On the constructive content of proofs in abstract analysis

Ulrich Berger
Swansea University

Helsinki Logic Seminar
December 19, 2018

Supported by a Royal Society grant on Team Semantics, and the EU project 'Computation with Infinite Data'

Overview

1. Computational content of proofs
2. Brouwer's thesis
3. Concurrency
4. Abstract bar induction
5. Proving uniform continuity
6. Extracting the fan functional

Computational content of proofs via realizability - Overview

Instead of defining when a formula is true or false one can define what it means to *realize* it, i.e. what it means to solve the *computational problem* it expresses:

$$p\,\mathbf{r}\,A \qquad \text{(program } p \text{ realizes the formula } A\text{)}$$

Depending on the variant of realizability, p can be

- a code of a Turing machine (Kleene 1945)
- a higher-type functional program (e.g. a term in Gödel's system T)
- an element of a combinatory algebra (e.g. Scott's D_∞)

Soundness Theorem. From a constructive proof of a formula one can extract a program realizing it.

Intuitionistic Fixed Point logic (IFP)

- Intuitionistic first-order logic with equality.
- Extra constants, function symbols and atomic predicates (not necessarily decidable), depending on applications.
- Free predicate variables $X, Y, \ldots$.
- Inductive and coinductive definitions as least and largest fixed points of monotone predicate transformers.
- Axioms consisting of *non-computational (nc)*, that is, disjunction-free, formulas (depending on applications).
- For the classically minded user it suffices for these nc axioms to be classically true in the intended model.

The rational for this system is to stay as close as possible to the axiomatic style common in mathematics while still being able to extract useful computational content from proofs.

Without nc axioms the proof-theoretic strength of IFP is that of Π^1_2-comprehension (Möllerfeld 2003, Tupailo 2004).

Induction and coinduction

Let $\Phi = \lambda X\, \lambda \vec{x}\, A(X, \vec{x})$ a monotone predicate transformer.

Monotonicity is usually guarantied by requiting X to occur at strictly positive positions in A.

The following rules express that $\mu(\Phi)$ is the least predicate X such that $\Phi(X) \subseteq X$ (hence $\Phi(\mu(\Phi)) = \mu\Phi$), and $\nu(\Phi)$ is the largest predicate X such that $X \subseteq \Phi(X)$ (hence $\Phi(\nu(\Phi)) = \nu\Phi$).

$$\frac{}{\Phi(\mu(\Phi)) \subseteq \mu(\Phi)}\ \text{cl} \qquad \frac{\Phi(P) \subseteq P}{\mu(\Phi) \subseteq P}\ \text{ind}$$

$$\frac{}{\nu(\Phi) \subseteq \Phi(\nu(\Phi))}\ \text{cocl} \qquad \frac{P \subseteq \Phi(P)}{P \subseteq \nu(\Phi)}\ \text{coind}$$

Example: Real and natural numbers

- Variables $x, y, \ldots$ are intended to range over abstract real numbers
- Constants and function symbols: $0, 1, +, -, *, /, |\cdot|, \ldots$.
- Atomic predicates: $<, \leq, \ldots$.
- Nc axioms: $\forall x \,.\, x + 0 = x$, ...
- Inductive predicate defining the natural numbers as a subset of the reals numbers: $\mathbf{N} \stackrel{\text{Def}}{=} \mu\, \Phi$, where
  $\Phi = \lambda X\, \lambda x \,.\, x = 0 \vee X(x - 1)$.
  We write this more intuitively as $\mathbf{N}(x) \stackrel{\mu}{=} x = 0 \vee \mathbf{N}(x - 1)$.
- Coinductive predicate defining those real numbers that can be approximated by dyadic rationals: $\mathbf{C} \stackrel{\text{Def}}{=} \nu\, \Psi$, where
  $\Psi = \lambda X\, \lambda x \,.\, \exists n \in \mathbf{N}\, |x - n| \leq 1 \wedge X(2x)$.
  Intuitive notation $\mathbf{C}(x) \stackrel{\nu}{=} \exists n \in \mathbf{N}\, |x - n| \leq 1 \wedge \mathbf{C}(2x)$.

One can prove $\mathbf{C}(x) \leftrightarrow \forall k \in \mathbf{N}\, \exists q \in \mathbf{Q}\, |x - q| \leq 2^{-k}$ where $\mathbf{Q}$ is the set of the rational numbers, defined as usual.

