# Lambda-Definable Functions

# Encoding data in $\lambda$-calculus

Computation in $\lambda$-calculus is given by $\beta$-reduction. To relate this to register/Turing-machine computation, or to partial recursive functions, we first have to see how to encode numbers, pairs, lists, . . . as $\lambda$-terms.

We will use the original encoding of numbers due to Church. . .

# Church's numerals

$$
\begin{aligned}
\underline{0} &\triangleq \lambda f\, x.x \\
\underline{1} &\triangleq \lambda f\, x.f\, x \\
\underline{2} &\triangleq \lambda f\, x.f(f\, x) \\
&\vdots \\
\underline{n} &\triangleq \lambda f\, x.\underbrace{f(\cdots(f}_{n \text{ times}}\, x)\cdots)
\end{aligned}
$$

**Notation:** $\begin{cases} M^0 N & \triangleq N \\ M^1 N & \triangleq M\, N \\ M^{n+1} N & \triangleq M(M^n N) \end{cases}$

so we can write $\underline{n}$ as $\lambda f\, x.f^n x$ and we have $\boxed{\underline{n}\, M\, N =_\beta M^n\, N}$.

# Church's numerals

$$
\begin{aligned}
\underline{0} &\triangleq \lambda f\, x.x \\
\underline{1} &\triangleq \lambda f\, x.f\, x \\
\underline{2} &\triangleq \lambda f\, x.f\,(f\, x) \\
&\vdots \\
\underline{n} &\triangleq \lambda f\, x.\underbrace{f\,(\cdots(f}_{n \text{ times}}\, x)\cdots)
\end{aligned}
$$

**Notation:** $\begin{cases} M^0 N & \triangleq N \\ M^1 N & \triangleq M\, N \\ M^{n+1} N & \triangleq M(M^n N) \end{cases}$

so we can write $\underline{n}$ as $\lambda f\, x.f^n x$ and we have $\boxed{\underline{n}\, M\, N =_\beta M^n\, N}$.

# $\lambda$-Definable functions

**Definition.** $f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ is $\lambda$-definable if there is a closed $\lambda$-term $F$ that represents it: for all $(x_1, \ldots, x_n) \in \mathbb{N}^n$ and $y \in \mathbb{N}$

- if $f(x_1, \ldots, x_n) = y$, then $F\,\underline{x_1} \cdots \underline{x_n} =_\beta \underline{y}$
- if $f(x_1, \ldots, x_n)\uparrow$, then $F\,\underline{x_1} \cdots \underline{x_n}$ has no $\beta$-nf.

For example, addition is $\lambda$-definable because it is represented by $P \triangleq \lambda x_1\, x_2.\lambda f\, x.\, x_1\, f\,(x_2\, f\, x)$:

$$
\begin{aligned}
P\,\underline{m}\,\underline{n} &=_\beta \lambda f\, x.\, \underline{m}\, f\,(\underline{n}\, f\, x) \\
&=_\beta \lambda f\, x.\, \underline{m}\, f\,(f^n x) \\
&=_\beta \lambda f\, x.\, f^m(f^n x) \\
&= \lambda f\, x. f^{m+n} x \\
&= \underline{m+n}
\end{aligned}
$$

# Computable = $\lambda$-definable

**Theorem.** A partial function is computable if and only if it is $\lambda$-definable.

We already know that

$$\begin{aligned} & \text{Register Machine computable} \\ = \ & \text{Turing computable} \\ = \ & \text{partial recursive.} \end{aligned}$$

Using this, we break the theorem into two parts:

- every partial recursive function is $\lambda$-definable
- $\lambda$-definable functions are RM computable

# $\lambda$-Definable functions

**Definition.** $f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ is $\lambda$-definable if there is a closed $\lambda$-term $F$ that represents it: for all $(x_1, \ldots, x_n) \in \mathbb{N}^n$ and $y \in \mathbb{N}$

- if $f(x_1, \ldots, x_n) = y$, then $F\,\underline{x_1} \cdots \underline{x_n} =_\beta \underline{y}$
- if $f(x_1, \ldots, x_n)\uparrow$, then $F\,\underline{x_1} \cdots \underline{x_n}$ has no $\beta$-nf.

This condition can make it quite tricky to find a $\lambda$-term representing a non-total function.

For now, we concentrate on total functions. First, let us see why the elements of PRIM (primitive recursive functions) are $\lambda$-definable.

# Basic functions

- Projection functions, $\texttt{proj}_i^n \in \mathbb{N}^n \to \mathbb{N}$:

$$\texttt{proj}_i^n(x_1, \ldots, x_n) \triangleq x_i$$

- Constant functions with value $0$, $\texttt{zero}^n \in \mathbb{N}^n \to \mathbb{N}$:

$$\texttt{zero}^n(x_1, \ldots, x_n) \triangleq 0$$

- Successor function, $\texttt{succ} \in \mathbb{N} \to \mathbb{N}$:

$$\texttt{succ}(x) \triangleq x + 1$$

# Basic functions are representable

- $\mathtt{proj}_i^n \in \mathbb{N}^n \to \mathbb{N}$ is represented by $\lambda x_1 \ldots x_n . x_i$
- $\mathtt{zero}^n \in \mathbb{N}^n \to \mathbb{N}$ is represented by $\lambda x_1 \ldots x_n . \underline{0}$
- $\mathtt{succ} \in \mathbb{N} \to \mathbb{N}$ is represented by

$$\mathrm{Succ} \triangleq \lambda x_1\, f\, x. f\,(x_1\, f\, x)$$

since

$$\begin{aligned} \mathrm{Succ}\,\underline{n} &=_\beta \lambda f\, x.\, f\,(\underline{n}\, f\, x) \\ &=_\beta \lambda f\, x.\, f\,(f^n\, x) \\ &= \lambda f\, x.\, f^{n+1}\, x \\ &= \underline{n+1} \end{aligned}$$

