Constructive Analysis and Experimental Mathematics using the Nuprl Proof Assistant

Mark Bickford

March 2, 2016

1 Introduction

In 1967, Errett Bishop's "Foundations of Constructive Analysis"[1] demonstrated that all of the real analysis normally taught in a first year calculus course (and much, much more) could be developed using only constructive methods. The methods used in this fundamental treatise inspired the development of automated proof assistants such as NuPrl and Coq. These proof assistants are now powerful tools that can be used both to develop software that is "correct-by-construction" and also carry out proofs in "pure mathematics" such as the Four Color Theorem and the Feit-Thompson Theorem. These tools support constructive, or intuitionistic, logic with a "proofs-as-programs" paradigm, in which there is an isomorphism between proofs and programs. When used constructively, as intended, these proof assistants will generate a program from a proof. In this article we will present a variant of Bishop's construction of the real numbers and discuss the programs that result from the construction.

To show that a constructive analysis is possible, Bishop was not concerned with the efficiency of the algorithms implicit in his proofs. Since our automated proof assistant can generate and execute the algorithms that result from our proofs, we try to make the algorithms efficient. It is for this reason that we use a variant of Bishop's construction. We also want to show how using a proof assistant that can compute helps us discover efficient algorithms when used in an "experimental" mode. To illustrate this point, after presenting enough of the basic structure of the real numbers, we will show how we discovered constructions for the square root, $\sqrt{x}$, and general kth root, $\sqrt[k]{x}$, of a real number, x, that compute with extreme efficiency.

2 The Constructive Real Numbers

If x is a constructive real number, then, intuitively, there should be an algorithm that computes x to a given accuracy $\frac{1}{n}$. This means that for each $n \in \mathbb{N}$ we can find a rational number q_n such that $q_n - \frac{1}{n} \leq x \leq q_n + \frac{1}{n}$ (in this article, we will take $\mathbb{N} = \{1, 2, 3 \ldots\}$).

We use the symbol $\div$ for integer division; it satisfies the division equation: $n = k(n \div k) + (n \text{ rem } k)$ (with $|n \text{ rem } k| < |k|$). We can use integer division to round off a rational $\frac{a}{b}$ to $\text{round}(k, \frac{a}{b}) = \frac{ka \div b}{k}$, and then $|\frac{a}{b} - \text{round}(k, \frac{a}{b})| \leq \frac{1}{k}$. Let $q'_n = \text{round}(2n, q_{2n})$, then $|q'_n - q_{2n}| \leq \frac{1}{2n}$, and since $q_{2n} - \frac{1}{2n} \leq x \leq q_{2n} + \frac{1}{2n}$, we have $q'_n - \frac{1}{n} \leq x \leq q'_n + \frac{1}{n}$. This shows that we can choose the denominator of the nth approximation to be $2n$.

Thus, a constructive real number, x, has the property that for each $n \in \mathbb{N}$ we can find an integer x_n for which $\frac{x_n}{2n} - \frac{1}{n} \leq x \leq \frac{x_n}{2n} + \frac{1}{n}$, or, equivalently, $x_n - 2 \leq 2nx \leq x_n + 2$. From this property we deduce a regularity of the sequence of integers $x_1, x_2, x_3, \ldots$ Multiplying the inequalities for x_n and x_m by m and n respectively, we get $mx_n - 2m \leq 2mnx \leq mx_n + 2m$ and $nx_m - 2n \leq 2mnx \leq nx_m + 2n$. Thus, $mx_n - 2m \leq nx_m + 2n$ and $nx_m - 2n \leq mx_n + 2m$, so $|mx_n - nx_m| \leq 2(n + m)$.

Definition 1. A sequence $x_1, x_2, x_3, \ldots$ of integers is regular if for all $n, m \in \mathbb{N}$, $|mx_n - nx_m| \leq 2(n + m)$. The sequence is k-regular if for all $n, m \in \mathbb{N}$, $|mx_n - nx_m| \leq 2k(n + m)$.

Our proof assistant speaks the language of type theory, so we will introduce the elements of type theory we need as we go along. The first type we need is the function type $A \to B$. Members of this type are the functions f such that $f(a) \in B$ whenever $a \in A$. This might seem to be the usual set-theoretic definition of a function with domain A and co-domain B, but in constructive logic, every function $f \in A \to B$ is an effective algorithm that computes output $f(a)$ from input a. Thus, to say that we have an algorithm that computes the numbers $x_1, x_2, x_3, \ldots$ is simply to say that $\lambda n.x_n \in \mathbb{N} \to \mathbb{Z}$.

The next type we need is the subset type $\{t : T \mid P(t)\}$. The members of the type $\{t : T | P(t)\}$ are the members of type T that satisfy the proposition P. We will say more about what propositions in type theory are as we proceed, but, for now, think of any definable property. Thus, to say that a (computable) sequence x of integers is regular is to say that
$x \in \{x : \mathbb{N} \to \mathbb{Z} \mid \forall n, m \colon \mathbb{N}.|mx(n) - nx(m)| \leq 2(n + m)\}$.

We have argued that associated to every constructive real number is a computable, regular sequence of integers. The principle of parsimony suggests that this is what a constructive real is: a computable, regular sequence of integers.

Definition 2. The constructive real numbers are the members of the type

$$\mathbb{R} = \{x : \mathbb{N} \to \mathbb{Z} \mid \forall n, m \colon \mathbb{N}.|mx(n) - nx(m)| \leq 2(n + m)\}$$

From now on, we will use the word real to mean constructive real number. So, a real x is a function that satisfies the regularity condition. We can think of $x(n)$ as the numerator of the n^{th} approximation $\frac{x(n)}{2n}$ of the real x, but, in fact, we won't need to introduce the rational numbers at all. We will prove all the properties of the reals using only integers and sequences of integers.

