Towards a Formal Theory of Computability: a Case Study

Basil A. Karádais
(joint work with H. Schwichtenberg and S. Huber)

Institute of Mathematics
Ludwig-Maximilian University of Munich

April 29, 2010

Intro

- We aim at a constructive formal theory of computability TCF^+, where the functionals are studied together with their *finite approximations*.
- The attempt is guided by the semantics of coherent, non-flat Scott information systems, induced by free algebras given by constructors; in this setting, the latter are injective and have disjoint ranges.
- We present here a case study, namely, an adaption of Plotkin's definability theorem:

"A functional is *computable*
if and only if
it is definable by a term in the language".

Scott Information Systems

A *(Scott) information system* is a triple $(T, \mathsf{Con}, \vdash)$ where T is a countable set of *tokens*, $\mathsf{Con} \subseteq \mathcal{P}_f(T)$ is a collection of *consistent sets* or *(formal) neighborhoods* and $\vdash \subseteq \mathsf{Con} \times T$ is an *entailment* relation, which obey the following:

1. $\forall_{a \in T}\, \{a\} \in \mathsf{Con}$,
2. $U \in \mathsf{Con} \wedge V \subseteq U \to V \in \mathsf{Con}$,
3. $U \vdash U$ (where $U \vdash V :\Leftrightarrow \forall_{b \in V}\, U \vdash b$),
4. $U \vdash V \wedge V \vdash c \to U \vdash c$,
5. $U \in \mathsf{Con} \wedge U \vdash b \to U \cup \{b\} \in \mathsf{Con}$.

Ideals

- An *ideal* or *object* is a set $u \subseteq T$ which is consistent and closed to entailment in the following sense:

  $$\forall_{U \subseteq^f u} U \in \mathsf{Con} \wedge \forall_{U \subseteq^f u} (U \vdash a \to a \in u) \enspace .$$

  Denote the empty ideal by $\perp$ and the collection of all ideals by Ide.

- The *(deductive) closure* of a consistent set $U \in \mathsf{Con}$ is defined by

  $$\overline{U} := \{a \in T \mid U \vdash a\} \enspace .$$

  It is $\overline{U} \in \mathsf{Ide}$ for every $U \in \mathsf{Con}$.

Function Spaces

Let $\alpha = (T_\alpha, \mathsf{Con}_\alpha, \vdash_\alpha)$ and $\beta = (T_\beta, \mathsf{Con}_\beta, \vdash_\beta)$ be information systems. Their *function space* $\alpha \to \beta = (T, \mathsf{Con}, \vdash)$ is defined by

$$\begin{aligned}
T \quad &:= \quad \mathsf{Con}_\alpha \times T_\beta \ , \\
\{(U_i, b_i)\}_{i\in I} \in \mathsf{Con} \quad &:\Leftrightarrow \quad \underset{J\subseteq I}{\forall} \, . \, \bigcup_{j\in J} U_j \in \mathsf{Con}_\alpha \to \{b_j\}_{j\in J} \in \mathsf{Con}_\beta \ , \\
\{(U_i, b_i)\}_{i\in I} \vdash (U, b) \quad &:\Leftrightarrow \quad \{(U_i, b_i)\}_{i\in I}\, U := \{b_i \mid U \vdash_\alpha U_i\} \vdash_\beta b \ ,
\end{aligned}$$

One can prove that $\alpha \to \beta$ is again an information system.

