Robust computability notions for higher types arising in classical analysis

John Longley

School of Informatics
University of Edinburgh
jrl@inf.ed.ac.uk

Domains XIII
Oxford, July 2018

Orientation

Earlier work (Normann 2000, Longley 2007):
A wide class of models/languages for higher-order computation (cast as typed partial combinatory algebras) gives rise to just a handful of total type structures over $\mathbb{N}$ (types $\mathbb{N}, \mathbb{N}^{\mathbb{N}}, \mathbb{N}^{\mathbb{N}^{\mathbb{N}}}, \ldots$).

- 'Continuous operations on continuous data' $\Rightarrow$ Ct (Kleene-Kreisel)
- 'Effective operations on continuous data' $\Rightarrow$ $\mathsf{Ct}^{\mathrm{eff}}$ $\quad (\subset \mathsf{Ct})$
- 'Effective operations on effective data' $\Rightarrow$ HEO

Orientation

Earlier work (Normann 2000, Longley 2007):
A wide class of models/languages for higher-order computation (cast as typed partial combinatory algebras) gives rise to just a handful of total type structures over $\mathbb{N}$ (types $\mathbb{N}, \mathbb{N}^{\mathbb{N}}, \mathbb{N}^{\mathbb{N}^{\mathbb{N}}}, \ldots$).

- 'Continuous operations on continuous data' $\Rightarrow$ Ct (Kleene-Kreisel)
- 'Effective operations on continuous data' $\Rightarrow$ $\mathrm{Ct}^{\mathrm{eff}}$ $(\subset \mathrm{Ct})$
- 'Effective operations on effective data' $\Rightarrow$ HEO

This work: Extend these 'ubiquity' results to other types more relevant to mathematical practice, e.g.

- Spaces of continuous functions on subsets of $\mathbb{R}^n$
- Spaces of analytic functions on subsets of $\mathbb{C}$.
- Operators on such spaces . . . [E.g. finite types over $\mathbb{R}$]

Also outline a cleaner, more axiomatic approach than that of (L 2007) — and widen the class of models in some ways.

Foundational issues

Recall how such 'mathematical types' are constructed within a foundational framework based on Church's simple types (as in e.g. Isabelle/HOL). Suppose types built up from $\mathbb{N}$ via '$\to$' (total function space constructor).

etc.

Foundational issues

Recall how such 'mathematical types' are constructed within a foundational framework based on Church's simple types (as in e.g. Isabelle/HOL). Suppose types built up from $\mathbb{N}$ via '$\rightarrow$' (total function space constructor).

'Days of Creation' for the mathematical universe . . .

Day 1: Integers / rationals are representable at type $\mathbb{N}$.

etc.

Foundational issues

Recall how such 'mathematical types' are constructed within a foundational framework based on Church's simple types (as in e.g. Isabelle/HOL). Suppose types built up from $\mathbb{N}$ via '$\rightarrow$' (total function space constructor).

'Days of Creation' for the mathematical universe ...

Day 1: Integers / rationals are representable at type $\mathbb{N}$.

Day 2: Real / complex numbers are representable at type $\mathbb{N} \rightarrow \mathbb{N}$.

etc.

Foundational issues

Recall how such 'mathematical types' are constructed within a foundational framework based on Church's simple types (as in e.g. Isabelle/HOL). Suppose types built up from $\mathbb{N}$ via '$\rightarrow$' (total function space constructor).

'Days of Creation' for the mathematical universe ...

Day 1: Integers / rationals are representable at type $\mathbb{N}$.

Day 2: Real / complex numbers are representable at type $\mathbb{N} \rightarrow \mathbb{N}$.

Day 3: Functions on $\mathbb{R}$ or $\mathbb{C}$ are representable at $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$.

etc.

Foundational issues

Recall how such ‘mathematical types’ are constructed within a foundational framework based on Church’s simple types (as in e.g. Isabelle/HOL). Suppose types built up from $\mathbb{N}$ via ‘$\rightarrow$’ (total function space constructor).

‘Days of Creation’ for the mathematical universe . . .

Day 1: Integers / rationals are representable at type $\mathbb{N}$.

Day 2: Real / complex numbers are representable at type $\mathbb{N} \rightarrow \mathbb{N}$.

Day 3: Functions on $\mathbb{R}$ or $\mathbb{C}$ are representable at $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$.

Day 4: Operators on such functions are representable at . . . .

etc.

Foundational issues

Recall how such 'mathematical types' are constructed within a foundational framework based on Church's simple types (as in e.g. Isabelle/HOL). Suppose types built up from $\mathbb{N}$ via '$\rightarrow$' (total function space constructor).

