A theory of computable functionals

Helmut Schwichtenberg

Mathematisches Institut, LMU, München

MCMP, 8. Dezember 2022

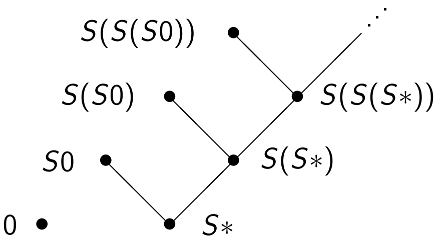
A theory of computable functionals (TCF)

Similar to HA^ω, but

- add inductively and coinductively defined predicates,
- distinguish computationally relevant (c.r.) and non-computational (n.c.) predicates,
- add realizability predicates (internal "meta"-step),
- allow partial functionals, defined by equations (possibly non-terminating, like corecursion),
- minimal logic: only $\to$, $\forall$ primitive. $\vee$, $\exists$, $\wedge$ inductively defined.

Minlog implements TCF.

- TCF has an intended model: partial continuous functionals.
- Defined via information systems (Scott). Has function spaces.
- It consists of ideals (infinite) approximated by tokens (finite).
- Ideals are consistent and deductively closed sets of tokens.
- Tokens are constructor trees with possibly $*$ at some leaves.
- Examples: natural numbers $\mathbb{N}$, binary trees $\mathbb{Y}$.



- $\{S0, S(S*)\}$ is inconsistent.
- $\{S*, S(S*)\}$ is an ideal.
- $\{S*, S(S*), S(S0)\}$ is an ideal ("total").
- $\{S*, S(S*), S(S(S*)), \dots\}$ is an infinite ideal ("cototal").

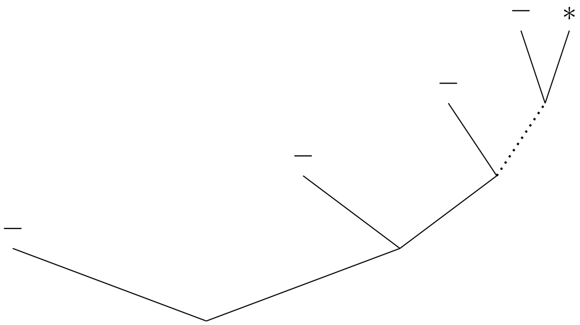
An ideal x in a closed base type

- is cototal if for each of its tokens $t(*)$ with a distinguished occurrence of $*$ there is another token of the form $t(\mathrm{C}\vec{*})$ in x,
- total if it is cototal and finite.

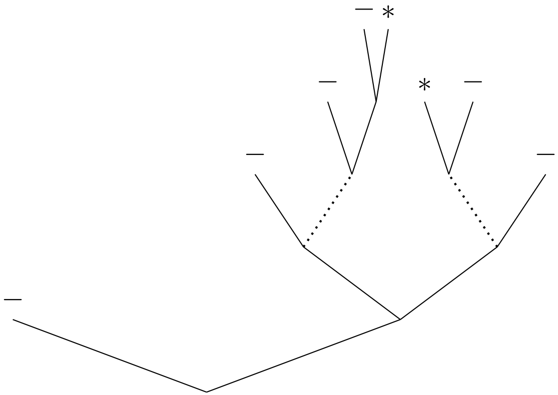
The base type $\mathbb{Y}$ (binary trees) is given by the constructors

$$\begin{aligned} -&\colon \mathbb{Y} && \text{(leaf)},\\ \mathrm{C}&\colon \mathbb{Y}\to\mathbb{Y}\to\mathbb{Y} && \text{(branch)}. \end{aligned}$$

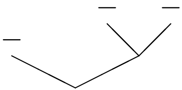
Example of a cototal ideal in $\mathbb{Y}$: all tokens



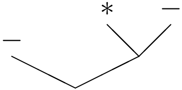
Another example of a cotota‌l ideal in $\mathbb{Y}$: all tokens



