Lambda-Definable Functions

Representing composition

If total function $f \in \mathbb{N}^n \to \mathbb{N}$ is represented by F and total functions $g_1, \ldots, g_n \in \mathbb{N}^m \to \mathbb{N}$ are represented by $G_1, \ldots, G_n$, then their composition $f \circ (g_1, \ldots, g_n) \in \mathbb{N}^m \to \mathbb{N}$ is represented simply by

$$\lambda x_1 \ldots x_m . F\,(G_1\,x_1 \ldots x_m) \ldots (G_n\,x_1 \ldots x_m)$$

because

$$\begin{array}{rl} & F\,(G_1\,\underline{a_1} \ldots \underline{a_m}) \ldots (G_n\,\underline{a_1} \ldots \underline{a_m}) \\ =_\beta & F\,\underline{g_1(a_1, \ldots, a_m)} \ldots \underline{g_n(a_1, \ldots, a_m)} \\ =_\beta & \underline{f(g_1(a_1, \ldots, a_m), \ldots, g_n(a_1, \ldots, a_m))} \\ = & \underline{f \circ (g_1, \ldots, g_n)(a_1, \ldots, a_m)} \end{array}$$

.

Representing composition

If total function $f \in \mathbb{N}^n \to \mathbb{N}$ is represented by F and total functions $g_1, \ldots, g_n \in \mathbb{N}^m \to \mathbb{N}$ are represented by $G_1, \ldots, G_n$, then their composition $f \circ (g_1, \ldots, g_n) \in \mathbb{N}^m \to \mathbb{N}$ is represented simply by

$$\lambda x_1 \ldots x_m . F\,(G_1\, x_1 \ldots x_m) \ldots (G_n\, x_1 \ldots x_m)$$

This does not necessarily work for partial functions. E.g. totally undefined function $u \in \mathbb{N} \rightharpoonup \mathbb{N}$ is represented by $U \triangleq \lambda x_1 . \Omega$ (why?) and $\mathtt{zero}^1 \in \mathbb{N} \to \mathbb{N}$ is represented by $Z \triangleq \lambda x_1 . \underline{0}$; but $\mathtt{zero}^1 \circ u$ is not represented by $\lambda x_1 . Z(U\, x_1)$, because $(\mathtt{zero}^1 \circ u)(n)\uparrow$ whereas $(\lambda x_1 . Z(U\, x_1))\, \underline{n} =_\beta Z\, \Omega =_\beta \underline{0}$. (What is $\mathtt{zero}^1 \circ u$ represented by?)

(see Ex. 12)

Primitive recursion

Theorem. Given $f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ and $g \in \mathbb{N}^{n+2} \rightharpoonup \mathbb{N}$, there is a unique $h \in \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}$ satisfying

$$\begin{cases} h(\vec{x}, 0) & \equiv f(\vec{x}) \\ h(\vec{x}, x+1) & \equiv g(\vec{x}, x, h(\vec{x}, x)) \end{cases}$$

for all $\vec{x} \in \mathbb{N}^n$ and $x \in \mathbb{N}$.

We write $\rho^n(f, g)$ for h and call it the partial function defined by primitive recursion from f and g.

Representing primitive recursion

If $f \in \mathbb{N}^n \to \mathbb{N}$ is represented by a λ-term F and

$g \in \mathbb{N}^{n+2} \to \mathbb{N}$ is represented by a λ-term G,

we want to show λ-definability of the unique
$h \in \mathbb{N}^{n+1} \to \mathbb{N}$ satisfying

$$\begin{cases} h(\vec{a}, 0) & = f(\vec{a}) \\ h(\vec{a}, a+1) & = g(\vec{a}, a, h(\vec{a}, a)) \end{cases}$$

or equivalently

$$\begin{aligned} h(\vec{a}, a) = \mathit{if}\ a = 0\ \mathit{then}\ & f(\vec{a}) \\ \mathit{else}\ & g(\vec{a}, a-1, h(\vec{a}, a-1)) \end{aligned}$$

