Computational content of proofs

Helmut Schwichtenberg
(j.w.w. Ulrich Berger, Nils Köpp, Kenji Miyamoto, Monika Seisenberger, Hideki Tsuiki and Franziskus Wiesnet)

Mathematisches Institut, LMU, München

Proof Society Workshop, Swansea, 13. September 2019

Proofs have two aspects:

1. they guarantee correctness, and
2. they may have computational content.

We address (2), and use a BHK-interpretation to extract programs from proofs. Features:

- The extract is a term in the underlying theory, hence we have a framework to formally prove its properties.
- Computational content in (co)inductive predicates only.
- From proofs in constructive analysis[1] we can extract programs operating on stream-represented real numbers.

[1] E. Bishop, Foundations of Constructive Analysis, 1967

In more detail:

$$M \mapsto \mathrm{et}(M) \in \mathrm{T}^+, \qquad \mathrm{sp}(M) \colon \mathrm{et}(M)\ \mathbf{r}\ A$$

where

- M proof of A in TCF,
- $\mathrm{et}(M)$ term of "cotype" $\varphi(A)$ in T^+,
- $\mathrm{sp}(M)$ soundness proof in TCF, of $\mathrm{et}(M)\ \mathbf{r}\ A$,
- $\mathbf{r}$ Keisel's modified realizability.

Related work, comparison.

- HA^ω and Martin-Löf style type theories use total functionals only. In contrast, TCF is based on the Scott[2]-Ershov[3] model of partial continuous functionals.
- Coq extracts programs in a programming language (OCaml, Scheme), and Agda uses whole proofs as programs. In both cases it is difficult to formally prove properties of these.

[2] D. Scott, Outline of a mathematical theory of computation, Oxford 1970

[3] Y. Ershov, Model C of partial continuous functionals, Logic. Coll. 1977

The model. An information system $\boldsymbol{A} = (A, \mathrm{Con}, \vdash)$ id given by:

- A countable set of "tokens",
- Con set of finite subsets of A,
- $\vdash$ ("entails") subset of $\mathrm{Con} \times A$.

such that

$$U \subseteq V \in \mathrm{Con} \to U \in \mathrm{Con},$$
$$\{a\} \in \mathrm{Con},$$
$$U \vdash a \to U \cup \{a\} \in \mathrm{Con},$$
$$a \in U \in \mathrm{Con} \to U \vdash a,$$
$$U, V \in \mathrm{Con} \to \forall_{a \in V}(U \vdash a) \to V \vdash b \to U \vdash b.$$

$x \subseteq A$ is an ideal if

$$U \subseteq x \to U \in \mathrm{Con} \quad (x \text{ is consistent}),$$
$$x \supseteq U \vdash a \to a \in x \quad (x \text{ is deductively closed}).$$

Function spaces

Let $\boldsymbol{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\boldsymbol{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems. Define $\boldsymbol{A} \to \boldsymbol{B} := (C, \mathrm{Con}, \vdash)$ where

- $C := \mathrm{Con}_A \times B$,
- $$\{ (U_i, b_i) \mid i \in I \} \in \mathrm{Con} := \forall_{J \subseteq I}(\bigcup_{j \in J} U_j \in \mathrm{Con}_A \to \{ b_j \mid j \in J \} \in \mathrm{Con}_B),$$
- $\{ (U_i, b_i) \mid i \in I \} \vdash (U, b)$ means $\{ b_i \mid U \vdash_A U_i \} \vdash_B b$.

$\boldsymbol{A} \to \boldsymbol{B}$ is an information system.

Application of an ideal x in $\boldsymbol{A} \to \boldsymbol{B}$ to an ideal y in $\boldsymbol{A}$ is defined by

$$x(y) := \{ b \in B \mid \exists_{U \subseteq y}((U, b) \in x) \}.$$

(Free) algebras given by constructors:

$$
\begin{array}{lll}
\mathbb{N} & \text{by} & 0^{\mathbb{N}}, S^{\mathbb{N}\to\mathbb{N}} \\
\mathbb{Y} & \text{by} & 0^{\mathbb{Y}}, C^{\mathbb{Y}\to\mathbb{Y}\to\mathbb{Y}} \qquad \text{(binary trees)} \\
\alpha\times\beta & \text{by} & \langle .,. \rangle^{\alpha\to\beta\to\alpha\times\beta} \\
\alpha+\beta & \text{by} & (\mathrm{InL}_{\alpha\beta})^{\alpha\to\alpha+\beta}, (\mathrm{InR}_{\alpha\beta})^{\beta\to\alpha+\beta} \\
\mathbb{L}(\alpha) & \text{by} & []^{\mathbb{L}(\alpha)}, \mathrm{cons}^{\alpha\to\mathbb{L}(\alpha)\to\mathbb{L}(\alpha)} \\
\mathbb{S}(\alpha) & \text{by} & \mathrm{SCons}^{\alpha\to\mathbb{S}(\alpha)\to\mathbb{S}(\alpha)}
\end{array}
$$

$\mathbb{S}(\alpha)$ has no nullary constructor, hence no "total" objects.