Scott Topology

- The *cone* over a neighborhood U is the set $\{u \in \mathrm{Ide}_\alpha \mid U \subseteq u\}$; the collection of all cones in an information system forms the basis of a topology on Ide, the *Scott topology*.
- The (Scott) continuous functions $f : \mathrm{Ide}_\alpha \to \mathrm{Ide}_\beta$ are in a bijective correspondence with the ideals $u \in \mathrm{Ide}_{\alpha\to\beta}$:

$$\begin{aligned} |u|(v) &:= \left\{ b \in T_\beta \mid \exists_{U \subseteq^f v} (U, b) \in u \right\} , \\ \hat{f} &:= \left\{ (U, b) \mid b \in f(\overline{U}) \right\} . \end{aligned}$$

The assignments are inverse to each other, i.e.,

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

Algebraic Information Systems

Consider an algebra given by three constructors, a nullary 0, a unary S, and a binary B. We induce an information system $(T, \mathrm{Con}, \vdash)$ as follows (write C for either constructor):

- It is $Cab \in T$ if $a, b \in T \cup \{*\}$, where $*$ means *least information* and a or b may be empty; a token is called *total* if it is $*$-free; so $0, B{*}{*}, B(S0)(SS*) \in T$, but $* \notin T$.
- For a finite U, it is $U \in \mathrm{Con}$ if (a) all tokens in it start with the same constructor, $U = \{Ca_1b_1, \ldots, Ca_nb_n\}$, and (b) every *component set* is consistent, in the sense that $\{b_1, \ldots, b_n\} - \{*\} \in \mathrm{Con}$ (similarly for a_i's); so $\{SB0*, SB{*}0\} \in \mathrm{Con}$ but $\{SB0*, S0\} \notin \mathrm{Con}$.
- For $n > 0$, it is $\{Ca_1b_1, \ldots, Ca_nb_n\} \vdash C'ab$ if (a) $C = C'$, and (b) each component set of the entailer entails the corresponding argument of the entailed, i.e., $\{a_1, \ldots, a_n\} \vdash a$, etc., where $U \vdash *$ is defined to be true; so $\{B0*, B{*}0\} \vdash B00$ but $\{B0*, B{*}0\} \nvdash S*$.

Coherence

An information system is *coherent* when

$$U \in \mathsf{Con} \leftrightarrow \forall_{a,b \in U} \{a, b\} \in \mathsf{Con} \ ,$$

for all finite U's. Write $a \asymp b$ for $\{a, b\} \in \mathsf{Con}$.

- Coherent information systems correspond to coherent domains, coherent precusl's, and coherent Scott-Ershov formal topologies.
- Algebraic information systems and their function spaces are coherent.

Injectivity and Range Disjointness of Constructors

Every constructor C induces a Scott continuous function in the function space, defined by

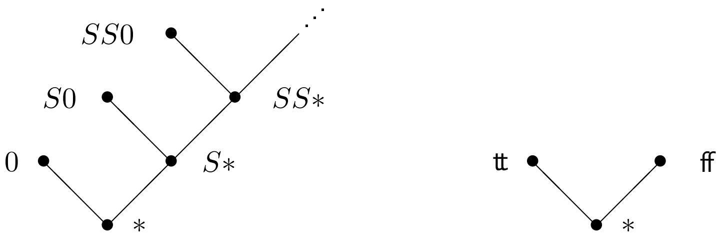
$$\tilde{C} := \left\{ (\vec{U}, C\vec{a}) \mid \vec{U} \vdash \vec{a} \right\} \ ,$$

where $(\vec{U}, b) := (U_1, \cdots (U_n, b) \cdots)$ and $\vec{U} \vdash \vec{a} :\Leftrightarrow \forall_i \, U_i \vdash a_i$.

- For an argument $\vec{u}$, it is $\tilde{C}(\vec{u}) = \left\{ C\vec{a} \mid \exists_{\vec{U} \subseteq^f \vec{u}} \vec{U} \vdash \vec{a} \right\}$.
- If $\tilde{C}(\vec{u}) = \tilde{C}(\vec{v})$ then $\vec{u} = \vec{v}$.
- If C and C' are distinct, then $\tilde{C}(\vec{u}) \neq \tilde{C}'(\vec{v})$.
- So constructors are injective and have disjoint ranges—not so in the flat case due to *strictness*, e.g.,
  $\tilde{B}(\perp, v) = \perp = \tilde{B}(u, \perp)$ and $\tilde{S}(\perp) = \perp = \tilde{S}'(\perp)$.

