Computing with partial continuous functionals

Helmut Schwichtenberg

Mathematisches Institut, LMU, München

University of Canterbury, Christchurch, 10 Feb 2016

Proof: 2 aspects

- provides insight (uniformity)
- may have computational content

For proofs we need: logic + data + inductive definitions

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

(a). The model of partial continuous functionals

- Information systems
- Algebras and types
- A common extension T^+ of Gödel's T and Plotkin's PCF
- Denotational semantics

(b). A theory of computable functionals

- Formulas and predicates
- A theory of computable functionals
- Brouwer - Heyting - Kolmogorov and decorations
- The type of a formula or predicate
- Realizability
- Extracted terms

(c). Extracting programs from proofs

- Computing with infinite data: Gray-coded reals
- Ishihara's trick

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 systems
- Algebras and types
- A common extension T^{+} of Gödel's T and Plotkin's PCF
- Denotational semantics

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

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

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

Characterizing ideals in $\mathbf{A} \to \mathbf{B}$

$\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ information systems.

Definition ($r \subseteq \mathrm{Con}_A \times B$ approximable map)

- If $r(U, b_1), \dots, r(U, b_n)$, then $\{b_1, \dots, b_n\} \in \mathrm{Con}_B$.
- If $r(U, b_1), \dots, r(U, b_n)$ and $\{b_1, \dots, b_n\} \vdash_B b$, then $r(U, b)$.
- If $r(U', b)$ and $U \vdash_A U'$, then $r(U, b)$.

Theorem

The ideals in $\mathbf{A} \to \mathbf{B}$ are the approximable maps from $\mathbf{A}$ to $\mathbf{B}$.

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

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

- Information systems
- Algebras and types
- A common extension T^+ of Gödel's T and Plotkin's PCF
- Denotational semantics

Concrete information systems, from free algebras.

- Types will be built from base types ι by $\rho \to \sigma$.
- Information systems for base types are built from non-flat free algebras, given by their constructors (reason: want constructors to be injective and with disjoint ranges).

Inductively define type forms:

$$\rho, \sigma ::= \alpha \mid \rho \to \sigma \mid \mu_\xi((\rho_{i\nu})_{\nu<n_i} \to \xi)_{i<k}$$

with α, ξ type variables and $k \geq 1$ (since we want our algebras to be inhabited). $(\rho_\nu)_{\nu<n} \to \sigma$ means $\rho_0 \to \ldots \to \rho_{n-1} \to \sigma$.

Strict positivity

We define α occurs at most strictly positive in ρ, by induction on ρ.

$$\mathrm{SP}(\alpha, \beta) \qquad \frac{\alpha \notin \mathrm{FV}(\rho) \quad \mathrm{SP}(\alpha, \sigma)}{\mathrm{SP}(\alpha, \rho \to \sigma)}$$

$$\frac{\mathrm{SP}(\alpha, \rho_{i\nu}) \text{ for all } i < k,\ \nu < n_i}{\mathrm{SP}(\alpha, \mu_\xi((\rho_{i\nu})_{\nu < n_i} \to \xi)_{i<k})}$$

We define $\mathrm{Ty}(\rho)$ "ρ is a type", again by induction on ρ.

$$\mathrm{Ty}(\alpha) \qquad \frac{\mathrm{Ty}(\rho) \quad \mathrm{Ty}(\sigma)}{\mathrm{Ty}(\rho \to \sigma)}$$

$$\frac{\mathrm{Ty}(\rho_{i\nu}) \text{ and } \mathrm{SP}(\xi, \rho_{i\nu}) \text{ for all } i < k,\ \nu < n_i}{\mathrm{Ty}(\mu_\xi((\rho_{i\nu})_{\nu < n_i} \to \xi)_{i<k})}$$

We call

$$\iota := \mu_\xi((\rho_{i\nu})_{\nu<n_i} \to \xi)_{i<k}$$

an algebra.

Let $(\rho_\nu(\xi))_{\nu<n} \to \xi$ be the i-th component of ι. Call

$$(\rho_\nu(\iota))_{\nu<n} \to \iota$$

the i-th constructor type of ι.

