Proof and computation

Helmut Schwichtenberg

Mathematisches Institut, LMU, München

FMV: Foundations in Mathematics – Modern views,
LMU, April 2018

Proofs are what distinguishes mathematics from other sciences.

A proof has two aspects

- provides insight (uniformity)
- may have computational content

Mathematics = logic + data + (co)inductive definitions

- Logic: minimal, intro and elim for $\to$, $\forall$
- Proof $\sim$ lambda-term (Curry-Howard correspondence)
- Can embed classical and intuitionistic logic

Constructive logic as an extension of classical logic

Observed by Gödel and Gentzen (1930s). Constructive logic:

- $\exists_x A$ — proof needs an instance.
- $A \vee B$ — proof must be for one of the two alternatives.

In addition we have the classical (weaker) versions

- $\tilde{\exists}_x A$ — assuming that there is no such x leads to a contradiction.
- $A \,\tilde{\vee}\, B$ — assuming that none of the alternatives holds leads to a contradiction.

Example of a non-constructive proof

Lemma

There are irrational numbers a, b such that a^b is rational.

Proof.

Case $\sqrt{2}^{\sqrt{2}}$ rational. Let $a = \sqrt{2}$ and $b = \sqrt{2}$. Then both a, b are irrational, and by assumption a^b is rational.

Case $\sqrt{2}^{\sqrt{2}}$ irrational. Let $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$. Then by assumption a, b are irrational, and

$$a^b = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \left(\sqrt{2}\right)^2 = 2$$

is rational. □

Constructive analysis

- Errett Bishop, Foundations of Constructive Analysis. McGraw-Hill, 1967.
- Needs appropriate definitions (example: compactness).
- To be done: constructive proofs with

  constructions $\sim$ good (e.g., efficient) algorithms.

Type theory

- Per Martin-Löf, Intuitionistic Type Theory. Bibliopolis, 1984.
- Very popular: basis for (i) proof assistants Nuprl, Coq, Agda (ii) homotopy type theory (Voevodsky).

For Bishop-style constructive mathematics

- simplify: finite types ($\iota \mid \rho \to \sigma$) rather than dependent types
- extend: partial rather than total functionals

Finite rather than dependent types

- Aczel & Gambino (2006): Logic enriched type theory
- Formulas need not be types, they just behave similarly.
- $\forall_x A$ and $A \to B$ primitive, $\exists_x A$ and $A \vee B$ defined inductively:

$$\exists^+ : \forall_x(x \in P \to \exists_x(x \in P))$$
$$\exists^- : \exists_x(x \in P) \to \forall_x(x \in P \to C) \to C \qquad (x \notin \mathrm{FV}(C))$$

$$\vee_i^+ : A_i \to A_0 \vee A_1$$
$$\vee^- : A \vee B \to (A \to C) \to (B \to C) \to C$$

Partial rather than total functionals

General view: computations are finite.

Arguments not only numbers and functions, but also functionals of any finite type.

- Principle of finite support. If $\mathcal{H}(\Phi)$ is defined with value n, then there is a finite approximation Φ_0 of Φ such that $\mathcal{H}(\Phi_0)$ is defined with value n.
- Monotonicity principle. If $\mathcal{H}(\Phi)$ is defined with value n and Φ' extends Φ, then also $\mathcal{H}(\Phi')$ is defined with value n.
- Effectivity principle. An object is computable iff its set of finite approximations is (primitive) recursively enumerable (or equivalently, Σ^0_1-definable).