Realizability

To every predicate variable X we assign a new predicate variable $\tilde{X}$ with an extra argument place for realizers.

$$
\begin{array}{rcl}
a\,\mathbf{r}\,P(\vec{t}) & = & P(\vec{t}) \wedge a = \mathbf{Nil} \quad P \text{ atomic predicate} \\
a\,\mathbf{r}\,X(\vec{t}) & = & \tilde{X}(\vec{t}, a) \quad X \text{ a predicate variable} \\
c\,\mathbf{r}\,(A \wedge B) & = & \mathbf{proj}_1(c)\,\mathbf{r}\,A \wedge \mathbf{proj}_2(c)\,\mathbf{r}\,B \\
c\,\mathbf{r}\,(A \vee B) & = & \exists a\,(c = \mathbf{Left}(a) \wedge a\,\mathbf{r}\,A) \vee \\
 & & \exists b\,(c = \mathbf{Right}(b) \wedge b\,\mathbf{r}\,B) \\
f\,\mathbf{r}\,(A \to B) & = & \forall a\,(a\,\mathbf{r}\,A \to (f\ a)\,\mathbf{r}\,B) \\
a\,\mathbf{r}\,\forall x\,A & = & \forall x\,(a\,\mathbf{r}\,A) \\
a\,\mathbf{r}\,\exists x\,A & = & \exists x\,(a\,\mathbf{r}\,A) \\
a\,\mathbf{r}\,(\mu(\lambda X \lambda \vec{x}\,.\,A))(\vec{t}) & = & (\mu(\lambda \tilde{X} \lambda \vec{x} \lambda b\,.\,b\,\mathbf{r}\,A))(\vec{t}, a) \\
a\,\mathbf{r}\,(\nu(\lambda X \lambda \vec{x}\,.\,A))(\vec{t}) & = & (\nu(\lambda \tilde{X} \lambda \vec{x} \lambda b\,.\,b\,\mathbf{r}\,A))(\vec{t}, a)
\end{array}
$$

Special treatment of nc formulas, e.g.

$$b\,\mathbf{r}\,(A \to B) \quad = \quad A \to b\,\mathbf{r}\,B \quad \text{if } A \text{ is nc}$$

Soundness

Soundness Theorem

From an IFP proof of a formula A from nc axioms Γ one can extract a program realizing A, provably from Γ in RIFP, the extension of IFP to the language of realizers.

$$\Gamma \vdash_{\mathrm{IFP}} d : A \quad \Longrightarrow \quad \Gamma \vdash_{\mathrm{RIFP}} \mathbf{ep}(d)\,\mathbf{r}\,A$$

The nc property (no disjunctions) can be weakened to requiring that axioms be *Harrop formulas*, that is, don't contain disjunctions at strictly positive positions and that these axioms imply their realizability translations.

Minlog

Realizability and program extraction is implemented in the interactive proof system *Minlog* developed by H Schwichtenberg in Munich.

`http://www.mathematik.uni-muenchen.de/~logik/minlog/`

Overview of existing case studies in program extraction

- Discrete structures
  - Quotient and remainder on natural numbers.
  - Dijkstra's algorithm (1997, Benl, Schwichtenberg):
    Reachable nodes in a weighted graph
  - Warshall Algorithm (2001, Schwichtenberg, Seisenberger, B):
    Transitive closure of a relation
- Programs from classical proofs
  - GCD (1995, B, Schwichtenberg):
    Uses the Friedman/Dragalin A-translation
  - Dickson's Lemma (2001, Schwichtenberg, Seisenberger, B):
    F/D A-translation in infinite combinatorics
  - Higman's Lemma (2008, Seisenberger):
    Uses F/D A-translation and classical countable choice
  - Fibonacci numbers from a classical proofs (2002, Buchholz, Schwichtenberg, B):
    Uses F/D A-translation to obtain fast program

Overview ctd.

- Lambda calculus:
  - Extraction of normalization-by-evaluation (NbE) (2006, Berghofer, Letouzey, Schwichtenberg, B):
    Extraction of NbE from Tait's proof of strong normalization for the typed lambda calculus (in Isabelle, Coq, Minlog)

- Real numbers
  - Cauchy sequences vs signed digit representation (SD):
    Cauchy sequences are functions.
    SD representations are streams defined by coinduction.
  - Arithmetic operations on reals w.r.t. SD
  - Integration w.r.t. SD (2011, B):
    Real functions are given by trees realizing a nested coinductive/inductive definition

Overview ctd.

- Lists
  - List reversal
    Uses F/D A-translation to extract linear program from naive proof
  - In-place Quicksort (2014, Seisenberger, Woods, B):
    Extracts an 'imperative' program

- Satisfiabilty testing
  - Extraction of a SAT-solver from completeness proof for DPLL (2015, B, Forsberg, Lawrence, Seisenberger)

- Ongoing: Extraction of
  - monadic parsers (Jones, Seisenberger, B)
  - concurrent programs (Miyamoto, Petrovska, Schwichtenberg, Spreen, Takayama, Tsuiki, B)
  - truly imperative programs (Reus, B)
  - modulus of uniform continuity from Fan Theorem (B)

The Archimedean property

The natural numbers are unbounded:

$$\forall\, x\, \exists n \in \mathbf{N}\, x < n \qquad \text{not realizable}$$

The rational numbers are dense:

$$\forall\, x\, \forall\, \epsilon > 0\, \exists q \in \mathbf{Q}\, |x - q| < \epsilon \qquad \text{not realizable}$$

There are no infinitesimals:

AP $\quad \forall\, x(\forall k \in \mathbf{N}\, |x| < 2^{-k} \to x = 0) \qquad$ a true Harrop formula

Apartness

$$x \mathrel{\#} y \stackrel{\text{Def}}{=} \exists k \in \mathbf{N}\, |x - y| \geq 2^{-k}$$

Proposition 1. $\forall x \in \mathbf{C}\,(x \neq 0 \to x \mathrel{\#} 0)$.