Information systems $\boldsymbol{C}_\rho = (C_\rho, \mathrm{Con}_\rho, \vdash_\rho)$

$\boldsymbol{C}_{\rho\to\sigma} := \boldsymbol{C}_\rho \to \boldsymbol{C}_\sigma$. At base types ι:

Tokens are type correct constructor expressions $\mathrm{C}a_1^* \dots a_n^*$.
(Examples: 0, $C{*}0$, $C0{*}$, $C(C{*}0)0$.)

$U = \{a_1, \dots, a_n\}$ is consistent if

- all a_i start with the same constructor,
- (proper) tokens at j-th argument positions are consistent.

(Example: $\{C{*}0, C0{*}\}$.)

$U \vdash a$ (entails) if

- all $a_i \in U$ and also a start with the same constructor,
- (proper) tokens at j-th argument positions of a_i entail j-th argument of a.

(Example: $\{C{*}0, C0{*}\} \vdash C00$).

Definition

- A partial continuous functional of type ρ is an ideal in $\boldsymbol{C}_\rho$.
- A partial continuous functional is computable if it is a (primitive) recursively enumerable set of tokens.

Ideals in $\boldsymbol{C}_\rho$: Scott-Ershov domain of type ρ.

- x^ι is total iff $x = \{\, a \mid \{b\} \vdash a \,\}$ for some token (i.e., constructor expression) b without $*$.
- x^ι is cototal iff every token $b(*) \in x$ has a "one-step extension" $b(\mathrm{C}\vec{*}) \in x$.

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

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

$$n + 0 = n$$
$$n + Sm = S(n + m)$$

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

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

Recursion operators

- Introduced by Hilbert[4] and Gödel[5].
- 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, $\mathcal{R}^{\mathbb{N}}_{\mathbb{N}} nm\lambda_{n,p}(Sp)$ defines addition $m + n$ by recursion on n.

[4] D. Hilbert, Über das Unendliche, Math. Ann. 1925

[5] K. Gödel, Über eine bisher noch nicht benützte Erweiterung des finiten Standpunkts, Dialectica 1958

Corecursion operators

Recall

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

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

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

A theory TCF of partial computable functionals

TCF can be seen as a variant of both HA^ω and Martin-Löf type theory. Features:

- Based on the model $\mathcal{C}$: partial functionals allowed.
- Logic enriched type theory[6]: formulas and types kept separate.
- Computational content only arises from (co)inductive definitions.

[6] N. Gambino & P. Aczel, The generalized type-theoretic interpretation of constructive set theory, JSL 2006

On closed base types (binary trees $\mathbb{Y}$) inductively define totality:

$$(T_{\mathbb{Y}})^+_0 : 0 \in T_{\mathbb{Y}},$$
$$(T_{\mathbb{Y}})^+_1 : \forall_{x,x'}(x, x' \in T_{\mathbb{Y}} \to \mathrm{C}xx' \in T_{\mathbb{Y}})$$

$$T^-_{\mathbb{Y}} : 0 \in X \to \forall_{x',x''}(x', x'' \in X \to \mathrm{C}x'x'' \in X) \to T_{\mathbb{Y}} \subseteq X$$

and coinductively define cototality, by the closure axiom

$${}^{\mathrm{co}}T^-_{\mathbb{Y}} : \forall_x(x \in {}^{\mathrm{co}}T_{\mathbb{Y}} \to x \equiv 0 \vee \exists_{x',x''}(x', x'' \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge x \equiv \mathrm{C}x'x''))$$

and the greatest-fixed-point (or coinduction) axiom ${}^{\mathrm{co}}T^+_{\mathbb{Y}}$:

$$\forall_x(x \in X \to x \equiv 0 \vee \exists_{x',x''}(x', x'' \in ({}^{\mathrm{co}}T_{\mathbb{Y}} \cup X) \wedge x \equiv \mathrm{C}x'x'')) \to X \subseteq {}^{\mathrm{co}}T_{\mathbb{Y}}.$$