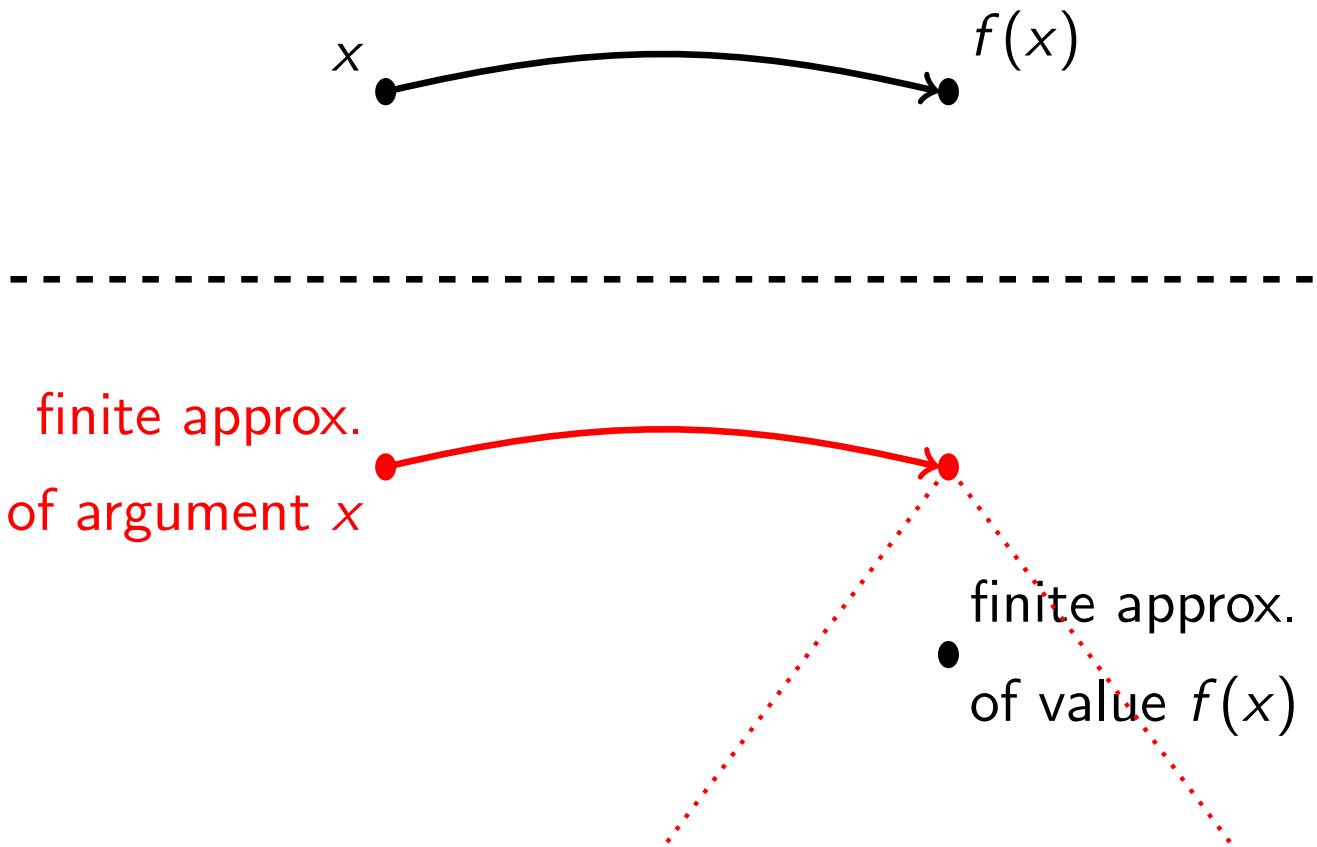
The Scott-Ershov model of partial continuous functionals

- $\mathcal{C} := (|\mathbf{C}_\rho|)_\rho$ provides the domains of computable functionals.
- Satisfies the three principles above.
- "x^ρ computable" for $x^\rho \in |\mathbf{C}_\rho|$ is defined by

  the set of finite approximations $U \subseteq x$ is recursively enumerable

- TCF (theory of partial computable functionals): variant of both HA^ω and Martin-Löf type theory.

Principle of finite support



A common extension T^+ of Gödel's T and Plotkin's PCF

Terms of T^+ are built from (typed) variables and (typed) constants (constructors C or defined constants D, see below) by (type-correct) application and abstraction:

$$M, N ::= x^\rho \mid \mathrm{C}^\rho \mid D^\rho \mid (\lambda_{x^\rho} M^\sigma)^{\rho\to\sigma} \mid (M^{\rho\to\sigma} N^\rho)^\sigma.$$

Every defined constant D comes with a system of computation rules, consisting of finitely many equations

$$D\vec{P}_i(\vec{y}_i) = M_i \qquad (i = 1, \ldots, n)$$

with free variables of $\vec{P}_i(\vec{y}_i)$ and M_i among $\vec{y}_i$, where the arguments on the left hand side must be "constructor patterns", i.e., lists of applicative terms built from constructors and distinct variables.