Representing primitive recursion

If $f \in \mathbb{N}^n \to \mathbb{N}$ is represented by a λ-term F and

$g \in \mathbb{N}^{n+2} \to \mathbb{N}$ is represented by a λ-term G,

we want to show λ-definability of the unique

$h \in \mathbb{N}^{n+1} \to \mathbb{N}$ satisfying $\boxed{h = \Phi_{f,g}(h)}$

where $\Phi_{f,g} \in (\mathbb{N}^{n+1} \to \mathbb{N}) \to (\mathbb{N}^{n+1} \to \mathbb{N})$ is given by

$$\begin{aligned}\Phi_{f,g}(h)(\vec{a}, a) \triangleq\ & \mathit{if}\ a = 0\ \mathit{then}\ f(\vec{a}) \\ & \mathit{else}\ g(\vec{a}, a-1, h(\vec{a}, a-1))\end{aligned}$$

Representing primitive recursion

If $f \in \mathbb{N}^n \to \mathbb{N}$ is represented by a λ-term F and

$g \in \mathbb{N}^{n+2} \to \mathbb{N}$ is represented by a λ-term G,

we want to show λ-definability of the unique

$h \in \mathbb{N}^{n+1} \to \mathbb{N}$ satisfying $\boxed{h = \Phi_{f,g}(h)}$

where $\Phi_{f,g} \in (\mathbb{N}^{n+1} \to \mathbb{N}) \to (\mathbb{N}^{n+1} \to \mathbb{N})$ is given by...

Strategy:

- show that $\Phi_{f,g}$ is λ-definable;
- show that we can solve fixed point equations $\boxed{X = M\,X}$ up to β-conversion in the λ-calculus.

Representing booleans

$$
\begin{aligned}
\textbf{True} &\triangleq \lambda x\, y.\, x \\
\textbf{False} &\triangleq \lambda x\, y.\, y \\
\textbf{If} &\triangleq \lambda f\, x\, y.\, f\, x\, y
\end{aligned}
$$

satisfy

- $\textbf{If}\ \textbf{True}\ M\ N =_\beta \textbf{True}\ M\ N =_\beta M$
- $\textbf{If}\ \textbf{False}\ M\ N =_\beta \textbf{False}\ M\ N =_\beta N$

Representing test-for-zero

$$\mathbf{Eq}_0 \triangleq \lambda x.\, x(\lambda y.\,\mathsf{False})\,\mathsf{True}$$

satisfies

- $$\begin{aligned} \mathsf{Eq}_0\,\underline{0} &=_\beta \underline{0}\,(\lambda y.\,\mathsf{False})\,\mathsf{True} \\ &=_\beta \mathsf{True} \end{aligned}$$
- $$\begin{aligned} \mathsf{Eq}_0\,\underline{n+1} &=_\beta \underline{n+1}\,(\lambda y.\,\mathsf{False})\,\mathsf{True} \\ &=_\beta (\lambda y.\,\mathsf{False})^{n+1}\,\mathsf{True} \\ &=_\beta (\lambda y.\,\mathsf{False})((\lambda y.\,\mathsf{False})^{n}\,\mathsf{True}) \\ &=_\beta \mathsf{False} \end{aligned}$$

Representing predecessor

Want λ-term **Pred** satisfying

$$\begin{aligned} \mathbf{Pred}\,\underline{n+1} &=_\beta \underline{n} \\ \mathbf{Pred}\,\underline{0} &=_\beta \underline{0} \end{aligned}$$

Have to show how to reduce the "$n+1$-iterator" $\underline{n+1}$ to the "n-iterator" $\underline{n}$.

Idea: given f, iterating the function

$$g_f : (x, y) \mapsto (f(x), x)$$

$n+1$ times starting from (x, x) gives the pair $(f^{n+1}(x), f^n(x))$. So we can get $f^n(x)$ from $f^{n+1}(x)$ *parametrically in f and x*, by building g_f from f, iterating $n+1$ times from (x, x) and then taking the second component.

Hence...