We have

$$T_{\mathbb{Y}} \subset {}^{\mathrm{co}}T_{\mathbb{Y}} \subset |\mathcal{C}_{\mathbb{Y}}| \qquad \text{(proper inclusions)}$$

Definition of predicates and formulas.

$$P, Q ::= X \mid \{\, \vec{x} \mid A \,\} \mid I(\vec{\rho}, \vec{P}) \mid {}^{\mathrm{co}}I(\vec{\rho}, \vec{P}) \qquad \text{(predicates)},$$
$$A, B ::= P\vec{t} \mid A \to B \mid \forall_x A \qquad \text{(formulas)}$$

Call a predicate or formula C computationally relevant (c.r.) or non-computational (n.c). if its final predicate is.

Definition of the type $\tau(C)$ of a c.r. predicate or formula C.
Assume a global injective assignment of type variables ξ to X^c.

$$\tau(X^c) := \xi,$$
$$\tau(\{\, \vec{x} \mid A \,\}) := \tau(A),$$
$$\tau((I/^{\mathrm{co}}I)(\vec{\rho}, \vec{P})) := \iota_I(\tau(\vec{P}^c)),$$
$$\tau(P\vec{t}) := \tau(P),$$
$$\tau(A \to B) := \begin{cases} \tau(A) \to \tau(B) & (A \text{ c.r.}) \\ \tau(B) & (A \text{ n.c.}) \end{cases}$$
$$\tau(\forall_x A) := \tau(A),$$

Call $\iota_I(\tau(\vec{P}^c))$ the algebra associated with $(I/^{\mathrm{co}}I)(\vec{\rho}, \vec{P})$.

The cotype $\varphi(C)$ has

$$\varphi(I(\vec{\rho}, \vec{P})) := \iota_I(\varphi(\vec{P}^c)),$$
$$\varphi({}^{\mathrm{co}}I(\vec{\rho}, \vec{P})) := {}^{\mathrm{co}}\iota_I(\varphi(\vec{P}^c)).$$

For arbitrary cotypes φ we define E_φ (exists) by

$$\begin{aligned}(x \in E_\iota) &:= (x \in T_\iota),\\ (x \in E_{\mathrm{co}\,\iota}) &:= (x \in {}^{\mathrm{co}}T_\iota),\\ (f \in E_{\varphi\to\psi}) &:= \forall_x(x \in E_\varphi \to fx \in E_\psi).\end{aligned}$$

Similary for $\doteq_\varphi$ (pointwise equal)

$$\begin{aligned}(x \doteq_\iota y) &:= (x \sim_\iota y),\\ (x \doteq_{\mathrm{co}\,\iota} y) &:= (x \approx_\iota y),\\ (f \doteq_{\varphi\to\psi} g) &:= \forall_{x,y}(x \doteq_\varphi y \to fx \doteq_\psi gy).\end{aligned}$$

What is the relation between $\doteq_\varphi$ and Leibniz equality $\equiv$? First consider closed base cotypes.

- Case ι. In the model $\mathcal{C}$ we have

$$(x \sim_\iota y) \leftrightarrow (x \in T_\iota \wedge x \equiv y).$$

  This is also provable in TCF.

- Case ${}^{\mathrm{co}}\iota$. In the model $\mathcal{C}$ we have

$$(x \approx_\iota y) \leftrightarrow (x \in {}^{\mathrm{co}}T_\iota \wedge x \equiv y).$$

  (Write $x = \bigcup_n (x\restriction n)$ and use induction on n). In TCF this is an axiom, called Bisimilarity Axiom.

Recall that at higher types, pointwise equality relative to a cotype is defined as a logical relation:

$$(f \doteq_{\varphi\to\psi} g) := \forall_{x,y}(x \doteq_\varphi y \to fx \doteq_\psi gy).$$

Then extensionality[7] relative to a cotype φ is defined by

$$\mathrm{Ext}_\varphi(f) := (f \doteq_\varphi f).$$

Lemma. For closed cotypes φ the relation $\doteq_\varphi$ is a partial equivalence relation (i.e., symmetric and transitive) whose domain is the set Ext_φ of objects extensional w.r.t. φ.

[7] R. Gandy, PhD 1953, and On the axiom of extensionality – part I. JSL 1956 and also G. Takeuti, On a generalized logic calculus, Jap. J. Math. 1953

Lemma. E_φ and Ext_φ are equivalent for closed cotypes φ of level ≤ 1.

