PHIL 222
Philosophical Foundations of Computer Science
Week 4, Thursday

Sept. 19, 2024

The Lambda Calculus
(cont'd)

A term of the lambda calculus is either

- a basic or "atomic" term, i.e., one of the variables $x, y, z, \ldots$ or the constants $a, b, c, \ldots$,
- (MN), obtained by the "application" of other terms M and N, or
- obtained from other terms by "abstraction".

A term of the lambda calculus is either

- a basic or "atomic" term, i.e., one of the variables $x, y, z, \ldots$ or the constants $a, b, c, \ldots,$
- (MN), obtained by the "application" of other terms M and N, or
- obtained from other terms by "abstraction".

Lambda abstraction.

is the function $f : x \mapsto M$, i.e., the f defined by $f(x) = M$.

A term of the lambda calculus is either

- a basic or "atomic" term, i.e., one of the variables $x, y, z, \ldots$ or the constants $a, b, c, \ldots,$
- (MN), obtained by the "application" of other terms M and N, or
- obtained from other terms by "abstraction".

Lambda abstraction.

$$x \;\text{——}\; \boxed{(\lambda x.\, M)} \;\text{——}\; M$$

is the function $f : x \mapsto M$, i.e., the f defined by $f(x) = M$.

A term of the lambda calculus is either

- a basic or "atomic" term, i.e., one of the variables $x, y, z, \ldots$ or the constants $a, b, c, \ldots,$
- (MN), obtained by the "application" of other terms M and N, or
- obtained from other terms by "abstraction".

Lambda abstraction.

$$x \;\text{---}\; \boxed{(\lambda x.\, M)} \;\text{---}\; M$$

is the function $f : x \mapsto M$, i.e., the f defined by $f(x) = M$.

E.g.,

$$x \;\text{---}\; \boxed{(\lambda x.\, (x + 3))} \;\text{---}\; x + 3$$

A term of the lambda calculus is either

- a basic or "atomic" term, i.e., one of the variables $x, y, z, \ldots$ or the constants $a, b, c, \ldots,$
- (MN), obtained by the "application" of other terms M and N, or
- obtained from other terms by "abstraction".

Lambda abstraction.

$$x \longrightarrow \boxed{(\lambda x.\, M)} \longrightarrow M$$

is the function $f : x \mapsto M$, i.e., the f defined by $f(x) = M$.

E.g.,

$$x \longrightarrow \boxed{(\lambda x.\, (x + 3))} \longrightarrow x + 3$$

What is the result of applying this function to 2?

A term of the lambda calculus is either

- a basic or "atomic" term, i.e., one of the variables $x, y, z, \ldots$ or the constants $a, b, c, \ldots,$
- (MN), obtained by the "application" of other terms M and N, or
- obtained from other terms by "abstraction".

Lambda abstraction.

$$x \longrightarrow \boxed{(\lambda x.\, M)} \longrightarrow M$$

is the function $f : x \mapsto M$, i.e., the f defined by $f(x) = M$.

E.g.,

$$x \longrightarrow \boxed{(\lambda x.\, (x + 3))} \longrightarrow x + 3$$

What is the result of applying this function to 2?

$$((\lambda x.\, (x + 3))\, 2) = \,?$$

A term of the lambda calculus is either

- a basic or "atomic" term, i.e., one of the variables $x, y, z, \ldots$ or the constants $a, b, c, \ldots,$
- (MN), obtained by the "application" of other terms M and N, or
- obtained from other terms by "abstraction".

Lambda abstraction.

$$x \;\text{—}\; \boxed{(\lambda x.\, M)} \;\text{—}\; M$$

is the function $f : x \mapsto M$, i.e., the f defined by $f(x) = M$.

E.g.,

$$x \;\text{—}\; \boxed{(\lambda x.\, (x + 3))} \;\text{—}\; x + 3$$

What is the result of applying this function to 2?

$$((\lambda x.\, (x + 3))\, 2) = \,?$$

A term of the lambda calculus is either

- a basic or “atomic” term, i.e., one of the variables $x, y, z, \ldots$ or the constants $a, b, c, \ldots,$
- (MN), obtained by the “application” of other terms M and N, or
- obtained from other terms by “abstraction”.

Lambda abstraction.

$$x \longrightarrow \boxed{(\lambda x.\, M)} \longrightarrow M$$

is the function $f : x \mapsto M$, i.e., the f defined by $f(x) = M$.

