Proof theory and computation

Helmut Schwichtenberg

Mathematisches Institut, LMU, München

Waseda University, Tokyo, March 2018

Proof: 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

Overview

- Initial cases of transfinite induction in arithmetic
- Partial continuous functionals
- Paths in trees
- Exact real arithmetic
- Linear two-sorted arithmetic

Goal: study "most complex" proofs in first-order arithmetic.

- The main tool for proving theorems in arithmetic is clearly the induction schema

$$A(0) \to \forall_x (A(x) \to A(Sx)) \to \forall_x A(x).$$

- An equivalent form of this schema is "course-of-values" or cumulative induction

$$\forall_x (\forall_{y<x} A(y) \to A(x)) \to \forall_x A(x).$$

- Both schemes refer to the standard ordering of $\mathbb{N}$. Tempting: strengthen arithmetic by allowing more general induction schemes, e.g., w.r.t. the lexicographical ordering of $\mathbb{N} \times \mathbb{N}$.
- Even more generally, let $\prec$ be a well-ordering of $\mathbb{N}$ and use transfinite induction:

$$\forall_x (\forall_{y \prec x} A(y) \to A(x)) \to \forall_x A(x).$$

Suppose the property $A(x)$ is "progressive", i.e., from the validity of $A(y)$ for all $y \prec x$ we can conclude that $A(x)$ holds. Then $A(x)$ holds for all x.

Recall transfinite induction:

$$\forall_x(\forall_{y \prec x} A(y) \to A(x)) \to \forall_x A(x).$$

For which well-orderings this schema is derivable in arithmetic?

- We will prove a classic result of Gentzen (1943) which in a sense answers this question completely.
- However, to state the result we have to be more explicit about the well-orderings used.

Ordinals below ε_0

- Need some knowledge and notations for ordinals.
- Do not want to assume set theory here.
- Introduce an initial segment of the ordinals (the ones $< \varepsilon_0$) in a formal, combinatorial way, i.e., via ordinal notations based on "Cantor normal form".
- From now on "ordinal" means "ordinal notation".

We define

- α is an ordinal
- $\alpha < \beta$ for ordinals α, β

simultaneously by induction.

1. If $\alpha_m, \dots, \alpha_0$ are ordinals, $m \geq -1$ and $\alpha_m \geq \cdots \geq \alpha_0$ (where $\alpha \geq \beta$ means $\alpha > \beta$ or $\alpha = \beta$), then
$$\omega^{\alpha_m} + \cdots + \omega^{\alpha_0}$$
is an ordinal. The empty sum (denoted by 0) is allowed.
2. If $\omega^{\alpha_m} + \cdots + \omega^{\alpha_0}$ and $\omega^{\beta_n} + \cdots + \omega^{\beta_0}$ are ordinals, then
$$\omega^{\alpha_m} + \cdots + \omega^{\alpha_0} < \omega^{\beta_n} + \cdots + \omega^{\beta_0}$$
iff there is an $i \geq 0$ such that $\alpha_{m-i} < \beta_{n-i}$, $\alpha_{m-i+1} = \beta_{n-i+1}$, $\dots$, $\alpha_m = \beta_n$, or else $m < n$ and $\alpha_m = \beta_n$, $\dots$, $\alpha_0 = \beta_{n-m}$.

- Notation:

$$\begin{aligned} 1 &:= \omega^0, \\ k &:= \omega^0 + \cdots + \omega^0 \quad \text{with } k \text{ copies of } \omega^0, \\ \omega^\alpha k &:= \omega^\alpha + \cdots + \omega^\alpha \quad \text{with } k \text{ copies of } \omega^\alpha. \end{aligned}$$

- $\mathrm{lev}(0) := 0$, $\mathrm{lev}(\omega^{\alpha_m} + \cdots + \omega^{\alpha_0}) := \mathrm{lev}(\alpha_m) + 1$.
- For ordinals of level $k+1$ we have $\omega_k \leq \alpha < \omega_{k+1}$, where $\omega_0 := 0$, $\omega_1 := \omega^1$, $\omega_{k+1} := \omega^{\omega_k}$.
- Easy (by induction on the levels): $<$ is a linear order with 0 the least element.

$$0 < 1 < 2 \cdots < \omega < \omega + 1 \cdots < \omega 2 < \omega 2 + 1 \cdots < \omega 3 \cdots < \omega^2$$
$$< \omega^2 + 1 \cdots < \omega^2 + \omega \cdots < \omega^3 \cdots < \omega^\omega = \omega_2 \cdots < \omega_3 \cdots$$

Addition for ordinals:

$$(\omega^{\alpha_m}+\cdots+\omega^{\alpha_0})+(\omega^{\beta_n}+\cdots+\omega^{\beta_0}) := \omega^{\alpha_m}+\cdots+\omega^{\alpha_i}+\omega^{\beta_n}+\cdots+\omega^{\beta_0}$$

where i is minimal such that $\alpha_i \geq \beta_n$.

Lemma

$+$ is an associative operation which is strictly monotonic in the second argument and weakly monotonic in the first argument.

$+$ is not commutative:

$$1+\omega = \omega \neq \omega + 1$$

There is also a commutative version on addition. The natural sum (or Hessenberg sum) of two ordinals is defined by

$$(\omega^{\alpha_m} + \cdots + \omega^{\alpha_0})\#(\omega^{\beta_n} + \cdots + \omega^{\beta_0}) := \omega^{\gamma_{m+n}} + \cdots + \omega^{\gamma_0},$$

where $\gamma_{m+n}, \ldots, \gamma_0$ is a decreasing permutation of $\alpha_m, \ldots, \alpha_0, \beta_n, \ldots, \beta_0$.

Easy: $\#$ is associative, commutative and strictly monotonic in both arguments.

How ordinals of the form $\beta + \omega^\alpha$ can be approximated from below?

- First note that
$$\delta < \alpha \to \beta + \omega^\delta k < \beta + \omega^\alpha.$$
- For any $\gamma < \beta + \omega^\alpha$ we can find a $\delta < \alpha$ and a k such that
$$\gamma < \beta + \omega^\delta k.$$

Enumerating ordinals

- To work with ordinals in a purely arithmetical system we set up some effective bijection between our ordinals $< \varepsilon_0$ and non-negative integers (i.e., a Gödel numbering).
- For its definition it is useful to refer to ordinals in the form

$$\omega^{\alpha_m} k_m + \cdots + \omega^{\alpha_0} k_0 \quad \text{with } \alpha_m > \cdots > \alpha_0 \text{ and } k_i \neq 0 \ (m \geq -1).$$

(By convention, $m = -1$ corresponds to the empty sum.)

- For every ordinal α we define its Gödel number $\ulcorner\alpha\urcorner$ by

$$\ulcorner\omega^{\alpha_m}k_m + \cdots + \omega^{\alpha_0}k_0\urcorner := \Big(\prod_{i\le m} p_{\ulcorner\alpha_i\urcorner}^{k_i}\Big) - 1,$$

  where p_n is the n-th prime (starting with $p_0 := 2$).
- For every integer $x \ge 0$ we define its ordinal $\mathrm{o}(x)$ by

$$\mathrm{o}\Big(\big(\prod_{i\le l} p_i^{q_i}\big) - 1\Big) := \sum_{i\le l} \omega^{\mathrm{o}(i)} q_i,$$

  where the sum is understood as the natural sum.

Lemma (Bijection between ordinals and non-negative integers)

1. $\mathrm{o}(\ulcorner\alpha\urcorner) = \alpha$,
2. $\ulcorner\mathrm{o}(x)\urcorner = x$.

Can transfer relations and operations on ordinals to computable relations and operations on non-negative integers. Abbreviations:

$$\begin{aligned}
x \prec y &:= \mathrm{o}(x) < \mathrm{o}(y), \\
\omega^x &:= \ulcorner \omega^{\mathrm{o}(x)} \urcorner, \\
x \oplus y &:= \ulcorner \mathrm{o}(x) + \mathrm{o}(y) \urcorner, \\
xk &:= \ulcorner \mathrm{o}(x)k \urcorner, \\
\omega_k &:= \ulcorner \omega_k \urcorner.
\end{aligned}$$

Provability of initial cases of transfinite induction

- We derive initial cases of transfinite induction in arithmetic:

$$\forall_x(\forall_{y \prec x} Py \to Px) \to \forall_{x \prec a} Px$$

  for some number a and a predicate symbol P, where $\prec$ is the standard order of order type ε_0 defined before.
- This result is optimal in the sense that for the full system of ordinals $< \varepsilon_0$ the principle

$$\forall_x(\forall_{y \prec x} Py \to Px) \to \forall_x Px$$

  of transfinite induction is underivable.
- All these results are due to Gentzen (1943).

