Current Research Topics: Computation with Real Numbers

Arno Pauly
Computer Laboratory
University of Cambridge, United Kingdom
Arno.Pauly@cl.cam.ac.uk

1 Motivation

Definition 1. The real numbers $\mathbb{R}$ are the closure of the rational numbers (fractions) $\mathbb{Q}$, i.e. everything that may occur as a limit of a Cauchy sequence of rationals. More applied, real numbers are the objects represented by infinite decimal fractions such as $3.14159\ldots$.

Computation with real numbers is not necessarily as ubiquitous as one might naively expect: Floating point numbers are definitely not an adequate representation of (mathematical) real numbers; so in those applications were floating point numbers **should** be used, a different mathematical model is required. Yet in many cases it is desirable to actually compute with real numbers, either to capitalize on the vast framework of mathematical analysis (including rather simple things such as calculating the area of a circle from its radius), or maybe just to allow the intuition developed during the programmer's mathematical education to be applied.

There is, of course, a significant problem attached to computation with real numbers: As there uncountably many real numbers, we cannot use finite words over a finite alphabet in order to represent them. As the usual decimal representation shows, **infinite** words are sufficient, but how do we compute with those? After all, computation is – to some extent – supposed to be a finite process?

It happens to be that we already know that: Programming languages such as ML offer lazy lists as a construct, which are nothing but infinite sequences over some data type. Also theory has the needed concepts available: A Turing functional (the things we use to define Turing-reducibility, i.e. oracle machines) defines a computable function from $\mathcal{P}(\mathbb{N})$ to $\mathcal{P}(\mathbb{N})$. The reader is invited to verify that both concepts induce the following computability concept:

Definition 2. A function $f :\subseteq \sum^{\mathbb{N}} \to \sum^{\mathbb{N}}$ is computable, if any finite prefix of $f(p)$ can uniformly be computed from some finite prefix of sufficient length of p.

2 Representing Real Numbers

So how exactly do we represent real numbers? A straight-forward choice would be to use the decimal representation. However, the resulting computability structure is not very nice:

Proposition 3. The function $x \mapsto 3x$ is not computable w.r.t. the decimal representation.

Proof. At some point we have to write the first digit of the output, and we have only read a finite prefix of the input so far. Assume this input-prefix is of the form $0.333\ldots3$. If we start the

output with 1., we might find that the input continues as $0.333\ldots30$, and we made a mistake. If we pick 0. instead, we could read $0.333\ldots37$ next, and would also be wrong. Hence, we cannot make a choice that is definitely correct. □

Instead we represent some $x \in \mathbb{R}$ by some sequence $(q_n)_{n\in\mathbb{N}} \in \mathbb{Q}^{\mathbb{N}}$ such that $|q_n - x| < 2^{-n}$, this is called the standard representation of $\mathbb{R}$. Implementation in ML could use lazy lists of pairs of int variables.

Proposition 4. Addition, multiplication, division, subtraction, exp, sin, cos are all computable w.r.t. the standard representation.

If we can **compute** finite output-prefixes from finite input-prefixes, then finite output-prefixes **are determined by** finite input-prefixes. This amounts to continuity on Cantor (or Baire) space. The standard representation of the real number is *admissible*, this allows us to transfer this result to the Euclidean topology:

Proposition 5. Any function on the real numbers that is computable w.r.t the standard representation is continuous w.r.t. the Euclidean topology.

Just as a reduction from the Halting problem is not the only possible way to prove incomputability for functions on the natural numbers, but by far the most common one; incomputability for naturally appearing real functions is almost always due to discontinuity. A typical example of this is:

Corollary 6. Equality of real numbers is undecidable.

3 An example: Zero finding

Our approach to computability on real numbers allows the formation of higher types, i.e. we have a straight-forward way to represent continuous functions on the real numbers. This representation allows evaluating functions on suitable input. With these higher types, problems such as a weak form of the Intermediate Value Theorem are open to a computational treatment:

Problem 7. Given a continuous and strictly monotone functions $f : [0,1] \to \mathbb{R}$ with $f(0) < 0 < f(1)$, find the value $x \in [0,1]$ such that $f(x) = 0$.

The traditional pseudo-algorithmic proof that such a function f indeed has a zero proceeds by bisection: Look at $f(0.5)$. If it's 0, you are done. If $f(0.5) < 0$, then continue on the interval $[0.5, 1]$. If $f(0.5) > 0$, then search in $[0, 0.5]$ next. Iteration of this procedure will either eventually find the zero, or the interval borders will converge to it.

However, as seen above we cannot decide whether some real number is negative, zero or positive, so this procedure is not actually computable. But we can solve the problem: We evaluate f both at 0.3 and at 0.7. Due to the assumptions about f, at least one of the two resulting values is non-zero. In this scenario, we can wait until one of the four cases $f(0.3) < 0$, $f(0.3) > 0$, $f(0.7) < 0$ or $f(0.7) > 0$ is confirmed. Then we pick one of the intervals $[0, 0.3]$, $[0, 0.7]$, $[0.3, 1]$ or $[0.7, 1]$ for the next step. The interval borders will now converge to the zero of f with some guaranteed speed, hence, we can actually compute the zero with this algorithm.

References

[1] Klaus Weihrauch (2000): *Computable Analysis.* Springer-Verlag.