Bishop defined a real to be sequence of rational numbers $q_1, q_2, q_3 \ldots$ such that $\forall n, m \colon \mathbb{N}.\ |q_n - q_m| \leq n^{-1} + m^{-1}$. Our definition of a real results from

Bishop's definition by normalizing the rationals q_n to always have denominator $2n$ and then clearing the denominators in Bishop's regularity condition. This allows us to define the reals, without constructing the rational numbers, directly from the integers and to define all operations on reals using computations on integers. Of course, computations on rationals also reduce to computations on integers, but, after first implementing Bishop's original definitions (which were not meant to be efficient) we discovered that because they used arithmetic on rationals, they were unnecessarily exact. In our variant definitions each operation on reals rounds off the n^{th} approximation so that it corresponds to a rational approximation with denominator of $2n$. This rounding avoids unnecessary accuracy and turns out to make the algorithms more efficient.

3 Equality in the Reals

Two different regular sequences x and y can represent the same real number. This happens when, for every n, the rational intervals $\left[\frac{x(n)}{2n} - \frac{1}{n}, \frac{x(n)}{2n} + \frac{1}{n}\right]$ and $\left[\frac{y(n)}{2n} - \frac{1}{n}, \frac{y(n)}{2n} + \frac{1}{n}\right]$ overlap. Then, clearing denominators, the integer intervals $[x(n) - 2, x(n) + 2]$ and $[y(n) - 2, y(n) + 2]$ overlap, equivalently,
$\forall n\colon N.\ |x(n) - y(n)| \leq 4$.

Definition 3. Real numbers x and y are equal, (x req y), if and only if
$\forall n\colon N.\ |x(n) - y(n)| \leq 4$

We are going to write (x req y), as Bishop does, as $x = y$ but this will give us two meanings for the $=$ symbol. The $x = y$ in the type $\mathbb{R}$ means that x and y are the same regular sequence, i.e. that for every $n \in \mathbb{N}$, $x(n) = y(n)$. This is stronger than the equality condition in definition 3, which says that $x(n)$ and $y(n)$ differ by at most four. We could resolve the ambiguity by changing the equality in the type $\mathbb{R}$ by quotienting with the relation req. However, because there is no algorithm to decide whether (x req y) for arbitrary reals x and y, and because there is no canonical member of an req- equivalence class, Bishop strongly advises against this approach. Luckily, since we are using a proof assistant, we can have two relations, req, and equality in type $\mathbb{R}$, that will both display as $x = y$ but are syntactically distinct relations. Thus, in the proof assistant, there is no ambiguity. In this article, we will clarify as needed, but, as in Bishop's book, the default meaning for $x = y$ will be the (x req y) defined above.

The astute reader will now be puzzled because the relation, req, does not seem to be a transitive relation, yet we are using the symbol $=$ for it. However, the relation req is an equivalence on $\mathbb{R}$, a fact which follows from the regularity condition.

Definition 4. The relation $\text{bnddiff}(x, y) \equiv \exists B\colon \mathbb{N}.\forall n\colon N.\ |x(n) - y(n)| \leq B$ says that the difference between x and y is bounded.

Lemma 5. *For all $x, y \in \mathbb{R}$, (x req y) $\Longleftrightarrow$ $\text{bnddiff}(x, y)$*

Proof. If (x req y) then we may take $B = 4$. In the other direction, suppose $\forall n\colon N.\ |x(n) - y(n)| \leq B$, then, for any $n, m \in \mathbb{N}$, if $5 \leq |x(n) - y(n)|$ then $5m \leq m|x(n) - y(n)| \leq |mx(n) - nx(m)| + |nx(m) - ny(m)| + |ny(m) - my(n)|$, by the triangle inequality. By regularity and our assumption, the right side is $\leq 4(n + m) + nB$. Thus, $5m \leq 4(n + m) + nB$, so, $m \leq (4 + B)n$ and we obtain a contradiction when $m = 1 + (4 + B)n$. Hence, $|x(n) - y(n)| \leq 4$. Note that this proof by contradiction is constructive because, since $|x(n) - y(n)| \in \mathbb{Z}$, we can prove $(5 \leq |x(n) - y(n)|) \vee (|x(n) - y(n)| \leq 4)$. □

Corollary 6. *req is an equivalence relation on* $\mathbb{R}$

Proof. The bounded-difference relation, bnddiff(x, y) is clearly an equivalence relation on integer sequences. □

We will define operations such as $x + y$, $x * y$, $\frac{1}{x}$, and $\sqrt{x}$ on regular integer sequences $x, y \in \mathbb{R}$, but in each case, we must prove the "functionality lemma" that says that equal inputs give equal outputs. For example, the functionality lemma for + states (where = is the req relation)

$$\forall x_1, x_2, y_1, y_2\colon \mathbb{R}.\ (x_1 = x_2\ \wedge\ y_1 = y_2) \Longrightarrow (x_1 + y_1 = x_2 + y_2)$$

Because of Lemma 5, once we prove that $x + y$ is regular, we only need to prove that on equal (i.e. req) inputs, the outputs have a bounded difference.

We first note the straightforward injection of the integers into the reals.

Definition 7. We write $r(k)$ for the sequence $\lambda n.\ 2kn$. It is the real number corresponding to the integer k (because $2kn \div 2n = k$).

