

# Lambda Calculus & Cartesian Closed Categories

Matilde Marcolli

Ma6c: Logic, Caltech, Spring 2026

## $\lambda$ -calculus

- Alonzo Church (1930s): another formulation of computability
- Kleene (1936): computable functions (recursive functions) are definable in Church's  $\lambda$ -calculus
- Turing (1937): Turing machines compute the same class of computable functions
- $\lambda$ -calculus based on three operations: application,  $\lambda$ -abstraction,  $\beta$ -reduction

## Syntax of $\lambda$ -calculus


- set of **terms**:  $\lambda$ -terms defined inductively by
  - 1 a variable  $x$  is a  $\lambda$ -term
  - 2 if  $A$  and  $B$  are  $\lambda$ -terms then **application** ( $AB$ ) is also a  $\lambda$ -term
  - 3 if  $A$  is a  $\lambda$ -terms and  $x$  variable, then the  **$\lambda$ -abstraction** ( $\lambda x.A$ ) is also a  $\lambda$ -term
- note:  $\lambda$ -abstraction is a way of declaring functions,  $(\lambda x.f(x))$  stands for  $x \mapsto f(x)$
- **depth** of a  $\lambda$ -term:
  - 1 if term is a variable  $x$  then  $d(x) = 0$
  - 2 if term is  $(AB)$  then  $d(AB) = \max\{d(A), d(B)\} + 1$
  - 3 if term is  $(\lambda x.A)$  then  $d(\lambda x.A) = d(A) + 1$

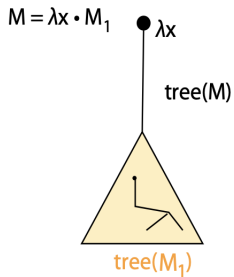
in a variable term  $x$  is a *free variable*, in a  $\lambda$ -abstraction  $(\lambda x, A)$  the variable  $x$  is a *bound variable*;  $\lambda$ -term is **combinator** (or **closed**) if no free variables


can written dropping parentheses:  $AB$  for  $(AB)$  and  $(\lambda xy.yx)zw$  for  $((((\lambda x.(\lambda y.(yx)))z)w)$

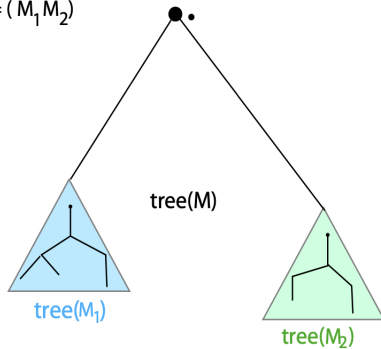
can represent  $\lambda$ -terms as trees

# $\lambda$ -terms as trees

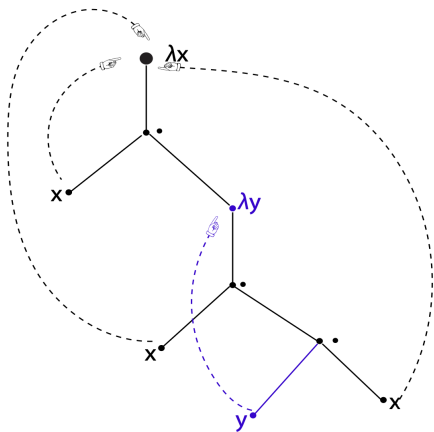
$M = x$    $x$   
tree(M)



$M = (M_1 M_2)$    $\cdot$



- can always **rename bound variables**: e.g.  $\lambda x(x(\lambda y.x(yx)))$  same as  $\lambda x(x(\lambda z.x(zx)))$  same as  $\lambda w(w(\lambda z.w(zw)))$
- can use the tree representation of  $\lambda$ -terms: look at every leaf that is marked by a bound variable  $x$  and draw an arc to its closest ancestor vertex (up in the tree) marked by  $\lambda x$
- useful to keep track of **substitutions**:  
 $\varphi = [x_1 := N_1, \dots, x_n := N_n]$  replaces variable  $x_i$  with  $\lambda$ -term  $N_i$
- write  $\varphi_{-x_i}$  for substitution as above, but where  $x_i$ 'th variable remains  $x_i$  and  $x_j := N_j$  for  $j \neq i$



$\text{tree}(\lambda x \cdot x(\lambda y \cdot x(yx)))$

backpointers in the tree of the  $\lambda$ -term  $\lambda x.x(\lambda y.(x(yx)))$

ref. Gallier and Quintance, 2025

- rules for substitutions in  $\lambda$ -terms

(1) If  $M = y$ , with  $y \neq x_i$  for  $i = 1, \dots, n$ , then  $M[\varphi] = y = M$ .

(2) If  $M = x_i$  for some  $i \in \{1, \dots, n\}$ , then  $M[\varphi] = N_i$ .

(3) If  $M = (PQ)$ , then  $M[\varphi] = (P[\varphi]Q[\varphi])$ .

(4) If  $M = \lambda x. N$  and  $x \neq x_i$  for  $i = 1, \dots, n$ , then  $M[\varphi] = \lambda x. N[\varphi]$ ,

(5) If  $M = \lambda x. N$  and  $x = x_i$  for some  $i \in \{1, \dots, n\}$ , then  
 $M[\varphi] = \lambda x. N[\varphi]_{-x_i}$ .