Proof. Uses **AP**, as well as *countable choice*:
$\forall n \in \mathbf{N}\, \exists x\, A(n, x) \to \exists f\, \forall n \in \mathbf{N}\, A(m, f(n))$,
and *Markov's principle* for decidable $A(n)$:
$\neg\neg\exists n \in \mathbf{N}\, A(n) \to \exists n \in \mathbf{N}\, A(n)$.

Assume $\mathbf{C}(x)$ and $x \neq 0$.

By countable choice there exists an infinite sequence of rational numbers q_k ($k \in \mathbf{N}$) such that $|x - q_k| \leq 2^{-k}$ for all $k \in \mathbf{N}$.

It is impossible that $|q_{k+1}| \leq 2^{-k}$ for all $k \in \mathbf{N}$ since this would imply that $|x| \leq 2^{-k}$ for all $k \in \mathbf{N}$ and therefore $x = 0$, by **AP**.

Since $|q_{k+1}| \leq 2^{-k}$ is a decidable property of k, by Markov's principle, we can find some $k \in \mathbf{N}$ with $|q_{k+1}| > 2^{-k}$. It follows that $|x| \geq 2^{-(k+1)}$.

Can countable choice be avoided?

Brouwer's thesis

Brouwer's thesis (**BT**) Every bar is inductive.

A predicate P on natural numbers is a *bar* if $\forall\alpha\exists n\, P(\bar{\alpha}\, n)$

P is an *inductive bar* if $\mathbf{IB}_P(\langle\rangle)$ holds where, inductively,

(i) If $P(s)$, then $\mathbf{IB}_P(s)$.

(ii) If $\mathbf{IB}_P(s * n)$ for all $n \in \mathbf{N}$, then $\mathbf{IB}_P(s)$.

More compactly,

$$\mathbf{IB}_P(s) \stackrel{\mu}{=} P(s) \vee \forall n\, \mathbf{IB}_P(s * n) \qquad (\mu \text{ means 'least'})$$

Hence **BT** can be written as the schema

$$\forall\alpha\exists n\, P(\bar{\alpha}\, n) \to \mathbf{IB}_P(\langle\rangle)$$

Recommended reading on **BT**: Wim Veldman: Brouwers Real Thesis on Bars, Philosophia Scientiae, CS 6, 2006.

Issues with **BT** (regarding applicability)

BT $\forall\alpha\exists n\, P(\bar{\alpha}\, n) \to \mathbf{IB}_P(\langle\rangle)$

- restricted to natural numbers
- talks about infinite sequences
- the premise has computational content which is often not available
- the conclusion has unwanted computational content
- to be realizable, the bar P must be decidable, that is, $\forall n\,(P(n) \vee \neg P(n))$ must be provable.

Therefore, we *weaken* and *generalize* premise and conclusion.

Paths and accessibility

Let $\prec$ be an arbitrary binary relation.

$$\begin{aligned} \mathbf{Path}_\prec(x) &\overset{\nu}{=} \exists y \prec x\, \mathbf{Path}_\prec(y) \qquad (\nu \text{ means 'greatest'}) \\ \mathbf{Acc}_\prec(x) &\overset{\mu}{=} \forall y \prec x\, \mathbf{Acc}_\prec(y) \end{aligned}$$

Classically, $\mathbf{Path}_\prec$ and $\mathbf{Acc}_\prec$ are complements of each other.

$\mathbf{Path}_\prec(x)$ means (with dependent choice) that there is an infinite $\prec$-descending sequence starting with x.

$\mathbf{Acc}_\prec(x)$ means that $\prec$-induction is valid at x.

Setting $s \prec_P t \overset{\text{Def}}{=} \exists n\, s = t * n \wedge \neg P(t)$:

$\neg\mathbf{Path}_{\prec_P}(\langle\rangle)$ means that P is a bar,

$\mathbf{Acc}_{\prec_P}(\langle\rangle)$ means that P is an inductive bar.

Brouwer's thesis without computational content

The implication $\mathbf{Acc}_{\prec}(x) \to \neg\mathbf{Path}_{\prec}(x)$ is intuitionistically valid (easy $\prec$-induction).

The converse is can be viewed as a version of Brouwer's thesis:

$\mathbf{BT}_0$ $\forall x\,(\neg\mathbf{Path}_{\prec}(x) \to \mathbf{Acc}_{\prec}(x))$

Both, the premise and conclusion of $\mathbf{BT}_0$, are Harrop formulas (do not contain $\vee$ at a strictly positive position).

Therefore, $\mathbf{BT}_0$ has no computational content and hence does not spoil program extraction.

Wellfounded induction

Combining $\mathbf{BT}_0$ and induction for $\mathbf{Acc}_\prec$ one obtains *wellfounded induction*

$$\frac{\forall x(\forall y \prec x\, P(y) \to P(x))}{\forall x\,(\neg \mathbf{Path}_\prec(x) \to P(x))}\ \text{wfind}$$

(progressive predicates hold at all wellfounded points).