E.g.,

$$\begin{array}{c} x \\ 2 \end{array} \longrightarrow \boxed{(\lambda x.\,(x + 3))} \longrightarrow x + 3$$

What is the result of applying this function to 2?

$$((\lambda x.\,(x + 3))\, 2) = ?$$

A term of the lambda calculus is either

- a basic or "atomic" term, i.e., one of the variables $x, y, z, \ldots$ or the constants $a, b, c, \ldots,$
- (MN), obtained by the "application" of other terms M and N, or
- obtained from other terms by "abstraction".

Lambda abstraction.

$$x \longrightarrow \boxed{(\lambda x.\, M)} \longrightarrow M$$

is the function $f : x \mapsto M$, i.e., the f defined by $f(x) = M$.

E.g.,

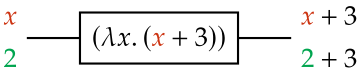
$$\begin{array}{c} x \\ 2 \end{array} \longrightarrow \boxed{(\lambda x.\, (x + 3))} \longrightarrow \begin{array}{c} x + 3 \\ 2 + 3 \end{array}$$

What is the result of applying this function to 2?

$$((\lambda x.\, (x + 3))\, 2) = 2 + 3.$$

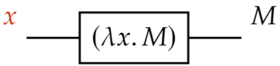
$$\begin{array}{ccc} x & & x + 3 \\ & \longrightarrow \boxed{(\lambda x.\,(x + 3))} \longrightarrow & \\ 2 & & 2 + 3 \end{array}$$

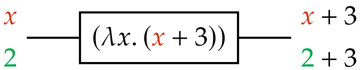
$$((\lambda x.\,(x + 3))\,2) = 2 + 3.$$



$$((\lambda x.\,(x+3))\,2) = 2+3.$$

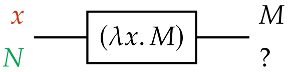
In general,





$$((\lambda x.\,(x + 3))\,2) = 2 + 3.$$

In general,



$$((\lambda x.\,M)\,N) = ?$$

$$\begin{array}{c} x \\ 2 \end{array} \;—\; \boxed{(\lambda x.\,(x+3))} \;—\; \begin{array}{c} x+3 \\ 2+3 \end{array}$$

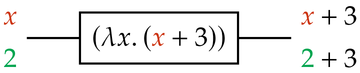
$$((\lambda x.\,(x+3))\,2) = 2+3.$$

In general,

$$\begin{array}{c} x \\ N \end{array} \;—\; \boxed{(\lambda x.\,M)} \;—\; \begin{array}{c} M \\ ? \end{array}$$

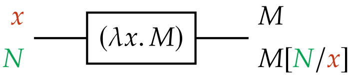
$$((\lambda x.\,M)\,N) = ?$$

where we write $M[N/x]$ for the result of substituting N for x in M (e.g., $(x+3)[2/x] = 2+3$).



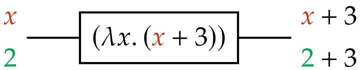
$$((\lambda x.\,(x+3))\,2) = 2+3.$$

In general,



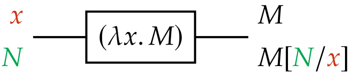
$$((\lambda x.\,M)\,N) = M[N/x]$$

where we write $M[N/x]$ for the result of substituting N for x in M (e.g., $(x+3)[2/x] = 2+3$).



$$((\lambda x.\,(x+3))\,2) = 2+3.$$

In general,



$$((\lambda x.\,M)\,N) = M[N/x]$$

where we write $M[N/x]$ for the result of substituting N for x in M (e.g., $(x+3)[2/x] = 2+3$).

***β*-reduction** is an application of this transformation

$$((\lambda x.\,M)\,N) \xrightarrow{\beta} M[N/x]$$

to terms and subterms. Computation means to keep applying this until it no longer applies.

(From now on, we omit outermost parens.)

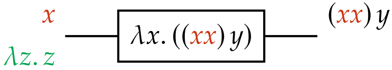
E.g.,

1. $(\lambda x. ((xx)\, y))(\lambda z. z)$

(From now on, we omit outermost parens.)

E.g.,

1. $(\lambda x.\,((xx)\,y))(\lambda z.\,z)$



(From now on, we omit outermost parens.)

E.g.,

1. $$(\lambda x.\,((xx)\,y))(\lambda z.\,z) \longrightarrow ((\lambda z.\,z)(\lambda z.\,z))\,y$$

$$\begin{array}{ccc} x & & (xx)\,y \\ & \text{—}\boxed{\lambda x.\,((xx)\,y)}\text{—} & \\ \lambda z.\,z & & ((\lambda z.\,z)(\lambda z.\,z))\,y \end{array}$$

(From now on, we omit outermost parens.)