Arithmetical systems

By an arithmetical system **Z** we mean a theory based on minimal logic in the $\forall\to$-language (including equality axioms) such that

- The language of **Z** consists of a fixed supply of function and relation constants assumed to denote computable functions and relations on the non-negative integers.
- Among the function constants there is a constant S for the successor function and 0 for (the 0-place function) zero.
- Among the relation constants we have $=$, P and also $\prec$ for the ordering of type ε_0 of $\mathbb{N}$, as introduced before.

- Terms are built up from object variables x, y, z by $f(t_1, \dots, t_m)$, where f is a function constant.
- We identify closed terms which have the same value; this expresses that each function constant is computable.
- Terms of the form $\mathrm{S}(\mathrm{S}(\dots \mathrm{S}0 \dots))$ are called numerals. Notation: $\mathrm{S}^n 0$ or $\overline{n}$ or just n.
- Formulas are built up from atomic formulas $R(t_1, \dots, t_m)$, with R a relation constant, by $A \to B$ and $\forall_x A$.

Axioms of **Z**

- Compatibility of equality

$$x = y \to A(x) \to A(y),$$

- the Peano axioms, i.e., the universal closures of

$$\begin{align} \mathrm{S}x = \mathrm{S}y \to x = y, \tag{1} \\ \mathrm{S}x = 0 \to A, \tag{2} \\ A(0) \to \forall_x (A(x) \to A(\mathrm{S}x)) \to \forall_x A(x), \tag{3} \end{align}$$

  with $A(x)$ an arbitrary formula.
- $R\vec{n}$ whenever $R\vec{n}$ is true (to express that R is computable).
- Irreflexivity and transitivity for $\prec$

$$x \prec x \to A,$$
$$x \prec y \to y \prec z \to x \prec z$$

Axioms of **Z** (continued)

Further axioms – following Schütte – are the universal closures of

$$x \prec 0 \to A, \tag{4}$$
$$z \prec y \oplus \omega^0 \to (z \prec y \to A) \to (z = y \to A) \to A, \tag{5}$$
$$x \oplus 0 = x, \tag{6}$$
$$x \oplus (y \oplus z) = (x \oplus y) \oplus z, \tag{7}$$
$$0 \oplus x = x, \tag{8}$$
$$\omega^x 0 = 0, \tag{9}$$
$$\omega^x(\mathrm{S}y) = \omega^x y \oplus \omega^x, \tag{10}$$
$$z \prec y \oplus \omega^{\mathrm{S}x} \to z \prec y \oplus \omega^{e(x,y,z)} m(x,y,z), \tag{11}$$
$$z \prec y \oplus \omega^{\mathrm{S}x} \to e(x,y,z) \prec \mathrm{S}x, \tag{12}$$

where $\oplus$, $\lambda_{x,y}(\omega^x y)$, e and m denote function constants and A is any formula. These axioms are formal counterparts to the properties of the ordinal notations observed above.

Theorem (Provable initial cases of transfinite induction in **Z**)

Transfinite induction up to ω_n, i.e., for arbitrary $A(x)$ the formula

$$\forall_x(\forall_{y \prec x} A(y) \to A(x)) \to \forall_{x \prec \omega_n} A(x),$$

is derivable in **Z**.

Proof. To every formula $A(x)$ we assign a formula $A^+(x)$ (with respect to a fixed variable x) by

$$A^+(x) := \forall_y(\forall_{z \prec y} A(z) \to \forall_{z \prec y \oplus \omega^x} A(z)).$$

We first show

$$\text{If } A(x) \text{ is progressive, then } A^+(x) \text{ is progressive,}$$

where "$B(x)$ is progressive" means $\forall_x(\forall_{y \prec x} B(y) \to B(x))$.

If $A(x)$ is progressive, then $A^+(x)$ is progressive

Assume that $A(x)$ is progressive and

$$\forall_{y \prec x} A^+(y). \tag{13}$$

Goal: $A^+(x) := \forall_y (\forall_{z \prec y} A(z) \to \forall_{z \prec y \oplus \omega^x} A(z))$. Assume

$$\forall_{z \prec y} A(z) \tag{14}$$

and $z \prec y \oplus \omega^x$. We have to show $A(z)$.

Case $x = 0$. Then $z \prec y \oplus \omega^0$. By (5):

$$z \prec y \oplus \omega^0 \to (z \prec y \to A) \to (z = y \to A) \to A$$

it suffices to derive $A(z)$ from $z \prec y$ as well as from $z = y$.
If $z \prec y$, then $A(z)$ follows from (14), and if $z = y$, then $A(z)$ follows from (14) and the progressiveness of $A(x)$.

If $A(x)$ is progressive, then $A^+(x)$ is progressive (ctd.)

Case $\mathrm{S}x$. From $z \prec y \oplus \omega^{\mathrm{S}x}$ we obtain $z \prec y \oplus \omega^{e(x,y,z)} m(x,y,z)$ by (11) and $e(x,y,z) \prec \mathrm{S}x$ by (12). By (13): $A^+(e(x,y,z))$, i.e.

$$\forall_{u \prec y \oplus \omega^{e(x,y,z)} v} A(u) \to \forall_{u \prec (y \oplus \omega^{e(x,y,z)} v) \oplus \omega^{e(x,y,z)}} A(u)$$

and hence, using (7) and (10)

$$\forall_{u \prec y \oplus \omega^{e(x,y,z)} v} A(u) \to \forall_{u \prec y \oplus \omega^{e(x,y,z)} (\mathrm{S}v)} A(u).$$

Also from (14) and (9), (6) we obtain

$$\forall_{u \prec y \oplus \omega^{e(x,y,z)} 0} A(u).$$

By induction:

$$\forall_{u \prec y \oplus \omega^{e(x,y,z)} m(x,y,z)} A(u)$$

and hence $A(z)$.

Next: show, by induction on n, how to derive

$$\forall_x(\forall_{y \prec x} A(y) \to A(x)) \to \forall_{x \prec \omega_n} A(x) \quad \text{for arbitrary } A(x).$$

Assume the left hand side, i.e., that $A(x)$ is progressive.

Case 0. Then $x \prec \omega^0$ and hence $x \prec 0 \oplus \omega^0$ by (8). By (5) it suffices to derive $A(x)$ from $x \prec 0$ as well as from $x = 0$. Now $x \prec 0 \to A(x)$ holds by (4), and $A(0)$ then follows from the progressiveness of $A(x)$.

Case $n+1$. Since $A(x)$ is progressive, by the above also $A^+(x)$ is. By IH: $\forall_{x \prec \omega_n} A^+(x)$, hence $A^+(\omega_n)$ since $A^+(x)$ is progressive. By definition of $A^+(x)$ (with (4): $x \prec 0 \to A$ and (8): $0 \oplus x = x$) we obtain $\forall_{z \prec \omega^{\omega_n}} A(z)$.

Definition (Level of a formula)

$$\begin{aligned}
\mathrm{lev}(R\vec{t}\,) &:= 0, \\
\mathrm{lev}(A \to B) &:= \max(\mathrm{lev}(A)+1, \mathrm{lev}(B)), \\
\mathrm{lev}(\forall_x A) &:= \max(1, \mathrm{lev}(A)).
\end{aligned}$$

- In the induction step we derived transfinite induction up to ω_{n+1} for $A(x)$ from transfinite induction up to ω_n for $A^+(x)$, and $\mathrm{lev}(A^+(x)) = \mathrm{lev}(A(x)) + 1$.
- Hence to prove transfinite induction up to ω_n, the induction scheme in **Z** is used for formulas of level n.
- Next: iteration operators of higher (type-)level have a similar relation to ordinals $< \varepsilon_0$.

The fast growing functions $(F_\alpha)_{\alpha<\varepsilon_0}$, are defined by

$$\begin{aligned} F_0(x) &:= 2^x, \\ F_{\alpha+1}(x) &:= F_\alpha^{(x)}(x) \quad (F_\alpha^{(x)} \text{ } x\text{-th iterate of } F_\alpha), \\ F_\lambda(x) &:= F_{\lambda[x]}(x). \end{aligned}$$

Here the fundamental sequence $\lambda[x]$ for a limit number $\lambda < \varepsilon_0$ and $x \in \mathbb{N}$ is defined in a natural way:

- Any such limit number can be written uniquely in the form $\lambda = \omega^{\alpha_n} + \ldots + \omega^{\alpha_0}$ with $\lambda > \alpha_n \geq \ldots \geq \alpha_0 > 0$.
- Then let

$$\lambda[x] := \begin{cases} \omega^{\alpha_n} + \ldots + \omega^{\alpha_1} + \omega^{\alpha_0 - 1} \cdot x & \text{if } \alpha_0 \text{ is a successor} \\ \omega^{\alpha_n} + \ldots + \omega^{\alpha_1} + \omega^{\alpha_0[x]} & \text{if } \alpha_0 \text{ is a limit.} \end{cases}$$

The extended Grzegorczyk hierarchy

Let $\mathcal{E}_\alpha$ be the elementary closure of F_α, i.e., the least class of functions containing F_α and the initial functions

$$
\begin{aligned}
&U_n^i := \lambda_{x_1,\dots,x_n} x_i \quad (\text{for } 1 \le i \le n),\\
&C_n^i := \lambda_{x_1,\dots,x_n} i \quad (\text{for } n \ge 0,\ i \ge 0),\\
&\lambda_{x,y}(x + y)\\
&\lambda_{x,y}(x \dot{-} y)
\end{aligned}
$$

closed against (simultaneous) substitution and bounded sums and products.

