A type of constructive real numbers for global optimisation

Todd Waugh Ambridge

1 Background and motivation

In recent years, data-types for exact real number computation have been developed in a variety of programming languages for different purposes.

iRRAM, for example, is an efficient, general-purpose library for exact reals implemented in C++ [8]. Leaving specific implementation details to one side, any iRRAM encoding $x\colon \mathbb{R}_i^*$ of a real number $[\![x]\!]\colon \mathbb{R}$ yields a stream of dyadic-rational intervals $[l_n, r_n]$ (i.e. $(l_n, r_n)\colon \mathbb{Z}[\frac{1}{2}] \times \mathbb{Z}[\frac{1}{2}] = (\frac{a_n}{2^{b_n}}, \frac{c_n}{2^{d_n}})$ for $a_n, b_n, c_n, d_n\colon \mathbb{Z}$) that converges to $[\![x]\!]$. Note that iRRAM deliberately imposes minimal restrictions on these streams – for example "nested intervals or a given convergence speed are not necessary" – in order to prioritise efficiency [8] [1]. Bauer & Kavkler note one reasoning for not requiring a given convergence speed: representing dyadic rationals as reals would require streams that purposefully feature inexact approximations [1].

A similar, more bespoke data-type was developed by Hans-J. Boehm in Java in the 90s, which has since been built on and now underlies Google's Android calculator app [2] [3]. A Boehm encoding $x\colon \mathbb{R}_b^*$ of a real number $[\![x]\!]\colon \mathbb{R}$ is a stream of integers $\{..., x_{-1}, x_0, x_1, ...\}\colon \mathbb{Z} \to \mathbb{Z}$ such that for any "precision-level" $n\colon \mathbb{N}$, $[\![x]\!] \in [\frac{2x_n - 1}{2^{n-1}}, \frac{2x_n + 1}{2^{n-1}}]$. Note that there is a trivial function of type $\mathbb{R}_b^* \to \mathbb{R}_i^*$, as we are still approximating via dyadic rational intervals, and that nested intervals are also not used by Boehm. The intervals here do however have a pre-determined width: each approximation x_n refers to an interval of width 2^n, and thus have a given convergence rate of $\frac{1}{2}$. Boehm resolves the aforementioned 'inexact approximations' situation by adding "flags" to the Java objects encoding real numbers that can inform the program that the given real number exactly represents, for example, a dyadic rational [3]. Boehm further preserves efficiency by having each object cache the best approximation requested of it previously [2].

We have developed an alternative encoding of real numbers in constructive type theory based on Boehm's encodings for the purpose of performing global optimisation of functions on compact intervals, and therefore we employ both a given convergence speed *and* nested intervals. We develop our work in constructive type theory (MLTT), the language of both homotopy type theory and the formal proof assistant Agda [9]. In order to formalise our proofs in Agda, we choose not to invoke Markov's principle (by repeatedly increasing the precision of our input encodings until we reach a desired output precision) and instead require explicit continuity information for our encodings of real functions.

2 Ternary Boehm encodings of real numbers

We employ Boehm's technique, but change the interval representation to ensure each dyadic rational interval is a strict subset of that on the previous precision-level.

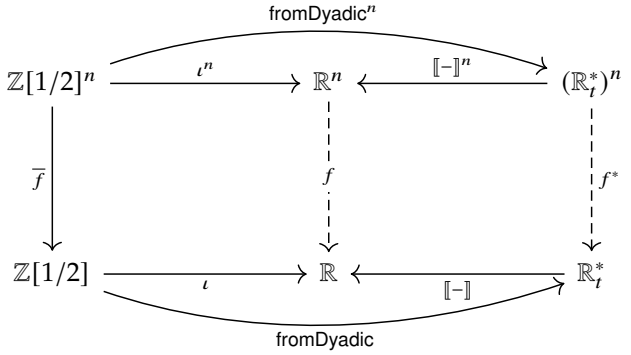
A ternary Boehm encoding $x\colon \mathbb{R}_t^*$ of a real number $[\![x]\!]\colon \mathbb{R}$ is a stream of integers $\{..., x_{-1}, x_0, x_1, ...\}\colon \mathbb{Z} \to \mathbb{Z}$ such that for any $n\colon \mathbb{N}$, $[\![x]\!] \in [\frac{x_n}{2^n}, \frac{x_n + 2}{2^n}]$. As an example, an approximation of π starting at precision-level 0 is $\{3, 6, 12, 25, ...\}$, which corresponds to the interval approximation $\{[3, 5], [3, 4], [3, 3.5], [3.125, 3.375], ...\}$.