E.g.,

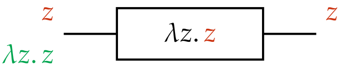
1. $$(\lambda x.\,((xx)\,y))(\lambda z.\,z) \longrightarrow ((\lambda z.\,z)(\lambda z.\,z))\,y$$

$$\begin{array}{rcl} x & & (xx)\,y \\ & \text{———}\boxed{\lambda x.\,((xx)\,y)}\text{———} & \\ \lambda z.\,z & & ((\lambda z.\,z)(\lambda z.\,z))\,y \end{array}$$

(From now on, we omit outermost parens.)

E.g.,

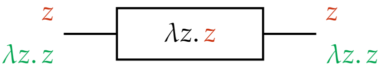
1. $$(\lambda x.\,((xx)\,y))(\lambda z.\,z) \to ((\lambda z.\,z)(\lambda z.\,z))\,y$$



(From now on, we omit outermost parens.)

E.g.,

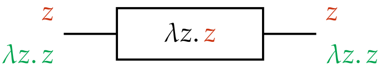
1. $(\lambda x.\,((xx)\,y))(\lambda z.\,z) \to ((\lambda z.\,z)(\lambda z.\,z))\,y \to (\lambda z.\,z)\,y$



(From now on, we omit outermost parens.)

E.g.,

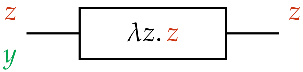
1. $$(\lambda x.\,((xx)\,y))(\lambda z.\,z) \to ((\lambda z.\,z)(\lambda z.\,z))\,y \to (\lambda z.\,z)\,y$$



(From now on, we omit outermost parens.)

E.g.,

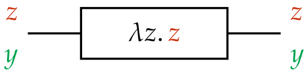
1. $(\lambda x.\, ((xx)\, y))(\lambda z.\, z) \longrightarrow ((\lambda z.\, z)(\lambda z.\, z))\, y \longrightarrow (\lambda z.\, z)\, y$



(From now on, we omit outermost parens.)

E.g.,

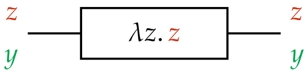
1. $(\lambda x.\,((xx)\,y))(\lambda z.\,z) \longrightarrow ((\lambda z.\,z)(\lambda z.\,z))\,y \longrightarrow (\lambda z.\,z)\,y \longrightarrow y$



(From now on, we omit outermost parens.)

E.g.,

1. $(\lambda x.\, ((xx)\, y))(\lambda z.\, z) \longrightarrow ((\lambda z.\, z)(\lambda z.\, z))\, y \longrightarrow (\lambda z.\, z)\, y \longrightarrow y$

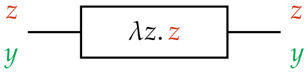


2. $(\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y))$

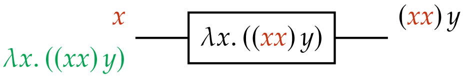
(From now on, we omit outermost parens.)

E.g.,

1. $(\lambda x.\, ((xx)\, y))(\lambda z.\, z) \longrightarrow ((\lambda z.\, z)(\lambda z.\, z))\, y \longrightarrow (\lambda z.\, z)\, y \longrightarrow y$



2. $(\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y))$



(From now on, we omit outermost parens.)

E.g.,

1. $(\lambda x.\,((xx)\,y))(\lambda z.\,z) \longrightarrow ((\lambda z.\,z)(\lambda z.\,z))\,y \longrightarrow (\lambda z.\,z)\,y \longrightarrow y$

$$\begin{array}{c} z \\ y \end{array} \;\text{—}\; \boxed{\lambda z.\,z} \;\text{—}\; \begin{array}{c} z \\ y \end{array}$$

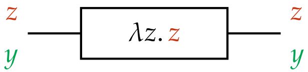
2. $(\lambda x.\,((xx)\,y))(\lambda x.\,((xx)\,y)) \longrightarrow ((\lambda x.\,((xx)\,y))(\lambda x.\,((xx)\,y)))\,y$

$$\begin{array}{r} x \\ \lambda x.\,((xx)\,y) \end{array} \;\text{—}\; \boxed{\lambda x.\,((xx)\,y)} \;\text{—}\; \begin{array}{l} (xx)\,y \\ ((\lambda x.\,((xx)\,y))(\lambda x.\,((xx)\,y)))\,y \end{array}$$

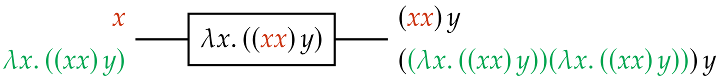
(From now on, we omit outermost parens.)