Representing ordered pairs

$$\begin{aligned}
\textsf{Pair} &\triangleq \lambda x\, y\, f.\, f\, x\, y \\
\textsf{Fst} &\triangleq \lambda f.\, f\, \textsf{True} \\
\textsf{Snd} &\triangleq \lambda f.\, f\, \textsf{False}
\end{aligned}$$

satisfy

- $$\begin{aligned}
\textsf{Fst}(\textsf{Pair}\, M\, N) &=_\beta \textsf{Fst}(\lambda f.\, f\, M\, N) \\
&=_\beta (\lambda f.\, f\, M\, N)\, \textsf{True} \\
&=_\beta \textsf{True}\, M\, N \\
&=_\beta M
\end{aligned}$$
- $\textsf{Snd}(\textsf{Pair}\, M\, N) =_\beta \cdots =_\beta N$

Representing predecessor

Want $\boldsymbol{\lambda}$-term **Pred** satisfying

$$\begin{aligned} \mathbf{Pred}\,\underline{n+1} &=_\beta \underline{n} \\ \mathbf{Pred}\,\underline{0} &=_\beta \underline{0} \end{aligned}$$

$$\mathbf{Pred} \triangleq \lambda y\, f\, x.\, \mathbf{Snd}(y\,(G\,f)\,(\mathbf{Pair}\,x\,x))$$

where

$$G \triangleq \lambda f\, p.\, \mathbf{Pair}(f\,(\mathbf{Fst}\,p))\,(\mathbf{Fst}\,p)$$

has the required $\boldsymbol{\beta}$-reduction properties.

Show

$$(\forall n \in \mathbb{N})\ \underline{n+1}(Gf)(\mathrm{Pair}\ x\, x) =_\beta \mathrm{Pair}\,(\underline{n+1}\, f x)(\underline{n}\, f x)$$

by induction on $n \in \mathbb{N}$:

Base case $n = 0$:

$$\begin{aligned}\underline{1}(Gf)(\mathrm{Pair}\, x\, x) &=_\beta Gf(\mathrm{Pair}\, x\, x)\\ &=_\beta \mathrm{Pair}\,(fx)\ x\\ &=_\beta \mathrm{Pair}\,(\underline{1}\, f x)(\underline{0}\, f x)\end{aligned}$$

✓

Show

$$(\forall n \in \mathbb{N})\ \underline{n+1}\,(Gf)(\mathrm{Pair}\ x\,x) =_\beta \mathrm{Pair}\,(\underline{n+1}\ f x)(\underline{n}\ f x)$$

by induction on $n \in \mathbb{N}$:

Induction step :

$$\underline{n+2}\,(Gf)(\mathrm{Pair}\ x\,x) =_\beta (Gf)\ \underline{n+1}\,(Gf)(\mathrm{Pair}\ x\,x)$$

by ind. hyp.

$$=_\beta (Gf)\ \mathrm{Pair}\,(\underline{n+1}\ f x)(\underline{n}\ f x)$$

Show

$$(\forall n \in \mathbb{N})\ \underline{n+1}\,(Gf)(\mathrm{Pair}\,x\,x) =_\beta \mathrm{Pair}\,(\underline{n+1}\,f\,x)(\underline{n}\,f\,x)$$

by induction on $n \in \mathbb{N}$:

Induction step:

$$\underline{n+2}\,(Gf)(\mathrm{Pair}\,x\,x) =_\beta (Gf)\,\underline{n+1}\,(Gf)(\mathrm{Pair}\,x\,x)$$

by ind. hyp.

$$=_\beta (Gf)\,\mathrm{Pair}\,(\underline{n+1}\,f\,x)(\underline{n}\,f\,x)$$

$$=_\beta \mathrm{Pair}\,(f(\underline{n+1}\,f\,x))(\underline{n+1}\,f\,x)$$

$$=_\beta \mathrm{Pair}\,(\underline{n+2}\,f\,x)(\underline{n+1}\,f\,x) \quad \checkmark$$

Show

$(\forall n \in \mathbb{N})\ \underline{n+1}(Gf)(\mathrm{Pair}\,x\,x) =_\beta \mathrm{Pair}\,(\underline{n+1}\,f\,x)(\underline{n}\,f\,x)$

