Does recursion help?

Gordon Plotkin

Edinburgh University

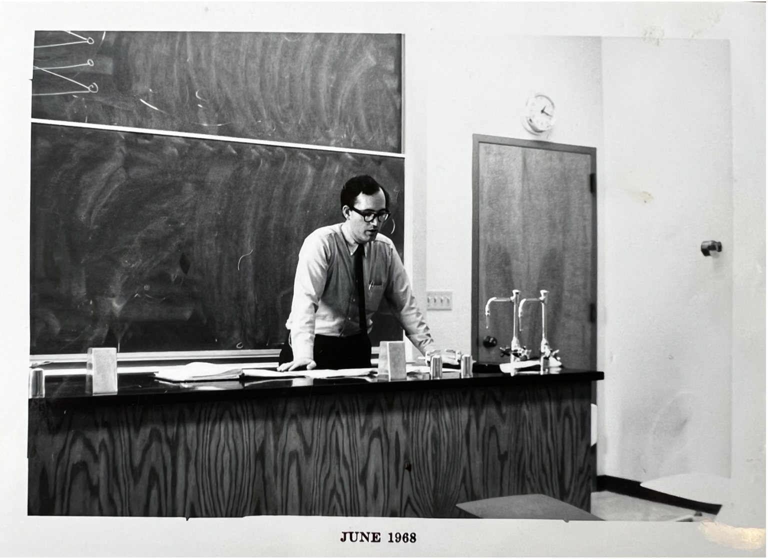
Dana's 90th Birthday Symposium

Topos Institute, Berkeley

October, 2022

Dana reading, with Monica

Dana 1968



Henk, Jan Willem, & Gordon $\sim$1977

The untyped λ-calculus

Representing the natural numbers with Church numerals

Use the coding function $\gamma : \mathbb{N} \to \Lambda$ where:

$$\gamma(n) = \lambda f.\, \lambda x.\, f^n(x)$$

Representing numerical functions

A (closed) term F $\lambda\beta$-represents $f : \mathbb{N} \rightharpoonup \mathbb{N}$ iff:

$$f(m) = n \implies \lambda\beta \vdash F\gamma(m) = \gamma(n)$$

$$f(m)\uparrow \implies F(\gamma(m)) \text{ has no } \beta\text{-normal form}$$

Theorem (Church, Kleene, Turing)

The following coincide:

1. *The $\lambda\beta$-representable functions*
2. *The Gödel-Herbrand partial recursive functions*
3. *The functions computable by a Turing machine*

The typed λ-calculus

Representing the natural numbers

Numeral type:

$$\underline{\mathbb{N}} = (o \to o) \to (o \to o)$$

Coding function $\gamma : \mathbb{N} \to \Lambda_{\underline{\mathbb{N}}}$ where:

$$\gamma(n) = \lambda f : o \to o.\, \lambda x : o.\, f^n(x)$$

Defining functions

A term $F : \underline{\mathbb{N}} \to \underline{\mathbb{N}}$ represents $f : \mathbb{N} \to \mathbb{N}$ iff:

$$f(m) = n \implies \lambda\beta\eta \vdash F\gamma(m) = \gamma(n)$$

Theorem (Schwichtenberg, Statman)

The representable functions are the extended polynomials.

The extended polynomials

The class of extended polynomials is the smallest class of numerical functions closed under composition which contains:

1. the constant functions: 0 and 1,
2. the projections,
3. addition,
4. multiplication, and
5. the function

$$\mathrm{ifzero}(l, m, n) = \begin{cases} m & (l = 0) \\ n & (l \neq 0) \end{cases}$$

Uniform and non-uniform representations

More general numeral types

$$\underline{\mathbb{N}}_\sigma = (\sigma \to \sigma) \to (\sigma \to \sigma)$$

Non-uniform representation

$$F : \underline{\mathbb{N}}_{\sigma_1} \to \ldots \to \underline{\mathbb{N}}_{\sigma_k} \to \underline{\mathbb{N}}_\sigma$$

Uniform representation

$$F : \underline{\mathbb{N}}_\sigma \to \ldots \to \underline{\mathbb{N}}_\sigma \to \underline{\mathbb{N}}_\sigma$$

Fact (Fortune, Leivant, and O'Donnell): Predecessor is non-uniformly representable.

