Proofs, computations and analysis

Helmut Schwichtenberg
(j.w.w. Kenji Miyamoto)

Mathematisches Institut, LMU, München

Computability Theory and Foundations of Mathematics, Tokyo,
19. February 2013

Motivation

Algorithms are viewed as one aspect of proofs in (constructive) analysis. A corresponding program (i.e., a term t in the underlying language) can be extracted from a proof of A, and a proof that t “realizes” A can be generated ($\Rightarrow$ automatic verification).

Data: From free algebras, given by their constructors. Examples:

- finite or infinite lists of signed digits -1, 0, 1 (i.e., reals as streams),
- possibly non well-founded alternating read-write trees (representing uniformly continuous functions).

Tools

- Decorations: $\to^{\mathrm{c}}, \forall^{\mathrm{c}}$ (short: $\to, \forall$) and $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$ for removal of abstract data, and fine-tuning.
- Nested inductive/coinductive definitions of predicates. Their clauses give rise to free algebras. Only here computational content arises.

Computable functionals

- Types: $\iota \mid \rho \to \sigma$. Base types ι: free algebras (e.g., **N**), given by their signature.
- Functionals seen as limits of finite approximations: ideals (Kreisel, Scott, Ershov).
- Computable functionals are r.e. sets of finite approximations (example: fixed point functional).
- Functionals are partial. Total functionals are defined (by induction over the types).

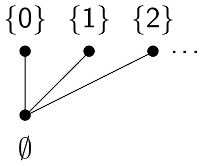
Information systems $\mathbf{C}_\rho$ for partial continuous functionals

- Types ρ, σ, τ: from algebras ι by $\rho \to \sigma$.
- $\mathbf{C}_\rho := (C_\rho, \mathrm{Con}_\rho, \vdash_\rho)$.
- Tokens $a \in C_\rho$ (= atomic pieces of information): constructor trees $\mathrm{C}a_1^*, \dots a_n^*$ with a_i^* a token or $*$. Example: $\mathrm{S}(\mathrm{S}*)$.
- Formal neighborhoods $U \in \mathrm{Con}_\rho$: $\{a_1, \dots, a_n\}$, consistent.
- Entailment $U \vdash_\rho a$.

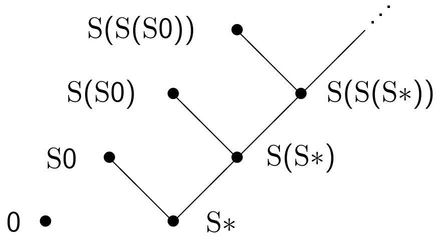
Ideals $x \in |\mathbf{C}_\rho|$ ("points", here: partial continuous functionals): consistent deductively closed sets of tokens.

Flat or non flat algebras?

- Flat:



- Non flat:



Non flat!

- Every constructor C generates an ideal in the function space: $r_{\mathrm{C}} := \{\, (U, \mathrm{C}a^*) \mid U \vdash a^* \,\}$. Associated continuous map:

$$|r_{\mathrm{C}}|(x) = \{\, \mathrm{C}a^* \mid \exists_{U \subseteq x}(U \vdash a^*) \,\}.$$

- Constructors are injective and have disjoint ranges:

$$|r_{\mathrm{C}}|(\vec{x}\,) \subseteq |r_{\mathrm{C}}|(\vec{y}\,) \leftrightarrow \vec{x} \subseteq \vec{y},$$
$$|r_{\mathrm{C}_1}|(\vec{x}\,) \cap |r_{\mathrm{C}_2}|(\vec{y}\,) = \emptyset.$$

- Both properties are false for flat information systems (for them, by monotonicity, constructors need to be strict).

$$|r_{\mathrm{C}}|(\emptyset, y) = \emptyset = |r_{\mathrm{C}}|(x, \emptyset),$$
$$|r_{\mathrm{C}_1}|(\emptyset) = \emptyset = |r_{\mathrm{C}_2}|(\emptyset).$$

A theory of computable functionals, TCF

- A variant of HA^ω.
- Variables range over arbitrary partial continuous functionals.
- Constants for (partial) computable functionals, defined by equations.
- Inductively and coinductively defined predicates. Totality for ground types inductively defined.
- Induction := elimination (or least-fixed-point) axiom for a totality predicate.
- Coinduction := greatest-fixed-point axiom for a coinductively defined predicate.