Examples

- $+ \colon \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ defined by

$$n + 0 = n$$
$$n + \mathrm{S}m = \mathrm{S}(n + m)$$

- $=_{\mathbb{N}} \colon \mathbb{N} \to \mathbb{N} \to \mathbb{B}$

$$(0 =_{\mathbb{N}} 0) = \mathrm{tt}, \qquad (\mathrm{S}m =_{\mathbb{N}} 0) = \mathrm{ff},$$
$$(0 =_{\mathbb{N}} \mathrm{S}n) = \mathrm{ff}, \qquad (\mathrm{S}m =_{\mathbb{N}} \mathrm{S}n) = (m =_{\mathbb{N}} n).$$

Recursion operators

- Introduced by Hilbert (1925) and Gödel (1958).
- Used to construct maps from the algebra ι to τ, by recursion on the structure of ι.
- Example: $\mathcal{R}^{\tau}_{\mathbb{N}}$ of type $\mathbb{N} \to \tau \to (\mathbb{N} \to \tau \to \tau) \to \tau$.
- The first argument is the recursion argument, the second one gives the base value, and the third one gives the step function, mapping the recursion argument and the previous value to the next value.
- For example, $\mathcal{R}^{\mathbb{N}}_{\mathbb{N}} nm\lambda_{n,p}(\mathrm{S}p)$ defines addition $m + n$ by recursion on n.

Corecursion operators

Streams:

$$\mathbb{S}(\alpha) \quad \text{by} \quad \mathrm{SCons}^{\alpha\to\mathbb{S}(\alpha)\to\mathbb{S}(\alpha)}.$$

The corecursion operator ${}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{S}(\rho)}$ of type

$$\tau \to (\tau \to \rho \times (\mathbb{S}(\rho) + \tau)) \to \mathbb{S}(\rho)$$

is defined by

$${}^{\mathrm{co}}\mathcal{R}xf := \begin{cases} \mathrm{SCons}(y, z) & \text{if } f(x) \equiv \langle y, \mathrm{InL}(z)\rangle, \\ \mathrm{SCons}(y, {}^{\mathrm{co}}\mathcal{R}x'f) & \text{if } f(x) \equiv \langle y, \mathrm{InR}(x')\rangle. \end{cases}$$

The Brouwer-Heyting-Kolmogorov interpretation

Also known as (modified) realizability.

Kolmogorov 1932: "Zur Deutung der intuitionistischen Logik"

- Proposed to view a formula A as a computational problem, of type $\tau(A)$, the type of a potential solution or "realizer" of A.
- Example: $\forall_{n \in T_{\mathbb{N}}} \exists_{m \in T_{\mathbb{N}}} (m > n \wedge m \in \mathrm{Prime})$ has type $\mathbb{N} \to \mathbb{N}$.

In TCF one can prove

Theorem (Soundness)

From a TCF-proof M of a (computationally relevant) formula A we can extract a term $\mathrm{et}(M)$ realizing A.

Application: exact real numbers

can be given in different formats:

- Cauchy sequences (of rationals, with Cauchy modulus).
- Infinite sequences (streams) of signed digits $\{-1, 0, 1\}$, or
- $\{-1, 1, \bot\}$ with at most one $\bot$ ("undefined"): Gray code.

Want formally verified algorithms on reals given as streams.

- Consider formal proofs M and apply realizability to extract their computational content.
- Switch between different formats of reals by relativising to coinductive predicates. Example:

$$\forall_x(x \in {}^{\mathrm{co}}I \to A) \quad \text{rather than} \quad \forall_x(x \in \mathrm{Real} \to A).$$

Computational content of $x \in {}^{\mathrm{co}}I$ is a stream representing x.