There are many characterizations of $\mathcal{E}_\alpha$, for instance

> *$\mathcal{E}_\alpha$ consists of all functions computable by a register machine with time (i.e. number of computation steps) bounded by a finite iteration of the function F_α.*

Let

$$\mathcal{E}_{\varepsilon_0} := \bigcup_{\alpha<\varepsilon_0} \mathcal{E}_\alpha \quad \text{“}\varepsilon_0\text{-recursive functions”}$$

Theorem (Ackermann 1940, Kreisel 1952)

$\mathcal{E}_{\varepsilon_0}$ consists of the functions “provably recursive” in arithmetic.

One can define natural subsystems of arithmetic whose provably recursive functions are exactly the ones in $\mathcal{E}_\alpha$.

Characterization of $(F_\alpha)_{\alpha<\varepsilon_0}$ by higher type iteration

- We extend the definition of the functions F_α into higher types.
- It is convenient here to introduce integer types ρ_n:

$$\rho_0 := \mathbb{N},$$
$$\rho_{n+1} := \rho_n \to \rho_n.$$

If $x_0, \dots, x_{n+1}$ are of integer types $\rho_0, \dots, \rho_{n+1}$, then we can form

- $x_{n+1}(x_n)$ (of type ρ_n) and so on, finally
- $x_{n+1}(x_n)\dots(x_0)$, abbreviated $x_{n+1}(x_n, \dots, x_0)$.

Note that $\mathrm{lev}(\rho_n) = n$.

We define F^{n+1}_α of type ρ_{n+1} for $\alpha < \varepsilon_0$:

$$F^{n+1}_0(x_n, \dots, x_0) := \begin{cases} 2^{x_0} & \text{if } n = 0 \\ x_n^{(x_0)}(x_{n-1}, \dots, x_0) & \text{otherwise.} \end{cases}$$
$$F^{n+1}_{\alpha+1}(x_n, \dots, x_0) := (F^{n+1}_\alpha)^{(x_0)}(x_n, \dots, x_0),$$
$$F^{n+1}_\lambda(x_n, \dots, x_0) := F^{n+1}_{\lambda[x_0]}(x_n, \dots, x_0).$$

Here $x_n^{(y)}(x_{n-1}, \dots, x_0)$ denotes $I(y, x_n, \dots, x_0)$ with an iteration functional I of type $\mathbb{N} \to \rho_n \to \rho_{n-1} \to \dots \to \rho_0 \to \rho_0$ defined by

$$I(0, y, z) := z,$$
$$I(x+1, y, z) := y(I(x, y, z)).$$

Lemma

$$F_{\alpha}^{n+1}(F_{\beta}^{n}) = F_{\beta+\omega^{\alpha}}^{n}$$

provided $\beta + \omega^{\alpha} = \beta \# \omega^{\alpha}$, i.e., in the Cantor normal form of β the last summand ω^{β_0} (if it exists) has an exponent $\beta_0 \geq \alpha$.

Proof. By induction on α. *Case* $\alpha = 0$.

$$\begin{aligned} F_{0}^{n+1}(F_{\beta}^{n}, x_{n-1}, \dots, x_0) &= (F_{\beta}^{n})^{(x_0)}(x_{n-1}, \dots, x_0) \\ &= F_{\beta+1}^{n}(x_{n-1}, \dots, x_0) \end{aligned}$$

Case α successor.

$$\begin{aligned} F_{\alpha}^{n+1}(F_{\beta}^{n}, x_{n-1}, \dots, x_0) &= (F_{\alpha-1}^{n+1})^{(x_0)}(F_{\beta}^{n}, x_{n-1}, \dots, x_0) \\ &= F_{\beta+\omega^{\alpha-1}\cdot x_0}^{n}(x_{n-1}, \dots, x_0) \quad \text{by IH} \\ &:= F_{(\beta+\omega^{\alpha})[x_0]}^{n}(x_{n-1}, \dots, x_0) \\ &:= F_{\beta+\omega^{\alpha}}^{n}(x_{n-1}, \dots, x_0) \end{aligned}$$

Recall the claim

$$F_\alpha^{n+1}(F_\beta^n) = F^n_{\beta+\omega^\alpha} \quad \text{if } \beta + \omega^\alpha = \beta \# \omega^\alpha.$$

Case α limit.

$$\begin{aligned} F_\alpha^{n+1}(F_\beta^n, x_{n-1}, \ldots, x_0) &= F^{n+1}_{\alpha[x_0]}(F^n_\beta, x_{n-1}, \ldots, x_0) \\ &= F^n_{\beta+\omega^{\alpha[x_0]}}(x_{n-1}, \ldots, x_0) \quad \text{by IH} \\ &= F^n_{(\beta+\omega^\alpha)[x_0]}(x_{n-1}, \ldots, x_0) \\ &= F^n_{\beta+\omega^\alpha}(x_{n-1}, \ldots, x_0). \end{aligned}$$

The strength of finite types

- $(F_\alpha^{n+1})_{\alpha<\varepsilon_0}$ and in particular $(F_\alpha^1)_{\alpha<\varepsilon_0}$ can be built from iteration functionals (and $F_0(x) = 2^x$) by application alone.
- In the resulting representation of the functions $(F_\alpha)_{\alpha<\varepsilon_0}$ we do not need the fundamental sequences $\lambda[x]$.
- The application pattern for F_α corresponds to the Cantor normal form of α.

Overview

- Initial cases of transfinite induction in arithmetic
- Partial continuous functionals
- Paths in trees
- Exact real arithmetic
- Linear two-sorted arithmetic

Computable 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).

Information system $\mathbf{A} = (A, \mathrm{Con}, \vdash)$:

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

such that

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

$x \subseteq A$ is an ideal if

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

Function spaces

Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems. Define $\mathbf{A} \to \mathbf{B} := (C, \mathrm{Con}, \vdash)$ 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 $\{ b_i \mid U \vdash_A U_i \} \vdash_B b$.

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

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

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

(Free) algebras given by constructors:

$\mathbb{N}$	by	$0^{\mathbb{N}}, \mathrm{S}^{\mathbb{N}\to\mathbb{N}}$
$\alpha \times \beta$	by	$\langle .,. \rangle^{\alpha\to\beta\to\alpha\times\beta}$
$\alpha + \beta$	by	$(\mathrm{InL}_{\alpha\beta})^{\alpha\to\alpha+\beta}, (\mathrm{InR}_{\alpha\beta})^{\beta\to\alpha+\beta}$
$\mathbb{L}(\alpha)$	by	$\mathrm{Nil}^{\mathbb{L}(\alpha)}, \mathrm{Cons}^{\alpha\to\mathbb{L}(\alpha)\to\mathbb{L}(\alpha)}$
$\mathbb{S}(\alpha)$	by	$\mathrm{SCons}^{\alpha\to\mathbb{S}(\alpha)\to\mathbb{S}(\alpha)}$

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

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

$\mathbf{C}_{\rho\to\sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma$. 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.

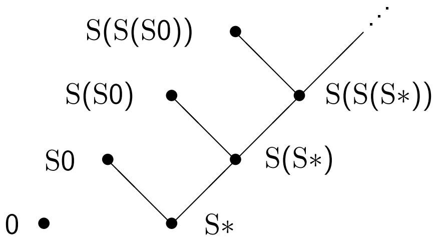
(Example: $\{C{*}0, C0{*}\}$.)

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

Tokens and entailment for $\mathbb{N}$



$\{a\} \vdash b$ iff there is a path from a (up) to b (down).

Definition

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

Ideals in $\mathbf{C}_\rho$: Scott-Ershov domain of type ρ.
Principles of finite support and monotonicity hold ("continuity").

- 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" $b(\mathrm{C}\vec{*}) \in x$.