The extracted program is *wellfounded recursion*.

Archimedean induction

$$\frac{\forall x \neq 0\,((|x| \leq 3 \to P(2x)) \to P(x))}{\forall x \neq 0\,P(x)}\ \mathbf{AI}$$

AI follows classically from **AP** and wellfounded induction and is realized by general recursion (least fixed point operator).

A useful variant of Archimedean induction is its relativization to **C**:

$$\frac{\forall x \in \mathbf{C} \setminus \{0\}\,((|x| \leq 3 \to P(2x)) \to P(x))}{\forall x \in \mathbf{C} \setminus \{0\}\,P(x)}\ \mathbf{AIC}$$

AIC follows from **AI** and is realized as follows:

Assume s realizes the premise of **AIC**. Then a realizer of the conclusion of **AIC** is extracted as the recursively defined function

$$\chi\, g = s\, g\, (\chi\, (d\, g))$$

where $d = \lambda g\, \lambda n\, 2 * (g(\mathbf{S}(n)))$ is the realizer extracted from the easy proof of $\mathbf{C}(x) \to \mathbf{C}(2x)$ and $2*$ implements doubling of (unary representations of) natural numbers.

Avoiding countable choice

Proposition 1. $\forall x \in \mathbf{C}\,(x \neq 0 \to x \mathrel{\#} 0)$.

Alternative proof. We show $\forall x \in \mathbf{C} \setminus \{0\}\, x \mathrel{\#} 0$ using **AIC**. Let $x \in \mathbf{C} \setminus \{0\}$ and assume, as induction hypothesis, $|x| \leq 3 \to 2x \mathrel{\#} 0$. Since $x \in \mathbf{C}$ there is $q \in \mathbf{Q}$ such that $|x - q| \leq 1$. If $|q| > 2$, then $|x| \geq 1$ and we are done. If $|q| \leq 2$, then $|x| \leq 3$ so we can apply the induction hypothesis to obtain $2x \mathrel{\#} 0$, which implies $x \mathrel{\#} 0$.

Concurrency (j.w.w. Hideki Tsuiki)

Given: Processes $p_1, \ldots, p_n$ such that

- at least one p_i is guaranteed to terminate,
- each terminating p_i will produce a correct result

Task: Combine the p_i to obtain a correct result.

Solution: Run $p_1, \ldots, p_n$ concurrently. As soon as one p_i terminates, deliver the result and kill all the other p_j.

`/papers/tsuiki/ccc_tsuiki.pdf`

We introduce an extension of intuitionistic logic enabling the extraction of such kind of programs (together with correctness proofs).

Concurrent disjunction

- We add a new form of disjunction $A_1 \overset{\mathsf{p}}{\vee} A_2$ which admits two concurrent processes as realizers.
- . . . and add a new program constructor $\mathbf{Amb}(a_1, a_2)$ for the concurrent execution of the processes a_1, a_2 (motivated by McCarthy's Amb).
- $\mathbf{Amb}(a_1, a_2)$ realizes $A_1 \overset{\mathsf{p}}{\vee} A_2$ iff at least one a_i is defined, and each defined a_j realizes A_j.

Concurrent law of excluded middle (failed attempt)

The following form of the law of excluded middle seems to be realizable provided B is nc:

$$\frac{B \to A_1 \quad \neg B \to A_2}{A_1 \overset{\mathsf{p}}{\vee} A_2}$$

If $a_1 \,\mathbf{r}\, (B \to A_1)$ and $a_2 \,\mathbf{r}\, (\neg B \to A_2)$, which means $B \to a_1 \,\mathbf{r}\, A_2$ and $\neg B \to a_2 \,\mathbf{r}\, A_2$, one might believe (classically) that $\mathbf{Amb}(a_1, a_2)$ realizes $A_1 \overset{\mathsf{p}}{\vee} A_2$.

However, if, for example, B is false, then the formula $B \to a_1 \,\mathbf{r}\, A_1$ says nothing about a_1, but a_1 might still be defined and be delivered as a result of $\mathbf{Amb}(a_1, a_2)$ and consequently, there is no guarantee that $\mathbf{Amb}(a_1, a_2)$ realizes $A_1 \overset{\mathsf{p}}{\vee} A_2$.

We need a variant of implication that avoids this.

Restriction $A \parallel B$ (a variant of $B \to A$)

$$a\,\mathbf{r}\,(A \parallel B) \quad \stackrel{\text{Def}}{=} \quad (B \to \mathbf{def}(a)) \wedge (\mathbf{def}(a) \to a\,\mathbf{r}\,A)$$

where B is nc and $\mathbf{def}(a)$ means that a is defined (i.e. terminates).

Proof rules:

$$\frac{B \to A_0 \vee A_1 \quad \neg B \to A_0 \wedge A_1}{A_0 \vee A_1 \parallel B} \parallel \mathsf{I}$$

where A_0, A_1 must be nc

...

