Computable functionals over non-flat data types

Helmut Schwichtenberg
(j.w.w. Basil Karadais, Simon Huber)

Mathematisches Institut, LMU, München

Kyoto University, Japan, 24. April 2009

Preview: logic for inductive definitions LID

- Typed language, with the partial continuous functionals as intended domains (cf. Peano arithmetic and $\mathbb{N}$).
- Terms are those of T^+.
- Natural deduction rules for $\to$ and $\forall$ ("minimal logic").
- All predicates are defined inductively.
  - (Leibniz) equality
  - Totality
  - $\exists$, $\wedge$, $\vee$.

Minlog

- Proof assistant for LID.
- Difference from Coq, Isabelle, Agda etc.: underlying theory of partial continuous functionals.
- Program extraction from (constructive and classical) proofs.
- `www.minlog-system.de`

Denotational semantics via continuous functionals

- Want: "meaning" of (typed) expressions or terms (in a functional programming language).
- What are the semantical objects /functionals?
- Numbers $0, 1, 2, \ldots$.
- Functions, like $n \mapsto 2n$.
- Functionals, like $f \mapsto f \circ f$ or $f \mapsto f0$.

Computations are to be finite. Hence: function arguments can only be called finitely many times (principle of finite support, PFS). Therefore computable functionals are continuous (Brouwer).

Denotational semantics via continuous functionals (ctd.)

- What are the domains of computable functionals?
- The full set theoretic hierarchy of functionals of finite types is problematic from a constructive point of view.
- Kreisel (1959) ("formal neighborhoods") and Kleene (1959) ("countable functionals") gave a more appropriate treatment.
- This was taken up and developed in a mathematically satisfactory way by Dana Scott and Yuri Ershov (70s), using partial functionals.
- Today usually in the framework of (classical) domain theory (initiated by Scott).

Here: concrete representations of domains, via information systems (Scott 1982).

(Coherent) information systems

$\mathbf{A} = (A, \smile, \vdash)$ such that

- A is a non-empty (countable) set (the "tokens").
- $\smile$ is a reflexive and symmetric relation on A ("consistency").
- $\vdash$ is a relation between Con and A ("entailment"), where

$$\mathrm{Con} := \{\, U \mid U \subseteq^{\mathrm{fin}} A \land \forall_{a,b \in U}(a \smile b) \,\}$$

such that

$$a \in U \to U \vdash a,$$
$$\forall_{b \in V}(U \vdash b) \land V \vdash a \to U \vdash a,$$
$$a \in U \land U \vdash b \to a \smile b.$$

Information systems (ctd.)

U, V, W denote finite sets. $U \vdash V$ means $\forall_{a \in V}(U \vdash a)$.

$$U \vdash V \to U \cup V \in \mathrm{Con},$$
$$U_1 \supseteq U \vdash V \supseteq V_1 \to U_1 \vdash V_1,$$
$$U \vdash V \vdash W \to U \vdash W.$$

The ideals or objects of an information system $\mathbf{A} = (A, \smile, \vdash)$ are subsets x of A which satisfy

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

We write $|\mathbf{A}|$ for the set of ideals of $\mathbf{A}$.

Examples

- Any (countable) set A can be turned into a flat information system. Tokens: all $a \in A$.

$$a \smile b :\leftrightarrow a = b \quad \text{and} \quad U \vdash a :\leftrightarrow a \in U,$$

  Then $\mathrm{Con} = \{\emptyset\} \cup \{\, \{a\} \mid a \in A \,\}$. Objects: elements of Con.

- Approximations of functions from A to B. Tokens: pairs (a, b) with $a \in A$ and $b \in B$.

$$(a_1, b_1) \smile (a_2, b_2) :\leftrightarrow (a_1 = a_2 \to b_1 = b_2),$$
$$U \vdash (a, b) :\leftrightarrow (a, b) \in U.$$

  Objects: (the graphs of) all partial functions from A to B.