A real number can be represented as a Cauchy sequence $(a_n)_n$ of rationals together with a Cauchy modulus M satisfying

$$|a_n - a_m| \leq \frac{1}{2^p} \quad \text{for } n, m \geq M(p).$$

Arithmetical operations on real numbers x, y are defined by

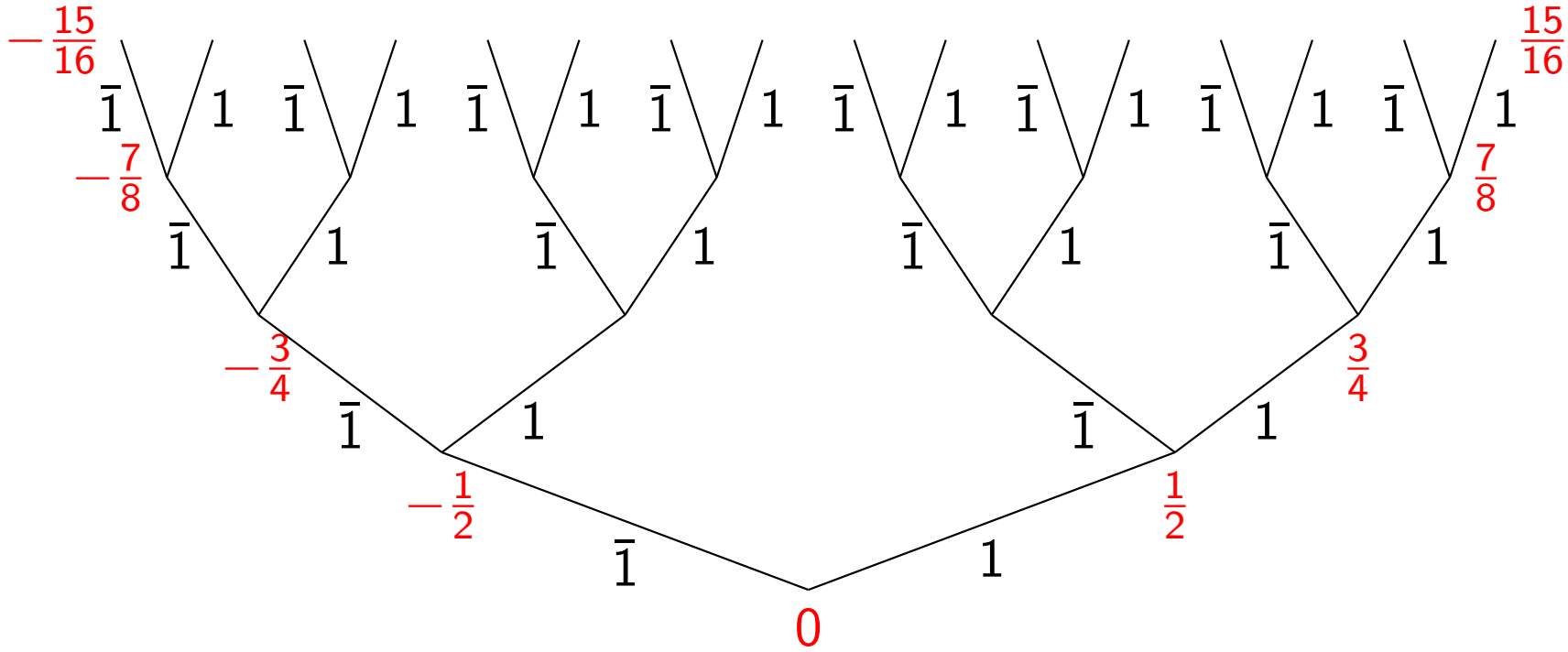
	c_n	$L(p)$
$x + y$	$a_n + b_n$	$\max\big(M(p+1), N(p+1)\big)$
$-x$	$-a_n$	$M(p)$
$\lvert x\rvert$	$\lvert a_n\rvert$	$M(p)$
$x \cdot y$	$a_n \cdot b_n$	$\max\big(M(p+1+p_y), N(p+1+p_x)\big)$
$\frac{1}{x}$ for $\lvert x\rvert \in_q \mathbb{R}^+$	$\begin{cases} \frac{1}{a_n} & \text{if } a_n \neq 0 \\ 0 & \text{if } a_n = 0 \end{cases}$	$M(2(q+1)+p)$

where 2^{p_x} is the upper bound of x provided by the Archimedian property.