Relation to type theory

- Main difference: partial functionals are first class citizens.
- Minimal logic: $\to, \forall$ only. $=$ (Leibniz), $\exists$, $\vee$, $\wedge$ (Martin-Löf) inductively defined.
- $\bot := (\mathrm{False} = \mathrm{True})$. Ex-falso-quodlibet: $\bot \to A$ provable.
- Classical logic as a fragment: $\tilde{\exists}_x A$ defined by $\neg \forall_x \neg A$.

Realizability interpretation

- Define a formula $t \mathbf{r} A$, for A a formula and t a term in T^+.
- From a proof M we can extract its computational content, a term $\mathrm{et}(M)$.
- Soundness theorem:
  If M proves A, then $\mathrm{et}(M) \mathbf{r} A$ can be proved.
- Decorations: $\to^{\mathrm{c}}, \forall^{\mathrm{c}}$ (short: $\to, \forall$) and $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$ for removal of abstract data, and fine-tuning:

$$\begin{aligned}
t \mathbf{r} (A \to^{\mathrm{c}} B) &:= \forall_x (x \mathbf{r} A \to tx \mathbf{r} B), \\
t \mathbf{r} (A \to^{\mathrm{nc}} B) &:= \forall_x (x \mathbf{r} A \to t \mathbf{r} B), \\
t \mathbf{r} (\forall^{\mathrm{c}}_x A) &:= \forall_x (tx \mathbf{r} A), \\
t \mathbf{r} (\forall^{\mathrm{nc}}_x A) &:= \forall_x (t \mathbf{r} A).
\end{aligned}$$

Example: decorating the existential quantifier

- $\exists_x A$ is inductively defined by the clause

$$\forall_x (A \to \exists_x A)$$

with least-fixed-point axiom

$$\exists_x A \to \forall_x (A \to P) \to P.$$

- Decoration leads to variants $\exists^{\mathrm{d}}, \exists^{\mathrm{l}}, \exists^{\mathrm{r}}, \exists^{\mathrm{u}}$ (d for "double", l for "left", r for "right" and u for "uniform").

$$\begin{array}{ll} \forall^{\mathrm{c}}_x (A \to^{\mathrm{c}} \exists^{\mathrm{d}}_x A), & \exists^{\mathrm{d}}_x A \to^{\mathrm{c}} \forall^{\mathrm{c}}_x (A \to^{\mathrm{c}} P) \to^{\mathrm{c}} P, \\ \forall^{\mathrm{nc}}_x (A \to^{\mathrm{c}} \exists^{\mathrm{r}}_x A), & \exists^{\mathrm{r}}_x A \to^{\mathrm{c}} \forall^{\mathrm{nc}}_x (A \to^{\mathrm{c}} P) \to^{\mathrm{c}} P. \end{array}$$

Practical aspects

- We need formalized proofs, to allow machine extraction.
- Can't take a proof assistant from the shelf: none fits TCF.

Minlog (`http://www.minlog-system.de`)

- Natural deduction for $\to, \forall$, plus inductively and coinductively defined predicates.
- Partial functionals are first class citizens.
- Allows type and predicate parameters (for abstract developments: groups, fields, reals, . . .).

Uniformly continuous functions

Based on work of Ulrich Berger (2009).

- Extraction from a proof dealing with abstract uniformly continuous functions.
- Data representing uniformly continuous functions: base type cototal ideals.
- The extracted term will involve corecursion.