4 Adding Reals

How should $x + y$ be defined? If $\frac{x(n)}{2n}$ is near x, and $\frac{y(n)}{2n}$ is near y, then $\frac{x(n)+y(n)}{2n}$ is near $x + y$, so, perhaps, $x + y$ should be the sequence $\lambda n.\ x(n) + y(n)$. The problem is that, for regular x and y, $|m(x(n) + y(n)) - n(x(m) + y(m))| \leq |mx(n) - nx(m)| + |my(n) - ny(m)| \leq 4(n + m)$; so, the sequence $\lambda n.\ x(n) + y(n)$ is 2-regular, but it may not be 1-regular. To get the correct 1-regular sequence, we must "accelerate" the sequence.

Definition 8. The sequence accel$(k, x) = \lambda n.\ x(2kn) \div 2k$ is called the k-acceleration of sequence x.

Lemma 9. *If sequence x is k-regular, then accel(k, x) is regular, and bnddiff(accel$(k, x), x)$.*

Proof. Because x is k-regular, $2k|mx(2kn) - nx(2km)| \leq 2k(2kn + 2km)$, so $|mx(2kn) - nx(2km)| \leq (2kn + 2km)$. Thus,

$$\begin{aligned}
& 2k|m(x(2kn) \div 2k) - n(x(2km) \div 2k)| \\
= \;& |m2k(x(2kn) \div 2k) - n2k(x(2km) \div 2k)| \\
\leq \;& |mx(2kn) - nx(2km)| + m|x(2kn) \text{ rem } 2k| + n|x(2km) \text{ rem } 2k| \\
\leq \;& (2kn + 2km) + 2k(n + m) \\
= \;& 4k(n + m)
\end{aligned}$$

and hence, $|m * \text{accel}(k, x)(n) - n * \text{accel}(k, x)(m)| \leq 2(n + m)$, so $\text{accel}(k, x)$ is regular.

To show that the difference between $\text{accel}(k, x)$ and x is bounded,

$$\begin{aligned}
& 2kn|\text{accel}(k, x)(n) - x(n)| \\
= \;& |n * 2k * x(2kn) \div 2k - 2kn * x(n)| \\
\leq \;& |nx(2kn) - 2knx(n)| + n|x(2kn) \text{ rem } 2k| \\
\leq \;& 2k(n + 2kn) + 2kn \\
= \;& 2kn(2 + 2k)
\end{aligned}$$

hence, $|\text{accel}(k, x)(n) - x(n)| \leq (2 + 2k)$, so we can take $B = (2 + 2k)$ in the definition of $\text{bnddiff}(\text{accel}(k, x), x)$. □

Definition 10. The sum of a list of reals (of length k) is $x_1 + \cdots + x_k = \lambda n.\ \text{accel}(k, x_1(n) + \cdots + x_k(n))$

By the triangle inequality, the point-wise sum of k reals is a k-regular sequence, so by Lemma 9 its k-acceleration is a real. Thus, the algorithm for the n^{th} approximation of the sum of k reals is: compute the $2kn$-th approximation of each real, add them, divide by $2k$. We observe a general pattern: to get the required accuracy in the output, we need sufficient extra accuracy from the inputs so that the final "rounding off" is still correct. Note that the n^{th}approximation of $a + b + c + d$ is

$$(a(8n) + b(8n) + c(8n) + d(8n)) \div 8$$

while the n^{th} approximation of $a + (b + (c + d))$ (where we iterate the sum of two reals, rather than adding all four at once) is

$$(a(4n) + ((b(8n) + ((c(16n) + d(16n)) \div 4) \div 4)) \div 4$$

If we only iterate the binary sum, then when summing many reals (for example, in the Taylor approximation of e^x) we would ask for ever greater accuracy from terms later in the list (and perform many divisions). Our definition of the general sum of k reals avoids this inefficiency, but we have to prove that $x_1 + x_2 \cdots + x_k = x_1 + (x_2 + \cdots + x_k)$. That fact is easy to prove (by induction on k) because, by Lemma 5, we only have to prove that the two sums have a bounded difference, and by Lemma 9, the accelerations used in the definition preserve bounded difference. Thus, the equality reduces to the equality of the point-wise sums, which is trivial. Commutativity and associativity of additions then follow from the fact that our definition of the sum of a list is invariant under permutation.

5 Multiplying Reals

How should $z = x * y$ be defined? We want $\frac{z(n)}{2n} = \frac{x(n)}{2n} * \frac{y(n)}{2n} = \frac{x(n)*y(n)}{2n*2n}$, so $z(n) = (x(n) * y(n)) \div 2n$ is the likely candidate. But, again, we will need to accelerate this sequence to get a regular sequence. For multiplication, unlike addition, the amount by which we must accelerate depends on the reals x and y, in particular, it depends on bounds on the integers $x(n)$ and $y(n)$ for $n \in \mathbb{N}$.

Definition 11. A natural number b bounds a real x if $\forall n\colon \mathbb{N}.\ |x(n)| \leq 2nb$. (Note that $2nb = r(b)(n)$.)

Lemma 12. *The "canonical bound", bound*$(x) = (|x(1)| + 4) \div 2$ *bounds real* x.

Proof. This follows easily from the regularity of x. □

We won't give the details, but the reader can check that $z = \lambda n.\ (x(n) * y(n)) \div 2n$ is $(2k + 1)$-regular provided that k bounds both x and y. So if we accelerate z by $2k + 1$ we get a real number.

Definition 13. The product of reals x and y, is the real number
$x * y = \mathrm{accel}(2(\max(\mathrm{bound}(x), \mathrm{bound}(y))) + 1, \lambda n.\ (x(n) * y(n)) \div 2n)$

For each of the commutative, associative, and distributive laws for multiplication, we have to prove that the difference between two expressions is bounded. Using Lemma 9, we can ignore the accelerations. Using the division equation, we can eliminate the integer divisions (by multiplying both sides by the divisor), introducing extra error terms coming from the remainders. But the remainders are bounded (by the size of the divisor), and the proofs go through easily. By carrying out all these proofs using a proof assistant, we let the proof assistant check all the details in these arguments, and we build up a library of lemmas and proofs. We can reuse parts of the proof of one lemma in the proof of another lemma. We automate reasoning steps that occur often, such as the use of the triangle inequality, with programs, called "tactics", that construct partial proofs of goals that match certain patterns.