E.g.,

1. $(\lambda x.\,((xx)\,y))(\lambda z.\,z) \to ((\lambda z.\,z)(\lambda z.\,z))\,y \to (\lambda z.\,z)\,y \to y$



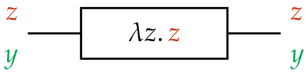
2. $(\lambda x.\,((xx)\,y))(\lambda x.\,((xx)\,y)) \to ((\lambda x.\,((xx)\,y))(\lambda x.\,((xx)\,y)))\,y$



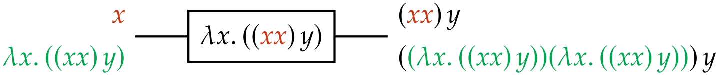
(From now on, we omit outermost parens.)

E.g.,

1. $$(\lambda x.\, ((xx)\, y))(\lambda z.\, z) \longrightarrow ((\lambda z.\, z)(\lambda z.\, z))\, y \longrightarrow (\lambda z.\, z)\, y \longrightarrow y$$



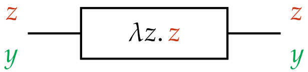
2. $$\begin{aligned}(\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y)) &\longrightarrow ((\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y)))\, y \\ &\longrightarrow (((\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y)))\, y)\, y\end{aligned}$$



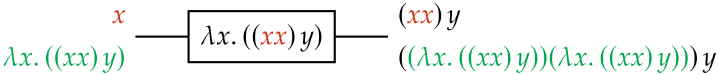
(From now on, we omit outermost parens.)

E.g.,

1. $$(\lambda x.\,((xx)\,y))(\lambda z.\,z) \longrightarrow ((\lambda z.\,z)(\lambda z.\,z))\,y \longrightarrow (\lambda z.\,z)\,y \longrightarrow y$$



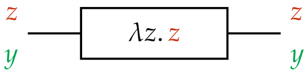
2. $$\begin{aligned}(\lambda x.\,((xx)\,y))(\lambda x.\,((xx)\,y)) &\longrightarrow ((\lambda x.\,((xx)\,y))(\lambda x.\,((xx)\,y)))\,y \\ &\longrightarrow (((\lambda x.\,((xx)\,y))(\lambda x.\,((xx)\,y)))\,y)\,y\end{aligned}$$



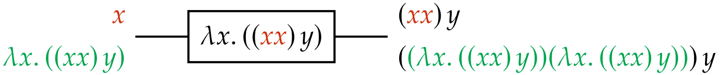
(From now on, we omit outermost parens.)

E.g.,

1. $$(\lambda x.\, ((xx)\, y))(\lambda z.\, z) \to ((\lambda z.\, z)(\lambda z.\, z))\, y \to (\lambda z.\, z)\, y \to y$$



2. $$\begin{aligned}(\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y)) &\to ((\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y)))\, y \\ &\to (((\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y)))\, y)\, y \to \cdots\end{aligned}$$



(From now on, we omit outermost parens.)

E.g.,

❶ $$(\lambda x.\, ((xx)\, y))(\lambda z.\, z) \longrightarrow ((\lambda z.\, z)(\lambda z.\, z))\, y \longrightarrow (\lambda z.\, z)\, y \longrightarrow y$$

$$\begin{array}{c} z \\ y \end{array} \;—\; \boxed{\lambda z.\, z} \;—\; \begin{array}{c} z \\ y \end{array}$$

❷ $$\begin{aligned}(\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y)) &\longrightarrow ((\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y)))\, y \\ &\longrightarrow (((\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y)))\, y)\, y \longrightarrow \cdots\end{aligned}$$

$$\begin{array}{r} x \\ \lambda x.\, ((xx)\, y) \end{array} \;—\; \boxed{\lambda x.\, ((xx)\, y)} \;—\; \begin{array}{l} (xx)\, y \\ ((\lambda x.\, ((xx)\, y))(\lambda x.\, ((xx)\, y)))\, y \end{array}$$

Thus, according to the model "computation = β-reduction", some computation halts (like ❶) but other computation fails to (like ❷).

Multiple inputs? In this calculus every function takes one and only one input, so how can we treat multi-input functions, such as +?

Multiple inputs? In this calculus every function takes one and only one input, so how can we treat multi-input functions, such as +?

— We obtain $2 + 3$ in two steps, using

$$\text{plus} := \lambda x.\, (\lambda y.\, (x + y)).$$

First apply this to 2 and obtain "$2 +$ (blank)", and then apply it to 3 and obtain $2 + 3$.

Multiple inputs? In this calculus every function takes one and only one input, so how can we treat multi-input functions, such as +?