Algebraic Information Systems with at most Unary Constructors

- The tokens and entailment of the algebras $\mathbb{N} = \{0, S\}$ and $\mathbb{B} = \{\mathbf{tt}, \mathbf{ff}\}$ can be depicted like this:



- **Comparability Lemma**. If an algebra is given by at most unary constructors, then every two consistent tokens are comparable:

$$a \asymp b \to (\{a\} \vdash b \vee \{b\} \vdash a) \ .$$

Partial Continuous Functionals

- Let ι denote either the algebraic information system $\mathbb{N}$ or $\mathbb{B}$. The ideals $\mathsf{Ide}_{\rho\to\sigma}$ of a function space built on ι's are the *partial continuous functionals of type* $\rho \to \sigma$.
- A functional $u \in \mathsf{Ide}_\rho$ is *computable* when its set of tokens is recursively enumerable.
- A number $x \in \mathsf{Ide}_\iota$ is *total* if it is of the form $C\vec{z}$, with $\vec{z}$ total; a functional $f \in \mathsf{Ide}_{\rho\to\sigma}$ is *total* if for any total argument $z \in \mathsf{Ide}_\rho$, the value $f(z) \in \mathsf{Ide}_\sigma$ is also total.
- *Equivalence* of total ideals is defined simultaneously with the above: it is $x \approx_\iota y$ if both are of the form $C\vec{z_i}$, and $\vec{z_1} \approx_{\vec{\iota}} \vec{z_2}$; it is $f \approx_{\rho\to\sigma} g$ if $\forall_{z\in G_\rho} f(z) \approx_\sigma g(z)$.
- **Theorem** (Longo & Moggi, 1984). If $x \approx_\rho y$ then $f(x) \approx_\sigma f(y)$.

Computable Functionals

We build (object) terms from variables and constants by application and abstraction:

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

- Every *defined constant* D is given by *computation rules* $D\vec{P}_i(\vec{y_i}) = M_i$, $i = 1, \dots, n$, where $\vec{P}_i(\vec{y_i})$ are *constructor patterns*.
- Gödel's primitive recursion operators $\mathcal{R}$ of type $\mathbb{N} \to \rho \to (\mathbb{N} \to \rho \to \rho) \to \rho$ have computation rules $\mathcal{R}0fg = f$ and $\mathcal{R}(Sn)fg = gn(\mathcal{R}nfg)$.
- The least fixed point operators Y of type $(\rho \to \rho) \to \rho$ have the computation rule $Yf = f(Yf)$.

Denotational Semantics

For every closed term $\lambda_{\vec{x}} M$ of type $\vec{\rho} \to \sigma$ we inductively define a set $[\![\lambda_{\vec{x}} M]\!]$ of tokens of type $\vec{\rho} \to \sigma$.

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

For every constructor C and defined constant D we have

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

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

Denotational Semantics (continued)

- **Theorem**. For every term M, $[\![\lambda_{\vec{x}} M]\!]$ is an ideal. Furthermore, if a term M converts to M' by $\beta\eta$-conversion or application of a computation rule, then its value is preserved, i.e., $[\![M]\!] = [\![M']\!]$.
- For a term M with free variables among $\vec{x}$ and an assignment $\vec{x} \mapsto \vec{u}$ of ideals $\vec{u}$ to $\vec{x}$ let

$$[\![M]\!]^{\vec{u}}_{\vec{x}} := \bigcup_{\vec{U} \subseteq \vec{u}} [\![M]\!]^{\vec{U}}_{\vec{x}} := \bigcup_{\vec{U} \subseteq \vec{u}} \left\{ b \mid (\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!] \right\} .$$

Then by (A) we have *continuity of application*:

$$c \in [\![MN]\!]^{\vec{u}}_{\vec{x}} \leftrightarrow \exists_{V \subseteq [\![N]\!]^{\vec{u}}_{\vec{x}}} \left((V, c) \in [\![M]\!]^{\vec{u}}_{\vec{x}} \right) .$$