Type-1 representation of uniformly continuous functions

For contrast: a type-1 represented function $f\colon [-1,1] \to [-1,1]$ is given by

- an approximating map $h\colon [-1,1] \cap \mathbb{Q} \to \mathbb{N} \to \mathbb{Q}$,
- bounds $N, M \in \mathbb{N}$ with $\forall_{a\in[-1,1]}\forall_n (N \leq h(a,n) \leq M)$, and
- a weakly increasing map $\alpha\colon \mathbb{N} \to \mathbb{N}$ such that $(h(a,n))_n$ is a Cauchy sequence with (uniform) modulus α, i.e.,

$$\forall_{a\in[-1,1]}\forall_k\forall_{n,m\geq\alpha(k)}(|h(a,n) - h(a,m)| \leq 2^{-k}).$$

f is (uniformly) continuous if we have a weakly increasing modulus $\omega\colon \mathbb{N} \to \mathbb{N}$ such that

$$\forall_k\forall_{a,b\in[-1,1]}\forall_{n\geq\alpha(k)}(|a-b| \leq 2^{-\omega(k)+1} \to |h(a,n)-h(b,n)| \leq 2^{-k}).$$

Application $f(x)$

Application of f given by h, α and modulus ω to $x := ((a_n)_n, M)$:

$$f(x) := (h(a_n, n))_n$$

with Cauchy modulus $\max(\alpha(k+2), M(\omega(k+1)-1))$.

Intermediate value theorem

Let $a < b$ be rationals. If $f\colon [a, b] \to \mathbb{R}$ is continuous with $f(a) \le 0 \le f(b)$, 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 $z \le y$ or $x \le z$.
2. IVTAux. Assume $a \le c < d \le b$, say $2^{-n} < d - c$, and $f(c) \le 0 \le f(d)$. Construct c_1, d_1 with $d_1 - c_1 = \frac{2}{3}(d - c)$, such that $a \le c \le c_1 < d_1 \le d \le b$ and $f(c_1) \le 0 \le f(d_1)$.
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 $x = y$. $\square$

Extracted term

```
[k0]
 left((cDC rat@@rat)(1@2)
      ([n1]
        (cId rat@@rat=>rat@@rat)
        ([cd3]
          [let cd4
            ((2#3)*left cd3+(1#3)*right cd3@
             (1#3)*left cd3+(2#3)*right cd3)
            [if (0<=(left cd4*left cd4-2+
                     (right cd4*right cd4-2))/2)
             (left cd3@right cd4)
             (left cd4@right cd3)]]))
      (IntToNat(2*k0)))
```

where cDC is a from of the recursion operator.

Free algebra $\mathbf{J}$ of intervals

- $\mathbf{SD} := \{-1, 0, 1\}$ signed digits (or $\{L, M, R\}$).
- $\mathbf{J}$ free algebra of intervals. Constructors

$\mathbb{I}$	the interval $[-1, 1]$,
$\mathrm{C} \colon \mathbf{SD} \to \mathbf{J} \to \mathbf{J}$	left, middle, right half.

  Write $\mathrm{C}_d x$ for $\mathrm{C} d x$.
- $\mathrm{C}_1 \mathbb{I}$ denotes $[0, 1]$.
- $\mathrm{C}_0 \mathbb{I}$ denotes $[-\frac{1}{2}, \frac{1}{2}]$.
- $\mathrm{C}_0(\mathrm{C}_{-1} \mathbb{I})$ denotes $[-\frac{1}{2}, 0]$.

$\mathrm{C}_{d_0}(\mathrm{C}_{d_1} \dots (\mathrm{C}_{d_{k-1}} \mathbb{I}) \dots)$ denotes the interval in $[-1, 1]$ whose reals have a signed digit representation starting with $d_0 d_1 \dots d_{k-1}$.

- We consider ideals $x \in |\mathbf{C}_{\mathbf{J}}|$.

Total and cototal ideals of base type

Generally:

- Cototal ideals x: every token (i.e., constructor tree) $P(*) \in x$ has a "$\succ_1$-successor" $P(\mathrm{C}\vec{*}) \in x$.
- Total ideals: the cototal ones with $\succ_1$ well-founded.

Examples:

- Total ideals of **J**:

$$\mathbb{I}_{\frac{i}{2^k},k} := [\frac{i}{2^k} - \frac{1}{2^k}, \frac{i}{2^k} + \frac{1}{2^k}] \qquad \text{for } -2^k < i < 2^k.$$

- Cototal ideals of **J**: reals in $[-1, 1]$, in (non-unique) stream representation using signed digits $-1, 0, 1$.