— We obtain $2 + 3$ in two steps, using

$$\text{plus} := \lambda x.\,(\lambda y.\,(x + y)).$$

First apply this to 2 and obtain "2 + (blank)", and then apply it to 3 and obtain $2 + 3$.

$$(\text{plus}\,2)\,3 = ((\lambda x.\,(\lambda y.\,(x + y)))\,2)\,3$$

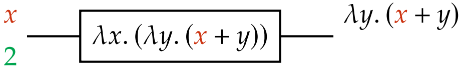
Multiple inputs? In this calculus every function takes one and only one input, so how can we treat multi-input functions, such as +?

— We obtain $2 + 3$ in two steps, using

$$\text{plus} := \lambda x. (\lambda y. (x + y)).$$

First apply this to 2 and obtain "2 + (blank)", and then apply it to 3 and obtain $2 + 3$.

$$(\text{plus}\, 2)\, 3 = ((\lambda x. (\lambda y. (x + y)))\, 2)\, 3$$



Multiple inputs? In this calculus every function takes one and only one input, so how can we treat multi-input functions, such as +?

— We obtain $2 + 3$ in two steps, using

$$\text{plus} := \lambda x.\,(\lambda y.\,(x + y)).$$

First apply this to 2 and obtain "2 + (blank)", and then apply it to 3 and obtain $2 + 3$.

$$(\text{plus}\,2)\,3 = ((\lambda x.\,(\lambda y.\,(x + y)))\,2)\,3 \longrightarrow (\lambda y.\,(2 + y))\,3$$

$$\begin{matrix} x \\ 2 \end{matrix} \;\text{---}\; \boxed{\lambda x.\,(\lambda y.\,(x + y))} \;\text{---}\; \begin{matrix} \lambda y.\,(x + y) \\ \lambda y.\,(2 + y) \end{matrix}$$

Multiple inputs? In this calculus every function takes one and only one input, so how can we treat multi-input functions, such as $+$?

— We obtain $2 + 3$ in two steps, using

$$\text{plus} := \lambda x.\,(\lambda y.\,(x + y)).$$

First apply this to 2 and obtain "$2 +$ (blank)", and then apply it to 3 and obtain $2 + 3$.

$$(\text{plus}\,2)\,3 = ((\lambda x.\,(\lambda y.\,(x + y)))\,2)\,3 \longrightarrow (\lambda y.\,(2 + y))\,3 \longrightarrow 2 + 3$$

$$\begin{array}{ccc} x & & \lambda y.\,(x + y) \\ & \text{---}\ \boxed{\lambda x.\,(\lambda y.\,(x + y))}\ \text{---} & \\ 2 & & \lambda y.\,(2 + y) \end{array}$$

Multiple inputs? In this calculus every function takes one and only one input, so how can we treat multi-input functions, such as +?

— We obtain $2 + 3$ in two steps, using

$$\text{plus} := \lambda x.\,(\lambda y.\,(x + y)).$$

First apply this to 2 and obtain "2 + (blank)", and then apply it to 3 and obtain $2 + 3$.

$$(\text{plus}\,2)\,3 = ((\lambda x.\,(\lambda y.\,(x + y)))\,2)\,3 \to (\lambda y.\,(2 + y))\,3 \to 2 + 3$$

$$\begin{array}{c} x \\ 2 \end{array} \;\text{———}\; \boxed{\lambda x.\,(\lambda y.\,(x + y))} \;\text{———}\; \begin{array}{l} \lambda y.\,(x + y) \\ \lambda y.\,(2 + y) \end{array}$$

In this way, we technically do not need multi-input functions.

Church numerals. In this calculus everything is a function, so we need functions that stand for natural numbers.

Church numerals. In this calculus everything is a function, so we need functions that stand for natural numbers.

$$\bar{n} := \lambda x.\,(\lambda y.\,(\underbrace{x(x(x(\cdots(x}_{n \text{ times}}\,y))))))$$

f — $\boxed{\bar{n}}$ — f^n, i.e., applying f n times

Church numerals. In this calculus everything is a function, so we need functions that stand for natural numbers.

$$\bar{n} := \lambda x.\,(\lambda y.\,(\underbrace{x(x(x(\cdots(x}_{n \text{ times}}\,y))))))$$

$$f \;\text{---}\; \boxed{\bar{n}} \;\text{---}\; f^n\text{, i.e., applying } f \; n \text{ times}$$

$$(\bar{3}f)\,a = ((\lambda x.\,(\lambda y.\,(x(x(xy)))))\,f)\,a$$

Church numerals. In this calculus everything is a function, so we need functions that stand for natural numbers.

$$\bar{n} := \lambda x.\,(\lambda y.\,(\underbrace{x(x(x(\cdots(x}_{n \text{ times}}\,y))))))$$

f —— $\boxed{\bar{n}}$ —— f^n, i.e., applying f n times

$$(\bar{3}f)\,a = ((\lambda x.\,(\lambda y.\,(x(x(xy)))))\,f)\,a$$

Church numerals. In this calculus everything is a function, so we need functions that stand for natural numbers.