Representation of real numbers $x \in [-1, 1]$

Dyadic rationals:

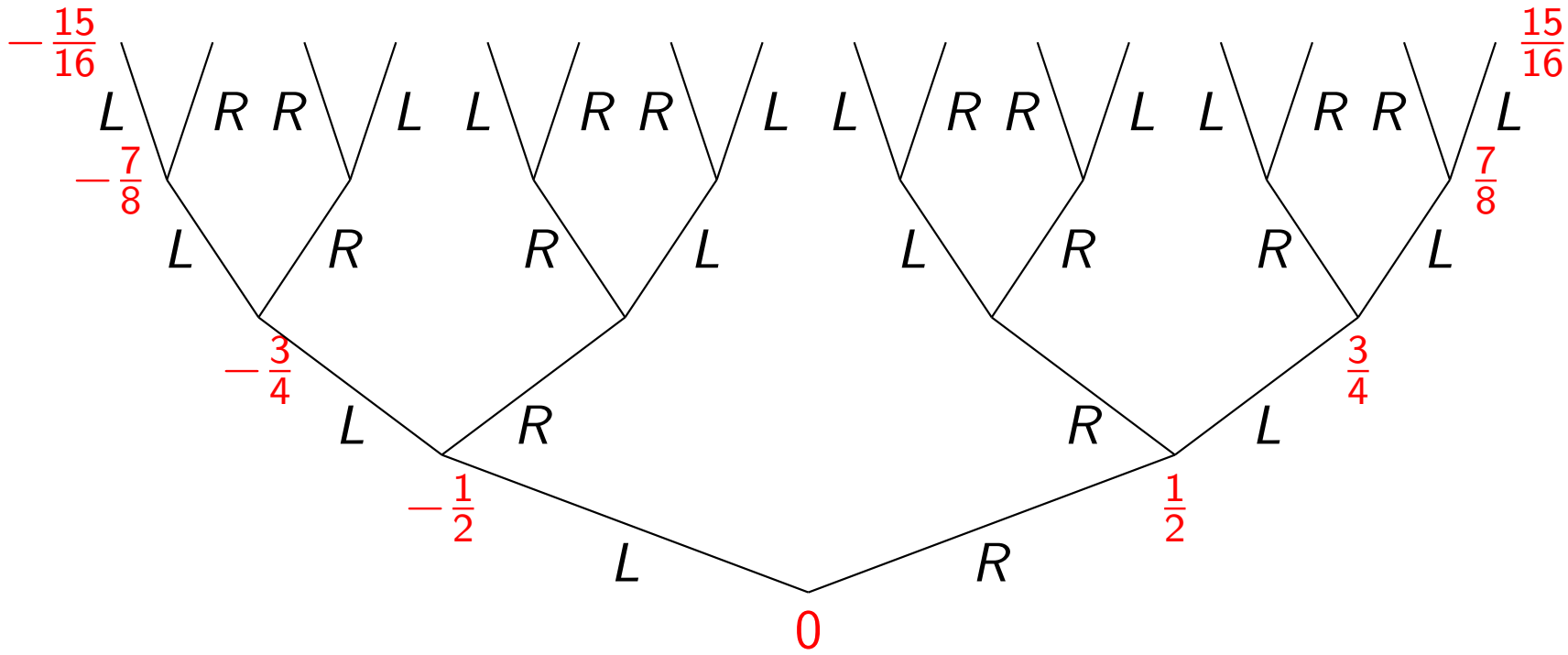
$$\sum_{i<k} \frac{a_i}{2^{i+1}} \qquad \text{with } a_i \in \{-1, 1\}$$



with $\bar{1} := -1$. Adjacent dyadics can differ in many digits:

$$\frac{7}{16} \sim 1\bar{1}11, \qquad \frac{9}{16} \sim 11\bar{1}\bar{1}.$$

Cure: flip after 1. Binary reflected (or Gray-) code.



$$\frac{7}{16} \sim \mathrm{RRRL}, \qquad \frac{9}{16} \sim \mathrm{RLRL}.$$

Problem with productivity:

$$\bar{1}111 + 1\bar{1}\bar{1}\bar{1}\cdots = ? \qquad (\text{or } \mathrm{LRLL}\ldots + \mathrm{RRRL}\cdots = ?)$$