'Days of Creation' for the mathematical universe ...

Day 1: Integers / rationals are representable at type $\mathbb{N}$.

Day 2: Real / complex numbers are representable at type $\mathbb{N} \rightarrow \mathbb{N}$.

Day 3: Functions on $\mathbb{R}$ or $\mathbb{C}$ are representable at $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$.

Day 4: Operators on such functions are representable at ....

etc.

So in one sense, the (hereditarily total) finite types over $\mathbb{N}$ already suffice for representing these mathematical objects.

But ...

Subset and quotient types

For 'practical' purposes, it's helpful to add subset and quotient types. E.g. $\mathbb{R}$ as a quotient of a subset of $\mathbb{N} \to \mathbb{N}$.

In the context of a classical logic (as in Isabelle/HOL), this is an inessential extension: e.g. a function with domain $S \subseteq \mathbb{N} \to \mathbb{N}$ can always be represented by a function on $\mathbb{N} \to \mathbb{N}$.

Subset and quotient types

For 'practical' purposes, it's helpful to add subset and quotient types. E.g. $\mathbb{R}$ as a quotient of a subset of $\mathbb{N} \to \mathbb{N}$.

In the context of a classical logic (as in Isabelle/HOL), this is an inessential extension: e.g. a function with domain $S \subseteq \mathbb{N} \to \mathbb{N}$ can always be represented by a function on $\mathbb{N} \to \mathbb{N}$.

Not so in constructive or computable settings. E.g. under any reasonable definition of 'computability' ...

Subset and quotient types

For 'practical' purposes, it's helpful to add subset and quotient types. E.g. $\mathbb{R}$ as a quotient of a subset of $\mathbb{N} \to \mathbb{N}$.

In the context of a classical logic (as in Isabelle/HOL), this is an inessential extension: e.g. a function with domain $S \subseteq \mathbb{N} \to \mathbb{N}$ can always be represented by a function on $\mathbb{N} \to \mathbb{N}$.

Not so in constructive or computable settings. E.g. under any reasonable definition of 'computability' ...

- $f \mapsto \min i.\ f(i) \neq 0$ is computable on $(\mathbb{N} \to \mathbb{N}) - \{\Lambda i.0\}$, but not extendable to a computable (or continuous) function on $\mathbb{N} \to \mathbb{N}$.

Subset and quotient types

For 'practical' purposes, it's helpful to add subset and quotient types. E.g. $\mathbb{R}$ as a quotient of a subset of $\mathbb{N} \to \mathbb{N}$.

In the context of a classical logic (as in Isabelle/HOL), this is an inessential extension: e.g. a function with domain $S \subseteq \mathbb{N} \to \mathbb{N}$ can always be represented by a function on $\mathbb{N} \to \mathbb{N}$.

Not so in constructive or computable settings. E.g. under any reasonable definition of 'computability' ...

- $f \mapsto \min\ i.\ f(i) \neq 0$ is computable on $(\mathbb{N} \to \mathbb{N}) - \{\Lambda i.0\}$, but not extendable to a computable (or continuous) function on $\mathbb{N} \to \mathbb{N}$.
- $x \mapsto 1/x : \mathbb{R} - \{0\} \to \mathbb{R}$ is computable, but not extendable to a computable (or continuous) function $\mathbb{R} \to \mathbb{R}$.

Subset and quotient types

For 'practical' purposes, it's helpful to add subset and quotient types. E.g. $\mathbb{R}$ as a quotient of a subset of $\mathbb{N} \to \mathbb{N}$.

In the context of a classical logic (as in Isabelle/HOL), this is an inessential extension: e.g. a function with domain $S \subseteq \mathbb{N} \to \mathbb{N}$ can always be represented by a function on $\mathbb{N} \to \mathbb{N}$.

Not so in constructive or computable settings. E.g. under any reasonable definition of 'computability' ...

- $f \mapsto \min i.\ f(i) \neq 0$ is computable on $(\mathbb{N} \to \mathbb{N}) - \{\Lambda i.0\}$, but not extendable to a computable (or continuous) function on $\mathbb{N} \to \mathbb{N}$.
- $x \mapsto 1/x : \mathbb{R} - \{0\} \to \mathbb{R}$ is computable, but not extendable to a computable (or continuous) function $\mathbb{R} \to \mathbb{R}$.
- Given a closed curve c in the plane and a point p not on c, can compute the winding number of c around p. Not extendable to a computable operation on arbitrary pairs (c, p).

Subset and quotient types

For 'practical' purposes, it's helpful to add subset and quotient types. E.g. $\mathbb{R}$ as a quotient of a subset of $\mathbb{N} \to \mathbb{N}$.