Example of a total ideal in $\mathbb{Y}$: deductive closure of



Example of a neither total nor cototal ideal: deductive closure of



Totality $T_{\mathbb{N}}$ is inductively defined as the least fixed point (lfp) of the clauses

$$0 \in T_{\mathbb{N}}, \qquad n \in T_{\mathbb{N}} \to Sn \in T_{\mathbb{N}}.$$

Cototality ${}^{\mathrm{co}}T_{\mathbb{N}}$ is coinductively defined as the greatest fixed point (gfp) of its closure axiom

$$n \in {}^{\mathrm{co}}T_{\mathbb{N}} \to n \equiv 0 \vee \exists_{n'}(n' \in {}^{\mathrm{co}}T_{\mathbb{N}} \wedge n \equiv Sn').$$

Similarity $\sim_{\mathbb{Y}}$ is a binary variant of totality. It is inductively defined as the least fixed point (lfp) of the clauses

$$
\begin{aligned}
&- \sim_{\mathbb{Y}} -,\\
&t_1 \sim_{\mathbb{Y}} t_1' \to t_2 \sim_{\mathbb{Y}} t_2' \to \mathrm{C} t_1 t_2 \sim_{\mathbb{Y}} \mathrm{C} t_1' t_2'.
\end{aligned}
$$

Bisimilarity $\approx_{\mathbb{Y}}$ is a binary variant of cototality. It is coinductively defined as the greatest fixed point (gfp) of its closure axiom

$$
\begin{aligned}
t \approx_{\mathbb{Y}} t' \to ((t \equiv -) \wedge (t' \equiv -)) \vee&\\
\exists_{t_1, t_2, t_1', t_2'} (t_1 \approx_{\mathbb{Y}} t_1' \wedge t_2 \approx_{\mathbb{Y}} t_2' \wedge t \equiv \mathrm{C} t_1 t_2 \wedge t' \equiv \mathrm{C} t_1' t_2')&
\end{aligned}
$$

Lemma

For every closed base type bisimilarity implies Leibniz equality.

- Example: $\mathbb{Y}$. Let a range over tokens, t over ideals.
- By induction on the height of extended tokens a^* we prove

$$\forall_{a^*,t,t'}(t \approx_{\mathbb{Y}} t' \to a^* \in t \to a^* \in t').$$

- It suffices to consider the case $\mathrm{C}a_1^* a_2^*$.
- From $t \approx t'$ by closure we have ideals t_1, t_2, t_1', t_2' with

$$t_1 \approx t_1' \wedge t_2 \approx t_2' \wedge t \equiv \mathrm{C}t_1 t_2 \wedge t' \equiv \mathrm{C}t_1' t_2'.$$

- Then $a_i^* \in t_i$, and by IH $a_i^* \in t_i'$. Thus $\mathrm{C}a_1^* a_2^* \in t'$.

Axioms for (co)inductive predicates: $I^{\pm}$, ${}^{\mathrm{co}}I^{\pm}$. Examples:

- Even. The introduction axioms (or clauses) are $\mathrm{Even}^{+}_{0,1}$:

$$0 \in \mathrm{Even}, \qquad n \in \mathrm{Even} \to S(Sn) \in \mathrm{Even}$$

  and the elimination axiom is Even^{-}:

$$0 \in X \to \forall_n (n \in \mathrm{Even} \to n \in X \to S(Sn) \in X) \to \mathrm{Even} \subseteq X.$$

  "Every competitor X satifying the clauses is above Even."

- Similar: $T^{\pm}_{\iota}$, ${}^{\mathrm{co}}T^{\pm}_{\iota}$, $\sim^{\pm}_{\iota}$ and $\approx^{\pm}_{\iota}$
- The n.c. Leibniz equality $\equiv$ is defined by

$$\equiv^{+}: x^{\tau} \equiv x^{\tau} \qquad \equiv^{-}: x \equiv y \to \forall_x Xxx \to Xxy$$

We can deduce the property Leibniz used as a definition.

Lemma (Compatibility of EqD)

$$x \equiv y \to A(x) \to A(y).$$

Proof: By the elimination axiom with
$X := \{\, x, y \mid A(x) \to A(y) \,\}$.