Corecursion

- The conversion rules for $\mathcal{R}$ with total ideals as recursion arguments work from the leaves towards the root, and terminate because total ideals are well-founded.
- For cototal ideals (streams) a similar operator is available to define functions with cototal ideals as values: corecursion.
- ${}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{J}} \colon \tau \to (\tau \to \mathbf{U} + \mathbf{SD} \times (\mathbf{J} + \tau)) \to \mathbf{J}$ (**U** unit type).
- Conversion rule

$$\begin{aligned}
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{J}} NM \mapsto [\textbf{case } (MN)^{\mathbf{U}+\mathbf{SD}\times(\mathbf{J}+\tau)} \textbf{ of} \\
&\mathrm{inl}\ \_ \mapsto \mathbb{I} \mid \\
&\mathrm{inr}\langle d, z\rangle \mapsto \mathrm{C}_d[\textbf{case } z^{\mathbf{J}+\tau} \textbf{ of} \\
&\qquad\qquad\qquad \mathrm{inl}\ \_ \mapsto \mathbb{I} \mid \\
&\qquad\qquad\qquad \mathrm{inr}\ u^{\tau} \mapsto {}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{J}} uM]].
\end{aligned}$$

W and continuous real functions

- Consider a well-founded "read tree", i.e., a constructor tree built from R (ternary) with R_d at its leaves.
- The digit d at a leaf means that, after reading all input digits on the path leading to the leaf, the output d is written.
- Let $R_{d_1}, \dots, R_{d_n}$ be all leaves. At a leaf R_{d_i} continue with W (i.e., write d_i), and continue reading.
- Result: a "nested **R**(**W**)-total **W**-cototal" ideal, representing a uniformly continuous real function $f \colon \mathbb{I} \to \mathbb{I}$.

A read-write instruction

```
W      W      W
-      -      -
d      d      d
 \     |     /
   \   |   /      W          W
     \ | /        -          -
       o          d          d
        \         |         /
          \       |       /
            \     |     /
              \   |   /
                \ | /
                  o
```

$\mathbf{R}(\alpha) := \mu_\xi(\alpha \to \xi, \alpha \to \xi, \alpha \to \xi, \xi \to \xi \to \xi \to \xi)$ labelled read-and-finally-write-one-digit trees. Constructors:

$R_d \colon \alpha \to \mathbf{R}(\alpha)$	$(d \in \{-1, 0, 1\})$	finally write d & continue,
$R \colon \mathbf{R}(\alpha) \to \mathbf{R}(\alpha) \to \mathbf{R}(\alpha) \to \mathbf{R}(\alpha)$		read.

Using $\mathbf{R}(\alpha)$ define nested alternating read-write trees

$$\mathbf{W} := \mu_\xi(\xi, \mathbf{R}(\xi) \to \xi)$$

with constructors

$W_0 \colon \mathbf{W}$	Stop,
$W \colon \mathbf{R}(\mathbf{W}) \to \mathbf{W}$	Branch by applying a read-write instruction, and continue.

Want finite read-write instructions, but infinitely many alternations, via a “nested inductive/coinductive” definition.

$\mathrm{Read}(X)$

We give an inductive definition of a unary predicate $\mathrm{Read}(X)$ on functions f; it depends on a parameter X:

$$f[\mathbb{I}] \subseteq \mathbb{I}_d \to X(\mathrm{out}_d \circ f) \to \mathrm{Read}(X)f \quad (d \in \{-1, 0, 1\}),$$
$$(\mathrm{Read}(X)(f \circ \mathrm{in}_d))_{d\in\{-1,0,1\}} \to \mathrm{Read}(X)f.$$

with $\mathrm{in}_d(a) := \frac{a+d}{2}$ and $\mathrm{out}_d(a) := 2a - d$. The corresponding least-fixed-point axiom is

$\mathrm{Read}(X)f \to$
$(\forall^{\mathrm{nc}}_f(f[\mathbb{I}] \subseteq \mathbb{I}_d \to X(\mathrm{out}_d \circ f) \to Pf))_{d\in\{-1,0,1\}} \to$
$\forall^{\mathrm{nc}}_f((\mathrm{Read}(X)(f \circ \mathrm{in}_d))_{d\in\{-1,0,1\}} \to (P(f \circ \mathrm{in}_d))_{d\in\{-1,0,1\}} \to Pf) \to$
$Pf)$.