So $\quad \mathrm{Pred}\,\underline{n+1} =_\beta \lambda f x.\,\mathrm{Snd}(\underline{n+1}\,(Gf)(\mathrm{Pair}\,x\,x))$

$\qquad =_\beta \lambda f x.\,\mathrm{Snd}\,(\mathrm{Pair}\,(\underline{n+1}\,f\,x)(\underline{n}\,f\,x))$

$$\mathrm{Pred}\ \underline{n+1} =_\beta \lambda fx.\,\mathrm{Snd}(\underline{n+1}\,(Gf)(\mathrm{Pair}\,x\,x))$$

$$=_\beta \lambda fx.\,\mathrm{Snd}(\mathrm{Pair}(\underline{n+1}\,fx)(\underline{n}\,fx))$$

$$=_\beta \lambda fx.\,\underline{n}\,fx$$

$$=_\beta \lambda fx.\,f^n x$$

$$= \underline{n}$$

Curry's fixed point combinator $\mathbf{Y}$

$$\mathbf{Y} \triangleq \lambda f.\,(\lambda x.\, f\,(x\,x))\,(\lambda x.\, f\,(x\,x))$$

$$\boxed{\mathbf{Y}\,M =_\beta M\,(\mathbf{Y}\,M)}$$

Origins of Y

Naive set theory	λ calculus
Russell set : $R \triangleq \{x \mid \neg(x \in x)\}$	$R \triangleq \lambda x.\, \text{not}\,(x\, x)$
	$\boxed{\text{not} \triangleq \lambda b.\, \text{If}\ b\ \text{False}\ \text{True}}$

Origins of Y

Naive set theory	λ calculus
Russell set : $R \triangleq \{x \mid \neg(x \in x)\}$	$R \triangleq \lambda x.\ \mathrm{not}\,(x\,x)$
Russell's Paradox : $R \in R \Leftrightarrow \neg(R \in R)$	$R\,R =_\beta \mathrm{not}\,(R\,R)$

Origins of Y

Naive set theory	λ calculus
Russell set : $R \triangleq \{x \mid \neg(x \in x)\}$	$R \triangleq \lambda x.\, \mathrm{not}\,(x\,x)$
Russell's Paradox : $R \in R \Leftrightarrow \neg(R \in R)$	$RR =_\beta \mathrm{not}\,(RR)$

$$Y\,\mathrm{not} =_\beta RR = (\lambda x.\, \mathrm{not}\,(x x))(\lambda x.\, \mathrm{not}\,(x x))$$

Origins of Y

Naive set theory	λ calculus
Russell set : $R \triangleq \{x \mid \neg(x \in x)\}$	$R \triangleq \lambda x.\, \mathrm{not}(x\,x)$
Russell's Paradox : $R \in R \Leftrightarrow \neg(R \in R)$	$RR =_\beta \mathrm{not}(R\,R)$

$$Y\mathrm{not} =_\beta RR = (\lambda x.\, \mathrm{not}(xx))(\lambda x.\, \mathrm{not}(xx))$$

$$Yf = (\lambda x.\, f(xx))(\lambda x.\, f(xx))$$

$$Y = \lambda f.\, (\lambda x.\, f(xx))(\lambda x.\, f(xx))$$

Curry's fixed point combinator $\mathbf{Y}$

$$\mathbf{Y} \triangleq \lambda f.\,(\lambda x.\, f(x\,x))(\lambda x.\, f(x\,x))$$

satisfies $\mathbf{Y}\,M \quad \to \quad (\lambda x.\, M(x\,x))(\lambda x.\, M(x\,x))$

Curry's fixed point combinator $\mathbf{Y}$

$$\mathbf{Y} \triangleq \lambda f. (\lambda x. f(x\,x))(\lambda x. f(x\,x))$$

satisfies
$$\begin{aligned} \mathbf{Y}\,M &\to (\lambda x. M(x\,x))(\lambda x. M(x\,x)) \\ &\to M((\lambda x. M(x\,x))(\lambda x. M(x\,x))) \end{aligned}$$

hence $\mathbf{Y}\,M \twoheadrightarrow M((\lambda x. M(x\,x))(\lambda x. M(x\,x))) \twoheadleftarrow M(\mathbf{Y}\,M)$.