Proof. For level 0 this holds be definition. For level 1 use induction on the height of the cotype. Let $\varphi \to \psi$ be a closed cotype of level 1. The following are equivalent.

$$f \in \mathrm{Ext}_{\varphi\to\psi}$$
$$f \doteq_{\varphi\to\psi} f$$
$$\forall_{x,y}(x \doteq_\varphi y \to fx \doteq_\psi fy)$$
$$\forall_x(x \in E_\varphi \to fx \doteq_\psi fx) \qquad \text{by definition, since } \mathrm{lev}(\varphi) = 0$$
$$\forall_x(x \in E_\varphi \to fx \in \mathrm{Ext}_\psi)$$

Now use the induction hypothesis and the definition of $E_{\varphi\to\psi}$.

Are E_φ and Ext_φ equivalent for levels ≥ 2? No: we would need $f \doteq_\varphi g \to f \equiv g$ for φ of level 1. But the following are equivalent:

$$f \doteq_{\mathbb{N}\to\mathbb{N}} g$$
$$\forall_{n,m}(n \doteq_{\mathbb{N}} m \to fn \doteq_{\mathbb{N}} gm)$$
$$\forall_n(n \in T_{\mathbb{N}} \to fn \doteq_{\mathbb{N}} gn)$$
$$\forall_n(n \in T_{\mathbb{N}} \to fn, gn \in T_{\mathbb{N}} \wedge fn \equiv gn)$$
$$f, g \in E_{\mathbb{N}\to\mathbb{N}} \wedge \forall_n(n \in T_{\mathbb{N}} \to fn \equiv gn)$$

This cannot be Leibniz equality, since nothing is said on the behaviour of f, g on non-total arguments.

Lemma (Extensionality of the recursion operator)

Let I be an inductive predicate and ι_I its associated algebra. Then the extracted term $\mathrm{et}(I^-) := \mathcal{R}^\tau_{\iota_I}$ of its least-fixed-point (or elimination) axiom I^- is extensional w.r.t. the formula of I^-.

Lemma (Extensionality of the corecursion operator)

Let ${}^{\mathrm{co}}I$ be a coinductive predicate and ι_I its associated algebra. Then the extracted term $\mathrm{et}({}^{\mathrm{co}}I^+) := {}^{\mathrm{co}}\mathcal{R}^\tau_{\iota_I}$ of its greatest-fixed-point (or coinduction) axiom ${}^{\mathrm{co}}I^+$ is extensional w.r.t. the formula of ${}^{\mathrm{co}}I^+$.

Realizers

Proofs have two aspects:

- they provide insight into why an argument is correct, and
- they can also have computational content.

The Brouwer-Heyting-Kolmogorov[8] (BHK)-interpretation gives a good analysis of the latter. Modification here: computational content only arises from (co)inductive predicates.

- p proves $A \to B$ if and only if p is a construction transforming any proof q of A into a proof $p(q)$ of B.
- p proves $\forall_x A$ if and only if p is a construction such that p proves A, irrespective of what x is.

[8] A. Kolmogorov, Zur Deutung der intuitionistischen Logik, Math. Z. 1932

Unexplained notions:

what is a "construction"?

what is a proof of a prime formula?

Here we propose to take

construction := computable functional,

proof of a prime formula $I\vec{t}$:= a "construction tree" for $I\vec{t}$,

proof of a prime formula ${}^{\mathrm{co}}I\vec{t}$:= a "destruction tree" for ${}^{\mathrm{co}}I\vec{t}$.

Such a construction or destruction tree can seen as a (co)total ideal in the algebra ι_I associated with the clauses of I.

Realizability extension $C^{\mathbf{r}}$ of c.r. predicates or formulas C

For n.c. C let $C^{\mathbf{r}} := C$. If C is c.r. $C^{\mathbf{r}}$ is a predicate of arity $(\vec{\sigma}, \tau(C))$ ($\vec{\sigma}$ arity of C). Write $z\ \mathbf{r}\ C$ for $C^{\mathbf{r}}z$ if C is a c.r. formula. For X^c let $X^{\mathbf{r}}$ be the n.c. predicate variable provided, and

$$\{\,\vec{x} \mid A\,\}^{\mathbf{r}} := \{\,\vec{x}, z \mid z\ \mathbf{r}\ A\,\}.$$