Concurrent law of excluded middle (correct)

$$\frac{A_1 \parallel B \quad A_2 \parallel \neg B}{A_1 \overset{\mathsf{p}}{\vee} A_2}\ \text{Conc-lem}$$

If a_1 realizes $A_1 \parallel B$ and a_2 realizes $A_2 \parallel \neg B$,

then $\mathbf{Amb}(a_1, a_2)$ realizes $A_1 \overset{\mathsf{p}}{\vee} A_2$.

Monotonicity (replacing disjunction elimination):

$$\frac{A_1 \to B_1 \quad A_2 \to B_2}{(A_1 \overset{\mathsf{p}}{\vee} A_2) \to (B_1 \overset{\mathsf{p}}{\vee} B_2)}\ \text{Conc-mon}$$

Infinite Gray code

Using the concurrent extension of IFP it is possible to extract programs operating on Tsuiki's *infinite Gray code for real numbers*.

Infinite Gray code admits representations of real numbers with possibly one undefined digit, which forces computation to be concurrent and nondeterministic.

In return, infinite Gray code has the remarkable property that is computable and *unique*, that is, *every real number has exactly one code*.

> Hideki Tsuiki. *Real number computation through Gray code embedding. Theoretical Computer Science, 284:467–485, 2002.*

Bar induction for decidable bars (**BI**)

If

(1) P is a bar,

(2) P decidable and $P \subseteq Q$,

(3) $\forall s\,(\forall n\, Q(s * n) \to Q(s))$,

then $Q(\langle\rangle)$.

It is easy to see that **BT** implies **BI**.

Abstract bar induction (**ABI**)

$$y \prec^* x \quad \overset{\mu}{=} \quad y = x \vee \exists z\,(y \prec^* z \wedge z \prec x) \qquad \text{(refl. trans. closure)}$$
$$y \prec_P x \quad \overset{\text{Def}}{=} \quad y \prec x \wedge \neg P(x)$$

Let x_0 be arbitrary (playing the role of the empty sequence).

ABI If
(1) $\neg\mathbf{Path}_{\prec_P}(x_0)$
(2) $\forall x \prec^* x_0\,(\neg P(x) \vee Q(x))$,
(3) $\forall x \prec^* x_0\,(\forall y \prec x\, Q(y) \to Q(x))$,
then $Q(x_0)$.

Lemma. $\mathbf{BT}_0$ implies **ABI**.

Proof. Assume (1), (2), (3). By $\mathbf{BT}_0$, $\mathbf{Acc}_{\prec_P}(x_0)$. We prove $\mathbf{Acc}_{\prec_P} \subseteq Q$ by wellfounded induction. By i.h., $\forall y \prec_P^* x\, Q(y)$. We have to show $Q(x)$. We do a case analysis according to (2). If $Q(x)$, we are done. If $\neg P(x)$ then the i.h. is equivalent to the premise of (3), hence, again $Q(x)$.

Bang!

If A is a formula, then $!A$ is a Harrop formula with $a\,\mathbf{r}\,!A \stackrel{\text{Def}}{=} a = \mathbf{Nil} \wedge \forall a\,(a\,\mathbf{r}\,A)$.

For example, $\mathbf{Nil}\,\mathbf{r}\,!(\bot \to A)$ since, $a\,\mathbf{r}\,(\bot \to A) \equiv \bot \to a\,\mathbf{r}\,A$.

Intuitively, $!A$ expresses that A is true (realizable) for trivial reasons.

Valid (realizable) rules we will use in the following:

$$\frac{A}{!A}\,!\mathbf{H} \qquad (A \text{ Harrop})$$

$$\frac{A \to !B}{!(A \to B)}\,!{\to} \qquad \frac{!A \wedge !B}{!(B \wedge A)}\,!{\wedge}$$

$$\frac{\forall x\,!A(x)}{!\forall x\,A(x)}\,!\forall \qquad \frac{\exists x\,!A(x)}{!\exists x\,A(x)}\,!\exists$$

!LEM

$$\frac{\neg A \to B \qquad A \to !B}{B}\ !\mathbf{LEM}$$

Lemma

The rules for bang are realizable.

Proof.

We only look at !**LEM**.

Assume $a\,\mathbf{r}\,(\neg A \to B)$ and $\mathbf{Nil}\,\mathbf{r}\,(A \to !B)$, that is, $\neg\exists c\; c\,\mathbf{r}\,A \to a\,\mathbf{r}\,B$ and $\exists c\; c\,\mathbf{r}\,A \to \forall b\; b\,\mathbf{r}\,B$.

Using the law of excluded middle, we conclude $a\,\mathbf{r}\,B$. □

Banged bar induction

!BI If

(1) $\neg\mathbf{Path}_{\prec_P}(x_0)$,
(2) $\forall x \prec^* x_0\,(P(x) \to\, !Q(x))$,
(3) $\forall x \prec^* x_0\,(\forall y \prec x\, Q(y) \to Q(x))$,

then $Q(x_0)$.

Lemma

$\mathbf{BT}_0$ *implies* **!BI**.

Proof.