Definability Theorem: prerequisites

- Assume that the base types ι are generated by at most unary constructors (hence, that the Comparability Lemma applies).
- Use total (i.e., $*$-free) tokens of $T_{\mathbb{N}}$ as *indices*. Write $n \in \mathbb{N}$ for $S^n 0 \in T_{\mathbb{N}}$; then $\overline{n}$ is a total ideal of type $\mathbb{N}$.
- Fix *enumerations* $(e_n)_{n\in\mathbb{N}}$ of tokens and $(E_n)_{n\in\mathbb{N}}$ of neighborhoods for each type.
- We need the following special functionals: pcond, $\cup_{\#}$, and $\asymp_{\#}$.

Parallel Conditional pcond

- The parallel conditional pcond of type $\mathbb{B} \to \rho \to \rho \to \rho$ is defined by the clauses

$$
\begin{aligned}
&U \vdash \mathsf{tt} \to V \vdash a \to (U, V, W, a) \in \mathsf{pcond},\\
&U \vdash \mathsf{ff} \to W \vdash a \to (U, V, W, a) \in \mathsf{pcond},\\
&V \vdash a \to W \vdash a \to (U, V, W, a) \in \mathsf{pcond}.
\end{aligned}
$$

  We also need the least-fixed-point axiom. It is easy to see that pcond is an ideal.

- Properties of pcond:

$$
\begin{aligned}
&\mathsf{tt} \in z \to \mathsf{pcond}(z, x, y) = x,\\
&\mathsf{ff} \in z \to \mathsf{pcond}(z, x, y) = y,\\
&a \in x \to a \in y \to a \in \mathsf{pcond}(z, x, y).
\end{aligned}
$$

Continuous Union $\cup_{\#}$

- The *continuous union* $\cup_{\#}$ has type $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$; its defining clauses are

$$U \vdash e_n \to V \vdash n \to U \vdash a \to (U, V, a) \in \cup_{\#},$$
$$\{e_n\} \vdash a \to V \vdash n \to (U, V, a) \in \cup_{\#},$$

  and again we require the least-fixed-point axiom; $\cup_{\#}$ is an ideal.
- Properties of $\cup_{\#}$:

$$\forall_{a \in x}(a \asymp e_n) \to x \cup_{\#} \overline{n} = x \cup \overline{e_n},$$
$$e_n \in x \cup_{\#} \overline{n}.$$

Continuous Consistency $\asymp_\#$

- We define $\asymp_\#$ of type $\rho \to \mathbb{N} \to \mathbb{B}$ by the clauses

$$U \vdash E_n \to V \vdash n \to (U, V, \mathtt{tt}) \in \asymp_\#,$$
$$a \in U \to b \in E_n \to V \vdash n \to a \not\asymp b \to (U, V, \mathtt{ff}) \in \asymp_\#.$$

  Again we require the least-fixed-point axiom; $\asymp_\#$ is an ideal.

- Properties of $\asymp_\#$:

$$\mathtt{tt} \in x \asymp_\# \overline{n} \leftrightarrow x \supseteq E_n,$$

$$\mathtt{ff} \in x \asymp_\# \overline{n} \leftrightarrow \exists_{a \in x, b \in E_n} (a \not\asymp b).$$

Definability Theorem

- A partial continuous functional Φ of type $\rho \to \iota$ is *recursive in* pcond, $\cup_{\#}$, *and* $\asymp_{\#}$ if it can be defined explicitly by a term involving the constructors of ι's, the fixed point operators, the parallel conditional, the continuous union and the continuous consistency.
- **Definability Theorem**. A partial continuous functional is computable if and only if it is recursive in pcond, $\cup_{\#}$ and $\asymp_{\#}$.