Write and its dual ${}^{\mathrm{co}}\mathrm{Write}$

Using $\mathrm{Read}(X)$ we give a nested inductive definition of another unary predicate Write by

$$\begin{aligned}&\mathrm{Write}(\mathrm{id}),\\ &\mathrm{Read}(\mathrm{Write})f \to \mathrm{Write}\, f.\end{aligned}$$

Its dual ${}^{\mathrm{co}}\mathrm{Write}$ is defined by

$${}^{\mathrm{co}}\mathrm{Write}\, f \to \mathrm{Eq}(f, \mathrm{id}) \vee \mathrm{Read}({}^{\mathrm{co}}\mathrm{Write})f.$$

The greatest-fixed-point axiom ${}^{\mathrm{co}}\mathrm{Write}^{+}$ is

$$Pf \to \forall_f^{\mathrm{nc}}(Pf \to \mathrm{Eq}(f, \mathrm{id}) \vee \mathrm{Read}({}^{\mathrm{co}}\mathrm{Write} \vee P)f) \to {}^{\mathrm{co}}\mathrm{Write}\, f.$$

${}^{\mathrm{co}}\mathrm{Write}$ is an example of a nested inductive/coinductive predicate.

Define

$$B_{l,k}f := \forall_{p \in \mathbb{I}} \exists_q (f[\mathbb{I}_{p,l}] \subseteq \mathbb{I}_{q,k}).$$
$$Cf := \forall_k \exists_l B_{l,k} f.$$

Theorem

$\forall^{\mathrm{nc}}_f (Cf \leftrightarrow {}^{\mathrm{co}}\mathrm{Write}\, f).$

Proof sketch for $\to$.

We use the greatest-fixed-point axiom ${}^{\mathrm{co}}\mathrm{Write}^+$ with $P := C$. Fix f; it suffices to show $Cf \to \mathrm{Read}({}^{\mathrm{co}}\mathrm{Write} \vee C)f$. Assume Cf. By definition we have an l such that $B_{l,2}f$. Prove

$$\forall_l \forall^{\mathrm{nc}}_f (B_{l,2}f \to Cf \to \mathrm{Read}({}^{\mathrm{co}}\mathrm{Write} \vee C)f)$$

by induction on l. □

Why is this useful?

Recall the Theorem: $\forall^{\mathrm{nc}}_f (Cf \leftrightarrow {}^{\mathrm{co}}\mathrm{Write}\, f)$.

A witness of ${}^{\mathrm{co}}\mathrm{Write}\, f$ is a nested alternating read-write tree. The theorem allows to switch to such (base type) data when proving properties of continuous functions.

Example: the composition $g \circ f$ of two continuous functions $f, g \colon \mathbb{I} \to \mathbb{I}$ is continuous.

The extracted term involves a corecursion operator with nested recursion operators.

Conclusion

TCF (theory of computable functionals) as a possible foundation for (constructive) exact real arithmetic.

- Simply typed theory, with “lazy” free algebras as base types ($\Rightarrow$ constructors are injective and have disjoint ranges).
- Variables range over partial continuous functionals.
- Constants denote computable functionals (:= r.e. ideals).
- Minimal logic ($\to$, $\forall$), plus inductive & coinductive definitions.
- Computational content in abstract theories.
- Decorations ($\to^{\mathrm{c}}, \forall^{\mathrm{c}}$ and $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$) for removal of abstract data, and fine-tuning.
- A nested inductive/coinductive definition of alternating read-write trees representing (uniformly) continuous functions.
- Base type representation of continuous functions when extracting computational content from proofs.

References

- U. Berger, From coinductive proofs to exact real arithmetic. CSL 2009.
- U. Berger, K. Miyamoto, H.S. and M. Seisenberger, The interactive proof system Minlog. Calco-Tools 2011.
- K. Miyamoto and H.S., Program extraction in exact real arithmetic. To appear in MSCS.
- K. Miyamoto, F. Nordvall Forsberg and H.S., Program extraction from nested definitions. Submitted.
- H.S., Realizability interpretation of proofs in constructive analysis. Theory of Computing Systems, 2008.
- H.S. and S.S. Wainer, Proofs and Computations. Perspectives in Logic, ASL & Cambridge UP, 2012.