Some operations on reals are particularly simple because they don't require any acceleration to generate a regular sequence. For reals x and y, the following operations all define real numbers:

Definition 14. $-x = \lambda n.\ -(x(n))$, $\min(x, y) = \lambda n.\ \min(x(n), y(n))$,
$\max(x, y) = \lambda n.\ \max(x(n), y(n))$, $|x| = \max(x, -x)$

6 Ordering and Completeness

To finish the construction of the real numbers we need to define their order structure, construct the multiplicative inverse $x^{-1} = \frac{1}{x}$ for non-zero x, and show that every Cauchy sequence converges. If real x is less than real y then for some $n \in \mathbb{N}$, $\frac{x(n)}{2n} + \frac{1}{n} < \frac{y(n)}{2n} - \frac{1}{n}$, or, equivalently, $x(n) + 4 < y(n)$. Also, $x \leq y \Leftrightarrow \neg(y < x)$. This motivates the definitions:

Definition 15. $x < y \equiv \exists n\colon \mathbb{N}.\ x(n) + 4 < y(n)$,
$x \leq y \equiv \forall n\colon \mathbb{N}.\ x(n) \leq y(n) + 4$

Note that $(x \leq y\ \wedge\ y \leq x) \Rightarrow x = y$, but $x \leq y$ does not constructively imply $x < y\ \vee\ x = y$. That is because to prove the latter we must either prove $x < y$ or prove $x = y$, and knowing only $x \leq y$ does not tell us which case is true. If we assume $\neg(x < y)$, then $x = y$ follows, but this proof by contradiction is not constructive because there is no algorithm to decide $x < y$, so we may not assume $x < y\ \vee\ \neg(x < y)$. Similarly, there is no algorithm to decide $x = y$, so we may not assume $x = y\ \vee\ \neg(x = y)$. We therefore sometimes need the following stronger evidence that reals are not equal.

Definition 16. Reals x and y are separated (written $x \neq y$) if $x < y\ \vee\ (y < x)$

If $x \in \{x\colon \mathbb{R} \mid x \neq r(0)\}$ then x is separated from zero, and this implies that there is a $k \in \mathbb{N}$ such that $4 < |x(k)|$. We define nonzero(x) to be the least such k; and we can find it by a search starting at $k = 1$ (unfortunately, if x is separated from zero, but very small, this search can be slow). Let $\overline{x} = \lambda n.\ (x(n)$ if $n <$ nonzero(x), 2 otherwise). This is a real equal (i.e req) to x but for which $\overline{x}(n)$ is never zero. We use this to define the inverse of x.

We want $\frac{x^{-1}(n)}{2n} = \frac{2n}{x(n)}$, so $x^{-1}(n)$ should be $4n^2 \div x(n)$, but we must not divide by zero, so we use $\overline{x}(n)$ instead. To get a regular sequence, we must also accelerate this sequence. It turns out that we can prove that accelerating by $4(4k^2 + 1)$ works, where $k =$ nonzero(x).

Definition 17. The inverse of real $x \in \{x\colon \mathbb{R} \mid x \neq r(0)\}$ is
$x^{-1} = \text{accel}(4(4\text{nonzero}(x)^2 + 1), \lambda n.\ 4n^2 \div \overline{x}(n))$

We can prove that if $x \in \{x\colon \mathbb{R} \mid x \neq r(0)\}$ then $x^{-1} \in \mathbb{R}$ and $x * x^{-1} = r(1)$. Because we need to search for nonzero(x), the algorithm for the inverse is the least efficient of the algorithms presented so far. It may be possible to find a better construction for the inverse. This is also one place where the constructive reals differ from the "classical" reals. Classically, every non-zero real has an inverse, but constructively, only the reals that are separated from zero have an inverse.

When $y \neq r(0)$, we define $\frac{x}{y} = x * y^{-1}$. From these definitions we can prove

Lemma 18. *For any real x and any $n \in \mathbb{N}$, $|\frac{r(x(n))}{r(2n)} - x| \leq \frac{r(1)}{r(n)}$*

This shows that what we have be using, informally, as motivation for our definitions, is formally true. Note that our real numbers contain the rational numbers in the form $\frac{r(a)}{r(b)}$ (when $b \neq 0$), and we can prove all the usual rules for rational arithmetic.

A sequence of real numbers is a member of the type $\mathbb{N} \to \mathbb{R}$. We will use capital letters X,Y,... for sequences of reals, so $X, Y \in \mathbb{N} \to \mathbb{R}$. Now we can define when a sequence X is a Cauchy sequence and when a sequence X converges to a real number x.

Definition 19. A sequence X of reals is Cauchy if

$\forall k\colon \mathbb{N}.\ \exists N\colon \mathbb{N}.\ \forall n, m\colon \mathbb{N}.\ (N \leq n\ \wedge\ N \leq m) \Rightarrow |X(n) - X(m)| \leq r(k)^{-1}$.

Sequence X converges to real x (written $\lim_{n\to\infty} X(n) = x$) if

$\forall k\colon \mathbb{N}.\ \exists N\colon \mathbb{N}.\ \forall n\colon \mathbb{N}.\ (N \leq n) \Rightarrow |X(n) - x| \leq r(k)^{-1}$.