Examples of algebras:

$$\begin{aligned}
&\mathbf{U} := \mu_\xi \xi \qquad \text{(unit)},\\
&\mathbf{B} := \mu_\xi(\xi, \xi) \qquad \text{(booleans)},\\
&\mathbf{N} := \mu_\xi(\xi, \xi \to \xi) \qquad \text{(natural numbers, unary)},\\
&\mathbf{D} := \mu_\xi(\xi, \xi \to \xi \to \xi) \qquad \text{(binary trees, or derivations)},
\end{aligned}$$

Examples of algebras strictly positive in their type parameters:

$$\begin{aligned}
\mathbf{L}(\alpha) &:= \mu_\xi(\xi, \alpha \to \xi \to \xi) \qquad \text{(lists)},\\
\alpha \times \beta &:= \mu_\xi(\alpha \to \beta \to \xi) \qquad \text{(product)},\\
\alpha + \beta &:= \mu_\xi(\alpha \to \xi, \beta \to \xi) \qquad \text{(sum)}.
\end{aligned}$$

Example of a nested algebra:

$$\mathbf{T} := \mu_\xi(\mathbf{L}(\xi) \to \xi) \quad \text{(finitely branching trees)}.$$

Note that **T** has a total inhabitant since $\mathbf{L}(\alpha)$ has one (Nil).

Standard names for constructors:

$$\mathrm{tt}^{\mathbf{B}}, \mathrm{ff}^{\mathbf{B}}$$
$$0^{\mathbf{N}}, \mathrm{S}^{\mathbf{N}\to\mathbf{N}}$$
$$0^{\mathbf{D}}, C^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}} \quad \text{for the type } \mathbf{D} \text{ of binary trees,}$$
$$\mathrm{Nil}^{\mathbf{L}(\rho)}, \mathrm{Cons}^{\rho\to\mathbf{L}(\rho)\to\mathbf{L}(\rho)} \quad \text{for the type } \mathbf{L}(\rho) \text{ of lists,}$$
$$(\mathrm{Inl}_{\rho\sigma})^{\rho\to\rho+\sigma}, (\mathrm{Inr}_{\rho\sigma})^{\sigma\to\rho+\sigma} \quad \text{for the sum type } \rho+\sigma,$$
$$\mathrm{Branch}\colon \mathbf{L}(\mathbf{T}) \to \mathbf{T} \quad \text{for the type } \mathbf{T} \text{ of finitely branching trees.}$$

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.

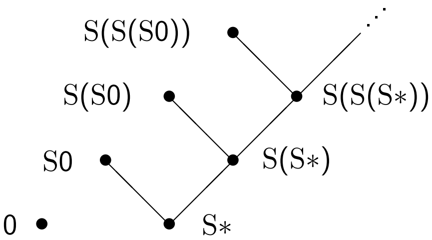
(Example: $\{C{*}0, C0{*}\}$.)

$U \vdash a$ (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 **N**



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

Why non-flat domains?

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

Lemma

(a) $r_{\mathrm{C}}(\vec{x}\,) \subseteq r_{\mathrm{C}}(\vec{y}\,) \leftrightarrow \vec{x} \subseteq \vec{y}$. *Hence* r_{C} *is injective.*

(b) $r_{\mathrm{C}_1}(\vec{x}\,) \neq r_{\mathrm{C}_2}(\vec{y}\,)$, *since the two ideals are non-empty and disjoint. Hence distinct constructors have disjoint ranges.*

Neither property holds for flat information systems, since for them, by monotonicity, constructors are strict (i.e., if one argument is the empty ideal, then the value is as well). But then

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

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").

Note.

- The set of all ideals of $\mathbf{A}$ is denoted by $|\mathbf{A}|$.
- Define $\mathcal{O}_U \subseteq |\mathbf{A}|$ by $\mathcal{O}_U := \{ x \in |\mathbf{A}| \mid U \subseteq x \}$.
- The system of all $\mathcal{O}_U$ with $U \in \mathrm{Con}$ forms the basis of a topology on $|\mathbf{A}|$, called the Scott topology.