Examples:

$$\{\, 01^m \mid m \leq 3 \,\} \cup \{\, 01^n * \mid n < 3 \,\} \quad \text{total,}$$
$$\{\, 01^m \mid m \geq 0 \,\} \cup \{\, 01^m * \mid m \geq 0 \,\} \quad \text{cototal.}$$

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

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

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

$$\begin{aligned} n + 0 &= n \\ n + \mathrm{S}m &= \mathrm{S}(n + m) \end{aligned}$$

- $Y \colon (\tau \to \tau) \to \tau$ defined by

$$Yf = f(Yf)$$

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

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

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: $\mathcal{R}_{\mathbb{N}}^{\tau}$ 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.

Examples

$$\begin{aligned}
\mathcal{R}^{\tau}_{\mathbb{B}} &\colon \mathbb{B} \to \tau \to \tau \to \tau, \\
\mathcal{R}^{\tau}_{\mathbb{N}} &\colon \mathbb{N} \to \tau \to (\mathbb{N} \to \tau \to \tau) \to \tau, \\
\mathcal{R}^{\tau}_{\mathbb{L}(\rho)} &\colon \mathbb{L}(\rho) \to \tau \to (\rho \to \mathbb{L}(\rho) \to \tau \to \tau) \to \tau, \\
\mathcal{R}^{\tau}_{\rho+\sigma} &\colon \rho + \sigma \to (\rho \to \tau) \to (\sigma \to \tau) \to \tau, \\
\mathcal{R}^{\tau}_{\rho\times\sigma} &\colon \rho \times \sigma \to (\rho \to \sigma \to \tau) \to \tau.
\end{aligned}$$

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

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

How to use computation rules to define a computable functional?
Inductively define $(\vec{U}, a) \in [\![\lambda_{\vec{x}} M]\!]$, where M is a term with free variables among $\vec{x}$.

Case $\lambda_{\vec{x},y,\vec{z}} M$ with $\vec{x}$ free in M, but not y.

$$\frac{(\vec{U}, \vec{W}, a) \in [\![\lambda_{\vec{x},\vec{z}} M]\!]}{(\vec{U}, V, \vec{W}, a) \in [\![\lambda_{\vec{x},y,\vec{z}} M]\!]}(K).$$

Case $\lambda_{\vec{x}} M$ with $\vec{x}$ the free variables in M.

$$\frac{U \vdash a}{(U, a) \in [\![\lambda_x x]\!]}(V), \quad \frac{(\vec{U}, V, a) \in [\![\lambda_{\vec{x}} M]\!] \quad (\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} N]\!]}{(\vec{U}, a) \in [\![\lambda_{\vec{x}} (MN)]\!]}(A).$$

For every constructor C and defined constant D:

$$\frac{\vec{U} \vdash \vec{a^*}}{(\vec{U}, \mathrm{C}\vec{a^*}) \in [\![\mathrm{C}]\!]}(\mathrm{C}), \quad \frac{(\vec{V}, a) \in [\![\lambda_{\vec{x}} M]\!] \quad \vec{U} \vdash \vec{P}(\vec{V})}{(\vec{U}, a) \in [\![D]\!]}(D),$$

with one rule (D) for every defining equation $D\vec{P}(\vec{x}\,) = M$.

Properties of the denotational semantics

- $[\![\lambda_{\vec{x}} M]\!]$ is a partial continuous functional.
- The value is preserved under β, η-conversion and computation rules.
- An adequacy theorem holds: whenever a closed term M^{ι} has a proper token in its denotation $[\![M]\!]$, then M (head) reduces to a constructor term entailing this token.

- $\mathcal{C} := (|\mathbf{C}_\rho|)_\rho$ Scott-Ershov model of partial continuous functionals.
- "x^ρ computable" for $x^\rho \in |\mathbf{C}_\rho|$ is defined.
- Goal: TCF, a theory of partial computable functionals.

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

- Partial functionals allowed.
- Formulas and types are kept separate (Aczel & Gambino (2006): "logic enriched type theory").

Predicates and formulas

$$P, Q ::= X \mid \{\, \vec{x} \mid A \,\} \mid \mu_X(\forall_{\vec{x}_i}((A_{i\nu})_{\nu<n_i} \to X\vec{r}_i))_{i<k} \mid \nu_X(\dots)$$
$$A, B ::= P\vec{r} \mid A \to B \mid \forall_x A$$

Example: $\mathrm{Even} := \mu_X(X0, \forall_n(Xn \to X(\mathrm{S}(\mathrm{S}n))))$.

Predicate variables X and (co)inductive predicates can be computationally relevant (c.r.) or non-computational (n.c).
Example: $T_{\mathbb{N}}$ (c.r.) and $T_{\mathbb{N}}^{\mathrm{nc}}$ (n.c.)

Clauses and least-fixed-point (induction) axiom for $T_{\mathbb{N}}$:

$$(T_{\mathbb{N}}^{+})_0 \colon 0 \in T_{\mathbb{N}}$$
$$(T_{\mathbb{N}}^{+})_1 \colon \forall_n(n \in T_{\mathbb{N}} \to \mathrm{S}n \in T_{\mathbb{N}})$$

$$T_{\mathbb{N}}^{-} \colon 0 \in X \to \forall_n(n \in T_{\mathbb{N}} \to n \in X \to \mathrm{S}n \in X) \to T_{\mathbb{N}} \subseteq X$$

and similar for the n.c. variant $T_{\mathbb{N}}^{\mathrm{nc}}$.

Coinductive predicates: ${}^{\mathrm{co}}T_{\mathbb{N}}$ (c.r.) and ${}^{\mathrm{co}}T_{\mathbb{N}}^{\mathrm{nc}}$ (n.c.)

Closure and greatest-fixed-point (coinduction) axioms for ${}^{\mathrm{co}}T_{\mathbb{N}}$:

$${}^{\mathrm{co}}T_{\mathbb{N}}^{-} : \forall_n (n \in {}^{\mathrm{co}}T_{\mathbb{N}} \to n \equiv 0 \vee \exists_{n'}(n' \in {}^{\mathrm{co}}T_{\mathbb{N}} \wedge n \equiv \mathrm{S}n'))$$

$$\begin{aligned}{}^{\mathrm{co}}T_{\mathbb{N}}^{+} : {} & \forall_n (n \in X \to n \equiv 0 \vee \exists_{n'}((n' \in {}^{\mathrm{co}}T_{\mathbb{N}} \vee n' \in X) \wedge n \equiv \mathrm{S}n')) \to \\ & X \subseteq {}^{\mathrm{co}}T_{\mathbb{N}}\end{aligned}$$

and similar for the n.c. variant ${}^{\mathrm{co}}T_{\mathbb{N}}^{\mathrm{nc}}$ (with X^{nc}, $\vee^{\mathrm{nc}}$ for X, $\vee$).

Example of a proof by coinduction

- We show cototality of corecursion: ${}^{\mathrm{co}}\mathcal{R}xf \in {}^{\mathrm{co}}T_{\mathbb{S}(\rho)}$.
- Recall: 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

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

Lemma (Cototality of corecursion)

Let $f : \tau \to \rho \times (\mathbb{S}(\rho) + \tau)$ *be of* InR*-form, i.e.,* $f(x)$ *has the form* $\langle y, \mathrm{InR}(x')\rangle$ *for all* x*. Then* ${}^{\mathrm{co}}\mathcal{R}xf \in {}^{\mathrm{co}}T_{\mathbb{S}(\rho)}$ *for all* x*.*

Proof.

By coinduction with competitor predicate

$$X := \{\, z \mid \exists_x (z \equiv {}^{\mathrm{co}}\mathcal{R}xf\,)\,\}.$$

Need to prove that X satisfies the clause defining ${}^{\mathrm{co}}T_{\mathbb{S}(\rho)}$:

$$\forall_z (z \in X \to \exists_y \exists_{z'} (z' \in X \wedge z \equiv \mathrm{SCons}(y, z'))).$$

Let $z \equiv {}^{\mathrm{co}}\mathcal{R}xf$ for some x. Since f is assumed to be of InR-form we have y, x' such that $f(x) \equiv \langle y, \mathrm{InR}(x')\rangle$. By the definition of ${}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{S}(\rho)}$ obtain ${}^{\mathrm{co}}\mathcal{R}xf \equiv \mathrm{SCons}(y, {}^{\mathrm{co}}\mathcal{R}x'f)$. Use ${}^{\mathrm{co}}\mathcal{R}x'f \in X$. $\square$

Existence $\exists$, conjunction $\wedge$, disjunction $\vee$, $\vee^{\mathrm{nc}}$

These are nullary inductive predicates with parameters

$$
\begin{aligned}
\exists^{+}&\colon \forall_x(x \in P \to \exists_x(x \in P))\\
\exists^{-}&\colon \exists_x(x \in P) \to \forall_x(x \in P \to C) \to C \qquad (x \notin \mathrm{FV}(C))\\
\wedge^{+}&\colon A \to B \to A \wedge B\\
\wedge^{-}&\colon A \wedge B \to (A \to B \to C) \to C\\
\vee_i^{+}&\colon A_i \to A_0 \vee A_1\\
\vee^{-}&\colon A \vee B \to (A \to C) \to (B \to C) \to C\\
(\vee_i^{\mathrm{nc}})^{+}&\colon A_i \to A_0 \vee^{\mathrm{nc}} A_1\\
(\vee^{\mathrm{nc}})^{-}&\colon A \vee^{\mathrm{nc}} B \to (A \to C) \to (B \to C) \to C \quad (C \text{ n.c.})
\end{aligned}
$$

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 $\mathbb{N} \to \mathbb{N}$.