The proof is almost identical to the proof for **ABI**. The only difference is that we use **!LEM** to do a case analysis, on whether $P(x)$ holds, using (2). □

The extracted program takes as input a realizer g of (3) (note that (2) is Harrop) and returns $h\,\langle\rangle$ where

$$h\,s = g\,s\,(\lambda a\,(h\,(s * a))).$$

Proving uniform continuity

We aim to prove that every total continuous functional F on Cantor space is uniformly continuous and extract from the proof the *fan functional* that computes the minimal modulus of uniform continuity of F.

Language:

Constants: $0, 1, \bot$, where $0, 1$ represent at the same time the first two natural numbers and the Booleans, and $\bot$ represents 'undefined' (not to be confused with the formula $\bot$).

Function symbols: $+, -$, application operation (written by juxtaposition), common (primitive recursive) operations to define finite and infinite sequences.

Relation symbol: $<$ (ordinary ordering of numbers).

Natural numbers: $\mathbf{N}(x) \stackrel{\mu}{=} x = 0 \vee \mathbf{N}(x - 1)$.

Partial functionals

We define the partial Booleans and partial Boolean-valued functionals of type 1 and 2:

$$\begin{aligned}
\mathbb{B}(x) &\overset{\text{Def}}{=} x = 0 \vee x = 1 \\
\mathbb{B}_\perp(x) &\overset{\text{Def}}{=} x \neq \perp \to \mathbb{B}(x) \\
\mathbb{B}^1_\perp(\alpha) &\overset{\text{Def}}{=} \forall n\,(\mathbf{N}(n) \to \mathbb{B}_\perp(\alpha\, n)) \\
\mathbb{B}^2_\perp(F) &\overset{\text{Def}}{=} \forall \alpha\,(\mathbb{B}^1_\perp(\alpha) \to \mathbb{B}_\perp(F\alpha))
\end{aligned}$$

For the following it wouldn't make much difference if the result predicate of F were $\mathbf{N}_\perp$ (instead of $\mathbb{B}_\perp$).

Continuity

Specialization order:

$$\begin{aligned} x \sqsubseteq y &\stackrel{\text{Def}}{=} x \neq \bot \to x = y \\ \alpha \sqsubseteq \beta &\stackrel{\text{Def}}{=} \forall n \in \mathbf{N}\,(\alpha\, n \sqsubseteq \beta\, n) \end{aligned}$$

Monotonicity, *finitarity*, *continuity*:

$$\begin{aligned} \mathbf{Mon}(F) &\stackrel{\text{Def}}{=} \forall \alpha, \beta \in \mathbb{B}^1_\bot\,(\alpha \sqsubseteq \beta \to F\,\alpha \sqsubseteq F\,\beta) \\ \mathbf{Fin}(F) &\stackrel{\text{Def}}{=} \forall \alpha \in \mathbb{B}^1_\bot\,(\forall n \in \mathbf{N}\, F\,(\alpha \upharpoonright n) = \bot \to F\,\alpha = \bot) \\ \mathbf{Cont}(F) &\stackrel{\text{Def}}{=} \mathbf{Mon}(F) \wedge \mathbf{Fin}(F) \end{aligned}$$

where $(\alpha \upharpoonright n)\,k = \mathbf{if}\ k < n\ \mathbf{then}\ \alpha\, k\ \mathbf{else}\ \bot$.

Totality

$$\begin{aligned}
\mathbf{Total}^1(\alpha) &\stackrel{\text{Def}}{=} \forall n\,(\mathbf{N}(n) \to \alpha\, n \neq \bot) \\
\mathbf{Total}^2(F) &\stackrel{\text{Def}}{=} \forall \alpha\,(\mathbf{Total}^1(\alpha) \to F\alpha \neq \bot))
\end{aligned}$$

$$\begin{aligned}
\mathbb{B}^1(\alpha) &\stackrel{\text{Def}}{=} \mathbb{B}^1_\bot(\alpha) \wedge \mathbf{Total}^1(\alpha) \\
\mathbb{B}^2(F) &\stackrel{\text{Def}}{=} \mathbb{B}^2_\bot(F) \wedge \mathbf{Total}^1(F)
\end{aligned}$$

Uniform continuity

A type 2 functional F is *uniformly continuous* if there is (a least) $n \in \mathbf{N}$ such that $F\,\alpha = F\,\beta$ for all total α, β agreeing below n.

$$\begin{aligned} \mathbf{UCont}(F, n) &\stackrel{\text{Def}}{=} \forall \alpha, \beta \in \mathbb{B}^1(\alpha =_n \beta \to F\,\alpha = F\,\beta) \\ \mathbf{UCont}(F) &\stackrel{\text{Def}}{=} \exists n \in \mathbf{N}\, \mathbf{UCont}(F, n) \end{aligned}$$

where $\alpha =_n \beta \stackrel{\text{Def}}{=} \forall k \in \mathbf{N}\,(k < n \to \alpha\, k = \beta\, k)$.

We aim to prove that every $F \in \mathbb{B}^2_\perp$ which is total and continuous is uniformly continuous.