Theorem (Zakrzewski): Predecessor is not uniformly representable.

Representable functions and schemas

Zakrzewski's conjecture

The class of uniformly representable numerical functions is the smallest class of numerical functions closed under composition which contains 1–5, as before, plus:

6. For any $l \geq 2$, the function
$$f_l(m, n_0, \ldots, n_{i-1}) = n_i$$
where $i = m \bmod l$

7. For any l, the function
$$\text{less-than}_l(m) = \begin{cases} 0 & (m \leq l) \\ 1 & (m \not\leq l) \end{cases}$$

Algebraic datatypes

Example T: the set of binary trees with leafs labelled by 0 or 1. Need two constants and a binary “cons" function. So set:

$$\underline{\mathrm{T}} = o \to o \to (o \to o \to o) \to o$$

Represent

$$\underline{0} = \lambda x : o, y : o, f : (o \to o \to o).\, x$$

$$\underline{1} = \lambda x : o, y : o, f : (o \to o \to o).\, y$$

$$\underline{cons} = \lambda t : \underline{\mathrm{B}}, u : \underline{\mathrm{B}}, x : o, y : o, f : (o \to o \to o).\, f(txyf)(uxyf)$$

Define $\gamma : \mathrm{T} \to \Omega_{\mathrm{T}}$ by:

$$\begin{array}{lcl} \gamma(0) & = & \underline{0} \\ \gamma(1) & = & \underline{1} \\ \gamma(cons(t,u)) & = & \underline{cons}\gamma(t)\gamma(u) \end{array}$$

Zaionc proved Schwichtenberg-Statman-type results for algebraic datatypes.

Representable functions and automata

Booleans

$$\underline{\mathrm{B}} = o \to o \to o$$

$$\begin{array}{rcl}\gamma(0) & = & \lambda x \lambda y.\, x \\ \gamma(1) & = & \lambda x \lambda y.\, y\end{array}$$

Binary words

$$\underline{\mathrm{W}}_\alpha = \alpha \to (\alpha \to \alpha) \to (\alpha \to \alpha) \to \alpha$$

Theorem (Hillebrand and Kanellakis)

The representable predicates $\mathrm{W}_\alpha \to \mathrm{B}$ (varying α) correspond exactly to the regular languages.

There is current work on automata and typed λ-calculi by Lê Thánh Düng Nguyên, Camille Noûs, and Pierre Pradic.

λ-representable functions in general

- Fix an extension λ^+ of the typed $\lambda\beta\eta$-calculus.
- A function $\gamma : X \to \Lambda_\sigma$ is λ^+-injective if

$$\lambda^+ \vdash \gamma(x) = \gamma(y) \implies x = y$$

- For non-empty sets $X_1, \ldots, X_k, X$ choose representing types $\underline{X_i}$ and $\underline{X}$, and λ^+-injective coding functions

$$\gamma_i : X_i \to \Lambda_{\underline{X_i}} \quad (i = 1, k) \qquad\qquad \gamma : X \to \Lambda_{\underline{X}}$$

- Then a λ-term

$$F : \underline{X_1} \to \ldots \to \underline{X_k} \to \underline{X}$$

represents

$$f : X_1 \times \ldots \times X_k \rightharpoonup X$$

iff

$$\mathrm{f}(x_1, \ldots, x_k) \simeq x_k \iff \lambda^+ \vdash F\gamma_1(x_1)\ldots\gamma_k(x_k) = \gamma(x)$$

Our extensions of the typed λ-calculus

- $\lambda\Omega$ This is $\lambda\beta\eta$ extended with a constant

$$\Omega : o$$

and no conversions.

- $\lambda\Omega^{+}$ This is $\lambda\beta\eta$ extended with constants

$$\Omega_\sigma : \sigma$$

and no conversions.

- $\lambda\mathrm{Y}$ This is $\lambda\beta\eta$ extended with recursion combinators, ie constants

$$\mathrm{Y}_\sigma : (\sigma \to \sigma) \to \sigma$$

It has conversions

$$\mathrm{Y}_\sigma F = F(\mathrm{Y}_\sigma F)$$

and reduction rules

$$\mathrm{Y}_\sigma \to \lambda f.f(\mathrm{Y}_\sigma f)$$

Recursion does not help

Suppose λ^{++} extends λ^{+}.

