation used in computable analysis.

More precisely, we work with a dependent type theory with basic types $0, 1, 2, \mathsf{N}, \mathsf{Z}$, and a universe of classical propositions and axiomatize the represented spaces of the Kleeneans and the reals and computable multivalued functions. To prove soundness of the set of axioms, we extend the standard realizability interpretation of extensional dependent type theories to the category of assemblies over Kleene's second algebra with computable morphisms [Reu99, § 4 and § 5].

We avoid a purely constructive approach since we want to allow a certain amount of classical reasoning in our proofs. That is, we often want to construct a real number that fulfills certain properties and while the construction itself should of course be computational, we do not require the same for proving the properties. To prove such properties, we would like to make use of the many classical mathematical results about the reals available in the Coq standard library instead of reproving them for our real type. To make this possible, we define a "relator" operation that relates results over a classical axiomatization of the reals with equivalent statements over our new type.

*This work has been partially supported by JSPS KAKENHI Grant Number JP20K19744, by the National Research Foundation of Korea (NRF) grants (No. NRF-2016K1A3A7A03950702, NRF-2017R1E1A1A03071032 (MSIT) & No. NRF-2017R1D1A1B05031658 (MOE)) and by the EU's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 731143.

```
Lemma Realmax : forall x y, {z | W_M x y z}.
Proof.
  intros.
  apply mslimit.
  + (* max is single valued predicate *) ...
  + (* construct limit *)
        intros.
        apply (mjoin (x>y - prec n)
                     (y > x - prec n)).
        ++ intros [c1|c2].
           +++ (* when x>y-2^n *)
         exists x. ...
           +++ (* when x<y+2^n *)
         exists y. ...
        ++ apply M_split.
           apply prec_pos.
Defined.
```

```
realmax :: CReal -> CReal -> CReal
realmax x y =
  mslimit (\n ->
    mjoin (\h -> case h of {
                   True -> x;
                   False -> y})
      (m_split x y (((creal 0.5)^) n)))
```

Figure 1: Outline of a Coq proof and corresponding extracted Haskell code

We implemented our axiomatization in the Coq proof assistant[1]. Coq is based on a constructive dependent type theory and therefore provides a natural program extraction mechanism, making it well-suited for our purpose. We extract Haskell code from our proofs where we map primitive operations on constructive reals to operations in the AERN frameworks. We also present some examples of programs generated from our proofs. The extracted programs are quite simple and behave similarly to native implementations in AERN in terms of running time (see Fig. 1 for an example).

References

[Kon21] Michal Konečný. aern2-real: A Haskell library for exact real number computation. `https://hackage.haskell.org/package/aern2-real`, 2021.

[KST20] Michal Konečný, Florian Steinberg, and Holger Thies. Computable analysis for verified exact real computation. In *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[Mül00] Norbert Th Müller. The iRRAM: Exact arithmetic in C++. In *International Workshop on Computability and Complexity in Analysis*, pages 222–252. Springer, 2000.

[Reu99] Bernhard Reus. Realizability models for type theories. *Electronic Notes in Theoretical Computer Science*, 23(1):128–158, 1999.

[STT21] Florian Steinberg, Laurent Thery, and Holger Thies. Computable analysis and notions of continuity in Coq. *Logical Methods in Computer Science*, Volume 17, Issue 2, May 2021.

[1]The source code can be found on `https://github.com/holgerthies/coq-aern`