Sequence X converges ($X\downarrow$) if $\exists x\colon \mathbb{R}.\ \lim_{n\to\infty} X(n) = x$.

Theorem 20. *(Completeness)* $(X\downarrow) \Leftrightarrow (X$ *is Cauchy*$)$

We won't give the proof of Theorem 20. Our proof comes from the proof in Bishop and Bridges[2] by "clearing denominators". Instead, we will discuss what the type-theoretic meaning of this theorem is and show the program that NuPrl generates from the proof of this theorem.

6.1 Some type theory

Suppose that for every $a \in A$, $B(a)$ is a type. Then we call B a *family of types*, indexed by type A. For any such family, we have the *dependent function type* $a\colon A \to B(a)$ and the *dependent pair type* $a\colon A \times B(a)$. A member f of $a\colon A \to B(a)$ is a function such that $f(a) \in B(a)$ whenever $a \in A$, and a member $\langle a, b\rangle$ of $a\colon A \times B(a)$ is a pair with first component $a \in A$ and second component $b \in B(a)$.

A constructive proof p of a statement $\forall a\colon A.\ P(a)$ constructs, for each $a \in A$, evidence that $P(a)$ is true. If we identify $P(a)$ with the type of evidence for $P(a)$, then the construction can be seen as a function p, defined for $a \in A$, such that $p(a) \in P(a)$. This is the same as saying $p \in a\colon A \to P(a)$. A constructive proof of $\exists a\colon A.\ P(a)$ must construct a witness $a \in A$ and evidence, $b \in P(a)$, that $P(a)$ is true. This is the same as saying that $\langle a, b\rangle \in a\colon A \times P(a)$.

We use the type Unit, which has only one member Ax, as the evidence for simple, checkable, true propositions about numbers, such as $3 < 7$, or $5 = 5$. The type Void, which has no members, is the evidence type for false propositions.

The logic of proof assistants like NuPrl and Coq is based on this "propositions as types" isomorphism:

$$\begin{array}{rcl}
A \wedge B & \longleftrightarrow & A \times B \\
A \Rightarrow B & \longleftrightarrow & (A \rightarrow B) \\
A \vee B & \longleftrightarrow & A + B \\
\forall a\colon A.\ B(a) & \longleftrightarrow & a\colon A \rightarrow B(a) \\
\exists a\colon A.\ B(a) & \longleftrightarrow & a\colon A \times B(a) \\
n = m & \longleftrightarrow & \text{Unit, if n=m, Void, otherwise} \\
n < m & \longleftrightarrow & \text{Unit, if n <m, Void, otherwise} \\
\text{False} & \longleftrightarrow & \text{Void}
\end{array}$$

6.2 Constructive content of Completeness

The proposition $(X \downarrow) \Leftrightarrow (X \text{ is Cauchy})$ is an iff statement of the form $A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$. So the evidence will be a pair of functions $\langle f, g\rangle$ where $f \in (X \downarrow) \Rightarrow (X \text{ is Cauchy})$ and $g \in (X \text{ is Cauchy}) \Rightarrow (X \downarrow)$. The proposition $(X$ is Cauchy) is $\forall k\colon \mathbb{N}.\ \exists N\colon \mathbb{N}.\ \forall n, m\colon \mathbb{N}.\ (N \leq n \wedge N \leq m) \Rightarrow |X(n) - X(m)| \leq r(k)^{-1}$, which has the form $\forall k\colon \mathbb{N}.\ \exists N\colon \mathbb{N}.\ R(k, N)$. The relation $R(k, N)$ is $\forall n, m\colon \mathbb{N}.\ (N \leq n \wedge N \leq m) \Rightarrow |X(n) - X(m)| \leq r(k)^{-1}$, which, by unfolding the definition of $\leq$, is

$$\forall n, m\colon \mathbb{N}.\ (N \leq n \wedge N \leq m) \Rightarrow \forall n'\colon \mathbb{N}.\ |X(n) - X(m)|(n') \leq r(k)^{-1}(n') + 4$$

A member of this type would have to be $\lambda n.\lambda m.\lambda p.\lambda n'$. Ax, because the inner proposition is a relation on numbers. We say that propositions like this have trivial constructive content. Thus, the non-trivial constructive content of (X is Cauchy) is the evidence for $\forall k\colon \mathbb{N}.\ \exists N\colon \mathbb{N}.\ R(k, N)$, and this is a function, $\text{cauchy}_X \in \mathbb{N} \rightarrow \mathbb{N}$. It is the Skolem function such that for $k \in \mathbb{N}$, $R(k, \text{cauchy}_X(k))$. Thus, $\text{cauchy}_X(k)$ is that N past which the reals in the sequence X are pairwise within $\frac{1}{k}$ of each other. This function is called the modulus of Cauchy convergence.

Similarly, the constructive content of $\lim_{n\rightarrow\infty} X(n) = x$ is a function $\mathrm{c}_{X,x}$ of type $\mathbb{N} \rightarrow \mathbb{N}$, the "modulus of convergence" that is the Skolem function for $\forall k\colon \mathbb{N}.\ \exists N\colon \mathbb{N}.\ \forall n\colon \mathbb{N}.\ (N \leq n) \Rightarrow |X(n) - x| \leq r(k)^{-1}$. Evidence for $X \downarrow$ is a pair $\langle x, \mathrm{c}_{X,x}\rangle$, the real x that X converges to, and the modulus of convergence.

When we stated and proved completeness in NuPrl, we used an alternate form for the existential quantifier. Instead of $a\colon A \times B(a)$, we can use the type $\{a\colon A \mid B(a)\}$. Evidence for this alternate form is just an $a \in A$ for which $B(a)$ is true–but the evidence for $B(a)$ is not provided. This form can be used when $B(a)$ has trivial constructive content.