$$\bar{n} := \lambda x.\,(\lambda y.\,(\underbrace{x(x(x(\cdots(x}_{n \text{ times}}\,y))))))$$

$$f \;—\!\!\!—\; \boxed{\bar{n}} \;—\!\!\!—\; f^n\text{, i.e., applying } f\ n \text{ times}$$

$$(\bar{3}f)\,a = ((\lambda x.\,(\lambda y.\,(x(x(xy)))))\,f)\,a$$

Church numerals. In this calculus everything is a function, so we need functions that stand for natural numbers.

$$\bar{n} := \lambda x.\,(\lambda y.\,(\underbrace{x(x(x(\cdots(x}_{n \text{ times}}\,y))))))$$

$$f \;\text{———}\; \boxed{\bar{n}} \;\text{———}\; f^n\text{, i.e., applying } f \; n \text{ times}$$

$$\begin{aligned}(\bar{3}f)\,a &= ((\lambda x.\,(\lambda y.\,(x(x(xy)))))\,f)\,a \\ &\to \quad (\lambda y.\,(f(f(fy))))\quad a\end{aligned}$$

Church numerals. In this calculus everything is a function, so we need functions that stand for natural numbers.

$$\bar{n} := \lambda x.\,(\lambda y.\,(\underbrace{x(x(x(\cdots(x}_{n \text{ times}}\,y))))))$$

$$f \;\text{———}\; \boxed{\bar{n}} \;\text{———}\; f^n, \text{ i.e., applying } f \; n \text{ times}$$

$$\begin{aligned}(\bar{3}f)\,a &= ((\lambda x.\,(\lambda y.\,(x(x(xy)))))\,f)\,a \\ &\to \qquad (\lambda y.\,(f(f(fy)))) \qquad a\end{aligned}$$

Church numerals. In this calculus everything is a function, so we need functions that stand for natural numbers.

$$\bar{n} := \lambda x.\,(\lambda y.\,(\underbrace{x(x(x(\cdots(x}_{n \text{ times}}\,y))))))$$

$$f \;\text{—}\boxed{\bar{n}}\text{—}\; f^n, \text{ i.e., applying } f \ n \text{ times}$$

$$\begin{aligned} (\bar{3}f)\,a &= ((\lambda x.\,(\lambda y.\,(x(x(xy)))))\,f)\,a \\ \to &\qquad (\lambda y.\,(f(f(f\,y))))\qquad a \end{aligned}$$

Church numerals. In this calculus everything is a function, so we need functions that stand for natural numbers.

$$\bar{n} := \lambda x.\,(\lambda y.\,(\underbrace{x(x(x(\cdots(x}_{n \text{ times}}\,y))))))$$

$$f \;—\!\!\boxed{\bar{n}}\!\!—\; f^n, \text{ i.e., applying } f \; n \text{ times}$$

$$\begin{aligned}
(\bar{3}f)\,a &= ((\lambda x.\,(\lambda y.\,(x(x(xy)))))\,f)\,a \\
&\to \quad (\lambda y.\,(f(f(fy))))\quad a \\
&\to \quad f(f(fa))
\end{aligned}$$

Church numerals. In this calculus everything is a function, so we need functions that stand for natural numbers.

$$\bar{n} := \lambda x.\,(\lambda y.\,(\underbrace{x(x(x(\cdots(x}_{n \text{ times}}\,y))))))$$

$$f \;—\; \boxed{\bar{n}} \;—\; f^n, \text{ i.e., applying } f \; n \text{ times}$$

$$\begin{aligned}(\bar{3}f)\,a &= ((\lambda x.\,(\lambda y.\,(x(x(xy)))))\,f)\,a \\ &\to \quad (\lambda y.\,(f(f(fy))))\quad a \\ &\to \quad f(f(fa))\end{aligned}$$

Definition. We say that a term F of the lambda calculus "λ-defines" a k-argument partial function f of natural numbers if

- whenever $f(n_1, \ldots, n_k)$ is defined and equals n,

$$((((F\overline{n_1})\,\overline{n_2})\cdots)\,\overline{n_k}) \to \cdots \to \bar{n},$$

- but when $f(n_1, \ldots, n_k)$ is undefined, the β-reduction of $((((F\overline{n_1})\,\overline{n_2})\cdots)$ never terminates.

E.g., Add := $\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines +.

E.g., Add := $\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines +.

Here is how the computation of $2 + 3 = 5$ goes:

E.g., Add := $\lambda u.\, (\lambda v.\, (\lambda x.\, (\lambda y.\, ((ux)((vx)\, y)))))$ defines +.