Definition (Totality)

- x^ι is total if it is generated from a total token (no $*$'s).
- $f^{\rho\to\sigma}$ is total if it maps total arguments to total values.

Definition (Cototality)

- x^ι is cototal if every token (i.e., constructor tree) $P(*) \in x$ has a "one-step extension" $P(\mathrm{C}\vec{*}) \in x$.
- $f^{\rho\to\sigma}$ is cototal if it maps cototal arguments to cototal values.

Similar: finite or infinite "locally correct" derivations [Mints 78].

- Information systems
- Algebras and types
- A common extension T^{+} of Gödel's T and Plotkin's PCF
- Denotational semantics

Computable functionals

Recall: a partial continuous functional f^ρ is computable if it is a (primitive) recursively enumerable set of tokens.

How to define computable functionals? By computation rules

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

with free variables of $\vec{P}_i(\vec{y}_i)$ and M_i among $\vec{y}_i$, where $\vec{P}_i(\vec{y}_i)$ are "constructor patterns".

Structural recursion operators

Important example for such D [Hilbert 1925, Gödel 1958]. The type of the recursion operator $\mathcal{R}_\iota^\tau$ for $\iota = \mu_\xi((\rho_{i\nu}(\xi))_{\nu<n_i} \to \xi)_{i<k}$ with result type τ is

$$\iota \to ((\rho_{i\nu}(\iota \times \tau))_{\nu<n_i} \to \tau)_{i<k} \to \tau.$$

- ι is the type of the recursion argument.
- Each $(\rho_{i\nu}(\iota \times \tau))_{\nu<n_i} \to \tau$ is called a step type.
- Usage of $\iota \times \tau$ (not τ) in the step types is a strengthening: more data are available to construct the value of type τ.
- We avoid the product type in $\vec{\sigma} \to \iota \times \tau$ and take the two argument types $\vec{\sigma} \to \iota$ and $\vec{\sigma} \to \tau$ instead.

Examples

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

Map operators

Let $\rho(\vec{\alpha})$ be a type and $\vec{\alpha}$ strictly positive type parameters. We define the map operator

$$\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\rho(\vec{\alpha})} \colon \rho(\vec{\sigma}) \to (\vec{\sigma} \to \vec{\tau}) \to \rho(\vec{\tau})$$

where $(\vec{\sigma} \to \vec{\tau}) \to \rho := (\sigma_1 \to \tau_1) \to \ldots \to (\sigma_n \to \tau_n) \to \rho$.

- If none of $\vec{\alpha}$ appears free in $\rho(\vec{\alpha})$ let

$$\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\rho(\vec{\alpha})} x\vec{f} = x.$$

- Otherwise we use an outer recursion on $\rho(\vec{\alpha})$ and if $\rho(\vec{\alpha})$ is $\iota(\vec{\alpha})$ an inner one on x.
- If $\rho(\vec{\alpha})$ is $\iota(\vec{\alpha})$ abbreviate $\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\iota(\vec{\alpha})}$ by $\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\iota}$ or $\mathcal{M}^{\vec{\tau}}_{\iota(\vec{\sigma})}$.

Immediate cases for the outer recursion:

$$\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\alpha_i} x\vec{f} = f_i x, \qquad \mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}(\sigma\to\rho)} h\vec{f}x = \mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\rho}(hx)\vec{f}.$$

It remains to consider $\iota(\vec{\pi}(\vec{\alpha}\,))$.