Thus, the evidence (i.e. the program) for completeness will be $\langle f, g\rangle$ where f takes an input $\langle x, \mathrm{c}_{X,x}\rangle$ and produces cauchy_X and g takes input cauchy_X and produces $\langle x, \mathrm{c}_{X,x}\rangle$.

When we ask Nuprl for the evidence for completeness generated from the proof we get $\lambda X.\ \langle f, g\rangle$ where

$$f = \lambda p.\ \text{let } x, c = p \text{ in } \lambda k.\ c(2k)$$

$$g = \lambda cy.\ \langle \text{accel}(2, \lambda n.\ X(cy(n))(n)), \lambda k.\ cy(4k) + 1\rangle\rangle$$

We see from the first function f that given the modulus of convergence c, the modulus of Cauchy convergence is $\lambda k.\ c(2k)$ (independent of the x that X converges to). The second function g shows that from X and its modulus of Cauchy convergence, cy, we construct the real that X converges to by accelerating (by two) the "diagonal" sequence $\lambda n.\ X(cy(n))(n)$. The modulus of convergence is $\lambda k.\ cy(4k) + 1$. (The factors of 2 and 4 that occur in the evidence, come from "$\frac{\epsilon}{2}$ - arguments" used in the proof.)

To construct a real number we often build a sequence of reals X that approximates any real with the desired properties, then prove that X is a Cauchy sequence, which therefore converges to a real r. We can then derive the properties of real r from the approximations in X. This method is used to prove Cantor's theorem (that in any non-trivial interval there are uncountably many reals), and (a version of) the intermediate value theorem. Because we can always generate the constructive content of these proofs, we can compute with all the real numbers we construct.

We have formalized almost all of the material in chapter two of Bishop and Brigdes, convergence tests for series, continuity, derivatives, the chain rule and product rule, Rolle's Theorem, Taylor's theorem, the sine, cosine, and exponential functions, and the Riemann integral. Rather than present more of this material (for which we encourage the interested reader to consult the original text of Bishop and Bridges[2]) we turn next to a discussion of how using a proof assistant gives us the ability to do "experimental mathematics".

7 Constructing roots

There are several ways to construct the square root of a real number b. We can use the intermediate value theorem on the polynomial $x^2 - b$, or we can use Newton's method, $\sqrt{b} = \lim_{n\to\infty} a_n$ where $a_1 = b$ and $2a_{n+1} = a_n + \frac{b}{a_n}$. Neither of these methods produces a very efficient algorithm in general because both methods iterate computations on reals. We wanted to give an efficient construction of the square root operation, and more generally, the k-th root (for $k \in \{2, 3, \dots\}$). The algorithms for addition and multiplication of reals are not iterative; they reduce directly to computations on integers. Could there be a similar algorithm for square root?

The first response is "no" because there is no square root function defined on the integers. But, although most integers do not have exact square roots, they do have approximate square roots–and these can be computed efficiently.

Definition 21. For $k \in \{2, 3, \dots\}$ and non-negative integers z and r, we say that r is the integer kth root of z (and write $r = \text{iroot}(k, z)$) if

$r^k \leq z < (r + 1)^k$

7.1 Fast Integer roots

Every non-negative integer has an integer kth root. The program generated from the following proof (by Christoph Kreitz) of this fact is very efficient.

Lemma 22. $\forall k\colon \{k\colon \mathbb{Z} \mid 2 \le k\}.\ \forall z\colon \mathbb{N}.\ \exists r\colon \mathbb{N}.\ r^k \le z < (r+1)^k$
(in this lemma $\mathbb{N} = \{n\colon \mathbb{Z} \mid 0 \le n\}$*)*

Proof. We use complete induction on z, so we assume that every $z' < z$ has an integer kth root. If $z = 0$ then we take $r = 0$. Otherwise, let $z' = z \div 2^k$, then $z' < z$ so there is an s such that $s^k \le z' < (s+1)^k$. Let $x = 2s$ and $y = 2s+1$. It is easy to check that if $y^k \le z$ then y is the integer kth root of z and otherwise x is the integer kth root. □

The program generated from this proof is

```
$\lambda k.$ let $b$   :=   $2^k$ in
letrec $f(i)$   =   if $i = 0$ then 0
                    else let $i' := i \div b$ in
                    let $x := 2f(i')$ in
                    let $y := x + 1$ in
                    if $y^k \le i$ then $y$ else $x$
    in   $\lambda z.\ f(z)$
```

The recursion in this program comes from the induction used in the proof, and all the other steps were explicit in the proof. This program runs in time proportional to $\frac{\log_2(z)}{k}$ so it is very fast. With $k = 2$ and $z = 2{*}10^{40}$ it computes to 141421356237309504880 in 1703 primitive computation steps.

7.2 A simple k-th root program

If k is even, then we can only construct a kth root for real x when $r(0) \le x$. So, we will first construct $z = \sqrt[k]{|x|}$ for any real x and any $k \in \{2, 3, \dots\}$. What should the nth approximation $z(n)$ be? We want $\frac{(z(n))^k}{(2n)^k} \approx |x| \approx \frac{|x(n^k)|}{2n^k}$, so $(z(n))^k \approx 2^{k-1}|x(n^k)|$. This suggests that the following simple program might construct $\sqrt[k]{|x|}$

$$\text{simpleroot}(k, x) = \lambda n.\ \text{iroot}(k,\ 2^{k-1}|x(n^k)|) =_? \sqrt[k]{|x|}$$