Here is how the computation of 2 + 3 = 5 goes:

$$(\mathrm{Add}\, \bar{2})\, \bar{3} = ((\lambda u.\, (\lambda v.\, (\lambda x.\, (\lambda y.\, ((ux)((vx)\, y))))))\, \bar{2})\, \bar{3}$$

E.g., Add $:= \lambda u.\, (\lambda v.\, (\lambda x.\, (\lambda y.\, ((ux)((vx)\, y)))))$ defines +.

Here is how the computation of $2 + 3 = 5$ goes:

$$(\text{Add}\, \bar{2})\, \bar{3} = ((\lambda u.\, (\lambda v.\, (\lambda x.\, (\lambda y.\, ((ux)((vx)\, y)))))\, \bar{2})\, \bar{3}$$

E.g., Add := $\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines +.

Here is how the computation of $2 + 3 = 5$ goes:

$$(\text{Add}\,\bar{2})\,\bar{3} = ((\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))\,\bar{2})\,\bar{3}$$

E.g., Add := $\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines +.

Here is how the computation of 2 + 3 = 5 goes:

$$
\begin{aligned}
(\text{Add}\,\bar{2})\,\bar{3} &= ((\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))\,\bar{2})\,\bar{3} \\
\to &\qquad (\lambda v.\,(\lambda x.\,(\lambda y.\,((\bar{2}x)((vx)\,y)))))\qquad \bar{3}
\end{aligned}
$$

E.g., Add := $\lambda u. (\lambda v. (\lambda x. (\lambda y. ((ux)((vx)\,y)))))$ defines +.

Here is how the computation of 2 + 3 = 5 goes:

$$\begin{aligned}(\mathrm{Add}\,\bar{2})\,\bar{3} &= ((\lambda u. (\lambda v. (\lambda x. (\lambda y. ((ux)((vx)\,y))))))\,\bar{2})\,\bar{3} \\ &\to \quad (\lambda v. (\lambda x. (\lambda y. ((\bar{2}x)((vx)\,y)))))\quad \bar{3}\end{aligned}$$

E.g., Add $:= \lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines +.

Here is how the computation of $2 + 3 = 5$ goes:

$$\begin{aligned}(\text{Add}\,\bar{2})\,\bar{3} &= ((\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y))))))\,\bar{2})\,\bar{3} \\ \longrightarrow &\qquad (\lambda v.\,(\lambda x.\,(\lambda y.\,((\bar{2}x)((vx)\,y)))))\qquad \bar{3}\end{aligned}$$

E.g., Add := $\lambda u. (\lambda v. (\lambda x. (\lambda y. ((ux)((vx)\, y)))))$ defines +.

Here is how the computation of 2 + 3 = 5 goes:

$$\begin{aligned}
(\mathrm{Add}\, \bar{2})\, \bar{3} &= ((\lambda u. (\lambda v. (\lambda x. (\lambda y. ((ux)((vx)\, y))))))\, \bar{2})\, \bar{3} \\
&\to (\lambda v. (\lambda x. (\lambda y. ((\bar{2}x)((vx)\, y)))))\quad \bar{3} \\
&\to \lambda x. (\lambda y. ((\bar{2}x)((\bar{3}x)\, y)))
\end{aligned}$$

E.g., Add $:= \lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines +.

Here is how the computation of $2 + 3 = 5$ goes:

$$
\begin{aligned}
(\mathrm{Add}\,\bar{2})\,\bar{3} &= ((\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y))))))\,\bar{2})\,\bar{3} \\
&\to \quad (\lambda v.\,(\lambda x.\,(\lambda y.\,((\bar{2}x)((vx)\,y)))))\quad \bar{3} \\
&\to \quad \lambda x.\,(\lambda y.\,((\bar{2}x)((\bar{3}x)\,y)))
\end{aligned}
$$

E.g., Add $:= \lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines +.

Here is how the computation of $2 + 3 = 5$ goes:

$$\begin{aligned}
(\mathrm{Add}\,\bar{2})\,\bar{3} &= ((\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y))))))\,\bar{2})\,\bar{3} \\
&\to \qquad (\lambda v.\,(\lambda x.\,(\lambda y.\,((\bar{2}x)((vx)\,y)))))\qquad \bar{3} \\
&\to \qquad \lambda x.\,(\lambda y.\,((\bar{2}x)((\bar{3}x)\,y))) \\
\to \cdots &\to \qquad \lambda x.\,(\lambda y.\,((\bar{2}x)(x(x(xy)))))
\end{aligned}$$

E.g., Add := $\lambda u.\, (\lambda v.\, (\lambda x.\, (\lambda y.\, ((ux)((vx)\, y)))))$ defines +.