Case

$$I/^{\mathrm{co}}I := (\mu/\nu)_X((K_i(X))_{i<k}$$

with algebra form $\iota_I = \mu_\xi(\kappa_i(\xi))_{i<k}$ where $\kappa_i(\xi) := \tau(K_i(X))$. The i-th constructor of ι_I is $\mathrm{C}_i \colon \kappa_i(\iota_I)$. Let

$$I^{\mathbf{r}}/^{\mathrm{co}}I^{\mathbf{r}} := (\mu/\nu)_{X^{\mathbf{r}}}(\mathrm{C}_i\ \mathbf{r}\ K_i(X))_{i<k}.$$

For c.r. formulas let

$$z \mathbf{r} P\vec{t} := P^{\mathbf{r}}\vec{t}z,$$

$$z \mathbf{r} (A \to B) := \begin{cases} \forall_w(w \mathbf{r} A \to zw \mathbf{r} B) & \text{if } A \text{ is c.r.} \\ A \to z \mathbf{r} B & \text{if } A \text{ is n.c.} \end{cases}$$

$$z \mathbf{r} \forall_x A := \forall_x(z \mathbf{r} A).$$

Extracted term $\mathrm{et}(M)$ of a proof M^A with A c.r.

$$\mathrm{et}(u^A) := z_u^{\tau(A)} \quad (z_u^{\tau(A)} \text{ uniquely associated to } u^A),$$

$$\mathrm{et}((\lambda_{u^A} M^B)^{A\to B}) := \begin{cases} \lambda_{z_u}^{\tau(A)}\mathrm{et}(M) & \text{if } A \text{ is c.r.} \\ \mathrm{et}(M) & \text{if } A \text{ is n.c.} \end{cases}$$

$$\mathrm{et}((M^{A\to B} N^A)^B) := \begin{cases} \mathrm{et}(M)\mathrm{et}(N) & \text{if } A \text{ is c.r.} \\ \mathrm{et}(M) & \text{if } A \text{ is n.c.} \end{cases}$$

$$\mathrm{et}((\lambda_x M^A)^{\forall_x A}) := \mathrm{et}(M),$$

$$\mathrm{et}((M^{\forall_x A(x)} t)^{A(t)}) := \mathrm{et}(M).$$

Extracted terms for the axioms. Let $\iota := \iota_I$.

$$\mathrm{et}(I_i^+) := \mathrm{C}_i \quad (\mathrm{C}_i \ i\text{-th constructor of } \iota,\ i < k)$$

$$\mathrm{et}(I^-) := \mathcal{R}_\iota^\tau$$

$$\mathrm{et}({}^{\mathrm{co}}I^-) := \mathrm{D}_\iota$$

$$\mathrm{et}({}^{\mathrm{co}}I^+) := {}^{\mathrm{co}}\mathcal{R}_\iota^\tau$$

Soundness

Recall: an n.c. part of a proof $M : A$ is a subproof $N : B$ with B n.c. Such n.c. parts will not contribute to the computational content of the whole proof: can ignore all decorations in those parts.

An assumption variable $u : C$ with C c.r. must be replaced by a corresponding realizability assumption $u : z_u \mathbf{r} C$. However, if $u : C$ appears in an n.c. part of the proof we can keep it (since computational content is ignored in such parts), and consider the two assumption variables $u : z_u \mathbf{r} C$ and $u : C$ to be the same.

Theorem (Soundness)

Let M be a proof of a c.r. formula A from assumptions $u : C$ (c.r.) and $v : D$ (n.c.) Then we can find a proof $\mathrm{sp}(M)$ of $\mathrm{et}(M) \mathbf{r} A$ from assumptions $u : z_u \mathbf{r} C$ and $v : D$.

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. Instead of $\forall_x(x \in \mathrm{Real}^c \to A)$ use

$$\forall_x(x \in \mathrm{Real}^{\mathrm{nc}} \to x \in {}^{\mathrm{co}}I \to A) \quad \text{or}$$
$$\forall_x(x \in \mathrm{Real}^{\mathrm{nc}} \to x \in {}^{\mathrm{co}}G \to A).$$

Computational content of $x \in {}^{\mathrm{co}}I$ ($x \in {}^{\mathrm{co}}G$) is a representation of x as stream (via Gray code).

Method works for average, multiplication and division:

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

both w.r.t. signed digits (${}^{\mathrm{co}}I$) and Gray code (${}^{\mathrm{co}}G$).

Slides:

http://www.math.lmu.de/∼schwicht/slides/swansea19.pdf

Course notes:

http://www.math.lmu.de/∼schwicht/lectures/logic/ss19/index.php

Implementation (Minlog, in examples/analysis):

git clone http://www.math.lmu.de/∼minlogit/git/minlog.git

(Use the development branch: git checkout dev).