Deciding constancy

Let $\mathbb{B}^*$ be the set of finite sequences of Booleans, that is,

$$\mathbb{B}^*(s) \stackrel{\mu}{=} s = \langle\rangle \vee \exists t \in \mathbb{B}^* \exists b \in \mathbb{B}\, s = t * b,$$

and set

$$\mathbf{Const}(F, s) \stackrel{\text{Def}}{=} \exists b \in \mathbb{B} \forall \alpha \in \mathbb{B}^1\, F\,(s * \alpha) = b$$

where $(s * \alpha)\, n = s_n$ if $n < |s|$ and $(s * \alpha)\, n = \alpha(n - |s|)$ if $n \geq |s|$.

Theorem (Decidability of constancy)

Let F be a total continuous functional, that is, $F \in \mathbb{B}^2$ and $\mathbf{Cont}(F)$. *Then for every $s \in \mathbb{B}^s$ it is decidable whether F is constant on total extensions of s, that is,* $\mathbf{Const}(F, s) \vee \neg\mathbf{Const}(F, s)$.

Deciding constancy

Proof.

We fix a total continuous functional F and define

$$\mathbf{sec}(s) \stackrel{\text{Def}}{=} F\,(s * \bot^\omega) \neq \bot \quad (\text{'}s \text{ is secured'})$$
$$s \prec t \stackrel{\text{Def}}{=} \exists b \in \mathbb{B}\, s = t * b$$

Hence $\mathbb{B}^*(s)$ iff $s \prec^* \langle\rangle$.

We define a version of the drinker formula:

$$\mathbf{Dr}(s, b, \alpha) \stackrel{\text{Def}}{=} F(s{*}\alpha) \neq \bot \wedge (\exists \beta \in \mathbb{B}^1_\bot\, F(s{*}\beta) = b \to F(s{*}\alpha) = b)$$

and set $Q_b(s) \stackrel{\text{Def}}{=} \exists \alpha \in \mathbb{B}^1_\bot\, \mathbf{Dr}(s, b, \alpha)$.

Claim: $\forall s \in \mathbb{B}^*\, Q_b(s)$ holds for every $b \in \mathbb{B}$.
Fix $b \in \mathbb{B}$. We prove the claim by banged bar induction on $\prec_{\mathbf{sec}}$.

Applying !**BI**

We have to show

(1) $\forall s \in \mathbb{B}^* \neg \mathbf{Path}_{\prec_{\mathbf{sec}}}(s)$,
(2) $\forall s \in \mathbb{B}^* (\mathbf{sec}(s) \to !Q_b(s))$,
(3) $\forall s \in \mathbb{B}^* \, (\forall a \in \mathbb{B} \, Q_b(s * a) \to Q_b(s))$,

(1) holds F since is total and continuous.

(2): By efq, $!\to$, and $!\forall$, $!\mathbb{B}^1_\perp(\perp^\omega)$. If $s \in \mathbb{B}^*$ is secured, then clearly $\mathbf{Dr}(s, b, \perp^\omega)$. Since this a Harrop formula, it follows $!\mathbf{Dr}(s, b, \perp^\omega)$, by $!\mathbf{H}$. With $!\wedge$ and $!\exists$ it follows $!Q_b(s)$.

(3): Let $s \in \mathbb{B}^*$ such that $\forall a \in \mathbb{B} \, Q_b(s * a)$, that is, we have $\alpha_0, \alpha_1 \in \mathbb{B}^1_\perp$ such that $\mathbf{Dr}(s * 0, b, \alpha_0)$ and $\mathbf{Dr}(s * 1, b, \alpha_1)$. We have to find $\alpha \in \mathbb{B}^1_\perp$ such that $\mathbf{Dr}(s, b, \alpha)$. Since $F \in \mathbb{B}^2_\perp$, we have $F(s * 0 * \alpha_0) \in \mathbb{B}$. If $F(s * 0 * \alpha_0) = b$, set $\alpha = 0 * \alpha_0$. Otherwise, set $\alpha = 1 * \alpha_1$. This completes the proof of the Claim.

To complete the proof of the theorem, let $\alpha_0, \alpha_1 \in \mathbb{B}^1_\perp$ with $\mathbf{Dr}(s, 0, \alpha_0)$ and $\mathbf{Dr}(s, 1, \alpha_1)$, according to the Claim. Let $a = F \, \alpha_0 \in \mathbb{B}$ and $b = F \, \alpha_1 \in \mathbb{B}$. Clearly, $\mathbf{Const}(F)$ iff $a = b$.

The proof of uniform continuity

Theorem

Every functional $F \in \mathbb{B}^2_\perp$ which is total and continuous is uniformly continuous.

Proof.

Let $F \in \mathbb{B}^2_\perp$ be total and continuous. We set

$$\begin{aligned} \mathbf{UCont}(s,n) &\stackrel{\text{Def}}{=} \forall \alpha,\beta \in \mathbb{B}^1(\alpha =_n \beta \to F(s*\alpha) = F(s*\beta)) \\ \mathbf{UCont}(s) &\stackrel{\text{Def}}{=} \exists n \in \mathbf{N}\, \mathbf{UCont}(s,n) \end{aligned}$$

and show $\forall s \in \mathbb{B}^*\, \mathbf{UCont}(s)$ by abstract bar induction, **ABI**, on $\prec_{\mathbf{Const}}$ where $\prec$ is as in the proof of the Constancy Theorem and $\mathbf{Const}(s) \stackrel{\text{Def}}{=} \mathbf{Const}(F,s)$.