In the context of a classical logic (as in Isabelle/HOL), this is an inessential extension: e.g. a function with domain $S \subseteq \mathbb{N} \to \mathbb{N}$ can always be represented by a function on $\mathbb{N} \to \mathbb{N}$.

Not so in constructive or computable settings. E.g. under any reasonable definition of 'computability' . . .

- $f \mapsto \min i.\ f(i) \neq 0$ is computable on $(\mathbb{N} \to \mathbb{N}) - \{\Lambda i.0\}$, but not extendable to a computable (or continuous) function on $\mathbb{N} \to \mathbb{N}$.
- $x \mapsto 1/x : \mathbb{R} - \{0\} \to \mathbb{R}$ is computable, but not extendable to a computable (or continuous) function $\mathbb{R} \to \mathbb{R}$.
- Given a closed curve c in the plane and a point p not on c, can compute the winding number of c around p. Not extendable to a computable operation on arbitrary pairs (c, p).
- If f is analytic on a disc $D_{1+\epsilon}$ and nonzero on ∂D_1, can compute the number of zeros (by multiplicity) of f within D_1. Not extendable to arbitrary continuous f, if codomain is taken to be $\mathbb{N}$ rather than $\mathbb{R}$.

Robust computability notions for mathematical types

Moral: Saying what 'computability' means at type $S \to T$ doesn't immediately fix what it should mean at $S' \to T$ where $S' \subseteq S$.

So a 'computability theory' for analysis should pay due attention to subset types. (Quotients then fall out for abstract reasons.)

Robust computability notions for mathematical types

Moral: Saying what ‘computability’ means at type $S \to T$ doesn't immediately fix what it should mean at $S' \to T$ where $S' \subseteq S$.

So a ‘computability theory’ for analysis should pay due attention to subset types. (Quotients then fall out for abstract reasons.)

From earlier work, we know that under quite mild conditions, two ‘higher-order computability models’ (TPCAs) yield same objects at all simple types over $\mathbb{N}$.

To what extent does this remain true when subset types are thrown in?

(In other words, how extensive are the subcategories shared by many different models $\mathcal{P}er(A)$?)

Robust computability notions for mathematical types

Moral: Saying what 'computability' means at type $S \to T$ doesn't immediately fix what it should mean at $S' \to T$ where $S' \subseteq S$.

So a 'computability theory' for analysis should pay due attention to subset types. (Quotients then fall out for abstract reasons.)

From earlier work, we know that under quite mild conditions, two 'higher-order computability models' (TPCAs) yield same objects at all simple types over $\mathbb{N}$.

To what extent does this remain true when subset types are thrown in?

(In other words, how extensive are the subcategories shared by many different models $\mathcal{P}er(A)$?)

Much existing work (e.g. in Type Two Effectivity) focuses on one particular underlying 'model of computation'.
Our contribution: the classes of functions we get are (largely) independent of the choice of computation model.

Models of higher-order computation

Types: $\sigma ::= \mathtt{N} \mid \sigma \to \sigma$. Pure types: $\overline{0} = \mathtt{N}$, $\overline{k+1} = \overline{k} \to \mathtt{N}$.

General setup: a typed partial combinatory algebra TPCA A with weak numerals and type 2 recursion. That is:

- a set $A(\sigma)$ for each type σ,
- partial 'application' functions $\cdot_{\sigma\tau} : A(\sigma \to \tau) \times A(\sigma) \rightharpoonup A(\tau)$
- ...such that there exist elements

$$k_{\sigma\tau}\ , \quad s_{\rho\sigma\tau}\ , \quad \widehat{0}, \widehat{1}, \ldots, \quad suc\ , \quad primrec\ , \quad Y_{\overline{2}}$$

satisfying familiar axioms.

There's an abundance of such structures, both 'syntactic' (term models for higher-order programming languages) and 'semantic' (arising from domain theory, game semantics, ...), embodying different flavours of higher-order computability.
These include untyped PCAs as a special case (K_1, K_2, $\mathcal{P}\omega$, ...).

Everything we do also works in the relative setting (TPCA A with designated 'computable substructure' $A^\sharp$), at a slight notational cost.

Our theory also works for a 'non-deterministic' variant of the above setup, so that we cover e.g. lattice models like $\mathcal{P}\omega$. (Fills a gap in (L 2007)).

Special axioms

We'll generalize the argument used for 'continuous' models in (L 2007). There, we assumed A came with a simulation in K_2 of a certain kind. Here, we replace this by some cleaner intrinsic conditions on A.

Let m, n, p range over $N = \{\widehat{0}, \widehat{1}, \ldots\} \subseteq A(\mathtt{N})$.
Let $N^N = \{g \in A(\overline{1}) \mid \forall n. \exists m.\ g \cdot n = m\}$.

