Inverting monotone continuous functions in constructive analysis

Helmut Schwichtenberg

Mathematisches Institut der Universität München

CiE, Swansea, 3. July 2006

Contents

Motivation

- "Mathematics as a numerical language".
- Extract programs from proofs, for exact real numbers.
- Special emphasis on low type level witnesses (making use of separability).

Tools

. . . for algorithmically reasonable proofs: Small variants of Bishop/Bridges' development of constructive analysis.

Idea: use separability to avoid high type levels. Where?

- "Order located" instead of "totally bounded".
- Continuity in $\mathbb{R}$, and $\mathbb{R}^2$.
- Uniformly convergent sequences of functions.

Reals

A real number x is a pair $((a_n)_{n\in\mathbb{N}}, \alpha)$ with $a_n \in \mathbb{Q}$ and $\alpha\colon \mathbb{N} \to \mathbb{N}$ such that $(a_n)_n$ is a Cauchy sequence with modulus α, that is

$$\forall_{k,n,m} \left(\alpha(k) \leq n, m \to |a_n - a_m| \leq 2^{-k}\right),$$

and α is weakly increasing.

Two reals $x := ((a_n)_n, \alpha)$, $y := ((b_n)_n, \beta)$ are equivalent (written $x = y$), if

$$\forall_k |a_{\alpha(k+1)} - b_{\beta(k+1)}| \leq 2^{-k}.$$

Nonnegative and positive reals

A real $x := ((a_n)_n, \alpha)$ is nonnegative (written $x \in \mathbb{R}^{0+}$) if

$$\forall_k \; -2^{-k} \leq a_{\alpha(k)}.$$

It is k-positive (written $x \in_k \mathbb{R}^+$) if

$$2^{-k} \leq a_{\alpha(k+1)}.$$

$x \in \mathbb{R}^{0+}$ and $x \in_k \mathbb{R}^+$ are compatible with equivalence.
Can define $x \mapsto k_x$ such that $a_n \leq 2^{k_x}$ for all n.
However, $x \mapsto k_x$ is not compatible with equivalence.

Arithmetical functions

Given $x := ((a_n)_n, \alpha)$ and $y := ((b_n)_n, \beta)$, define

z	c_n	$\gamma(k)$
$x + y$	$a_n + b_n$	$\max(\alpha(k+1), \beta(k+1))$
$-x$	$-a_n$	$\alpha(k)$
$\lvert x \rvert$	$\lvert a_n \rvert$	$\alpha(k)$
$x \cdot y$	$a_n \cdot b_n$	$\max(\alpha(k+1+k_{\lvert y \rvert}), \beta(k+1+k_{\lvert x \rvert}))$
$\frac{1}{x}$ for $\lvert x \rvert \in_I \mathbb{R}^+$	$\begin{cases} \frac{1}{a_n} & \text{if } a_n \neq 0 \\ 0 & \text{if } a_n = 0 \end{cases}$	$\alpha(2(I+1)+k)$

Cleaning up a real

After some computations involving reals, rationals in the Cauchy sequences may become complex. Hence: clean up a real, as follows.

Lemma

For every real $x = ((a_n)_n, \alpha)$ we can construct an equivalent real $y = ((b_n)_n, \beta)$ where the rationals b_n are of the form $c_n/2^n$ with integers c_n, and with modulus $\beta(k) = k + 2$.

Proof.

$c_n := \lfloor a_{\alpha(n)} \cdot 2^n \rfloor$. □

Redundant dyadic representation of reals

The existence of the usual b-adic representation of reals cannot be proved constructively ($1.000\ldots$ vs $.999\ldots$). Cure: in addition to $0, \ldots, b-1$ also admit -1 as a numeral. For $b = 2$:

Lemma

Every real x can be represented in the form

$$\sum_{n=-k}^{\infty} a_n 2^{-n} \quad \textit{with } a_n \in \{-1, 0, 1\}.$$

Notice: uniqueness is lost (this is not a problem).

Comparison of reals

Write $x \leq y$ for $y - x \in \mathbb{R}^{0+}$ and $x < y$ for $y - x \in \mathbb{R}^{+}$.

$$x \leq y \leftrightarrow \forall_k \exists_p \forall_{n \geq p}\, a_n \leq b_n + 2^{-k}$$

$$x < y \leftrightarrow \exists_{k,q} \forall_{n \geq q}\, a_n + 2^{-k} \leq b_n$$

Write $x <_{k,q} y$ (or simply $x <_k y$ if q is not needed) when we want to call these witnesses. Notice:

$$x \leq y \leftrightarrow y \not< x.$$

Continuous functions

A continuous function $f : I \to \mathbb{R}$ on a compact interval I with rational end points is given by

- an approximating map $h_f : (I \cap \mathbb{Q}) \times \mathbb{N} \to \mathbb{Q}$,
- a (uniform) modulus map $\alpha_f : \mathbb{N} \to \mathbb{N}$ such that $(h_f(c, n))_n$ is a real with modulus α_f;
- $\omega_f : \mathbb{N} \to \mathbb{N}$ (uniform) modulus of continuity:

$$|a - b| \leq 2^{-\omega_f(k)+1} \to |h_f(a, n) - h_f(b, n)| \leq 2^{-k}$$

for $n \geq \alpha_f(k)$. α_f, ω_f required to be weakly increasing.

Notice: h_f, α_f, ω_f are of type level 1 only.

Application of a continuous function to a real

Definition

Given a continuous function f (by h_f, α_f, ω_f) and a real $x := ((a_n)_n, \alpha)$, application $f(x)$ is defined to be

$$(h_f(a_n, n))_n$$

with modulus $k \mapsto \max(\alpha_f(k+2), \alpha(\omega_f(k+1)-1))$.