Then λ^{++} is conservative over λ^{+} for a class of functions $X_1 \times \ldots \times X_k \to X$ and coding scheme if such functions are λ^{++}-representable iff they are λ^{+}-representable.

Theorem (Total functions)

$\lambda\mathrm{Y}$ is conservative over $\lambda\beta\eta$ for total functions $X_1 \times \ldots \times X_k \to X$ and any coding scheme.

Theorem (Partial functions)

$\lambda\mathrm{Y}$ is conservative over $\lambda\Omega$ for all functions $X_1 \times \ldots \times X_k \rightharpoonup X$ and any coding scheme.

A corollary

Corollary

1. *(Zakrzewski) Predecessor is not uniformly representable*
2. *(Statman) Equality and inequality ($\leq$) are not uniformly representable.*

Proof.

Fixing $\underline{\mathbb{N}}_\alpha$, the three functions are interdefinable in $\lambda\mathrm{Y}$ via suitable recursions.
So if one of them were uniformly definable, so would be every total recursive function in $\lambda\mathrm{Y}$.
This contradicts the conservativity of $\lambda\mathrm{Y}$ over $\lambda\beta\eta$. $\square$

Going up is easy

An extension $\lambda^+ \subseteq \lambda^{++}$ is *conservative*, if, for all λ^+ terms M and N we have:

$$\lambda^+ \vdash M = N \iff \lambda^{++} \vdash M = N$$

The extensions $\lambda\beta\eta \subseteq \lambda\Omega \subseteq \lambda\Omega^+$ are conservative (use CR).

Lemma

If $\lambda^+ \subseteq \lambda^{++}$ is conservative, then every λ^+-representable function is λ^{++}-representable.

Proof.

Proof. For a defining λ^+-term F and codes $\gamma_i(x_i)$ we have

$$\lambda^+ \vdash F\gamma_1(x_1)\ldots\gamma_k(x_k) = \gamma(x) \iff \lambda^{++} \vdash F\gamma_1(x_1)\ldots\gamma_k(x_k) = \gamma(x)$$

So F also λ^+-represents. $\square$

Going up is easy (cntnd)

Lemma

Every $\lambda\Omega^{++}$-representable function is $\lambda\mathrm{Y}$-representable

Idea.

If M $\lambda\Omega^{++}$-represents a function then $\overline{M}$ $\lambda\mathrm{Y}$-represents it too, where $\overline{M}$ is obtained from M by replacing every Ω_σ by $\mathrm{Y}_\sigma(\lambda x : \sigma.\, x)$.
Note the reduction sequence

$$Y(\lambda x.x) \to (\lambda f.f(Yf))\lambda x.x \to (\lambda x.x)(Y(\lambda x.x)) \to Y(\lambda x.x)$$

$\square$

The Sierpiński type hierarchy and recursion depth

- Set

$$\mathcal{O}_o = \mathbb{O} = \{\bot \leq \top\} \qquad \mathcal{O}_{\sigma\to\tau} = \mathcal{O}_\sigma \xrightarrow{\text{mon}} \mathcal{O}_\tau$$

- Obtain semantics $\mathcal{O}[\![M]\!](\rho)$ for any of our λ-calculi, taking

$$\mathcal{O}[\![\Omega_\sigma]\!] = \bot \qquad \mathcal{O}[\![\mathrm{Y}_\sigma]\!] = \lambda f \in \mathcal{O}_{\sigma\to\sigma}.\ \bigvee_n f^n(\bot)$$

- Setting $h(\sigma)$ to be the height of $\mathcal{O}_\sigma$, we have

$$\mathcal{O}[\![\mathrm{Y}_\sigma]\!] = f^{h(\sigma)}(\bot)$$

- For any $\lambda\mathrm{Y}$-term M, let $\widetilde{M}$ be the $\lambda\Omega^+$-term obtained by replacing every Y_σ in M by $\lambda f.f^{h(\sigma)}(\Omega_{\sigma\to\sigma} f)$.
  Note that
  a) $\mathcal{O}[\![M]\!] = \mathcal{O}[\![\widetilde{M}]\!]$
  b) $\lambda\mathrm{Y} \vdash M = \widetilde{M}[\mathrm{Y}_\sigma/\Omega_{\sigma\to\sigma}]$
- Then, as we shall see, $\widetilde{M}$ represents any function M does.