Note that given a dyadic interval of the form $[\frac{k}{2^p}, \frac{k+2}{2^p}]$, there are exactly three nested sub-intervals at the next precision-level $p + 1$: $[\frac{2k}{2^{p+1}}, \frac{2k+2}{2^{p+1}}]$, $[\frac{2k+1}{2^{p+1}}, \frac{2k+3}{2^{p+1}}]$ and $[\frac{2k+2}{2^{p+1}}, \frac{2k+4}{2^{p+1}}]$. This ternary structure is by design, and is inspired by using signed-digit representations for real numbers in compact intervals [4]. The type of ternary Boehm reals can be nicely defined in constructive type theory:

$$\mathbb{R}_t^* := \sum_{(x\colon \mathbb{Z} \to \mathbb{Z})} \left(\prod_{(n\colon \mathbb{Z})} (2x_{n-1} \leq x_n \leq 2x_{n-1} + 2) \right).$$

We define the conversion map for all dyadic rationals $\mathsf{fromDyadic}\colon \mathbb{Z}[\frac{1}{2}] \to \mathbb{R}_t^*$ as $\mathsf{fromDyadic}(\frac{k}{2^p})_n := \mathsf{floor}(k2^{n-p})$, which does of course yield the sort of inexact approximations we discussed earlier – though we do not worry about this in our formal type-theoretic setting where inefficiency is less important. For our less formal Java implementation, we utilise Boehm's techniques of flags and caching best approximations.

By taking a function defined on dyadic rationals $\overline{f}\colon \mathbb{Z}[\frac{1}{2}]^n \to \mathbb{Z}[\frac{1}{2}]$, along with some explicit continuity information, we immediately get functions $f\colon \mathbb{R}^n \to \mathbb{R}$ and $f^*\colon (\mathbb{R}_t^*)^n \to \mathbb{R}_t^*$ such that the following diagram commutes:



The continuity information we must provide (e.g. for a unary function) corresponds to the ϵ-δ relationship between the δ-approximation $[\frac{x_\delta}{2^\delta}, \frac{x_\delta+2}{2^\delta}]$ of the input $x\colon \mathbb{R}_t^*$ required to give us the desired ϵ-approximation $[\frac{f(x)_\epsilon}{2^\epsilon}, \frac{f(x)_\epsilon+2}{2^\epsilon}]$ of the output $f(x)$. These functions can then be arbitrarily composed and combined to give new functions; continuity information of these new functions is constructed from those of the constituent functions.

We also have a complete (and unsound) up-to-ϵ order on our encodings $\leq^\epsilon\colon \mathbb{R}_t^* \times \mathbb{R}_t^* \to \Omega$ for any $\epsilon\colon \mathbb{Z}$, where Ω is our type of truth values, defined $x \leq^\epsilon y := x_\epsilon \leq y_\epsilon$. We have that $x \leq^\epsilon y \Leftrightarrow [\![x]\!] - 2^{-e} \leq [\![y]\!]$. For a fixed $y\colon \mathbb{R}_t^*$, if we wish to verify whether the predicate $p(x) := f^*(x) \leq^\epsilon f^*(y)$ holds for some given $x\colon \mathbb{R}_t^*$ and function $f^*\colon \mathbb{R}_t^* \to \mathbb{R}_t^*$ built as above, then we only need to evaluate the δ-approximation x_δ of x, which in turn evaluates $f^*(x)_\epsilon$ which is used in the predicate. We call the highest approximation $\delta\colon \mathbb{Z}$ required of an encoding x in order to judge whether $p(x)$ holds for some predicate $p\colon \mathbb{R}_t^* \to \Omega$ the '*modulus of continuity*' of the predicate.