If this construction is correct, then simpleroot(k, x) is a regular sequence, whenever x is regular. If not, then maybe simpleroot(k, x) is a K-regular sequence for some $K = K(k, x)$ in which case the correct program is

$$\text{accel}(K, \text{simpleroot}(k, x))$$

Our first attempts to prove that simpleroot(k, x) was regular or K-regular were unsuccessful so we reluctantly gave up on this construction and, instead, constructed the root as the limit of a Cauchy sequence.

```
near-root(k;p;q;n)  ==
        if  q=1
        then  if  n=1
              then  eval  c  =  2^k  -  1  in
                    eval  a  =  p  *  2  *  c  in
                    eval  d  =  (2  *  c)  -  1  in
                    eval  M  =  iroot(k;|a|  +  d)  +  1  in
                    eval  y  =  ((k  *  2  *  M^k  -  1)  ÷  d)  +  1  in
                    eval  x  =  iroot(k;(|a|  +  d)  *  y^k)  ÷  2  in
                          <if  (p)  <  (0)   then  -x   else  x,  y>
              else  eval  b  =  q  *  n  in
                    eval  c  =  b^k  -  1  in
                    eval  a  =  p  *  n  *  c  in
                    eval  d  =  c  -  1  in
                    eval  M  =  iroot(k;|a|  +  d)  +  1  in
                    eval  y  =  ((k  *  b  *  M^k  -  1)  ÷  d)  +  1  in
                    eval  x  =  iroot(k;(|a|  +  d)  *  y^k)  ÷  b  in
                          <if  (p)  <  (0)   then  -x   else  x,  y>
        else  eval  b  =  q  *  n  in
                        eval  c  =  b^k  -  1  in
                        eval  a  =  p  *  n  *  c  in
                        eval  d  =  c  -  1  in
                        eval  M  =  iroot(k;|a|  +  d)  +  1  in
                        eval  y  =  ((k  *  b  *  M^k  -  1)  ÷  d)  +  1  in
                        eval  x  =  iroot(k;(|a|  +  d)  *  y^k)  ÷  b  in
                            <if  (p)  <  (0)   then  -x   else  x,  y>
```

Figure 1: Program generated from proof that near roots exists

7.3 k-th root program generated from a proof

We used the existence of the integer kth root to prove the existence of a near kth root of a rational number. So, we proved that for any $p \in \mathbb{Z}$ and any $q, n \in \mathbb{N}$ there exists integers a and b such that $|(\frac{r(a)}{r(b)})^k - \frac{r(p)}{r(q)}| < \frac{1}{n}$. We call the program generated from this lemma nearroot(k, p, q, n). It is shown in figure 1. We used the near roots to prove that there is a sequence of (rational) numbers $Q \in \mathbb{N} \to \mathbb{R}$ such that $\lim_{n\to\infty} Q(n)^k = x$. Then we proved ($Q$ is Cauchy). From this we deduce that $\exists z\colon \mathbb{R}.\ \lim_{n\to\infty} Q = z$, and using the properties of limits, that $z^k = x$ and hence $z = \sqrt[k]{x}$ (when k is even, we assumed $r(0) \leq x$ and showed $r(0) \leq z$). We call the program generated from this proof genroot(k, x). It is shown in figure 2.

7.4 Experimental math

To compute (say) 50 decimal digits of a real number x, we let $n = 5 * 10^{49}$ and compute $x(n)$. Because $2n = 10^{50}$, $|x(n) - r(10^{50}) * x| \leq r(1)$, so the last digit of $x(n)$ may be off by one. The efficiency of genroot(k, x) was reasonably good. To

```
λi,x.  eval  x1  = x  in
  λn.eval  m  = 4*n  in
      eval  p = x1(2*(((2*m^i)-1)+1)) in
            if  (p*2*(((2*m^i)-1)+1)) < ((-1)*4*(((2*m^i)-1)+1))
            then if (1*4*(((2*m^i)-1)+1))<(p*2*(((2*m^i)-1)+1))
                 then  eval  r=near-root(i;p;4*(((2*m^i)-1)+1);
                      let  r1,r2=r         2 *(((2*m^i)-1)+1))  in
                      in  eval  k=r2  in
                          if  (k)<(0)
                          then  (-((2*(2*m)*r1)÷(-2)*k))÷4
                          else  ((2*(2*m)*r1)÷2*k÷4)
                  else  eval  r=near-root(i;p;4*(((2*m^i)-1)+1);
                        let  r1,r2=r       2 *(((2*m^i)-1)+1))  in
                        in  eval  k=r2  in
                          if  (k)<(0)
                          then  (-((2*(2*m)*r1)÷(-2)*k))÷4
                          else  ((2*(2*m)*r1)÷2*k÷4)
            else  if  (1*4*(((2*m^i)-1)+1))<(p*2*(((2*m^i)-1)+1))
                  then  eval  r=near-root(i;p;4*(((2*m^i)-1)+1);
                        let  r1,r2=r       2 *(((2*m^i)-1)+1))  in
                        in  eval  k=r2  in
                          if  (k)<(0)
                          then  (-((2*(2*m)*r1)÷(-2)*k))÷4
                          else  ((2*(2*m)*r1)÷2*k÷4)
                 else  ((2*m*0)÷4)
```

Figure 2: Program generated from proof that k-th root exists

compute 50 decimal digits of the 12th root of $r(2)$ took about 530,000 primitive computation steps[1]. However, the same computation using $\text{simpleroot}(12, r(2))$ used only about 7,000 primitive computation steps (two orders of magnitude fewer) and gave the same result! We know that $\text{genroot}(k, x)$ is a provably correct construction of $\sqrt[k]{x}$, but maybe the program $\text{simpleroot}(k, x)$ always computes the same real as $\text{genroot}(k, |x|)$?