Applying **ABI**

We have to show:

(1) $\mathbf{Wf}_{\prec_{\mathbf{Const}}}(\langle\rangle)$,

(2) $\forall s \in \mathbb{B}^* \, (\neg \mathbf{Const}(s) \vee \mathbf{UCont}(s))$,

(3) $\forall s \in \mathbb{B}^* \, (\forall a \in \mathbb{B} \, \mathbf{UCont}(s * a) \to \mathbf{UCont}(s))$.

(1) holds again by continuity.

(2): By the Constancy Theorem, we may assume $\mathbf{Const}(s)$. Then clearly $\mathbf{UCont}(s, 0)$.

(3): Assume $\mathbf{UCont}(s * 0, n)$ and $\mathbf{UCont}(s * 1, m)$. Then, clearly, $\mathbf{UCont}(s, 1 + \max(n, m))$.

Program extraction

Declarations:

```
type N = Int
type B = Int
type B1 = N -> B
type B2 = B1 -> B

(***) :: [B] -> B1 -> B1
s *** alpha = \n-> if n < length s
                  then s !! n
                  else alpha (n - length s)
```

Testing constancy

Testing whether a type 2 functional is constant on extensions of s:

```
thm1 :: B2 -> [B] -> Bool
thm1 f s = f (s *** (claim 0 s)) == f (s *** (claim 1 s))

  where

--  Computing the drinker
--  claim :: B -> [B] -> B1
    claim b s = let {
                      alpha0 = claim b (s++[0]) ;
                      alpha1 = claim b (s++[1])
                    }
                in if f ((s++[0]) *** alpha0) == b
                   then [0] *** alpha0
                   else [1] *** alpha1
```

Computing the mod. of unif. cont. (fan functional)

```
thm2 :: B2 -> N
thm2 f = aux []

  where

--  aux :: [B] -> N
    aux s = if thm1 f s
            then 0
            else 1 + max (aux (s++[0])) (aux (s++[1]))
```

In-class test

```
*Main> thm2 (\alpha-> 0)
0
*Main> thm2 (\alpha-> 1)
0
*Main> thm2 (\alpha-> alpha 1)
2
*Main> thm2 (\alpha-> alpha (sum [alpha n | n <- [0..5]]) )
7
*Main> thm2 (\alpha-> alpha (sum [2 * alpha n | n <- [0..7]]) )
17
*Main> thm2 (\alpha-> alpha (sum [alpha (2*n) | n <- [0..7]]) )
15
```

Conclusion

- The fine grained control of computational content not only optimizes extracted programs but also provides access to new kinds of algorithms by program extraction.
- Limited use of classical logic seems to be required to verify the correctness of these new algorithms.
- The Harrop version of Brouwer's thesis and banged bar induction might open ways to extract programs such as the Berard-Bezem-Coquand realizer of dependent choice from a proof.

References

Hideki Tsuiki. Real Number Computation through Gray Code Embedding.
Theor. Comput. Sci., *284(2):467–485, 2002.*

B., Kenji Miyamoto, Helmut Schwichtenberg, Hideki Tsuiki: Logic for Gray-code computation. In: Concepts of Proof in Mathematics, Philosophy, and Computer Science, de Gruyter, 2016.

B., Extracting Non-Deterministic Concurrent Programs. CSL 2016, LIPICS

L. E. J. Brouwer, Beweis dass jede volle Funktion gleichmässig stetig ist.
Nederlandse Akademie van Wetenschappen Verslagen 27, 189193, 1924.

L. E. J. Brouwer, Über Definitionsbereiche von Funktionen,
Math. Annalen 97, 6075, 1927.

References

V. Veldman. Brouwer's Real Thesis on Bars.
Philosophia Scientiæ, CS 6, *Constructivism: Mathematics, Logic, 21-42 Philosophy and Linguistics, 2006.*

H. Schwichtenberg. Minlog.
The Seventeen Provers of the World, Lecture Notes in Artificial Intell., *3600, 151–157, 2006.*
`http://www.mathematik.uni-muenchen.de/~logik/minlog/`

M. Escardó. Exhaustible sets in higher-type computation,
Logical Methods in Comput. Sci. *4 (3), 2008.*

B. Totale Objekte und Mengen in der Bereichstheorie,
PhD thesis, LMU Munich, *1990.*

References

B. From coinductive proofs to exact real arithmetic: theory and applications.
Logical Methods in Comput. Sci., *7(1):1–24, 2011.*

M. Escardo, P. Oliva. Bar recursion and products of selection functions, *JSL, 80(1):1-28, 2015.*

B, O. Petrovska Optimized program extraction for induction and coinduction
CiE 2018, LNCS 10936, 70-80, 2018.