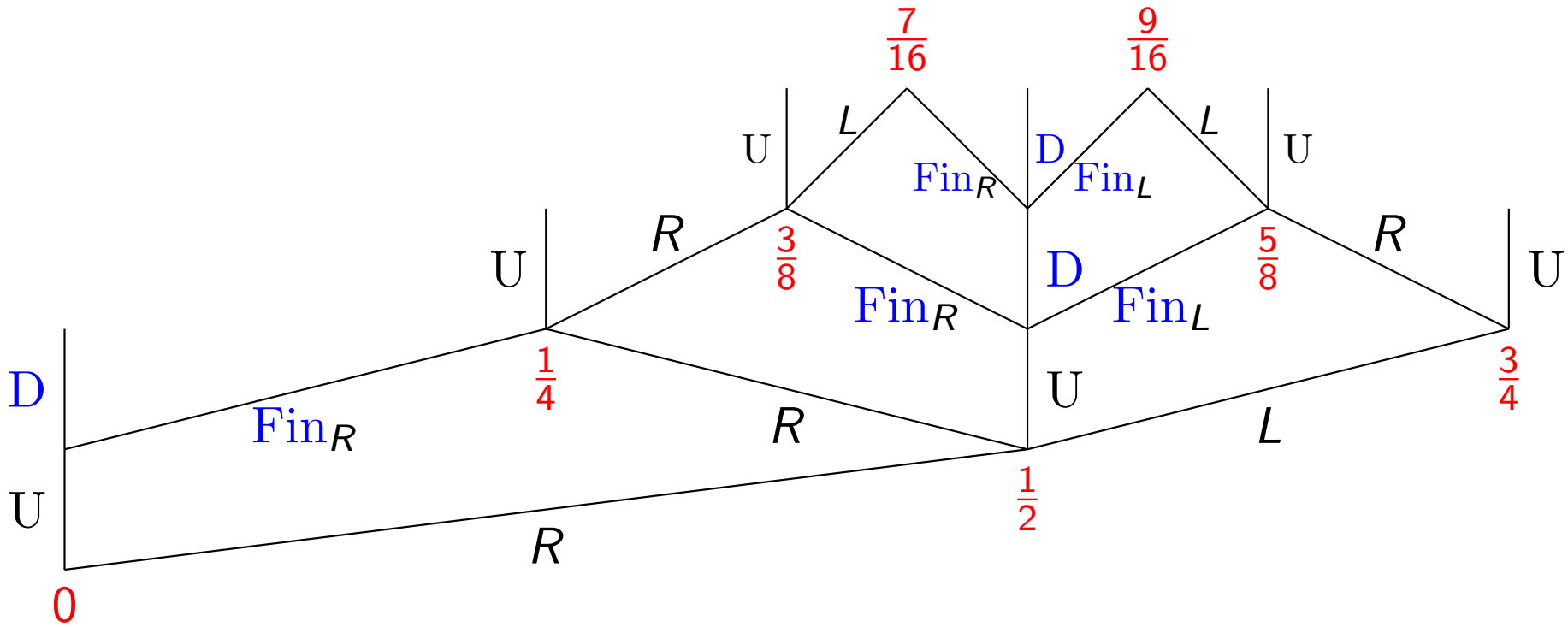
What is the first digit? Cure: delay.

- For binary code: add 0. Signed digit code

$$\sum_{i<k} \frac{d_i}{2^{i+1}} \qquad \text{with } d_i \in \{-1, 0, 1\}.$$

  Widely used for real number computation. There is a lot of redundancy: $\bar{1}1$ and $0\bar{1}$ both denote $-\frac{1}{4}$.
- For Gray-code: add U (undefined), D (delay), $\mathrm{Fin}_{L/R}$ (finally left / right).

Gray code



After computation in Gray code, one can remove Fin_a by

$$\mathrm{U} \circ \mathrm{Fin}_a \mapsto a \circ \mathrm{R}, \qquad \mathrm{D} \circ \mathrm{Fin}_a \mapsto \mathrm{Fin}_a \circ \mathrm{L}.$$

Average for signed digit streams

Goal:

$$\underbrace{x, y \in {}^{\mathrm{co}}I}_{x,y\in[-1,1]} \to \underbrace{\frac{x+y}{2} \in {}^{\mathrm{co}}I}_{\frac{x+y}{2}\in[-1,1]}.$$

- Streams appear only implicit in our logical framework.
- Model streams as cototal objects in the (free) algebra $\mathbb{S}(\mathbb{D})$ given by the constructor $\mathrm{C}\colon \mathbb{D} \to \mathbb{S}(\mathbb{D}) \to \mathbb{S}(\mathbb{D})$.