Function spaces

Let $\mathbf{A} = (A, \frown_A, \vdash_A)$ and $\mathbf{B} = (B, \frown_B, \vdash_B)$ be information systems. Then $\mathbf{A} \to \mathbf{B} = (C, \frown, \vdash)$ defined by

$$C := \mathrm{Con}_A \times B,$$
$$(U, b) \frown (V, c) :\leftrightarrow (U \frown_A V \to b \frown_B c),$$
$$\{ (U_i, a_i) \mid i \in I \} \vdash (V, b) :\leftrightarrow \{ a_i \mid V \vdash_A U_i \} \vdash_B b.$$

is an information system again.

Approximable maps (Scott 1982)

Given **A** and **B**, call a relation $r \subseteq \mathrm{Con}_A \times B$ an approximable map from **A** to **B** (written $r\colon \mathbf{A} \to \mathbf{B}$) iff

- if $r(U, b_1)$ and $r(U, b_2)$, then $b_1 \frown_B b_2$, and
- if $r(U, b)$, $V \vdash_A U$ and $b \vdash_B c$, then $r(V, c)$.

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

Scott topology

The set $|\mathbf{A}|$ of ideals for $\mathbf{A}$ carries a natural topology. Basis: cones $\tilde{U} := \{ z \mid z \supseteq U \}$ generated by the formal neighborhoods U. The continuous maps $f : |\mathbf{A}| \to |\mathbf{B}|$ and the ideals $r \in |\mathbf{A} \to \mathbf{B}|$ are in a bijective correspondence:

$$|r|(x) := \{ b \in B \mid \exists_{U \subseteq x}((U, b) \in r) \},$$
$$(U, b) \in \hat{f} :\leftrightarrow b \in f(\overline{U}) \quad \text{with } \overline{U} := \{ a \in A \mid U \vdash_A a \}.$$

These assignments are inverse to each other:

$$f = |\hat{f}| \quad \text{and} \quad r = \widehat{|r|}.$$

Examples of data types

$$\begin{aligned}
\mathbf{B} &:= \mu_\xi(\xi, \xi) && \text{(booleans)},\\
\mathbf{N} &:= \mu_\xi(\xi, \xi \to \xi) && \text{(natural numbers, unary)},\\
\mathbf{E} &:= \mu_\xi(\xi, \xi \to \xi \to \xi) && \text{(expressions, i.e., binary trees)},\\
\mathbf{O} &:= \mu_\xi(\xi, \xi \to \xi, (\mathbf{N} \to \xi) \to \xi) && \text{(ordinals)},\\
\mathbf{T}_0 &:= \mathbf{N}, \quad \mathbf{T}_{n+1} := \mu_\xi(\xi, (\mathbf{T}_n \to \xi) \to \xi) && \text{(trees)}.
\end{aligned}$$

Data types with type parameters

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

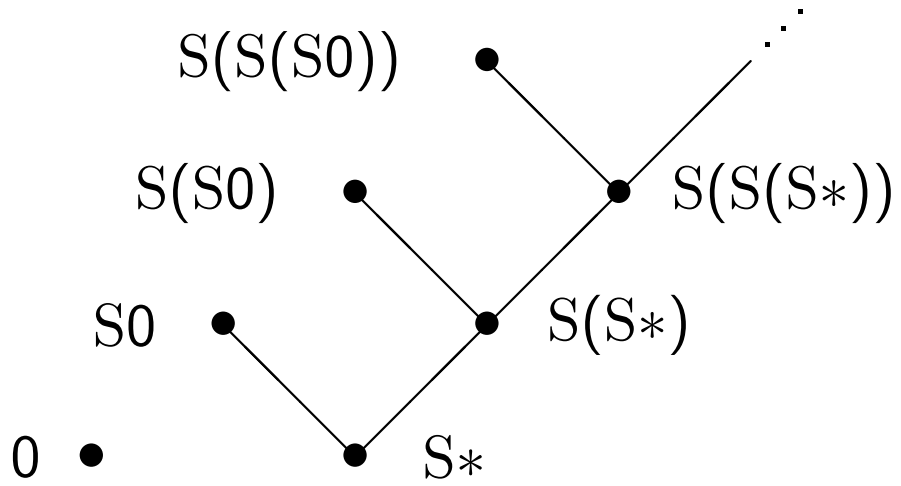
Information system $\mathbf{A}_\iota$ for a data type ι