Lemma

$$x = y \to f(x) = f(y),$$
$$|x - y| \leq 2^{-\omega_f(k)} \to |f(x) - f(y)| \leq 2^{-k}.$$

Intermediate value theorem

Let $a < b$ be rationals. If $f : [a, b] \to \mathbb{R}$ is continuous with $f(a) \leq 0 \leq f(b)$, and with a uniform lower bound on its slope, then we can find $x \in [a, b]$ such that $f(x) = 0$.

Proof sketch.

1. Approximate Splitting Principle. Let x, y, z be given with $x < y$. Then either $z \leq y$ or $x \leq z$.
2. IVTAux. Assume $a \leq c < d \leq b$, say $2^{-n} < d - c$, and $f(c) \leq 0 \leq f(d)$. Construct c_1, d_1 with $d_1 - c_1 = \frac{2}{3}(d - c)$, such that $a \leq c \leq c_1 < d_1 \leq d \leq b$ and $f(c_1) \leq 0 \leq f(d_1)$.
3. IVTcds. Iterate the step $c, d \mapsto c_1, d_1$ in IVTAux.

Let $x = (c_n)_n$ and $y = (d_n)_n$ with the obvious modulus. As f is continuous, $f(x) = 0 = f(y)$ for the real number $x = y$. □

Inverse functions

Theorem

Let $f : [a, b] \to \mathbb{R}$ be continuous with a uniform lower bound on its slope. Let $f(a) \leq a' < b' \leq f(b)$. We can find a continuous $g : [a', b'] \to \mathbb{R}$ such that $f(g(y)) = y$ for every $y \in [a', b']$ and $g(f(x)) = x$ for every $x \in [a, b]$ such that $a' \leq f(x) \leq b'$.

Proof sketch.

Let $f(a) \leq a' < b' \leq f(b)$. Construct a continuous $g : [a', b'] \to \mathbb{R}$ by the Intermediate Value Theorem. $\square$

Example: squaring $f : [1, 2] \to [1, 4]$

Given by

- the approximating map $h_f(a, n) := a^2$,
- the uniform Cauchy modulus $\alpha_f(k) := 1$, and
- the modulus $k \mapsto k + 1$ of uniform continuity.

The lower bound on its slope is $l := 0$, because for all $c, d \in [1, 2]$

$$2^{-m} \leq d - c \to c^2 <_m d^2.$$

Then $h_g(u, n) := c_n^{(u)}$, as constructed in the IVT for $x^2 - u$, iterating IVTAux. The Cauchy modulus α_g is such that $(2/3)^n \leq 2^{-k+3}$ for $n \geq \alpha_g(k)$, and the modulus of uniform continuity is $\omega_f(k) := k + 2$.

Program extraction

Formalization: many details. Important: representation of data. Here: direct approach, by explicitely building the required number systems (natural numbers in binary, rationals, reals as Cauchy sequences of rationals with a modulus, continuous functions in the sense of the type-1 representation described above, etc.)

Method of program extraction based on modified realizability

Animation

Suppose a proof of a theorem uses a lemma.

- Then the proof term contains the name of the lemma, say `L`.
- In the term extracted from this proof we want to preserve the structure of the original proof. So we use a new constant `cL` at places where the computational content of the lemma is needed.
- When we want to execute the program, we have to replace the constant `cL` corresponding to a lemma `L` by the extracted program of its proof. This can be achieved by adding computation rules for `cL`.
- We can be rather flexible here and enable/block rewriting by using `animate`/`deanimate` as desired.

Let

It often happens that a subterm has many occurrences in a term, which leads to unwanted recomputations when evaluating it.

- Cure: "optimize" the term after extraction, and replace for instance $M[x := N]$ with many occurrences of x in M by $(\lambda x M)N$ (or a corresponding "let"-expression).
- This can already be done at the proof level: When an object (value of a variable or realizer of a premise) is used more than once, make sure (if necessary by a cut) that the goal has the form $A \to B$ or $\forall_x A$.
- Now use the "identity lemma" $\mathtt{Id}\colon \hat{P} \to \hat{P}$, with a predicate variable $\hat{P}$. Its realizer then has the form $\lambda f, x.fx$.
- If $\mathtt{cId}$ is not animated, the extracted term has the form $\mathtt{cId}(\lambda x M)N$, which is printed as $[\mathtt{let}\ x\ N\ M]$.

Quantifiers without computational content

Besides the usual quantifiers, $\forall$ and $\exists$, Minlog has so-called non-computational quantifiers, $\forall^{\mathrm{nc}}$ and $\exists^{\mathrm{nc}}$, which allow for the extraction of simpler programs.

- The nc-quantifiers, which were first introduced by Berger (1993), can be viewed as a refinement of the Set/Prop distinction in constructive type systems like Coq or Agda.
- Intuitively, a proof of $\forall^{\mathrm{nc}}_x A(x)$ ($A(x)$ non-Harrop) represents a procedure that assigns to every x a proof $M(x)$ of $A(x)$ where $M(x)$ does not make "computational use" of x, i.e., the extracted program $[\![M(x)]\!]$ does not depend on x.
- Dually, a proof of $\exists^{\mathrm{nc}}_x A(x)$ is a proof of $M(x)$ for some x where the witness x is "hidden", that is, not available for computational use.

Conclusion

- Constructive analysis with witnesses of low type level. Type level 1 representation of continuous functions.
- Extraction of reasonable programs is possible.