Detecting proper normal forms (pnfs)

Long $\beta\eta$-normal $\lambda\Omega^{++}$- forms.

$$\lambda x_1 : \sigma_1 \ldots x_k : \sigma_k . x_i M_1 \ldots M_l$$

$$\lambda x_1 : \sigma_1 \ldots x_k : \sigma_k . \Omega_\sigma M_1 \ldots M_l$$

of type $\sigma_1 \to \ldots \to \sigma_k \to o$. (This type is written $(\sigma_1, \ldots, \sigma_k)$.) They are *proper* if they contain no Ω_σ, i.e. they are λ-terms.

We can use the Sierpiński hierarchy to detect properness.

For $\sigma = (\sigma_1, \ldots, \sigma_k)$ and $\tau = (\tau_1, \ldots, \tau_l)$ define:

$$t_\sigma \in \mathcal{O}_{(\sigma_1, \ldots, \sigma_k) \to o} \quad s_\tau \in \mathcal{O}_{(\tau_1, \ldots, \tau_l)}$$

by

$$t_\sigma(f) = f s_{\sigma_1} \ldots s_{\sigma_k} \qquad\qquad s_\tau g_1 \ldots g_l = \bigwedge_j t_{\tau_j}(g_j)$$

Setting $\sigma' = (\sigma_2, \ldots, \sigma_k)$ we have

$$t_{\sigma_1 \to \sigma'} f = t_\sigma f = f s_{\sigma_1} \ldots s_{\sigma_k} = t_{\sigma'}(f s_{\sigma_1})$$

More readably, we have: $t_{\sigma \to \tau} f = t_\tau(f s_\sigma)$.

The central lemma

Lemma

Let

$$M = \lambda f_1 \dots f_k . f_{i_0} M_1 \dots M_k : (\sigma_1, \dots, \sigma_k)$$

be a long $\beta\eta$-normal form in $\lambda\Omega^+$. Then:

$$M \text{ is proper } \iff t_\sigma(\mathcal{O}[\![M]\!]) = \top$$

Proof.

Set $\sigma_{i_0} = (\tau_1, \dots, \tau_l)$ and $N_i =_{\mathrm{def}} \lambda f_1 \dots f_k . M_i$.
For proper M we have:

$$\begin{array}{rcl}
t_\sigma(\mathcal{O}[\![M]\!]) & = & s_{\sigma_{i_0}}(\mathcal{O}[\![N_1]\!] s_{\sigma_1} \dots s_{\sigma_n}) \dots (\mathcal{O}[\![N_k]\!] s_{\sigma_1} \dots s_{\sigma_k}) \\
 & = & \bigwedge_i t_{\tau_i}(\mathcal{O}[\![N_i]\!] s_{\sigma_1} \dots s_{\sigma_k}) \\
 & = & \bigwedge_i t_{\sigma_k \to \tau_i}(\mathcal{O}[\![N_i]\!] s_{\sigma_1} \dots s_{\sigma_{k-1}}) \text{ (by remark above)} \\
 & = & \dots \\
 & = & \bigwedge_i t_{\sigma_1 \to \dots \to \sigma_k \to \tau_i}(\mathcal{O}[\![N_i]\!]) \\
 & = & \top \quad \text{(by induction hypothesis)}
\end{array}$$

□

Coming down: step 1

Lemma

Every $\lambda\mathrm{Y}$-representable function is $\lambda\Omega^+$-representable.

Proof.

In one direction, assume $\widetilde{M}$ represents. If

$$\lambda\Omega^+ \vdash \widetilde{M}A_1 \ldots A_n = A$$

for λ-terms A_i, A, then:

$$\lambda\mathrm{Y} \vdash MA_1 \ldots A_n = \widetilde{M}[Y_\sigma/\Omega_{\sigma\to\sigma}]A_1 \ldots A_n = A$$

So M represents. $\square$

Coming down:step1 (the other direction)

Suppose

$$\lambda\mathrm{Y} \vdash MA_1 \ldots A_n = A$$

Then as

$$\mathcal{O}[\![\widetilde{M}]\!] = \mathcal{O}[\![M]\!]$$

and A has a pnf we have:

$$\mathcal{O}[\![t(\widetilde{M}A_1 \ldots A_n)]\!] = \mathcal{O}[\![t(MA_1 \ldots A_n)]\!] = \mathcal{O}[\![t(A)]\!] = \top$$