So for all λ-terms M we have

$$\boxed{\mathbf{Y}\,M =_\beta M(\mathbf{Y}\,M)}$$

Turing's fixed point combinator

$$\Theta \triangleq A\,A$$

where $A \triangleq \lambda xy.\, y(xxy)$

Turing's fixed point combinator

$$\Theta \triangleq A\,A$$

where $A \triangleq \lambda xy.\, y(xxy)$

$$\Theta M = AAM = (\lambda xy.\, y(xxy))\,A\,M$$

Turing's fixed point combinator

$$\Theta \triangleq AA$$

where $A \triangleq \lambda xy.\, y(xxy)$

$$\Theta M = AAM = (\lambda xy.\, y(xxy))AM$$

$$\twoheadrightarrow M(AAM)$$

$$= M(\Theta M)$$

Representing primitive recursion

If $f \in \mathbb{N}^n \to \mathbb{N}$ is represented by a λ-term F and

$g \in \mathbb{N}^{n+2} \to \mathbb{N}$ is represented by a λ-term G,

we want to show λ-definability of the unique
$h \in \mathbb{N}^{n+1} \to \mathbb{N}$ satisfying $\boxed{h = \Phi_{f,g}(h)}$

where $\Phi_{f,g} \in (\mathbb{N}^{n+1} \to \mathbb{N}) \to (\mathbb{N}^{n+1} \to \mathbb{N})$ is given by

$$\begin{aligned} \Phi_{f,g}(h)(\vec{a}, a) \triangleq\ & \mathit{if}\ a = 0\ \mathit{then}\ f(\vec{a}) \\ & \mathit{else}\ g(\vec{a}, a-1, h(\vec{a}, a-1)) \end{aligned}$$

We now know that h can be represented by

$Y(\lambda z\vec{x}x.\,\mathbf{If}(\mathbf{Eq}_0\, x)(F\,\vec{x})(G\,\vec{x}\,(\mathbf{Pred}\,x)(z\,\vec{x}\,(\mathbf{Pred}\,x))))$.

Representing primitive recursion

Recall that the class **PRIM** of primitive recursive functions is the smallest collection of (total) functions containing the basic functions and closed under the operations of composition and primitive recursion.

Combining the results about λ-definability so far, we have: **every $f \in$ PRIM is λ-definable**.

So for λ-definability of all recursive functions, we just have to consider how to represent minimization. Recall...

Minimization

Given a partial function $f \in \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}$, define $\mu^n f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ by

$$\mu^n f(\vec{x}) \triangleq \begin{array}{l} \text{least } x \text{ such that } f(\vec{x}, x) = 0 \text{ and} \\ \text{for each } i = 0, \ldots, x-1,\ f(\vec{x}, i) \\ \text{is defined and } > 0 \\ \text{(undefined if there is no such } x\text{)} \end{array}$$

So $\mu^n f(\vec{x}) = g(\vec{x}, 0)$ where in general $g(\vec{x}, x)$ satisfies

$$g(\vec{x}, x) = \text{if } f(\vec{x}, x) = 0 \text{ then } x$$
$$\text{else } g(\vec{x}, x+1)$$

Minimization

Given a partial function $f \in \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}$, define $\mu^n f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ by

$$\mu^n f(\vec{x}) \triangleq \text{least } x \text{ such that } f(\vec{x}, x) = 0 \text{ and for each } i = 0, \ldots, x-1,\ f(\vec{x}, i) \text{ is defined and } > 0 \text{ (undefined if there is no such } x\text{)}$$

Can express $\mu^n f$ in terms of a fixed point equation:

$\mu^n f(\vec{x}) \equiv g(\vec{x}, 0)$ where g satisfies $\boxed{g = \Psi_f(g)}$
with $\Psi_f \in (\mathbb{N}^{n+1} \rightharpoonup \mathbb{N}) \to (\mathbb{N}^{n+1} \rightharpoonup \mathbb{N})$ defined by

$$\Psi_f(g)(\vec{x}, x) \equiv \mathit{if}\ f(\vec{x}, x) = 0\ \mathit{then}\ x\ \mathit{else}\ g(\vec{x}, x+1)$$

Representing minimization

Suppose $f \in \mathbb{N}^{n+1} \to \mathbb{N}$ (totally defined function) satisfies $\forall \vec{a}\, \exists a\, (f(\vec{a}, a) = 0)$, so that $\mu^n f \in \mathbb{N}^n \to \mathbb{N}$ is totally defined.