- In case $\vec{\pi}(\vec{\alpha}\,)$ is not $\vec{\alpha}$ let
$$\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\iota(\vec{\pi}(\vec{\alpha}\,))} x\vec{f} = \mathcal{M}^{\vec{\pi}(\vec{\sigma}\,)\to\vec{\pi}(\vec{\tau}\,)}_{\iota} x(\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\pi_i(\vec{\alpha}\,)} \cdot \vec{f}\,)_{i<|\vec{\pi}\,|}$$
with $\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\pi_i(\vec{\alpha}\,)} \cdot \vec{f} = \lambda_x \mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\pi_i(\vec{\alpha}\,)} x\vec{f}$.
- In case $\vec{\pi}(\vec{\alpha}\,)$ is $\vec{\alpha}$ we use recursion on x and define for a constructor $\mathrm{C}_i \colon (\rho_\nu(\vec{\sigma}, \iota(\vec{\sigma}\,)))_{\nu<n} \to \iota(\vec{\sigma}\,)$
$$\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\iota}(\mathrm{C}_i\vec{x}\,)\vec{f}$$
to be the result of applying C'_i of type $(\rho_\nu(\vec{\tau}, \iota(\vec{\tau}\,)))_{\nu<n} \to \iota(\vec{\tau}\,)$ (the same constructor as C_i with only the type changed) to, for each $\nu < n$,
$$\mathcal{M}^{\vec{\sigma},\iota(\vec{\sigma}\,)\to\vec{\tau},\iota(\vec{\tau}\,)}_{\lambda_{\vec{\alpha},\beta}\rho_\nu(\vec{\alpha},\beta)} x_\nu \vec{f}(\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\iota} \cdot \vec{f}\,).$$

The final function argument provides the recursive call w.r.t. the recursion on x.

Example: $\mathcal{M}^{\tau}_{\mathbf{L}(\sigma)} \colon \mathbf{L}(\sigma) \to (\sigma \to \tau) \to \mathbf{L}(\tau)$ is defined by

$$\mathcal{M}^{\tau}_{\mathbf{L}(\sigma)} \, \mathrm{Nil} \, f^{\sigma \to \tau} = \mathrm{Nil},$$
$$\mathcal{M}^{\tau}_{\mathbf{L}(\sigma)} (x^{\sigma} :: l^{\mathbf{L}(\sigma)}) f^{\sigma \to \tau} = (fx) :: (\mathcal{M} \, l \, f).$$

Definition

Terms of Gödel's T (for nested algebras) are generated from typed variables x^{ρ} and constants for

- constructors C^{ι}_{i},
- recursion operators $\mathcal{R}^{\tau}_{\iota}$ and
- map operators $\mathcal{M}^{\vec{\rho} \to \vec{\tau}}_{\lambda_{\vec{\alpha}} \pi}$

by abstraction $\lambda_{x^{\rho}} M^{\sigma}$ and application $M^{\rho \to \sigma} N^{\rho}$.

Computation rules for $\mathcal{R}_\iota^\tau$:

$$\mathcal{R}_\iota^\tau(\mathrm{C}_i^\iota \vec{N})\vec{M} = M_i(\mathcal{M}_{\lambda_\alpha \rho_\nu(\alpha)}^{\iota \to \iota \times \tau} N_\nu \lambda_x \langle x^\iota, \mathcal{R}_\iota^\tau x \vec{M} \rangle)_{\nu < n}$$

where $(\rho_\nu(\iota))_{\nu<n} \to \iota$ is the type of the i-th constructor C_i.

In the special case $\rho_\nu(\alpha) = \alpha$ we can avoid the product type and instead of the pair

$$\mathcal{M}_{\lambda_\alpha \alpha}^{\iota \to \iota \times \tau} N_\nu \lambda_x \langle x^\iota, \mathcal{R}_\iota^\tau x \vec{M} \rangle \quad \text{i.e.,} \quad \langle N_\nu^\iota, \mathcal{R}_\iota^\tau N_\nu \vec{M} \rangle$$

take its components N_ν^ι and $\mathcal{R}_\iota^\tau N_\nu \vec{M}$ as separate arguments of M_i.