Continuity: For any $F \in A(\overline{2})$, if $F \cdot g = p$ for all $g \in N^N$ such that $\forall n.\ g \cdot n = \widehat{0}$, then $F \cdot g = p$ for some $g \in N^N$ such that $\exists n.\ g \cdot n \neq \widehat{0}$.

Enumeration: For any $f \in A(\overline{1})$ there exists $g \in N^N$ such that

$$\forall m, n.\ f \cdot n = m \Leftrightarrow \exists p.\ g \cdot p = `\langle n, m\rangle + 1`$$

Normalizability: There exists $norm \in A(\overline{1} \rightarrow \overline{1})$ such that

$$\forall f \in N^N.\ norm \cdot f \sim f\ , \qquad \forall g, g' \in N^N.\ f \sim g \Rightarrow norm \cdot f = norm \cdot g$$

where $f \sim g$ means $\forall n.\ f \cdot n = g \cdot n$. (Excludes very intensional models like K_1.)

These will hold in all 'continuous' models covered in (L 2007), most 'effective' ones, and others besides.

Key idea: graphs and regular types

A key role will be played by the set Δ of functions $\mathbb{N} \to \mathbb{N}$ representable in A (by an element of N^N). Contents of Δ will completely determine contents of many other types.
(E.g. Finite types over $\mathbb{N}$ are Ct if $\Delta = \mathbb{N}^{\mathbb{N}}$, or HEO if $\Delta = \mathbb{N}^{\mathbb{N}}_{\mathrm{eff}}$.)

More specifically, for many types X, we shall have $\Phi \in X$ iff Φ has a 'graph' within Δ. We say X is regular if this is the case.

Example: Second-order functions (defined on subsets of Δ).
Think of Δ as a modest set over A.
Let X be any regular (in categorical sense!) subobject of Δ.

We say $g : \mathbb{N} \to \mathbb{N}$ is a graph of $F : |X| \to \mathbb{N}$ if g enumerates a set of elements $\langle\langle\langle n_1, m_1\rangle, \ldots, \langle n_r, m_r\rangle\rangle, p\rangle$ that form a 'graph' of F in the expected sense.

Theorem: Under our axioms, F is present in the modest set $(X \Rightarrow N)$ iff F has a graph in Δ. So all such $(X \Rightarrow N)$ are regular.

(Abstract version of Kreisel-Lacombe-Shoenfield theorem.)
NB. Normalizability means we needn't assume $X \subseteq \Delta$ is separable.

Higher types

Consider the modest sets over A we can reach by starting from N and alternately:

- picking any regular subobject
- applying $(- \Rightarrow N)$.

Thus $Q_0 \subseteq N$, $Q_1 \subseteq (Q_0 \Rightarrow N)$, $\ldots$, $Q_k \subseteq (Q_{k-1} \Rightarrow N)$.

From these types, we can abstractly reconstruct all modest sets reachable from N via $\Rightarrow$, regular subobjects and regular quotients.

Main theorem: Suppose $Q_0, \ldots, Q_{k-1}$ above are all Δ-separable subobjects (Q_k need not be). Then the type $(Q_k \Rightarrow N)$ is regular. So if A, B are two models with $\Delta_A = \Delta_B$, they agree at this type.

Here, suitable notions of graph and Δ-separable subset are defined by induction for the relevant types.

At type level 2, we require KLS methods but only weak computing power (ground-type iteration).

At type levels $k \geq 3$, we require the Normann algorithms to get from a graph in Δ to a realizer in $A(\overline{k})$.

Specific cases

If $\Delta = \mathbb{N}^{\mathbb{N}}$, all subsets are Δ-separable!
So we get ubiquity for all types generated from N by $\Rightarrow$, regular subobjects and regular quotients.

E.g. the finite types $\mathbb{R}, \mathbb{R}^{\mathbb{R}}, \mathbb{R}^{\mathbb{R}^{\mathbb{R}}}, \ldots$: get ubiquity theorem for the intensional hierarchy (cf. Bauer, Escardó, Simpson, Normann, Schröder).

The 'relative' case $(\mathbb{N}^{\mathbb{N}}; \mathbb{N}^{\mathbb{N}}_{\mathrm{eff}})$ is also interesting: separability questions become non-trivial. Nevertheless:

- The examples from analysis given earlier are covered.
- Get ubiquity for $\mathbb{R}$-hiearchy at least for levels ≤ 4 (where $\mathbb{R}$ has level 0), and probably all the way.

Lots more to explore (e.g. particular problems in analysis; relationship to Type Two Effectivity).