Type $\tau(C)$ of a c.r. predicate or formula C

$$\tau(X) := \xi \qquad (\xi \text{ uniquely assigned to } X)$$

$$\tau(\{\, \vec{x} \mid A \,\}) := \tau(A)$$

$$\tau(\underbrace{\mu_X(\forall_{\vec{x}_i}((A_{i\nu})_{\nu<n_i} \to X\vec{r}_i))_{i<k}}_{I}) := \underbrace{\mu_\xi((\tau(A_{i\nu})_{\nu<n_i}) \to \xi)_{i<k}}_{\iota_I}$$

(similar for ${}^{\mathrm{co}}I$)

$$\tau(P\vec{r}\,) := \tau(P)$$

$$\tau(A \to B) := \begin{cases} \tau(A) \to \tau(B) & (A \text{ c.r.}) \\ \tau(B) & (A \text{ n.c.}) \end{cases}$$

$$\tau(\forall_x A) := \tau(A)$$

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

We write $z \mathbin{\mathbf{r}} C$ for $C^{\mathbf{r}} z$ if C is a formula.

$$\begin{aligned}
X^{\mathbf{r}} \quad & \text{(uniquely assigned to } X\text{: } (\vec{\rho}), \text{ of arity } (\tau(X), \vec{\rho})) \\
\{\, \vec{x} \mid A \,\}^{\mathbf{r}} &:= \{\, z, \vec{x} \mid z \mathbin{\mathbf{r}} A \,\} \\
I^{\mathbf{r}}, {}^{\mathrm{co}}I^{\mathbf{r}} \quad & \text{(by examples)}
\end{aligned}$$

$$\begin{aligned}
z \mathbin{\mathbf{r}} P\vec{r} &:= P^{\mathbf{r}}(z, \vec{r}\,) \\
z \mathbin{\mathbf{r}} (A \to B) &:= \begin{cases} \forall_w (w \mathbin{\mathbf{r}} A \to zw \mathbin{\mathbf{r}} B) & \text{if } A \text{ is c.r.} \\ A \to z \mathbin{\mathbf{r}} B & \text{if } A \text{ is n.c.} \end{cases} \\
z \mathbin{\mathbf{r}} \forall_x A &:= \forall_x (z \mathbin{\mathbf{r}} A)
\end{aligned}$$

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

$$
\begin{array}{ll}
\mathrm{et}(u^A) & := z_u^{\tau(A)} \quad (z_u^{\tau(A)} \text{ uniquely associated to } u^A) \\
\mathrm{et}((\lambda_{u^A} M^B)^{A\to B}) & := \begin{cases} \lambda_{z_u}^{\tau(A)} \mathrm{et}(M) & \text{if } A \text{ is c.r.} \\ \mathrm{et}(M) & \text{if } A \text{ is n.c.} \end{cases} \\
\mathrm{et}((M^{A\to B} N^A)^B) & := \begin{cases} \mathrm{et}(M)\mathrm{et}(N) & \text{if } A \text{ is c.r.} \\ \mathrm{et}(M) & \text{if } A \text{ is n.c.} \end{cases} \\
\mathrm{et}((\lambda_x M^A)^{\forall_x A}) & := \mathrm{et}(M) \\
\mathrm{et}((M^{\forall_x A(x)} t)^{A(t)}) & := \mathrm{et}(M) \\
\mathrm{et}(I_i^+) & := \mathrm{C}_i \quad (i\text{-th constructor of } \iota \text{ associated to } I) \\
\mathrm{et}(I^-) & := \mathcal{R}_\iota^\tau \quad (\text{recursor for } \iota) \\
\mathrm{et}({}^{\mathrm{co}}I^-) & := \mathrm{D}_\iota \quad (\text{destructor of } \iota \text{ associated to } {}^{\mathrm{co}}I) \\
\mathrm{et}({}^{\mathrm{co}}I^+) & := {}^{\mathrm{co}}\mathcal{R}_\iota^\tau \quad (\text{corecursor for } \iota)
\end{array}
$$

An n.c. part of M is a subderivation with an n.c. end formula. In n.c. parts in we identify any I with I^{nc}.

Theorem (Soundness)

Let M be a derivation of a formula A from assumptions $u\colon C$ (c.r.) and $v\colon D$ (n.c.) Then we can find a derivation of

$$\begin{cases} \mathrm{et}(M) \mathbf{r} A & \text{if } A \text{ is c.r.} \\ A & \text{if } A \text{ is n.c.} \end{cases}$$

from assumptions $z_u \mathbf{r} C$ and D.

Proof.

By induction on M. Cases: $\to^{\pm}$, $\forall^{\pm}$ and c.r. axioms. $\square$

Overview

- Initial cases of transfinite induction in arithmetic
- Partial continuous functionals
- Paths in trees
- Exact real arithmetic
- Linear two-sorted arithmetic

Paths in trees

Paths x are functions of type $\mathbb{N} \to \mathbb{B}$.

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

$$\begin{array}{ll} \forall_x (x \in T_{\mathbb{N}\to\mathbb{B}} \to A) & \text{of type } (\mathbb{N} \to \mathbb{B}) \to \tau(A), \\ \forall_x (x \in {}^{\mathrm{co}}I \to A) & \text{of type } \mathbb{S}(\mathbb{B}) \to \tau(A). \end{array}$$

Why?

- Type level 1 rather than 2.
- Theory uses functions, but extracted terms use streams.

What is ${}^{\mathrm{co}}I$?

$$\Phi(X) := \{\, x \mid \forall_{n \in T_{\mathbb{N}}}(x(n) \in T_{\mathbb{B}}^{\mathrm{nc}}) \wedge \exists_{p \in T_{\mathbb{N}}} \exists_{x' \in X}(x \equiv px') \,\}.$$

Then

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

satisfy the (strengthened) axioms

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

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

- View trees as sets of nodes u, v, w of type $\mathbb{L}(\mathbb{B})$ (lists of booleans), which are downward closed.
- Sets of nodes are given by (not necessarily total) functions b of type $\mathbb{L}(\mathbb{B}) \to \mathbb{B}$. To be or not to be in b is expressed by saying that $b(u)$ is defined with 1 or 0 as its value.
- A set b of nodes is a bar if every path s hits the bar, i.e., there is an n such that $\bar{x}(n) \in b$.

For simplicity assume that bars b are upwards closed:

$$\forall_{u,p}(u \in b \to pu \in b).$$

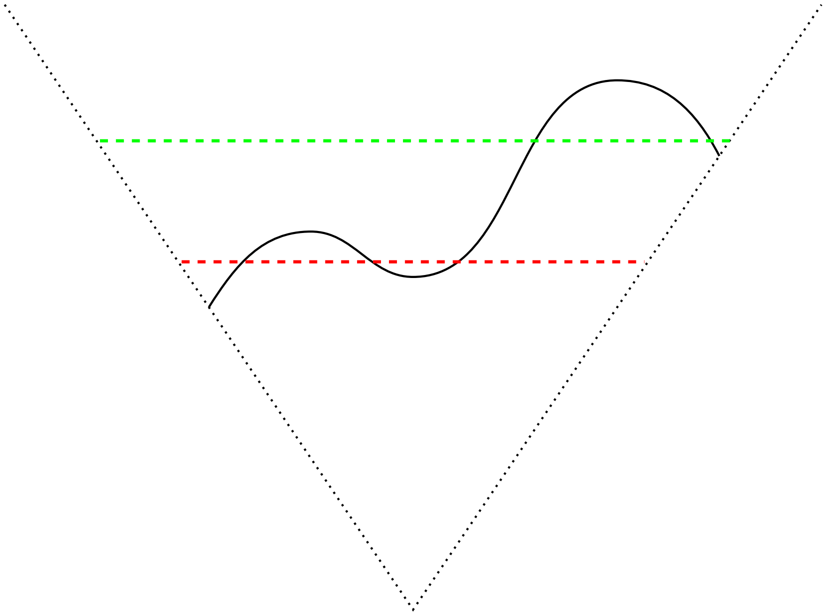
- Josef Berger and Gregor Svindland recently gave a constructive proof of the fan theorem for "coconvex" bars.
- They call a set $b \subseteq \{0,1\}^*$ coconvex if for every n and path x

  $$\bar{x}(n) \in b \to \exists_m \big(\forall_{v \le \bar{x}(m)}(v \in b) \vee \forall_{v \ge \bar{x}(m)}(v \in b)\big),$$

  where $v \le w$ means $|v| = |w|$ and v is left of w. Equivalently

  $$\begin{aligned} \bar{x}(n) \in b \to \exists_{p,m}\big((p = 0 \to \forall_{v \le \bar{x}(m)}(v \in b)) \wedge \\ (p = 1 \to \forall_{v \ge \bar{x}(m)}(v \in b))\big). \end{aligned}$$

  Two "moduli" p and m, depending on x, n and b.
- Better "finally coconvex", with coconvex in the sense that the b-nodes at height n form the complement of a convex set.