Using compatibility of $\equiv$ one proves symmetry and transitivity.
Define falsity by $\mathbf{F} := (\mathtt{ff} \equiv \mathtt{tt})$.

Theorem (Ex-falso-quodlibet)

We can derive $\mathbf{F} \to A$ from assumptions $\mathrm{Ef}_Y \colon \forall_{\vec{x}}(\mathbf{F} \to Y\vec{x})$ for predicate variables Y strictly positive in A, and $\mathrm{Ef}_I \colon \forall_{\vec{x}}(\mathbf{F} \to I\vec{x})$ for inductive predicates I without a nullary clause.

Bisimilarity axioms:

For every closed base type bisimilarity implies Leibniz equality.

Justification: holds in the intended model.

For closed base types ι it follows that

$$t \sim_\iota t' \leftrightarrow t, t' \in T_\iota \wedge t \equiv t',$$
$$t \approx_\iota t' \leftrightarrow t, t' \in {}^{\mathrm{co}}T_\iota \wedge t \equiv t'.$$

This is helpful because it gives us a tool (induction, coinduction) to prove equalities $t \equiv t'$, which otherwise would be difficult.

Corollary

$$t \sim_\iota t \leftrightarrow t \in T_\iota,$$
$$t \approx_\iota t \leftrightarrow t \in {}^{\mathrm{co}}T_\iota,$$

$\sim_\iota$ *is an equivalence relation on* T_ι,

$\approx_\iota$ *is an equivalence relation on* ${}^{\mathrm{co}}T_\iota$.

Definition (Pointwise equality[1])

$$(x \doteq_\iota y) := \begin{cases} x \approx_\iota y & \text{if } \iota \text{ is a ``cotype''} \\ x \sim_\iota y & \text{else} \end{cases}$$

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

Definition (Extensionality)

$$(x \in \mathrm{Ext}_\tau) := (x \doteq_\tau x).$$

[1] Robin Gandy, On the axiom of extensionality – Part I, JSL 1956 and Gaisi Takeuti, On a generalized logic calculus, Jap. J. Math. 1953

Example of a non-extensional functional:

- Define f, g of type $\mathbb{N} \to \mathbb{N}$ by the computation rules $fn = 0$ and $g0 = 0$, $g(Sn) = gn$.
- Then $f\bot_{\mathbb{N}} = 0$ by the computation rules for f.
- For $g\bot_{\mathbb{N}}$ no computation rule fits, but by the definition of $[\![\lambda_{\vec{x}}M]\!]$ we have that $[\![g\bot_{\mathbb{N}}]\!]$ is the empty ideal $[\![\bot_{\mathbb{N}}]\!]$.
- Hence $f \doteq g$, i.e., $\forall_{n,m}(n \doteq_{\mathbb{N}} m \to fn \doteq_{\mathbb{N}} gm)$, since $n \doteq_{\mathbb{N}} m$ implies $n \in T_{\mathbb{N}}$ and $n \equiv m$.
- Therefore the functional F defined by $Fh = h\bot_{\mathbb{N}}$ maps the pointwise equal f, g to different values.

Lemma

Ext_τ *and* ${}^{\mathrm{co}}T_\tau$ *are equivalent for closed types of level* ≤ 1.

Proof.

For closed base types this has been proved above. In case of level 1 we use induction on the height of the type. Let $\tau \to \sigma$ be a closed type of level 1. The following are equivalent.

$$
\begin{aligned}
& f \in \mathrm{Ext}_{\tau\to\sigma} \\
& f \doteq_{\tau\to\sigma} f \\
& \forall_{x,y}(x \doteq_\tau y \to fx \doteq_\sigma fy) \\
& \forall_{x \in {}^{\mathrm{co}}T_\tau}(fx \doteq_\sigma fx) \qquad\qquad \text{since } \mathrm{lev}(\tau) = 0 \\
& \forall_{x \in {}^{\mathrm{co}}T_\tau}(fx \in \mathrm{Ext}_\sigma).
\end{aligned}
$$

By IH the final formula is equivalent to $f \in {}^{\mathrm{co}}T_{\tau\to\sigma}$. □

For arbitrary closed types the relation $\doteq_\tau$ is a "partial equivalence relation", which means the following.