3 Optimising functions using ternary Boehm reals

Our work utilises the concept of general *searchable types* as introduced by Martín Escardó [5]. A type X is searchable if its elements can be searched by some algorithm $\mathcal{E}_X\colon X \to \Omega$ to find some element that satisfies a given predicate – i.e. for any $p\colon X \to \Omega$ if there is some $x\colon X$ such that $p(x)$, then $p(\mathcal{E}_X(p))$. We view the global optimisation of a function $f\colon X \to Y$ – i.e. finding its minimum or maximum in a given compact interval up to some degree of precision – as an instance of a search problem where $p(x) := \forall(y\colon X).f(x) \leq^\epsilon f(y)$ for some up-to-ϵ ordering of Y [6].

The type $3^{\mathbb{N}} := (\mathbb{N} \to \{-1, 0, 1\})$ of signed-digit encodings is a searchable type; we have previously optimised functions that encode functions on the reals by using signed-digit encodings of real numbers in the compact interval $[-1, 1]$ [6]. Optimising functions using $3^{\mathbb{N}}$ was troublesome, however: the type's encoding of functions must be changed relative to the interval being encoded, the ordering was cumbersome to work with and the search algorithm had to check every combination of elements of a prefix of the stream to check the predicate against that element. We instead developed $\mathbb{R}_t^*$ specifically for function opitmisation.

The type of ternary Boehm encodings that represent real numbers in a given compact dyadic interval $[\frac{k}{2^p}, \frac{k+2}{2^p}]$, where $(k, p)\colon \mathbb{Z} \times \mathbb{Z}$ is defined[1] $\mathbb{R}_t^*(k, p) := \sum_{((x,b)\colon \mathbb{R}_t^*)} (x_p = k)$. The type $\mathbb{R}_t^*(k, p)$ is isomorphic to $3^{\mathbb{N}}$ for any $(k, p)\colon \mathbb{Z} \times \mathbb{Z}$, and is therefore a searchable type. However, the resulting search algorithm from this isomorphism isn't ideal as it will check every combination of elements of a prefix of the encoding.

One advantage[2] of encoding with Boehm reals over signed-digit is that n-approximating an encoding $x\colon \mathbb{R}_t^*$ only requires us to evaluate $x_n\colon \mathbb{Z}$, rather than the first n elements of the signed-digit encoding. Using this, we can construct a more direct search algorithm for $\mathbb{R}_t^*(k, p)$ that simply iterates through the finitely-many candidates $x_\delta\colon \mathbb{Z}$, where $\delta\colon \mathbb{Z}$ is the predicate's modulus of continuity, and checks whether an arbitrary real passing through that approximation satisfies the predicate.

Using this exhaustive searcher as a basis, we then introduce *heuristics* that manipulate the order in which the candidates are searched. Finally, a more advanced *branch-and-bound* search algorithm is developed for $\mathbb{R}_t^*(k, p)$ [7]. This algorithm searchers $\mathbb{R}_t^*(k, p)$ in a tree-like structure similar to the original signed-digit searcher, but has the ability to prune this search tree based upon heuristic values to totally avoid invalid search paths.

[1] This does not in fact capture *all* encodings for numbers in that interval, but this does not cause problems for our search algorithms.

[2] Other advantages to ternary Boehm encodings include being able to use functions on $\mathbb{R}_t^*$, a type for encoding *all* real numbers, and therefore not having to specialise them for particular choices of compact intervals.

References

[1] A. Bauer and I. Kavkler. A constructive theory of continuous domains suitable for implementation. *Annals of Pure and Applied Logic*, 159(3):251–267, 2009. Joint Workshop Domains VIII — Computability over Continuous Data Types, Novosibirsk, September 11–15, 2007.

[2] H.-J. Boehm. Small-data computing: Correct calculator arithmetic. *Commun. ACM*, 60(8):44–49, Jul 2017.

[3] H.-J. Boehm. Towards an API for the Real Numbers. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2020, page 562–576, New York, NY, USA, 2020. Association for Computing Machinery.

[4] A. Ciaffaglione and P. Di Gianantonio. A certified, corecursive implementation of exact real numbers. *Theoretical Computer Science*, 351(1):39–51, 2006. Real Numbers and Computers.

[5] M. Escardo. Infinite sets that admit fast exhaustive search. pages 443 – 452, 08 2007.

[6] D. R. Ghica and T. W. Ambridge. Global optimisation with constructive reals. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, 2021.

[7] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[8] N. T. Müller. The iRRAM: Exact Arithmetic in C++. In J. Blanck, V. Brattka, and P. Hertling, editors, *Computability and Complexity in Analysis*, pages 222–252, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[9] T. Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.