Examples

- $\mathcal{R}_{\mathbf{N}}^{\tau}\colon \mathbf{N} \to \tau \to (\mathbf{N} \to \tau \to \tau) \to \tau$ defined by

$$\begin{aligned} \mathcal{R}_{\mathbf{N}}^{\tau} 0xf &= x, \\ \mathcal{R}_{\mathbf{N}}^{\tau} (\mathrm{S}n)xf &= fx(\mathcal{R}_{\mathbf{N}}^{\tau} nxf). \end{aligned}$$

- $\mathcal{R}_{\mathbf{T}}^{\tau}\colon \mathbf{T} \to (\mathbf{L}(\mathbf{T} \times \tau) \to \tau) \to \tau$ defined by

$$\mathcal{R}_{\mathbf{T}}^{\tau}(\mathrm{Branch}\, as) f^{\mathbf{L}(\mathbf{T}\times\tau)\to\tau} = f(\mathcal{M}_{\mathbf{L}(\mathbf{T})}^{\mathbf{T}\times\tau} as \lambda_a \langle a^{\mathbf{T}}, \mathcal{R}_{\mathbf{T}}^{\tau} af \rangle).$$

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

- $+ \colon \mathbf{N} \to \mathbf{N} \to \mathbf{N}$ 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

$$Yf = f(Yf)$$

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

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

Corecursion

The rules for $\mathcal{R}$ work from the leaves towards the root, and terminate because total ideals are well-founded.

For cototal ideals a similar operator defines functions with cototal ideals as values: corecursion. Consider $\iota = \mu_\xi(\kappa_0, \ldots, \kappa_{k-1})$.

constructor type:

$$\sum_{i<k} \prod_{\nu<n_i} \rho_{i\nu}(\iota) \to \iota$$

destructor type:

$$\iota \to \sum_{i<k} \prod_{\nu<n_i} \rho_{i\nu}(\iota)$$

type of recursion operator:

$$\iota \to (\sum_{i<k} \prod_{\nu<n_i} \rho_{i\nu}(\iota \times \tau) \to \tau) \to \tau$$

type of corecursion operator:

$$\tau \to (\tau \to \sum_{i<k} \prod_{\nu<n_i} \rho_{i\nu}(\iota + \tau)) \to \iota$$

Examples

$$
\begin{aligned}
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{B}} &\colon \tau \to (\tau \to \mathbf{U} + \mathbf{U}) \to \mathbf{B},\\
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{N}} &\colon \tau \to (\tau \to \mathbf{U} + (\mathbf{N} + \tau)) \to \mathbf{N},\\
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{D}} &\colon \tau \to (\tau \to \mathbf{U} + (\mathbf{D} + \tau) \times (\mathbf{D} + \tau)) \to \mathbf{D},\\
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{L}(\rho)} &\colon \tau \to (\tau \to \mathbf{U} + \rho \times (\mathbf{L}(\rho) + \tau)) \to \mathbf{L}(\rho).
\end{aligned}
$$

For $f \colon \rho \to \tau$, $g \colon \sigma \to \tau$ define $[f, g]^{\rho+\sigma\to\tau} := \lambda_x(\mathcal{R}^{\tau}_{\rho+\sigma} xfg)$. Let x_1, x_2 denote the two projections of x of type $\rho \times \sigma$.

$$
\begin{aligned}
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{B}} NM &= [\lambda\_\mathrm{tt}, \lambda\_\mathrm{ff}](MN),\\
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{N}} NM &= [\lambda\_0, \lambda_x(\mathrm{S}([\mathrm{id}^{\mathbf{N}\to\mathbf{N}}, \lambda_y({}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{N}} yM)]x))](MN),\\
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{D}} NM &= [\lambda\_0, \lambda_x(\mathrm{C}([\mathrm{id}, P_{\mathbf{D}}]x_1)([\mathrm{id}, P_{\mathbf{D}}]x_2))](MN),\\
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{L}(\rho)} NM &= [\lambda\_\mathrm{Nil}, \lambda_x(x_1 :: [\mathrm{id}, \lambda_y({}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{L}(\rho)} yM)]x_2)](MN).
\end{aligned}
$$

- Information systems
- Algebras and types
- A common extension T^+ of Gödel's T and Plotkin's PCF
- Denotational semantics

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 standard β, η-conversion and the computation rules.
- An adequacy theorem (Plotkin) 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.