Lemma

For every closed type τ the relation $\doteq_\tau$ is an equivalence relation on Ext_τ.

Lemma (Compatibility of terms)

For every term $t(\vec{x})$ with extensional constants and free variables among $\vec{x}$ we have

$$\vec{x} \doteq_{\vec{\rho}} \vec{y} \to t(\vec{x}) \doteq_\tau t(\vec{y}).$$

Lemma (Extensionality of terms)

For every term $t(\vec{x})$ with extensional constants and free variables among $\vec{x}$ we have

$$\vec{x} \in \mathrm{Ext}_{\vec{\rho}} \to t(\vec{x}) \in \mathrm{Ext}_{\tau}.$$

Need "realizability extensions" of c.r. predicates and formulas:

- Assume that we have a global assignment giving for every c.r. predicate variable X of arity $\vec{\rho}$ an n.c. predicate variable $X^{\mathbf{r}}$ of arity $(\vec{\rho}, \xi)$ where ξ is the type variable associated with X.
- We introduce $I^{\mathbf{r}}/{}^{\mathrm{co}}I^{\mathbf{r}}$ for c.r. (co)inductive predicates $I/{}^{\mathrm{co}}I$, e.g.,

$$\mathrm{Even}^{\mathbf{r}}00 \qquad \mathrm{Even}^{\mathbf{r}}nm \to \mathrm{Even}^{\mathbf{r}}(S(Sn))(Sm).$$

- A predicate or formula C is **r**-free if it does not contain any of these $X^{\mathbf{r}}$, $I^{\mathbf{r}}$ or ${}^{\mathrm{co}}I^{\mathbf{r}}$.
- A derivation M is **r**-free if it contains **r**-free formulas only.

Definition ($C^{\mathbf{r}}$ for $\mathbf{r}$-free c.r. formulas C)

Let $z \mathbin{\mathbf{r}} C$ mean $C^{\mathbf{r}} z$.

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

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

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

Definition (Extracted term for an $\mathbf{r}$-free proof M of a c.r. A)

$$
\begin{aligned}
\mathrm{et}(u^A) &:= z_u^{\tau(A)} \quad (z_u^{\tau(A)} \text{ uniquely associated to } u^A),\\
\mathrm{et}((\lambda_{u^A} M^B)^{A\to B}) &:= \begin{cases} \lambda_{z_u}\mathrm{et}(M) & \text{if } A \text{ is c.r.}\\ \mathrm{et}(M) & \text{if } A \text{ is n.c.}\end{cases}\\
\mathrm{et}((M^{A\to B} N^A)^B) &:= \begin{cases} \mathrm{et}(M)\mathrm{et}(N) & \text{if } A \text{ is c.r.}\\ \mathrm{et}(M) & \text{if } A \text{ is n.c.}\end{cases}\\
\mathrm{et}((\lambda_x M^A)^{\forall_x A}) &:= \mathrm{et}(M),\\
\mathrm{et}((M^{\forall_x A(x)} t)^{A(t)}) &:= \mathrm{et}(M).
\end{aligned}
$$

It remains to define extracted terms for the axioms. Consider a (c.r.) inductively defined predicate I.

- $\mathrm{et}(I_i^+) := \mathrm{C}_i$ and $\mathrm{et}(I^-) := \mathcal{R}$, where the constructor C_i and the recursion operator $\mathcal{R}$ refer to ι_I associated with I.
- $\mathrm{et}({}^{\mathrm{co}}I^-) := \mathrm{D}$ and $\mathrm{et}({}^{\mathrm{co}}I_i^+) := {}^{\mathrm{co}}\mathcal{R}$, where the destructor D and the corecursion operator ${}^{\mathrm{co}}\mathcal{R}$ refer to ι_I again.