Intuitively, $d_0, d_1, d_2 \ldots$ represents

$$\sum_{i=0}^{\infty} \frac{d_i}{2^{i+1}} \qquad \text{with } d_i \in \mathrm{D} := \{-1, 0, 1\}.$$

Definition of ${}^{\mathrm{co}}I$

$$\Phi(X) := \{\, x \mid x \in [-1,1] \wedge \exists_{d \in \mathrm{D}} \exists_{x' \in X} (x = \frac{x' + d}{2}) \,\}.$$

Then

$$\begin{aligned} I &:= \mu_X \Phi(X) && \text{least fixed point} \\ {}^{\mathrm{co}}I &:= \nu_X \Phi(X) && \text{greatest fixed point} \end{aligned}$$

satisfy the (strengthened) axioms

$$\begin{aligned} \Phi(I \cap X) \subseteq X \to I \subseteq X && \text{induction} \\ X \subseteq \Phi({}^{\mathrm{co}}I \cup X) \to X \subseteq {}^{\mathrm{co}}I && \text{coinduction} \end{aligned}$$

("strengthened" because their hypotheses are weaker than the fixed point property $\Phi(X) = X$).

Goal: compute the average of two stream-coded reals. Prove

$$x, y \in {}^{\mathrm{co}}I \to \frac{x+y}{2} \in {}^{\mathrm{co}}I.$$

Computational content of this proof will be the desired algorithm.

Informal proof (from Ulrich Berger & Monika Seisenberger 2006). Define sets P, Q of averages, Q with a "carry" $i \in \mathbb{Z}$:

$$P := \{ \frac{x+y}{2} \mid x, y \in {}^{\mathrm{co}}I \},$$
$$Q := \{ \frac{x+y+i}{4} \mid x, y \in {}^{\mathrm{co}}I, i \in \mathrm{D}_2 \} \quad (\mathrm{D}_2 := \{-2, -1, 0, 1, 2\}).$$

Suffices: Q satisfies the clause coinductively defining ${}^{\mathrm{co}}I$. Then by the greatest-fixed-point axiom for ${}^{\mathrm{co}}I$ we have $Q \subseteq {}^{\mathrm{co}}I$. Since also $P \subseteq Q$ we obtain $P \subseteq {}^{\mathrm{co}}I$, which is our claim.

Q satisfies the ${}^{\mathrm{co}}I$-clause:

$$i \in \mathrm{D}_2 \to x, y \in {}^{\mathrm{co}}I \to \exists_{j \in \mathrm{D}_2} \exists_{d \in \mathrm{D}} \exists_{x', y' \in {}^{\mathrm{co}}I} \left(\frac{x + y + i}{4} = \frac{\frac{x' + y' + j}{4} + d}{2} \right).$$

Proof. Write $x = \frac{x' + d}{2}$ and $y = \frac{y' + e}{2}$ $\quad (d, e \in \mathrm{D},\ x', y' \in {}^{\mathrm{co}}I)$.
Then

$$\frac{x + y + i}{4} = \frac{x' + y' + d + e + 2i}{8}.$$

Since $|d + e + 2i| \leq 6$ we can write $d + e + 2i = j + 4k$ with $|j| \leq 2$ and $|k| \leq 1$. Therefore

$$\frac{x + y + i}{4} = \frac{x' + y' + j + 4k}{8} = \frac{\frac{x' + y' + j}{4} + k}{2}.$$

Implicit algorithm.

$q\colon \mathbb{D}_2 \to \mathbb{S}(\mathbb{D}) \to \mathbb{S}(\mathbb{D}) \to \mathbb{D}_2 \times \mathbb{D} \times \mathbb{S}(\mathbb{D}) \times \mathbb{S}(\mathbb{D})$ defined by

$$q(i, \mathrm{C}_d(u), \mathrm{C}_e(v)) = (J(d+e+2i), K(d+e+2i), u, v)$$

with $J, K\colon \mathbb{Z} \to \mathbb{Z}$ such that

$$i = J(i) + 4K(i), \qquad |J(i)| \leq 2, \qquad |i| \leq 6 \to |K(i)| \leq 1.$$

By coinduction we obtain $Q \subseteq {}^{\mathrm{co}}I$:

$$\exists_{i \in \mathrm{D}_2} \exists_{x,y \in {}^{\mathrm{co}}I} (z = \frac{x + y + i}{4}) \to z \in {}^{\mathrm{co}}I.$$