Definability Theorem: Proofsketch

- For the left direction: the constants involved are defined in such a way that their denotations are clearly recursively enumerable, i.e., computable. The right direction is the nontrivial one.
- Let Φ be a computable functional of type $\rho \to \iota$; then its tokens (E_{fn}, e_{gn}) are enumerated by primitive recursive functions f and g on indices.
- For an arbitrary argument ϕ of type ρ define a term w_ϕ of type $(\mathbb{N} \to \iota) \to \mathbb{N} \to \iota$ by

$$w_\phi \psi x := \mathsf{pcond}(\phi \asymp_{\#} fx, \psi(Sx) \cup_{\#} gx, \psi(Sx)) \ .$$

- Show that $\Phi\phi = Y w_\phi 0$.

Definability Theorem: Proofsketch (continued)

In the proof, among others, we made heavy use of the following:

- basic definitions and properties of information systems and of the involved special functionals;
- induction over indices, ex falso quod libet, decidability of membership, consistency and entailment in base types;
- continuity of application, Comparability Lemma.

Syntax: TCF^+

- TCF^+ addresses computable functionals plus their finite approximations, i.e., their tokens and neighborhoods.
- Its development so far draws not only from the definability theorem, but also from an adaption of Berger's proof of Kreisel's *density theorem* [Berger 1993, Schwichtenberg 2006].

TCF^+: types

- We have *object types*, $\rho, \sigma ::= \iota \mid \rho \to \sigma$, as already explained.
- We also have *token types*: Tok_ρ for tokens, and LTok_ρ for lists of tokens, for every object type ρ.
- Both tokens and neighborhoods of a base type are generated inductively as expected.
- For tokens of a function type $\rho \to \sigma$ we have pairing (U, b) of LTok_ρ and Tok_σ, along with projections π_1, π_2. Consistency and entailment in function types are defined as expected.

TCF^+: token functions

We allow *token functions*, i.e., primitive recursive functions on token types; in particular, we define

- membership of a token in a neighborhood by $\dot{\in}_\iota \colon \mathrm{Tok}_\iota \to \mathrm{LTok}_\iota \to \mathrm{Tok}_\mathbb{B}$;
- equality of tokens by $=_\iota \colon \mathrm{Tok}_\iota \to \mathrm{Tok}_\iota \to \mathrm{Tok}_\mathbb{B}$;
- entailment by $\vdash_\iota \colon \mathrm{LTok}_\iota \to \mathrm{Tok}_\iota \to \mathrm{Tok}_\mathbb{B}$;
- consistency by $\mathsf{Con}_\iota : \mathrm{LTok}_\iota \to \mathrm{Tok}_\mathbb{B}$; write $a \asymp b$ for $\mathsf{Con}_\rho(a ::_\rho b ::_\rho \mathrm{nil}_\rho)$;
- totality by $G_\iota : \mathrm{Tok}_\iota \to \mathrm{Tok}_\mathbb{B}$; we use total tokens of $T_\mathbb{N}$ as *indices*, and write $n \in \mathbb{N}$.

Finally, we fix *enumerations* $(e_n)_{n\in\mathbb{N}}$ of tokens and $(E_n)_{n\in\mathbb{N}}$ of neighborhoods for each type, through appropriate Gödel numbering, so that we primitive recursively obtain $e_{\ulcorner a \urcorner} = a$ and $E_{\ulcorner U \urcorner} = U$.

TCF^+: Δ-formulas

- For each object type ρ, we have token variables a for tokens of Tok_ρ and list variables U of LTok_ρ. We build *token terms* from variables, constructor symbols and token function symbols.
- *Prime Δ-formulas*, or *decidable prime formulas*, are of the form $\mathrm{atom}(p)$ (for simplicity p), p of type $\mathrm{Tok}_\mathbb{B}$; these are decidable, in the sense that for each closed token term we can prove either $p =_\mathbb{B} \mathtt{tt}$ or $p =_\mathbb{B} \mathtt{ff}$, e.g., $a \asymp b$, $a \mathbin{\dot\in} U$, $U \vdash a$.
- *Δ-formulas* are built from prime Δ-formulas by $\to$, $\wedge$, $\vee$ and bounded quantifiers $\forall_{a \dot\in U}$ and $\exists_{a \dot\in U}$.