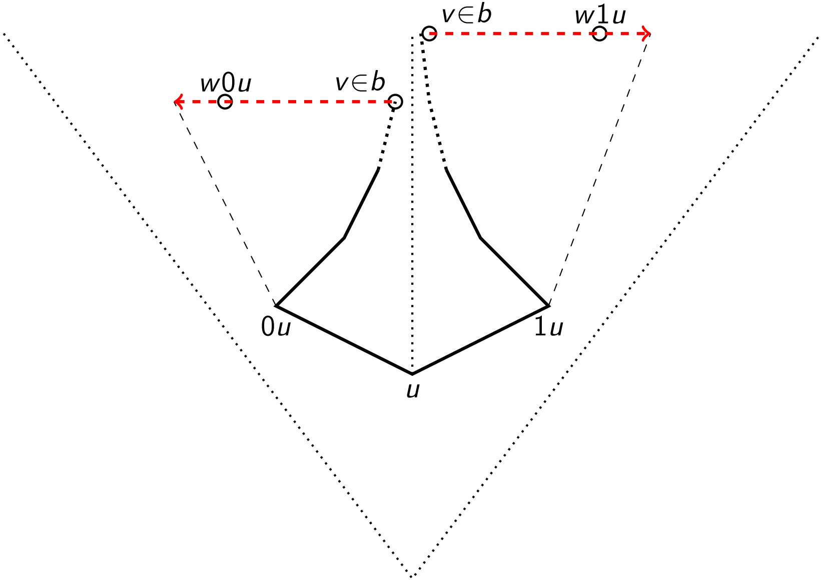
Uniform coconvexity with modulus d (direction)

- Simplification: p only, depending on node u (i.e., $p = d(u)$).
- Special case of the B&S concept. Goal: better algorithm.

Definition

A set $b \subseteq \{0,1\}^*$ is uniformly coconvex with modulus d if for all u we have: if the innermost path from pu (where $p := d(u)$) hits b in some node $v \in b$, then

$$\begin{cases} \forall_w (wpu \leq v \to wpu \in b) & \text{if } p = 0, \\ \forall_w (wpu \geq v \to wpu \in b) & \text{if } p = 1. \end{cases}$$


$v{\in}b$
$w1u$
$w0u$
$v{\in}b$
$0u$
$1u$
u

The escape path

- $N_d(u) := d^-(u)u$ extends u in the opposite direction to what d says.
- $U_d(n)$ is the nth node in the escape path

$$\begin{aligned} U_d(0) &:= \mathrm{Nil}, \\ U_d(n+1) &:= N_d(U_d(n)). \end{aligned}$$

- $A_d \colon \mathbb{N} \to \mathbb{B}$ is the escape path:

$$A_d(n) := d^-(U_d(n)).$$

Definition (Distance)

$$D_b n u := \forall_v (|v| = n \to vu \in b)$$

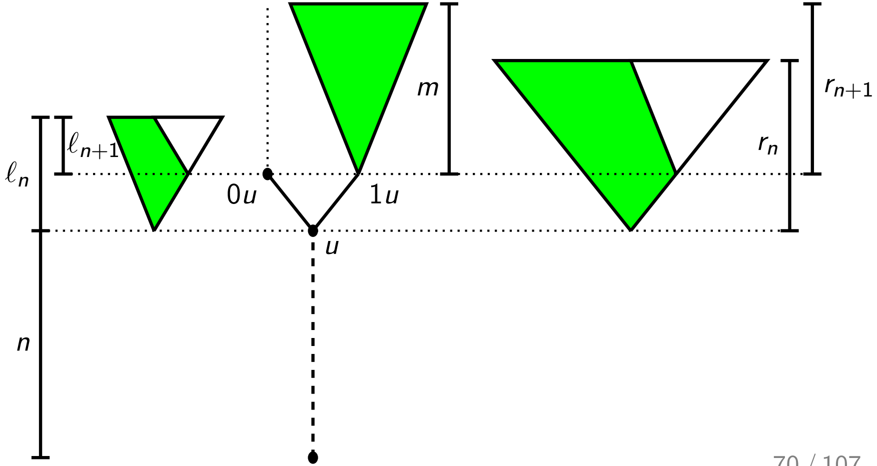
"u has distance n from the bar b".

Lemma (BoundL, BoundR)

Let b be a uniformly coconvex bar with modulus d. Then for every n there are bounds ℓ_n, r_n for the b-distances of all nodes of the same length n that are left / right of $U_d n$

Proof. For $n = 0$ there are no such nodes.

Consider $U_d(n+1) = A_d(n)u$ of length $n+1$. Assume $A_d(n) = 0$. Then every node to the left of $0u$ is a successor node of one to the left of u, and hence $\ell_{n+1} = \ell_n - 1$. The nodes to the right of $0u$ are $1u$ and then nodes which are all successor nodes of one to the right of u. Since $1u$ is $d(u)u$, by assumption we have its b-distance m. Let $r_{n+1} = \max(m, r_n - 1)$.



C.r. premises of BoundL with their types:

$\forall_{u \in T_{\mathbb{L}(\mathbb{B})}} \forall_{x \in {}^{\mathrm{co}}I} \exists_{n \in T_{\mathbb{N}}} (\mathrm{Rev}(\bar{x}(n))u \in b))$	of type $\mathbb{L}(\mathbb{B}) \to \mathbb{S}(\mathbb{B}) \to \mathbb{N}$
$d \in T_{\mathbb{L}(\mathbb{B}) \to \mathbb{B}}$	of type $\mathbb{L}(\mathbb{B}) \to \mathbb{B}$
$n \in T_{\mathbb{N}}$	of type $\mathbb{N}$

Conclusion of BoundL with its type:

$$\exists_{m \in T_{\mathbb{N}}} \forall_{w \in T_{\mathbb{L}(\mathbb{B})}} (w < U_d(n) \to Dbmw) \quad \text{of type } \mathbb{N}$$

Extracted term for `BoundL`

```
[hit,d,n](Rec nat=>nat)n 0
 ([n0,n1][case (d(U d n0))
  (True -> Pred n1 max hit(True::U d n0)cCoIConstFalse)
  (False -> Pred n1)])
```

and for `BoundR`

```
[hit,d,n](Rec nat=>nat)n 0
 ([n0,n1][case (d(U d n0))
  (True -> Pred n1)
  (False -> Pred n1 max hit(False::U d n0)cCoIConstTrue)])
```

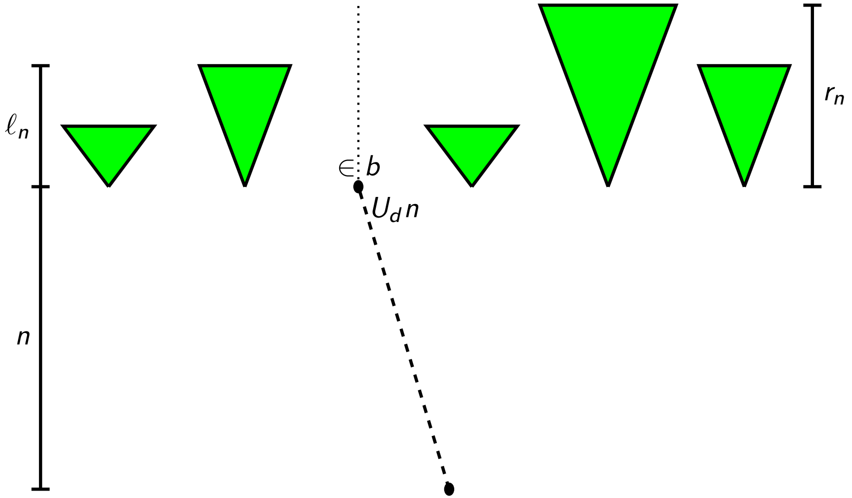
with `hit` of type $\mathbb{L}(\mathbb{B}) \to \mathbb{S}(\mathbb{B}) \to \mathbb{N}$.