This gives our claim

$$x, y \in {}^{\mathrm{co}}I \to \frac{x + y}{2} \in {}^{\mathrm{co}}I.$$

Implicit algorithm. $P \subseteq Q$ computes the first "carry" $i \in \mathrm{D}_2$ and the tails of the inputs. Then $f : \mathbb{D}_2 \times \mathbb{S}(\mathbb{D}) \times \mathbb{S}(\mathbb{D}) \to \mathbb{S}(\mathbb{D})$ defined corecursively by

$$f(i, \mathrm{C}_d(u), \mathrm{C}_e(v)) = \mathrm{C}_{K(k+l+2i)}(f(J(k + l + 2i), u, v))$$

is called repeatedly and computes the average step by step. (Here $(d, k), (e, l) \in \mathrm{D}^{\mathbf{r}}$).

```
[u,u0][let tuv
   (IntToSdtwo(SdToInt clft(cCoIClosure u)+
               SdToInt clft(cCoIClosure u0))pair
   crht(cCoIClosure u)pair crht(cCoIClosure u0))
   ((CoRec sdtwo yprod str yprod str=>str)tuv
   ([tuv0][let tsuv
       (IntToSdtwo
       (J(SdToInt clft(cCoIClosure clft crht tuv0)+
         SdToInt clft(cCoIClosure crht crht tuv0)+
         SdtwoToInt clft tuv0*2))pair
       IntToSd
       (K(SdToInt clft(cCoIClosure clft crht tuv0)+
         SdToInt clft(cCoIClosure crht crht tuv0)+
         SdtwoToInt clft tuv0*2))pair
       crht(cCoIClosure clft crht tuv0)pair
       crht(cCoIClosure crht crht tuv0))
       (clft crht tsuv pair
        InR(clft tsuv pair crht crht tsuv))]))]
```

From the proof M of

$$x, y \in {}^{\mathrm{co}}I \to \frac{x+y}{2} \in {}^{\mathrm{co}}I$$

extract a term $\mathrm{et}(M)$. The Soundness theorem gives a proof of

$$\mathrm{et}(M)\ \mathbf{r}\ \forall_{x,y}(x, y \in {}^{\mathrm{co}}I \to \frac{x+y}{2} \in {}^{\mathrm{co}}I).$$

Brouwer-Heyting-Kolmogorov interpretation:

$$u\ \mathbf{r}\ (x \in {}^{\mathrm{co}}I) \to v\ \mathbf{r}\ (y \in {}^{\mathrm{co}}I) \to \mathrm{et}(M)(u, v)\ \mathbf{r}\ (\frac{x+y}{2} \in {}^{\mathrm{co}}I)$$

This is a formal verification that $\mathrm{et}(M)$ computes the average w.r.t. signed digit streams.

Average for Gray code

Method essentially the same as for signed digit streams.

- Only need to insert a different computational content to the predicates expressing how a real x is given.
- Instead of ${}^{\mathrm{co}}I$ for signed digit streams we now need two such predicates ${}^{\mathrm{co}}G$ and ${}^{\mathrm{co}}H$, corresponding to the two "modes" in Gray code.

Method also works for multiplication and division:

$$x, y \in {}^{\mathrm{co}}I \to \frac{x+y}{2} \in {}^{\mathrm{co}}I,$$
$$x, y \in {}^{\mathrm{co}}I \to x \cdot y \in {}^{\mathrm{co}}I,$$
$$x, y \in {}^{\mathrm{co}}I \to \frac{1}{4} \leq y \to \frac{x}{y} \in {}^{\mathrm{co}}I,$$

both w.r.t. signed digit and Gray code.

References

E. Bishop, *Foundations of Constructive Analysis*. McGraw-Hill, 1967

H.S. and S.S. Wainer, *Proofs and Computations*, Perspectives in Logic. Association for Symbolic Logic and Cambridge University Press, 2012.

U. Berger, K. Miyamoto, H.S. and H. Tsuiki, *Logic for Gray-code computation*. In: Concepts of Proof in Mathematics, Philosophy, and Computer Science (eds. Probst, Schuster). De Gruyter, 2016, pp. 69-110