- An extended token is a token or the symbol $*$ (notation a^*).
- $a^* \smile b^* :\leftrightarrow (a = *) \vee (b = *) \vee (a = b)$.
- $U \cup \{*\} \vdash a :\leftrightarrow U \vdash a$, and $U \vdash *$ is true.

$\mathbf{A}_\iota = (A, \smile, \vdash)$ is defined by $A := \{\, \mathrm{C}\vec{a^*} \mid \mathrm{C} \text{ constructor of } \iota \,\}$,

$$\frac{\vec{a^*} \smile \vec{b^*}}{\mathrm{C}\vec{a^*} \smile \mathrm{C}\vec{b^*}}, \qquad \frac{\{a^*_{11}, \dots, a^*_{1n}\} \vdash b^*_1 \;\; \dots \;\; \{a^*_{n1}, \dots, a^*_{nn}\} \vdash b^*_n}{\{\mathrm{C}\vec{a^*_1}, \dots, \mathrm{C}\vec{a^*_n}\} \vdash \mathrm{C}\vec{b^*}}.$$

Tokens and entailment for **N**



Another example: **E** with constructors 0 and C. Then

$$C0* \smile C*0, \qquad \{C0*, C*0\} \vdash C00.$$

Constructors as continuous functions

- Every constructor C generates an ideal in the function space:

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

- By definition of the continuous function associated to an ideal in a function space, the continuous map $|r_{\mathrm{C}}|$ satisfies

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

- Every non-empty ideal in $\mathbf{A}_\iota$ has the from $|r_{\mathrm{C}}|(\vec{x}\,)$ with a constructor C and ideals $\vec{x}$.

Constructors are injective and have disjoint ranges

- Let C be a constructor of ι. Then

$$|r_{\mathrm{C}}|(\vec{x}\,) = |r_{\mathrm{C}}|(\vec{y}\,) \to \vec{x} = \vec{y}.$$

- If $\mathrm{C}_1, \mathrm{C}_2$ are distinct constructors of ι then

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

These properties are mandatory for a smooth equational reasoning (cf. the Peano axioms $\mathrm{S}n \neq 0$ and $\mathrm{S}n = \mathrm{S}m \to n = m$). — Notice that they would not hold for flat $\mathbf{A}_\iota$'s:

$$|r_{\mathrm{C}}|(\bot, y) = \bot = |r_{\mathrm{C}}|(x, \bot),$$
$$|r_{\mathrm{C}_1}|(\bot) = \bot = |r_{\mathrm{C}_2}|(\bot)$$

(with $\bot := \emptyset$), since for flat $\mathbf{A}_\iota$'s constructors need to be strict.

Partial continuous functionals

For types ρ over base types ι, define the information system $\mathbf{C}_\rho$ inductively by

$$\mathbf{C}_\iota := \mathbf{A}_\iota \text{ defined as above,} \qquad \mathbf{C}_{\rho\to\sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma.$$

The ideals in $\mathbf{C}_\rho$ are called partial continuous functionals of type ρ (Dana Scott, Yuri Ershov).

Computable functionals

- Recall that computable functionals need to be continuous (PFS), and are viewed as limits of finite approximations.
- Hence there is an easy way to define computability: F is computable iff the set of its finite approximations is recursively enumerable.
- However, in practice this would be inconvenient. We want to present computable functionals by expressions (or terms) of a functional programming language.