Thus for all $\vec{a} \in \mathbb{N}^n$, $\mu^n f(\vec{a}) = g(\vec{a}, 0)$ with $g = \Psi_f(g)$ and $\Psi_f(g)(\vec{a}, a)$ given by
$\textit{if}\ (f(\vec{a}, a) = 0)\ \textit{then}\ a\ \textit{else}\ g(\vec{a}, a + 1)$.

So if f is represented by a λ-term F, then $\mu^n f$ is represented by

$$\lambda \vec{x}.\mathbf{Y}(\lambda z\, \vec{x}\, x.\, \mathbf{If}(\mathbf{Eq}_0(F\, \vec{x}\, x))\, x\, (z\, \vec{x}\, (\mathbf{Succ}\, x)))\, \vec{x}\, \underline{0}$$

Recursive implies λ-definable

Fact: every partial recursive $f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ can be expressed in a standard form as $f = g \circ (\mu^n h)$ for some $g, h \in \mathbf{PRIM}$. (Follows from the proof that computable = partial-recursive.)

Hence every (total) recursive function is λ-definable.

More generally, every partial recursive function is λ-definable, but matching up $\uparrow$ with $\nexists \beta{-}\mathbf{nf}$ makes the representations more complicated than for total functions: see [Hindley, J.R. & Seldin, J.P. (CUP, 2008), chapter 4.]

Computable = λ-definable

Theorem. A partial function is computable if and only if it is λ-definable.

We already know that computable = partial recursive $\Rightarrow$ λ-definable. So it just remains to see that **λ-definable functions are RM computable**. To show this one can

- code λ-terms as numbers (ensuring that operations for constructing and deconstructing terms are given by RM computable functions on codes)
- write a RM interpreter for (normal order) β-reduction.

Computable = λ-definable

Theorem. A partial function is computable if and only if it is λ-definable.

We already know that computable = partial recursive $\Rightarrow$ λ-definable. So it just remains to see that **λ-definable functions are RM computable**. To show this one can

- code λ-terms as numbers (ensuring that operations for constructing and deconstructing terms are given by RM computable functions on codes)
- write a RM interpreter for (normal order) β-reduction.

Numerical coding of λ-terms

Fix an enumeration $x_0, x_1, x_2, \ldots$ of the set of variables.

For each λ-term M, define $\ulcorner M \urcorner \in \mathbb{N}$ by

$$\boxed{\begin{aligned} \ulcorner x_i \urcorner &= \ulcorner [0, i] \urcorner \\ \ulcorner \lambda x_i . M \urcorner &= \ulcorner [1, i, \ulcorner M \urcorner] \urcorner \\ \ulcorner M N \urcorner &= \ulcorner [2, \ulcorner M \urcorner, \ulcorner N \urcorner] \urcorner \end{aligned}}$$

(where $\ulcorner [n_0, n_1, \ldots, n_k] \urcorner$ is the numerical coding of lists of numbers from p 43).

Computable = λ-definable

Theorem. A partial function is computable if and only if it is λ-definable.

We already know that computable = partial recursive $\Rightarrow$ λ-definable. So it just remains to see that **λ-definable functions are RM computable**. To show this one can

- code λ-terms as numbers (ensuring that operations for constructing and deconstructing terms are given by RM computable functions on codes)
- write a RM interpreter for (normal order) β-reduction.

The details are straightforward, if tedious.

Summary

- Formalization of intuitive notion of ALGORITHM in several equivalent ways

  cf. "Church-Turing Thesis"

- Limitative results : $\begin{cases} \text{undecidable problems} \\ \text{uncomputable functions} \end{cases}$

"programs as data" + diagonalization