We could test this hypothesis experimentally by computation inside the proof assistant. For various choices of k (odd and even) and various choices of x (large and small), we searched for the first n for which the result of the two programs, $\text{genroot}(k, |x|)(n)$ and $\text{simpleroot}(k, x)(n)$, differ by more than one (if they never differ by more than four, then they compute the same real number). We observed them differ by one in some cases, but never by more than one!

The result of this experiment gave us confidence that the simple root construction was correct after all. Previous attempts to prove it correct were hampered by not knowing that it was correct. When several inequalities that would have sufficed to prove the regularity of the sequence turned out to be unprovable, we gave up because we concluded that the sequence might not be regular. Since our experiments now showed that that was unlikely, we were encouraged to find the proof that it is regular.

7.5 Proof of simple root construction

We have to show that if x is a real number (i.e. a regular sequence) then $z = \text{simpleroot}(k, x)$ is a real number, and we have to show that $z^k = |x|$.

To show that z is regular we must show:

$$|m * \text{iroot}(k,\ 2^{k-1}|x(n^k)|) - n * \text{iroot}(k,\ 2^{k-1}|x(m^k)|)| \leq 2(n+m)$$

Let $s = 2^{k-1}$, $i = n^k$, $j = m^k$, $c = j * s * |x(i)|$, and $d = i * s * |x(j)|$. Let $a = m * \text{iroot}(k,\ s * |x(i)|)$ and $b = n * \text{iroot}(k,\ s * |x(j)|)$. Then we must prove $|a - b| \leq 2(n+m)$. Since

$$\text{iroot}(k,\ s * |x(i)|)^k \leq s * |x(i)| < (\text{iroot}(k,\ s * |x(i)| + 1)^k$$

we have, multiplying by $j = m^k$,

$$a^k \leq j * s * |x(i)| < (a+m)^k$$

So, $a^k \leq c < (a+m)^k$, and similarly, $b^k \leq d < (b+n)^k$. By the regularity of $|x|$ we have $|c - d| \leq 2s(i+j)$. By symmetry, we may assume $a \leq b$. Then either $|a - b| \leq 2(n+m)$ (in which case we are done) or $(2n+m) + (a+m) < b$. In the latter case, $((2n+m) + (a+m))^k < b^k \leq d$, so

$$((2n+m) + (a+m))^k - c \leq d - c \leq |d - c| \leq 2^k(i+j)$$

[1]Inside NuPrl, the programs are interpreted (rather than compiled) and this computation takes about 30 seconds.

Now, if $a = 0$, then also $c = 0$, and therefore

$$2^k(i+j) \leq 2^k(n+m)^k < b^k \leq d = |c-d| \leq 2^k(i+j)$$

which is impossible. Hence, $a \neq 0$, and this implies that $m \leq a$, so $2m \leq (a+m)$. The last term in the binomial expansion of $((2n+m)+(a+m))^k$ is $(a+m)^k$ which is greater than c, and all the other terms are $\geq$ the corresponding term in the binomial expansion of $((2n+m)+2m)^k$. Thus,

$$(2n+3m)^k - (2m)^k \leq ((2n+m)+(a+m))^k - c \leq 2^k(i+j) = (2n)^k + (2m)^k$$

Then

$$(2n)^k + (3m)^k \leq (2n+3m)^k \leq (2n)^k + 2(2m)^k$$

so, $(3m)^k \leq 2(2m)^k$, which implies $3^k \leq 2^{k+1}$, and it is easy to prove that this is false when $k \geq 2$; so we have shown that $z = \text{simpleroot}(k,x)$ is a real number.

To show that $z^k = |x|$ we only have to show $\text{bnddiff}(z^k, |x|)$. With some work, it follows from the definition of multiplication that it is enough to find a bound B such that, for all $n \in \mathbb{N}$,

$$|(z(n))^k \div (2n)^{k-1} - |x(n)|| \leq B$$

As above, set $s = 2^{k-1}$ and $i = n^k$; and, in addition, set $u = |x(i)|$, $v = |x(n)|$, and $R = z(n) = \text{iroot}(k,\ s*u)$. Then

$$\begin{aligned}
s*i*|R^k \div (2n)^{k-1} - v| &= \\
|n*(R^k - (R^k \text{ rem } (2n)^{k-1})) - s*i*v| &\leq \\
|nR^k - snu| + |snu - siv| + n|R^k \text{ rem } (2n)^{k-1}| &\leq \\
n|R^k - su| + 2s(i+n) + si &\leq \\
n((R+1)^k - R^k) + 5si &\leq \\
n*k*(R+1)^{k-1} + 5si &\leq \\
n*k*(2R)^{k-1} + 5si &\leq \\
si*k*(2^*\text{bound}(z))^{k-1} + 5si &
\end{aligned}$$

Thus, $|R^k \div (2n)^{k-1} - v| \leq k*(2^*\text{bound}(z))^{k-1} + 5$, so we can choose $B = k*(2^*\text{bound}(z))^{k-1} + 5$.

That completes the proof that $\text{simpleroot}(k,x) = \sqrt[k]{|x|}$. We extended this construction to get $\sqrt[k]{x}$ (i.e. to handle x, rather than $|x|$ when k is odd) but we won't give the details here. While the proof just given is completely elementary, it is a bit tricky, which explains why we missed it initially. Because our experimental results convinced us that the result was true, we were able to work harder and find the proof. Our proof assistant does not automatically find proofs like this, but it does automatically verify some of the arithmetic reasoning. And, it checks all the details and does not allow us to leave out any cases.

References

[1] E. Bishop. *Foundations of Constructive Analysis.* McGraw Hill, NY, 1967.

[2] E. Bishop and D. Bridges. *Constructive Analysis.* Springer, New York, 1985.