

# 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

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

# Church's numerals

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

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  $\underline{n} M N =_{\beta} M^n N$ .

# Church's numerals

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

N.B. not  $ffx$ , which stands for  $(ff)x$

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  $\underline{n} M N =_{\beta} M^n N$ .

# $\lambda$ -Definable functions

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

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

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

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

# $\lambda$ -Definable functions

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

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

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

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

can  
prove  
this equality  
by induction  
on  $n$

# Computable = $\lambda$ -definable

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

We already know that

Register Machine computable  
= Turing computable  
= partial recursive.

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

- ▶ if  $f(x_1, \dots, x_n) = y$ , then  $F \underline{x_1} \cdots \underline{x_n} =_{\beta} \underline{y}$
- ▶ if  $f(x_1, \dots, x_n) \uparrow$ , 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,  $\text{proj}_i^n \in \mathbb{N}^n \rightarrow \mathbb{N}$ :

$$\text{proj}_i^n(x_1, \dots, x_n) \triangleq x_i$$

- ▶ **Constant** functions with value  $\mathbf{0}$ ,  $\text{zero}^n \in \mathbb{N}^n \rightarrow \mathbb{N}$ :

$$\text{zero}^n(x_1, \dots, x_n) \triangleq \mathbf{0}$$

- ▶ **Successor** function,  $\text{succ} \in \mathbb{N} \rightarrow \mathbb{N}$ :

$$\text{succ}(x) \triangleq x + \mathbf{1}$$

# Basic functions are representable

- ▶  $\text{proj}_i^n \in \mathbb{N}^n \rightarrow \mathbb{N}$  is represented by  $\lambda x_1 \dots x_n. x_i$
- ▶  $\text{zero}^n \in \mathbb{N}^n \rightarrow \mathbb{N}$  is represented by  $\lambda x_1 \dots x_n. \underline{0}$
- ▶  $\text{succ} \in \mathbb{N} \rightarrow \mathbb{N}$  is represented by

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

since

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

# Basic functions are representable

- ▶  $\text{proj}_i^n \in \mathbb{N}^n \rightarrow \mathbb{N}$  is represented by  $\lambda x_1 \dots x_n. x_i$
- ▶  $\text{zero}^n \in \mathbb{N}^n \rightarrow \mathbb{N}$  is represented by  $\lambda x_1 \dots x_n. \underline{0}$
- ▶  $\text{succ} \in \mathbb{N} \rightarrow \mathbb{N}$  is represented by

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

since

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

( $\lambda x_1 f x. x_1 f(fx)$  also represents  $\text{succ}$ )

# Representing composition

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

$$\lambda x_1 \dots x_m. F (G_1 x_1 \dots x_m) \dots (G_n x_1 \dots x_m)$$

because

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

# Representing composition

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

$$\lambda x_1 \dots x_m. F (G_1 x_1 \dots x_m) \dots (G_n x_1 \dots x_m)$$

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

(see Ex. 12)

# Primitive recursion

**Theorem.** Given  $f \in \mathbb{N}^n \rightarrow \mathbb{N}$  and  $g \in \mathbb{N}^{n+2} \rightarrow \mathbb{N}$ , there is a unique  $h \in \mathbb{N}^{n+1} \rightarrow \mathbb{N}$  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  $f \in \mathbb{N}^n \rightarrow \mathbb{N}$  is represented by a  $\lambda$ -term  $F$  and  $g \in \mathbb{N}^{n+2} \rightarrow \mathbb{N}$  is represented by a  $\lambda$ -term  $G$ , we want to show  $\lambda$ -definability of the unique  $h \in \mathbb{N}^{n+1} \rightarrow \mathbb{N}$  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

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



# Representing primitive recursion

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

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

we want to show  $\lambda$ -definability of the unique

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

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

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

# Representing primitive recursion

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

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

we want to show  $\lambda$ -definability of the unique

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

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

## Strategy:

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

# Representing booleans

**True**  $\triangleq$   $\lambda x y. x$

**False**  $\triangleq$   $\lambda x y. y$

**If**  $\triangleq$   $\lambda f x y. f x y$

satisfy

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

# Representing test-for-zero

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

satisfies

- ▶  $\mathbf{Eq}_0 \underline{0} =_{\beta} \underline{0} (\lambda y. \mathbf{False}) \mathbf{True}$   
 $=_{\beta} \mathbf{True}$
- ▶  $\mathbf{Eq}_0 \underline{n + 1} =_{\beta} \underline{n + 1} (\lambda y. \mathbf{False}) \mathbf{True}$   
 $=_{\beta} (\lambda y. \mathbf{False})^{n+1} \mathbf{True}$   
 $=_{\beta} (\lambda y. \mathbf{False}) ((\lambda y. \mathbf{False})^n \mathbf{True})$   
 $=_{\beta} \mathbf{False}$

# Representing predecessor

Want  $\lambda$ -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

**Pair**  $\triangleq \lambda x y f. f x y$

**Fst**  $\triangleq \lambda f. f \text{ True}$

**Snd**  $\triangleq \lambda f. f \text{ False}$

satisfy

- ▶ **Fst(Pair M N)**  $=_{\beta}$  **Fst**( $\lambda f. f M N$ )  
 $=_{\beta}$  ( $\lambda f. f M N$ ) **True**  
 $=_{\beta}$  **True M N**  
 $=_{\beta}$  **M**
- ▶ **Snd(Pair M N)**  $=_{\beta} \cdots =_{\beta}$  **N**

# Representing predecessor

Want  $\lambda$ -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  $\beta$ -reduction properties.

Show

$$(\forall n \in \mathbb{N}) \ \underline{n+1}(Gf)(\text{Pair } xx) =_{\beta} \text{Pair}(\underline{n+1}fx)(\underline{n}fx)$$

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

Base case  $n=0$  :

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





Show

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

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

Induction step :

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

*by ind. hyp.*

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

Show

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

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

Induction step :

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

by ind. hyp.

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

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

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

Show

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

So

$$\text{Pred } \underline{n+1} =_{\beta} \lambda f x. \text{Snd} (\underline{n+1} (Gf) (\text{Pair } x x))$$
$$\rightarrow =_{\beta} \lambda f x. \text{Snd} (\text{Pair } (\underline{n+1} f x) (\underline{n} f x))$$

$$\begin{aligned}
\text{Pred } \underline{n+1} &=_{\beta} \lambda f x. \text{Snd}(\underline{n+1} (Gf) (\text{Pair } x x)) \\
&=_{\beta} \lambda f x. \text{Snd}(\text{Pair}(\underline{n+1} f x)(\underline{n} f x)) \\
&=_{\beta} \lambda f x. \underline{n} f x \\
&=_{\beta} \lambda f x. f^n x \\
&=_{\beta} \lambda x. x
\end{aligned}$$

# Representing primitive recursion

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

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

we want to show  $\lambda$ -definability of the unique

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

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

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

# Representing primitive recursion

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

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

we want to show  $\lambda$ -definability of the unique

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

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

**Strategy:**

► show that  $\Phi_{f,g}$  is  $\lambda$ -definable;

$\lambda z \vec{x} x. If (Eq_0 x) (F \vec{x}) (G \vec{x} (Pred x) (z \vec{x} (Pred x)))$