Theorem

Let b be a uniformly coconvex bar with modulus d. Then b is a uniform bar, i.e.,

$$\exists_{m \in T_{\mathbb{N}}} \forall_{u \in T_{\mathbb{L}(\mathbb{B})}} (|u| = m \to u \in b).$$

Let A_d be the escape path. Since b is a bar, the escape path A_d hits b at some length n. Use lemma Bounds: the uniform bound is $n + \max(\ell_n, r_n)$



Extracted term

```
[hit,d]
 cBoundL hit d(hit Nil(cCoIA d))max
 cBoundR hit d(hit Nil(cCoIA d))+
 hit Nil(cCoIA d)
```

with `hit` of type $\mathbb{L}(\mathbb{B}) \to \mathbb{S}(\mathbb{B}) \to \mathbb{N}$.

Overview

- Initial cases of transfinite induction in arithmetic
- Partial continuous functionals
- Paths in trees
- Exact real arithmetic
- Linear two-sorted arithmetic

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. Here

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

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

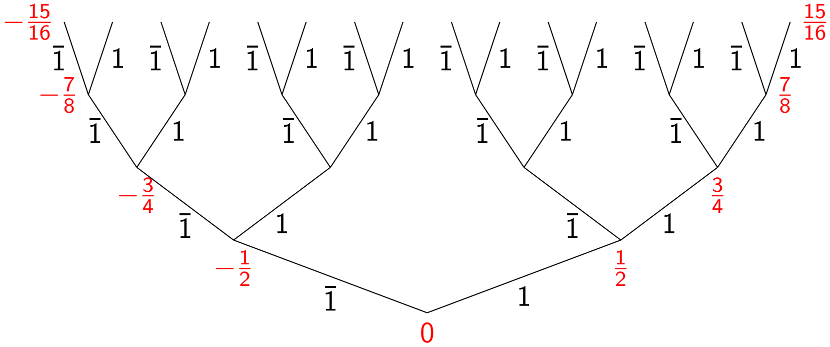
	c_n	$L(p)$
$x + y$	$a_n + b_n$	$\max\big(M(p+1), N(p+1)\big)$
$-x$	$-a_n$	$M(p)$
$\lvert x\rvert$	$\lvert a_n\rvert$	$M(p)$
$x \cdot y$	$a_n \cdot b_n$	$\max\big(M(p+1+p_y), N(p+1+p_x)\big)$
$\frac{1}{x}$ for $\lvert x\rvert \in_q \mathbb{R}^+$	$\begin{cases} \frac{1}{a_n} & \text{if } a_n \neq 0 \\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

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:

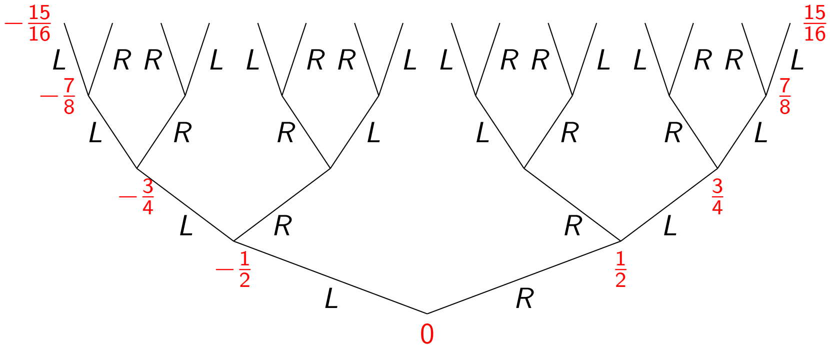
$$\sum_{i<k} \frac{a_i}{2^{i+1}} \qquad \text{with } a_i \in \{-1, 1\}$$



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:

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

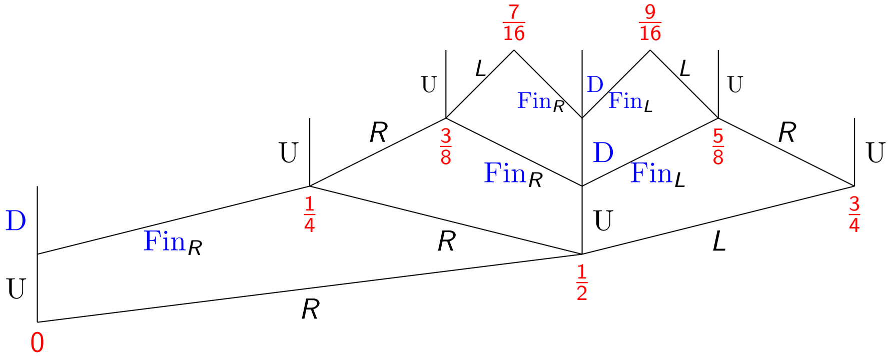
What is the first digit? Cure: delay.

- 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). Pre-Gray code.

Pre-Gray code



After computation in pre-Gray code, one can remove Fin_a by

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

$$\mathrm{RRRLLL}\dots \qquad \mathrm{RLRLLL}\dots \qquad \mathrm{RUDDDD}\dots$$

all denote $\frac{1}{2}$. Only keep the latter to denote $\frac{1}{2}$. Then, generally,

- U occurs in a context $\mathrm{UDDDD}\dots$ only, and
- U appears iff we have a dyadic rational.

Result: unique representation, called pure Gray code.