Here is how the computation of $2 + 3 = 5$ goes:

$$\begin{aligned}
(\mathrm{Add}\, \bar{2})\, \bar{3} &= ((\lambda u.\, (\lambda v.\, (\lambda x.\, (\lambda y.\, ((ux)((vx)\, y))))))\, \bar{2})\, \bar{3} \\
&\to (\lambda v.\, (\lambda x.\, (\lambda y.\, ((\bar{2}x)((vx)\, y)))))\quad \bar{3} \\
&\to \lambda x.\, (\lambda y.\, ((\bar{2}x)((\bar{3}x)\, y))) \\
\to \cdots &\to \lambda x.\, (\lambda y.\, ((\bar{2}x)(x(x(xy)))))
\end{aligned}$$

E.g., Add := $\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines +.

Here is how the computation of $2 + 3 = 5$ goes:

$$
\begin{aligned}
(\mathrm{Add}\,\bar{2})\,\bar{3} &= ((\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y))))))\,\bar{2})\,\bar{3} \\
&\to (\lambda v.\,(\lambda x.\,(\lambda y.\,((\bar{2}x)((vx)\,y)))))\quad \bar{3} \\
&\to \lambda x.\,(\lambda y.\,((\bar{2}x)((\bar{3}x)\,y))) \\
\to \cdots &\to \lambda x.\,(\lambda y.\,((\bar{2}x)(x(x(xy))))) \\
\to \cdots &\to \lambda x.\,(\lambda y.\,(x(x(x(x(xy))))))
\end{aligned}
$$

E.g., Add := $\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines +.

Here is how the computation of $2+3=5$ goes:

$$
\begin{aligned}
(\mathrm{Add}\,\bar{2})\,\bar{3} &= ((\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y))))))\,\bar{2})\,\bar{3} \\
&\to \quad (\lambda v.\,(\lambda x.\,(\lambda y.\,((\bar{2}x)((vx)\,y)))))\quad \bar{3} \\
&\to \quad \lambda x.\,(\lambda y.\,((\bar{2}x)((\bar{3}x)\,y))) \\
\to \cdots &\to \quad \lambda x.\,(\lambda y.\,((\bar{2}x)(x(x(xy))))) \\
\to \cdots &\to \quad \lambda x.\,(\lambda y.\,(x(x(x(x(xy))))))
\end{aligned}
$$

E.g., Add := $\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines +.

Here is how the computation of 2 + 3 = 5 goes:

$$\begin{aligned}
(\mathrm{Add}\,\bar{2})\,\bar{3} &= ((\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))\,\bar{2})\,\bar{3} \\
\to\quad & (\lambda v.\,(\lambda x.\,(\lambda y.\,((\bar{2}x)((vx)\,y)))))\quad \bar{3} \\
\to\quad & \lambda x.\,(\lambda y.\,((\bar{2}x)((\bar{3}x)\,y))) \\
\to \cdots \to\quad & \lambda x.\,(\lambda y.\,((\bar{2}x)(x(x(xy))))) \\
\to \cdots \to\quad & \lambda x.\,(\lambda y.\,(x(x(x(x(xy)))))) \\
&= \bar{5}
\end{aligned}$$

E.g., $\mathrm{Add} := \lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y)))))$ defines $+$.

Here is how the computation of $2 + 3 = 5$ goes:

$$\begin{aligned}
(\mathrm{Add}\,\bar{2})\,\bar{3} &= ((\lambda u.\,(\lambda v.\,(\lambda x.\,(\lambda y.\,((ux)((vx)\,y))))))\,\bar{2})\,\bar{3} \\
\to\quad & \quad (\lambda v.\,(\lambda x.\,(\lambda y.\,((\bar{2}x)((vx)\,y)))))\quad \bar{3} \\
\to\quad & \quad \lambda x.\,(\lambda y.\,((\bar{2}x)((\bar{3}x)\,y))) \\
\to \cdots \to\quad & \quad \lambda x.\,(\lambda y.\,((\bar{2}x)(x(x(xy))))) \\
\to \cdots \to\quad & \quad \lambda x.\,(\lambda y.\,(x(x(x(x(xy)))))) \\
&= \bar{5}
\end{aligned}$$

Theorem. For any partial function f of natural numbers,

$$\begin{tikzcd}
 & f \text{ is Turing computable} \arrow[dl, Leftrightarrow] \arrow[dr, Leftrightarrow] & \\
f \text{ is partial recursive} \arrow[rr, Leftrightarrow] & & f \text{ is } \lambda\text{-definable.}
\end{tikzcd}$$