# Basic functions are representable

- $\mathtt{proj}_i^n \in \mathbb{N}^n \to \mathbb{N}$ is represented by $\lambda x_1 \ldots x_n . x_i$
- $\mathtt{zero}^n \in \mathbb{N}^n \to \mathbb{N}$ is represented by $\lambda x_1 \ldots x_n . \underline{0}$
- $\mathtt{succ} \in \mathbb{N} \to \mathbb{N}$ is represented by

$$\mathrm{Succ} \triangleq \lambda x_1\, f\, x . f\,(x_1\, f\, x)$$

since

$$\begin{aligned} \mathrm{Succ}\, \underline{n} &=_\beta \lambda f\, x.\, f\,(\underline{n}\, f\, x) \\ &=_\beta \lambda f\, x.\, f\,(f^n\, x) \\ &= \lambda f\, x.\, f^{n+1}\, x \\ &= \underline{n+1} \end{aligned}$$

$\lambda x_1\, f\, x . x_1 f\,(f x)$ also represents $\mathtt{succ}$

# 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

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

.

# 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?)

# 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 $\lambda$-term $F$ and $g \in \mathbb{N}^{n+2} \to \mathbb{N}$ is represented by a $\lambda$-term $G$, we want to show $\lambda$-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 $\lambda$-term $F$ and
$g \in \mathbb{N}^{n+2} \to \mathbb{N}$ is represented by a $\lambda$-term $G$,
we want to show $\lambda$-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 $\lambda$-term $F$ and
$g \in \mathbb{N}^{n+2} \to \mathbb{N}$ is represented by a $\lambda$-term $G$,
we want to show $\lambda$-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 $\lambda$-definable;
- show that we can solve fixed point equations $\boxed{X = M\,X}$ up to $\beta$-conversion in the $\lambda$-calculus.

# Representing booleans

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

satisfy

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

# Representing test-for-zero

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

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 ordered pairs

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

satisfy

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

# Representing predecessor

Want $\lambda$-term Pred satisfying

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

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 predecessor

Want $\lambda$-term Pred satisfying

$$\begin{array}{lll} \text{Pred}\ \underline{n+1} & =_\beta & \underline{n} \\ \text{Pred}\ \underline{0} & =_\beta & \underline{0} \end{array}$$

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

where

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

has the required $\beta$-reduction properties. [Exercise]

# Representing primitive recursion

If $f \in \mathbb{N}^n \to \mathbb{N}$ is represented by a $\lambda$-term $F$ and
$g \in \mathbb{N}^{n+2} \to \mathbb{N}$ is represented by a $\lambda$-term $G$,
we want to show $\lambda$-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 $\lambda$-definable;
- show that we can solve fixed point equations $\boxed{X = M\,X}$ up to $\beta$-conversion in the $\lambda$-calculus.

# Curry's fixed point combinator Y

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

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

$\to M((\lambda x.\, M(x\,x))(\lambda x.\, M(x\,x)))$

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

So for all $\lambda$-terms $M$ we have

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

# Representing primitive recursion

If $f \in \mathbb{N}^n \to \mathbb{N}$ is represented by a $\lambda$-term $F$ and $g \in \mathbb{N}^{n+2} \to \mathbb{N}$ is represented by a $\lambda$-term $G$, we want to show $\lambda$-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 $\lambda$-term $F$ and
$g \in \mathbb{N}^{n+2} \to \mathbb{N}$ is represented by a $\lambda$-term $G$,
we want to show $\lambda$-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
$Y(\lambda z \vec{x} x.\, \mathrm{If}(\mathrm{Eq}_0\, x)(F\, \vec{x})(G\, \vec{x}\, (\mathrm{Pred}\, x)(z\, \vec{x}\, (\mathrm{Pred}\, x))))$.

# Representing primitive recursion

Recall that the class $\mathrm{PRIM}$ 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 $\lambda$-definability so far, we have: **every $f \in \mathrm{PRIM}$ is $\lambda$-definable**.

So for $\lambda$-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

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

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 *if* $(f(\vec{a}, a) = 0)$ *then* $a$ *else* $g(\vec{a}, a+1)$.
So if $f$ is represented by a $\lambda$-term $F$, then $\mu^n f$ is represented by

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

# Recursive implies $\lambda$-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 $g, h \in \mathrm{PRIM}$. (Follows from the proof that computable = partial-recursive.)

Hence every (total) recursive function is $\lambda$-definable.

More generally, every partial recursive function is $\lambda$-definable, but matching up $\uparrow$ with $\not\exists \beta\mathrm{-nf}$ makes the representations more complicated than for total functions: see [Hindley, J.R. & Seldin, J.P. (CUP, 2008), chapter 4.]

# Computable = $\lambda$-definable

**Theorem.** A partial function is computable if and only if it is $\lambda$-definable.

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

- code $\lambda$-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) $\beta$-reduction.

The details are straightforward, if tedious.

# Summary

# Summary

- Formalization of intuitive notion of *algorithm* in several *equivalent* ways.

# Summary

- Formalization of intuitive notion of *algorithm* in several *equivalent* ways.
  *Church-Turing Thesis*

# Summary

- Formalization of intuitive notion of *algorithm* in several *equivalent* ways.
  *Church-Turing Thesis*
- Limitative results:
  - undecidable problems
  - uncomputable functions

# Summary

- Formalization of intuitive notion of *algorithm* in several *equivalent* ways.
  *Church-Turing Thesis*
- Limitative results:
  - undecidable problems
  - uncomputable functions

  *"programs as data" and diagonalization*