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

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

Then

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

satisfy the (strengthened) axioms

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

("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

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

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 $f \colon \mathbb{D}_2 \times \mathbb{S}(\mathbb{D}) \times \mathbb{S}(\mathbb{D}) \to \mathbb{S}(\mathbb{D})$ 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 $(d,k), (e,l) \in \mathrm{D}^{\mathsf{r}}$).

```
[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 pre-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 pre-Gray code.

Method also works for multiplication and division:

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

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

Overview

- Initial cases of transfinite induction in arithmetic
- Partial continuous functionals
- Paths in trees
- Exact real arithmetic
- Linear two-sorted arithmetic

Feasible computation with higher types

Gödel's T (1958): finitely typed λ-terms with structural recursion.

$\mathrm{LT}(;)$ (linear two-sorted λ-terms) restricts T s.t. that the definable functions are the polynomial time (ptime) computable ones. $\mathrm{LT}(;)$ generalizes Bellantoni & Cook (1992) to finite types.

$\mathrm{LA}(;)$ solves

$$\frac{\text{Heyting Arithmetic}}{\text{Gödel's }\mathrm{T}} = \frac{?}{\mathrm{LT}(;)}$$

Its provably recursive functions are the ptime computable ones.

Problem: how to cover ptime algorithms (not only functions), e.g. divide-and-conquer ones (quicksort, treesort): they are not linear.

Sources of exponential complexity. (i) Two recursions

We define a function D doubling a natural number and – using D – a function $E(n)$ representing 2^n:

$$\begin{aligned} D(0) &:= 0, & E(0) &:= 1, \\ D(S(n)) &:= S(S(D(n))), & E(S(n)) &:= D(E(n)). \end{aligned}$$

Problem: previous value $E(n)$ taken as recursion argument for D.
Cure: mark argument positions in arrow types as input or output.
Recursion arguments are always input positions.

(ii) Double use of higher type values

Define F as the 2^n-th iterate of D:

$$\begin{aligned} F(0,m) &:= D(m), \\ F(S(n),m) &:= F(n,F(n,m)) \end{aligned} \qquad \text{or} \qquad \begin{aligned} F(0) &:= D, \\ F(S(n)) &:= F(n)\circ F(n). \end{aligned}$$

Problem: in the recursion equation previous value is used twice.
Cure: linearity restriction. No double use of higher type output.

(iii) Marked value types

Define $I(n, f)$ as the n-th iterate f^n of f. Thus $I(n, D)(m) = 2^n m$.

$$\begin{aligned} I(0, f, m) &:= m, \\ I(S(n), f, m) &:= f(I(n, f, m)) \end{aligned} \qquad \text{or} \qquad \begin{aligned} I(0, f) &:= \mathrm{id}, \\ I(S(n), f) &:= f \circ I(n, f). \end{aligned}$$

Problem: since $D \colon \mathbb{N} \hookrightarrow \mathbb{N}$, I's value type is $(\mathbb{N} \hookrightarrow \mathbb{N}) \to \mathbb{N} \hookrightarrow \mathbb{N}$.
Cure: only allow "safe" types as value types of a recursion (no marked argument positions).

(I will be admitted is our setting. This is not the case in Cook and Kapron's PV^ω, since PV^ω is closed under substitution.)

Linear two-sorted terms

Types with input arrow $\hookrightarrow$ and output arrow $\to$:

$$\rho, \sigma ::= \iota \mid \rho \hookrightarrow \sigma \mid \rho \to \sigma \quad \text{with } \iota \text{ base type } (\mathbb{B}, \mathbb{N}, \rho \times \sigma, \mathbb{L}(\rho)).$$

ρ is safe if it does not involve the input arrow $\hookrightarrow$.
Input variables $\bar{x}^\rho$ and output variables x^ρ (typed).
Constants are (i) constructors, (ii) recursion operators

$$\left.\begin{aligned} \mathcal{R}^\tau_{\mathbb{N}} &\colon \mathbb{N} \hookrightarrow \tau \to (\mathbb{N} \hookrightarrow \tau \to \tau) \hookrightarrow \tau \\ \mathcal{R}^\tau_{\mathbb{L}(\rho)} &\colon \mathbb{L}(\rho) \hookrightarrow \tau \to (\rho \hookrightarrow \mathbb{L}(\rho) \hookrightarrow \tau \to \tau) \hookrightarrow \tau \end{aligned}\right. \quad (\tau \text{ safe}),$$

and (iii) cases operators (τ safe)

$$\begin{aligned} \mathcal{C}^\tau_{\mathbb{N}} &\colon \mathbb{N} \to \tau \to (\mathbb{N} \hookrightarrow \tau) \to \tau, \\ \mathcal{C}^\tau_{\mathbb{L}(\rho)} &\colon \mathbb{L}(\rho) \to \tau \to (\rho \hookrightarrow \mathbb{L}(\rho) \hookrightarrow \tau) \to \tau, \\ \mathcal{C}^\tau_{\rho\times\sigma} &\colon \rho \times \sigma \to (\rho \hookrightarrow \sigma \hookrightarrow \tau) \to \tau. \end{aligned}$$

$\mathrm{LT}(;)$-terms built from variables and constants by introduction and elimination rules for the two type forms $\rho \hookrightarrow \sigma$ and $\rho \to \sigma$:

$$\begin{array}{ll}
\bar{x}^\rho \mid x^\rho \mid C^\rho \text{ (constant)} \mid & \\
(\lambda_{\bar{x}^\rho} r^\sigma)^{\rho \hookrightarrow \sigma} \mid (r^{\rho \hookrightarrow \sigma} s^\rho)^\sigma & (s \text{ an input term}) \mid \\
(\lambda_{x^\rho} r^\sigma)^{\rho \to \sigma} \mid (r^{\rho \to \sigma} s^\rho)^\sigma & (\text{higher type output vars in } r, s \text{ distinct,} \\
 & \quad r \text{ does not start with } \mathcal{C}_\iota^\tau) \mid \\
\mathcal{C}_\iota^\tau t \vec{r} & (\text{h.t. output vars in } \mathrm{FV}(t) \text{ not in } \vec{r}\,)
\end{array}$$

with as many r_i as there are constructors of ι. s is an input term if

- all its free variables are input variables, or else
- s is of higher type and all its higher type free variables are input variables.

The parse dag computation model

Represent terms as directed acyclic graphs (dag), where only nodes for terms of base type can have in-degree > 1. Nodes can be

- terminal nodes labelled by a variable or constant,
- abstraction nodes with 1 successor, labelled with an (input or output) variable and a pointer to the successor node, or
- application nodes with 2 successors, labelled with 2 pointers.

A parse dag is a parse tree for a term.

The treesort algorithm

$$
\begin{aligned}
\mathrm{TreeSort}(l) &= \mathrm{Flatten}(\mathrm{MakeTree}(l)),\\
\mathrm{MakeTree}([]) &= \diamond,\\
\mathrm{MakeTree}(a :: l) &= \mathrm{Insert}(a, \mathrm{MakeTree}(l)),\\
\mathrm{Insert}(a, \diamond) &= C_a(\diamond, \diamond),\\
\mathrm{Insert}(a, C_b(u, v)) &= \begin{cases} C_b(\mathrm{Insert}(a, u), v) & \text{if } a \leq b\\ C_b(u, \mathrm{Insert}(a, v)) & \text{if } b < a,\end{cases}\\
\mathrm{Flatten}(\diamond) &= [],\\
\mathrm{Flatten}(C_b(u, v)) &= \mathrm{Flatten}(u) * (b :: \mathrm{Flatten}(v)).
\end{aligned}
$$

Problem: two recursive calls in $\mathrm{Flatten}$, not allowed in $\mathrm{LT}(;)$.
Cure: analysis of $\mathrm{Flatten}$ in the parse dag computation model.

We estimate the number $\#t$ of steps it takes to reduce a term t to its normal form $\mathrm{nf}(t)$.

Lemma. Let l be a numeral of type $\mathbb{L}(\mathbb{N})$. Then $\#(l * l') = O(|l|)$.

For $\#\mathrm{Flatten}(u)$ use this size function for numerals u of type $\mathbb{T}$:

$$\begin{aligned} \|\diamond\| &:= 0, \\ \|C_a(u, v)\| &:= 2\|u\| + \|v\| + 3. \end{aligned}$$

Lemma. Let u be a numeral of type $\mathbb{T}$. Then

$$\#\mathrm{Flatten}(u) = O(\|u\|).$$

Goal: all functions definable in $\mathrm{LT}(;) + \mathrm{Flatten}$ are polytime computable. Call a term

- $\mathcal{RD}$-free: no recursion constant $\mathcal{R}$, no $\mathrm{Flatten}$.
- simple: no higher type input variables.

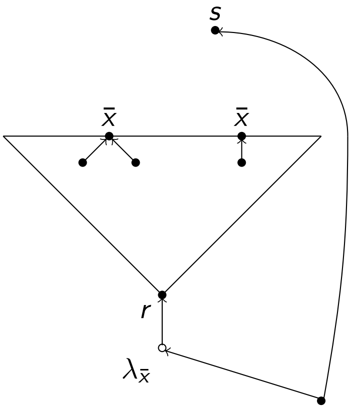
Lemma (Sharing normalization)

Let t be an $\mathcal{RD}$-free simple term. Then a parse dag for $\mathrm{nf}(t)$, of size at most $\|t\|$, can be computed from t in time $O(\|t\|^2)$.

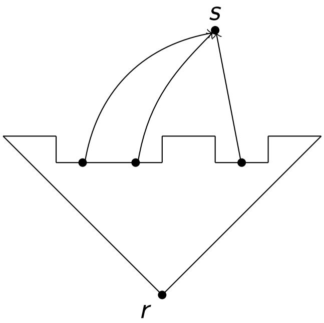
Corollary (Base normalization)

Let t be a closed $\mathcal{RD}$-free simple term of type $\mathbb{N}$ or $\mathbb{L}(\mathbb{N})$. Then $\mathrm{nf}(t)$ can be computed from t in time $O(\|t\|^2)$, and $\|\mathrm{nf}(t)\| \leq \|t\|$.

$(\lambda_{\bar{x}} r(\bar{x}))s$ with $\bar{x}$ of base type



$\mapsto$



Lemma ($\mathcal{RD}$-elimination)

Let $t(\vec{x})$ be a simple term of safe type. There is a polynomial P_t such that: if $\vec{r}$ are safe type $\mathcal{RD}$-free closed simple terms and the free variables of $t(\vec{r})$ are output variables, then in time $P_t(\|\vec{r}\|)$ one can compute an $\mathcal{RD}$-free simple term $\mathrm{rdf}(t;\vec{x};\vec{r})$ such that $t(\vec{r}) \to^* \mathrm{rdf}(t;\vec{x};\vec{r})$.

Proof.

By induction on $\|t\|$ (cf. Chapter 8 of H.S. & S.Wainer, Proofs and Computations, 2012). Need an additional case for $\mathrm{Flatten}$, and $\#\mathrm{Flatten}(u) = O(\|u\|)$. □

Theorem (Normalization)

Let $t\colon \mathbb{N} \twoheadrightarrow \dots \mathbb{N} \twoheadrightarrow \mathbb{N}$ (with $\twoheadrightarrow \in \{\hookrightarrow, \to\}$) be a closed term in $\mathrm{LT}(;) + \mathrm{Flatten}$. Then t denotes a polytime function.

Conclusion

- $\mathrm{LA}(;) \sim \mathrm{LT}(;)$ via Curry-Howard correspondence.
- $$\frac{\text{Heyting Arithmetic}}{\text{Gödel's T}} = \frac{\mathrm{LA}(;)}{\mathrm{LT}(;)} = \frac{\mathrm{LA}(;) + \mathrm{Flatten}}{\mathrm{LT}(;) + \mathrm{Flatten}}$$
- $\mathrm{LA}(;) + \mathrm{Flatten} \vdash \forall_{l,\bar{n}}(|l| \leq \bar{n} \to \exists_u S(l, u))$
- Computational content of this proof: $(\mathrm{LT}(;) + \mathrm{Flatten})$-term. Can be extracted by realizability. $\sim$ treesort algorithm.