So $\widetilde{M}A_1 \ldots A_n$ has a pnf say B. By the argument in the first part, as we now have

$$\lambda\Omega^+ \vdash \widetilde{M}A_1 \ldots A_n = B$$

we also have

$$\lambda\mathrm{Y} \vdash MA_1 \ldots A_n = B$$

But then we have $\lambda\mathrm{Y} \vdash B = A$ and so $\lambda\Omega^+ \vdash B = A$ and so

$$\lambda\Omega^+ \vdash \widetilde{M}A_1 \ldots A_n = A$$

as required.

From $\lambda\Omega^+$ to $\lambda\Omega$

If

$$M[\Omega_\sigma]$$

represents a partial function f then so does

$$M[\lambda x_1 : \sigma_1 \ldots x_n : \sigma_n . \Omega_o]$$

where $\sigma = (\sigma_1, \ldots, \sigma_n)$.

From $\lambda\Omega$ to $\lambda\beta\eta$

Suppose we have a $\lambda\Omega$ term M representing a function f using a coding scheme

$$\gamma_i : X_i \to \Lambda_{\sigma_i} \quad \gamma : X \to \Lambda_\sigma$$

with $\sigma = (\tau_1, \ldots, \tau_l)$.

Choose $x \in X$, and set $E = \gamma(x) : (\tau_1, \ldots, \tau_l)$.

Then M has a $\lambda\Omega$ long normal form

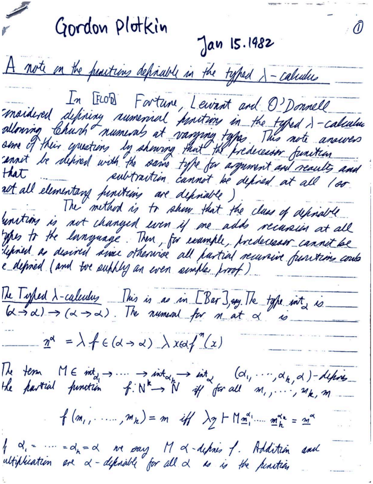
$$\lambda x_1 : \sigma_1 \ldots x_k : \sigma_k., \lambda y_1 : \tau_1 \ldots y_l : \tau_l.N[\Omega]$$

Replacing Ω by $Ey_1 \ldots y_l$ we obtain a λ-term

$$\lambda x_1 : \sigma_1 \ldots x_k : \sigma_k., \lambda y_1 : \tau_1 \ldots y_l : \tau_l.N[Ey_1 \ldots y_l]$$

representing f.

Acknowledgement



Gordon Plotkin ①

Jan 15. 1982

A note on the functions definable in the typed λ-calculus

In [FLO'D] Fortune, Leivant and O'Donnell considered defining numerical functions in the typed λ-calculus allowing Church numerals at varying types. This note answers some of their questions by showing that the predecessor function cannot be defined with the same type for argument and results and that subtraction cannot be defined at all (so not all elementary functions are definable).

The method is to show that the class of definable functions is not changed even if one adds recursion at all types to the language. Then, for example, predecessor cannot be defined as desired since otherwise all partial recursive functions could be defined (and we supply an even simple proof).

The Typed λ-calculus This is as in [Bar], say. The type int_α is $(\alpha\to\alpha)\to(\alpha\to\alpha)$. The numeral for n at α is

$$\underline{n}^\alpha = \lambda f \in (\alpha\to\alpha)\ \lambda x \in \alpha f^n(x)$$

The term $M \in int_{\alpha_1} \to \cdots \to int_{\alpha_k} \to int_\alpha$ $(\alpha_1, \ldots, \alpha_k, \alpha)$-defines the partial function $f: N^k \to N$ iff for all $m_1, \ldots, m_k, m$

$$f(m_1, \ldots, m_k) = m \text{ iff } \lambda\eta \vdash M\underline{m}_1^{\alpha_1} \ldots \underline{m}_k^{\alpha_k} = \underline{m}^\alpha$$

If $\alpha_1 = \cdots = \alpha_k = \alpha$ we say M α-defines f. Addition and multiplication are α-definable for all α as is the function

Thanks to Paweł Urzyczyn!

Happy Birthday Dana!

Thank You Dana!