TCF^+: Σ-formulas and formulas

- We have set variables x for each type ρ and we build (object) terms from the variables as well as constants by application and abstraction, as explained. In particular, we have the constants $[\![\lambda_{\vec{x}} M]\!]$ of type $\vec{\rho} \to \sigma$, pcond of type $\mathbb{B} \to \rho \to \rho \to \rho$, $\cup_{\#}$ of type $\rho \to \mathbb{N} \to \rho$, $\asymp_{\#}$ of type $\rho \to \mathbb{N} \to \mathbb{B}$.
- *Prime Σ-formulas* are prime Δ-formulas (i.e., decidable ones) or have the form $a \in_\rho x$, with a a token variable or constant of type Tok_ρ.
- Σ*-formulas* are either (a) prime Σ-formulas, or (b) of the form $A_0 \to B$, for A_0 a Δ-formula and B a Σ-formula, and (c) they are closed under $\wedge$, $\vee$, bounded quantifiers, and existential quantifiers over token variables.
- Furthermore, *prime formulas* are either prime Σ-formulas or of the form $G_\rho x$ or $x \approx_\rho y$, for object variables x, y.
- *Formulas* are built from prime formulas by $\to$, $\wedge$, $\vee$, and all kinds of $\forall$ and $\exists$.

TCF^+: axioms and properties

- The theory is based on intuitionistic logic.
- Adapt the axioms of Heyting arithmetic to indices.
- Assume ordinary induction schemes for arbitrary formulas A. E.g.,

$$A(*) \to A(0) \to \forall_a (A(a) \to A(Sa)) \to A(a)$$

  is needed to prove the Comparability Lemma.
- Assume clauses of all inductive definitions of $[\![\lambda_{\vec{x}} M]\!]$, pcond, $\cup_\#$, $\asymp_\#$, G_ρ, $\approx_\rho$, together with the corresponding least-fixed-point axioms.

TCF^+: axioms and properties (continued)

- For object types, assume Σ-*comprehension*:

$$\exists_x \forall_a (a \in_\rho x \leftrightarrow A) \ ,$$

  where A is a Σ-formula. Write $x = \{a \mid A\}$ for terms defined by Σ-comprehension.

- Define $r \in_\rho t$, for r of type Tok_ρ and t of type ρ, by

$$\begin{aligned} r \in_\rho \{a \mid A(a)\} \quad &:\Leftrightarrow \quad A(r) \ , \\ r \in ts \quad &:\Leftrightarrow \quad \exists_{U \subseteq s} (U, r) \in t \ ; \end{aligned}$$

  the latter formalizes continuity of application.

Future Work

- Explain TCF^+ in a rigorous and systematic way; test it against further case studies (e.g., computational adequacy).
- Implementation on a theorem prover, which would allow for handling functionals and finite approximations alike (e.g., MINLOG, `http://www.math.lmu.de/~minlog/`).

References

- U. Berger, Total Sets and Objects in Domain Theory, *Annals of Pure and Applied Logic* **60**, 1993.
- S. Huber, B. A. K., H. Schwichtenberg, Towards a Formal Theory of Computability, to appear in R. Schindler, ed.: *Ways of Proof Theory (Wolfram Pohler's Festschrift)*.
- G. Plotkin, Gordon, LCF considered as a programming language, *Theoretical Computer Science* **5**(3), 1997.
- H. Schwichtenberg, Recursion on the partial continuous functionals, in C. Dimitracopoulos et al., eds.: Logic Colloquium '05, *Lecture Notes in Logic*, vol. 28, ASL, 2006.
- D. Scott, Domains for denotational semantics, in Nielsen, E. and Schmidt, E. M., eds.: *Automata, languages, and programming*, Volume 140 of Lecture Notes in Computer Science, Springer, 1982.
- V. Stoltenberg-Hansen, E. Griffor, I. Lindström, *Mathematical Theory of Domains*, Cambridge University Press, 1994.