Let I be an inductive predicate and ι_I its associated algebra. One can show that

- every constructor of ι_I is extensional w.r.t. its clause I_i^+,
- $\mathcal{R}_{\iota_I}^{\alpha}$ is extensional w.r.t. the least-fixed-point axiom I^-,
- the destructor of ι_I is extensional w.r.t. the closure axiom ${}^{\mathrm{co}}I^-$, and
- ${}^{\mathrm{co}}\mathcal{R}_{\iota_I}^{\alpha}$ is extensional w.r.t. the greatest-fixed-point axiom ${}^{\mathrm{co}}I^+$.

Since the term $\mathrm{et}(M)$ extracted from a closed proof M of a c.r. formula A is built from these constants by abstraction and application, by the lemma on extensionality of terms we can conclude that $\mathrm{et}(M)$ is extensional w.r.t. A.

Theorem (Soundness)

Let M be an **r***-free derivation of a formula A from assumptions $u_i\colon C_i$ $(i < n)$. Then we can derive*

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

from assumptions

$$\begin{cases} z_{u_i}\ \mathbf{r}\ C_i & \text{if } C_i \text{ is c.r.} \\ C_i & \text{if } C_i \text{ is n.c.} \end{cases}$$

We express

- Kolmogorov's view of "formulas as problems"[2]
- Feferman's dictum "to assert is to realize"[3]

by invariance axioms:

For **r**-free c.r. formulas A we require as axioms

$$\mathrm{InvAll}_A\colon \forall_z (z \mathbf{\,r\,} A \to A).$$
$$\mathrm{InvEx}_A\colon A \to \exists_z (z \mathbf{\,r\,} A).$$

[2] Zur Deutung der intuitionistischen Logik, Math. Zeitschr., 1932

[3] Constructive theories of functions and classes, Logic Colloquium 78, p.208

Invariance axioms used in the proof of soundness (1):

Case $(\lambda_{u^A} M^B)^{A \to B}$ with B n.c. We need a derivation of $A \to B$.
Subcase A c.r. By IH we have a derivation of B from $z \,\mathbf{r}\, A$. Using the invariance axiom $A \to \exists_z (z \,\mathbf{r}\, A)$ we get the required derivation of B from A:

$$\dfrac{\dfrac{A \to \exists_z (z \,\mathbf{r}\, A) \qquad A}{\exists_z (z \,\mathbf{r}\, A)} \qquad \begin{array}{c}[z \,\mathbf{r}\, A] \\ \mid \text{IH} \\ B\end{array}}{B}\ \exists^-$$

Invariance axioms used in the proof of soundness (2):

Case $(M^{A \to B} N^A)^B$ with B n.c. Goal: find a derivation of B.
Subcase A c.r. By IH we have derivations of $A \to B$ and of $\mathrm{et}(N) \mathbf{r} A$. From the invariance axiom $\forall_z (z \mathbf{r} A \to A)$ we obtain the required derivation of B by $\to^-$ from the derivation of $A \to B$ and

$$\dfrac{\dfrac{\forall_z (z \mathbf{r} A \to A) \qquad \mathrm{et}(N)}{\mathrm{et}(N) \mathbf{r} A \to A} \qquad \begin{array}{c} \mid \mathrm{IH} \\ \mathrm{et}(N) \mathbf{r} A \end{array}}{A}$$

Issues

- Strong language, but controlled existence axioms (Kreisel).
- Functions (other than constructors) can only be defined by computation rules, e.g.,

$$n + 0 = n,$$
$$n + S(m) = S(n + m).$$

  No termination proof is required, hence partial functions.
- Predicates can only be defined inductively or coinductively.
- Bisimilarity and invariance axioms justified: hold in a model.

Conclusion

- In TCF the computational content of a proof M is represented by an extracted term $\mathrm{et}(M)$ in the language of TCF.
- The Soundness theorem provides a formal verfication in TCF that the extracted term realizes the formula ("specification"). This is automated in Minlog.
- Since extraction ignores n.c. parts of the proof, $\mathrm{et}(M)$ is much shorter than M.
- For efficiency, in a second step one can translate the extracted term to a functional programming language. Minlog does this for Scheme and Haskell.