The typed λ-calculus T^+

Common extension of Gödel's T and Plotkin's PCF.

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

Every defined constant D^ρ comes with a system of computation rules, consisting of equations

$$D\vec{P}_i(\vec{y}_i) = M_i(\vec{y}_i)$$

with constructor patterns $\vec{P}_i$.

Examples of constants D with their computation rules

- $\begin{cases} D(0) := 0, \\ D(\mathrm{S}(n)) := \mathrm{S}(\mathrm{S}(n)). \end{cases}$
- Addition, multiplication.
- Gödel's higher type primitive recursion operators $\mathcal{R}$:

$$\begin{cases} \mathcal{R}(0, r, s) = r, \\ \mathcal{R}(\mathrm{S}n, r, s) = s(n, \mathcal{R}(n, r, s)). \end{cases}$$

- The fixed point operators Y:

$$Yw = w(Yw).$$

Inductive definition of $(\vec{U}, a) \in [\![\lambda_{\vec{x}} M]\!]$

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

For every constructor C and defined constant D

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

with one such rule (D) for every computation rule $D\vec{P}(\vec{y}\,) = M$.

Properties of $[\![M]\!]$

- $[\![M]\!]$ is an ideal, i.e., consistent and deductively closed.
- $[\![MN]\!] = [\![M]\!][\![N]\!]$.
- $[\![M]\!]$ is preserved under reduction, generated from

$(\lambda_x M(x))N \mapsto M(N)$ β-conversion,

$\lambda_x(Mx) \mapsto M \quad (x \notin \mathrm{FV}(M))$ η-conversion,

$D\vec{P}(\vec{N}) \mapsto M(\vec{N})$ for $D\vec{P}(\vec{y}\,) = M(\vec{y}\,)$ a computation rule.

Questions

- Can all computable functionals be represented?
- Suppose $[\![M]\!]$ is a "numeral". Does M reduce to it? (Operational semantics "adequate" for the denotational one).
- How can totality be defined?

Plotkin's definability theorem

Plotkin (1977) proved that a continuous functional is computable iff it can be defined by a PCF term involving additional constants for "parallel" computable functionals:

- the parallel conditional pcond, and
- a continuous approximation $\exists$ to the existential quantifier.

His proof makes essential use of the fact that his base type **N** is flat. Goal: extend the proof to the non-flat setting (j.w.w. Basil Karadais).

Properties of pcond, ∃ and valmax

$$[\![\mathrm{pcond}(\mathrm{tt}, x, y)]\!]^{u,v}_{x,y} = u, \qquad [\![\mathrm{pcond}(\mathrm{ff}, x, y)]\!]^{u,v}_{x,y} = v,$$

$$[\![\mathrm{pcond}(p, x, x)]\!]^{z,u}_{p,x} = u,$$

$$[\![\exists f]\!]^{w}_{f} = \{\mathrm{ff}\} \quad \text{if } (\emptyset, \mathrm{ff}) \in w,$$

$$[\![\exists f]\!]^{w}_{f} = \{\mathrm{tt}\} \quad \text{if } (\{\mathrm{S}*\}, \mathrm{tt}) \in w \text{ or } (\{0\}, \mathrm{tt}) \in u,$$

$$[\![\mathrm{valmax}(x, y)]\!]^{u,v}_{x,y} = u \quad \text{if } \mathrm{S}^n 0 \in v \text{ and } a_n \in u,$$

$$[\![\mathrm{valmax}(x, y)]\!]^{u,v}_{x,y} = \overline{\{a_n\}} \quad \text{if } \mathrm{S}^n 0 \in v \text{ and } a_n \notin u.$$

Notice that an n with $\mathrm{S}^n 0 \in v$ is uniquely determined if it exists. (a_n) is a fixed enumeration of the tokens in $\mathbf{A_N}$.

Point-free definition of pcond, ∃ and valmax

For readability the leading $\vec{U}$ and $\lambda_{\vec{x}}$ are omitted. pcond:

$$\frac{U \vdash \mathrm{tt} \qquad V \vdash a}{(U, V, W, a) \in [\![\mathrm{pcond}]\!]}(P_1) \qquad \frac{U \vdash \mathrm{ff} \qquad W \vdash a}{(U, V, W, a) \in [\![\mathrm{pcond}]\!]}(P_2)$$

$$\frac{V \vdash a \qquad W \vdash a}{(U, V, W, a) \in [\![\mathrm{pcond}]\!]}(P_3).$$

∃:

$$\frac{U \vdash (\emptyset, \mathrm{ff})}{(U, \mathrm{ff}) \in [\![\exists]\!]}(E_1) \qquad \frac{U \vdash (\{\mathrm{S}*\}, \mathrm{tt})}{(U, \mathrm{tt}) \in [\![\exists]\!]}(E_2) \qquad \frac{U \vdash (\{0\}, \mathrm{tt})}{(U, \mathrm{tt}) \in [\![\exists]\!]}(E_3).$$

valmax:

$$\frac{U \vdash a_n \qquad U \vdash a}{(U, \{\mathrm{S}^n 0\}, a) \in [\![\mathrm{valmax}]\!]}(M_1) \qquad \frac{\{a_n\} \vdash a}{(U, \{\mathrm{S}^n 0\}, a) \in [\![\mathrm{valmax}]\!]}(M_2).$$

Definability theorem

A partial continuous functional Φ of type $\rho_1 \to \cdots \to \rho_p \to \mathbf{N}$ is said to be recursive in pcond and $\exists$ if it can be defined explicitly by a term involving the constructors $0, \mathrm{S}$ and the constants predecessor, the fixed point operators Y_ρ, pcond, $\exists$ and valmax.

Theorem

A partial continuous functional is computable if and only if it is recursive in pcond *and* $\exists$.

Proof of the definability theorem

- Since the constants are defined by rules: the ideals they denote are recursively enumerable. Hence every functional recursive in pcond and $\exists$ is computable.
- Conversely, let Φ be computable of type $\rho_1 \to \cdots \to \rho_p \to \mathbf{N}$. Then Φ is a primitive recursively enumerable set of tokens

$$\Phi = \{\, (U^1_{f_1 n}, \ldots, U^p_{f_p n}, a_{g n}) \mid n \in \mathbb{N} \,\}$$

where for each type ρ_j, $(U^j_i)_{i\in\mathbb{N}}$ is an enumeration of Con_{ρ_j}, and $f_1, \ldots, f_p$ and g are fixed primitive recursive functions.

Proof of the definability theorem (ctd.)

Let $\vec{\varphi} = \varphi_1, \ldots, \varphi_p$ be arbitrary continuous functionals of types $\rho_1, \ldots, \rho_p$ respectively. One shows that Φ is definable by

$$\Phi\vec{\varphi} = Yw_{\vec{\varphi}}0$$

with $w_{\vec{\varphi}}$ of type $(\mathbf{N} \to \mathbf{N}) \to \mathbf{N} \to \mathbf{N}$ given by

$$w_{\vec{\varphi}}\psi n := \mathrm{pcond}(\mathrm{incons}_{\rho_1}(\varphi_1, f_1 n) \vee \cdots \vee \mathrm{incons}_{\rho_p}(\varphi_p, f_p n), \psi(n+1), \mathrm{valmax}(\psi(n+1), gn)).$$

incons_{ρ_i} of type $\rho_i \to \mathbf{N} \to \mathbf{B}$ is continuous with

$$\mathrm{incons}(\varphi, n) = \begin{cases} \mathrm{tt} & \text{if } \varphi \cup U_n \text{ is inconsistent} \\ \mathrm{ff} & \text{if } \varphi \supseteq U_n \\ \bot & \text{otherwise.} \end{cases}$$

One can prove that there are such functionals recursive in pcond and $\exists$; their definition involves $\exists$.