- also need that the free variables in the substitution do *not* become bound (only substitute terms in variables that are free)
- no non-trivial substitutions in closed terms  $M[\varphi] = M$

## $\alpha$ -conversion

- defining an equivalence under substitutions
- equivalence relation generated by  $\alpha$ -conversion  $\rightarrow_\alpha$

$$\lambda x. M \rightarrow_\alpha \lambda y. M[x := y], \quad \text{for all } y \notin FV(M) \cup BV(M)$$

$$\text{if } M \rightarrow_\alpha N \text{ then } MQ \rightarrow_\alpha NQ \text{ and } PM \rightarrow_\alpha PN$$

$$\text{if } M \rightarrow_\alpha N \text{ then } \lambda x. M \rightarrow_\alpha \lambda x. N.$$

for  $\lambda$ -terms  $M, N, P, Q$

- $\equiv_\alpha$  min symmetric, reflexive, transitive 2-ary relation extending  $\rightarrow_\alpha$
- $M \equiv_\alpha N$  considered identical

## $\beta$ -reduction

- key operation of  $\lambda$ -calculus
- $\beta$  reduction (for terms  $M, N, P, Q$ )

$$(\lambda x. M)N \longrightarrow_{\beta} M[x := N], \quad \text{where } M \text{ is safe for } [x := N]$$

$$\text{if } M \longrightarrow_{\beta} N \text{ then } MQ \longrightarrow_{\beta} NQ \text{ and } PM \longrightarrow_{\beta} PN$$

$$\text{if } M \longrightarrow_{\beta} N \text{ then } \lambda x. M \longrightarrow_{\beta} \lambda x. N.$$

- $\overset{+}{\longrightarrow}_{\beta}$  transitive closure of  $\longrightarrow_{\beta}$   $\overset{*}{\longrightarrow}_{\beta}$  reflexive, transitive closure, and  $\overset{*}{\leftrightarrow}_{\beta}$  equivalence relation generated by  $\longrightarrow_{\beta}$  (also symmetric)
- subterm  $(\lambda x. M)N$  occurring in a  $\lambda$ -term called a  $\beta$ -redex (to which  $\beta$ -reduction can be applied);  $\lambda$ -term is in  $\beta$ -normal form if it contains no  $\beta$ -redex
- Example:

$$(\lambda xy. x)uv = ((\lambda x. (\lambda y. x)u)v \longrightarrow_{\beta} ((\lambda y. x)[x := u])v = (\lambda y. u)v \longrightarrow_{\beta} u[y := v] = u$$

- note that  $\beta$  reduction does not always “reduce/simplify”  $\lambda$ -terms
- they can stay the same: for  $\omega = \lambda x.(xx)$

$$\Omega = \omega\omega = (\lambda x.(xx))(\lambda x.(xx)) \longrightarrow_{\beta} (xx)[x := \lambda x.(xx)] = \omega\omega = \Omega.$$

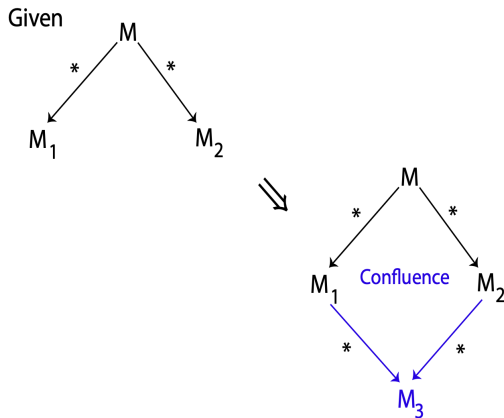
- they can become more complex

$$\begin{aligned} (\lambda x. xxx)(\lambda x. xxx) &\xrightarrow{+}_{\beta} (\lambda x. xxx)(\lambda x. xxx)(\lambda x. xxx) \\ &\xrightarrow{+}_{\beta} (\lambda x. xxx)(\lambda x. xxx)(\lambda x. xxx)(\lambda x. xxx) \\ &\xrightarrow{+}_{\beta} \dots \end{aligned}$$

- if a  $\lambda$ -term contains several  $\beta$ -redex subterms, can apply different sequences of  $\beta$ -reductions so get  $M \rightarrow_{\beta} M_1$  and  $M \rightarrow_{\beta} M_2$  with  $M_1 \neq M_2$ : how are these related?
- **Church-Rosser theorem**: order of application of  $\beta$ -reductions does not matter

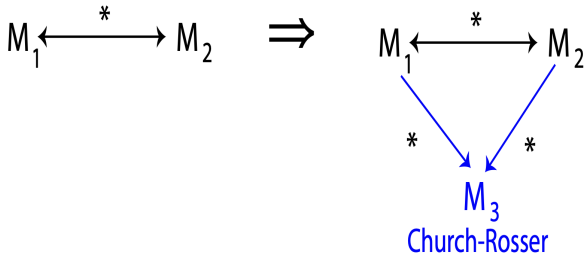
# Church-Rosser theorem

(1) The  $\lambda$ -calculus is **confluent**: for any three  $\lambda$ -terms  $M, M_1, M_2$ , if  $M \xrightarrow{*}_\beta M_1$  and  $M \xrightarrow{*}_\beta M_2$ , then there is some  $\lambda$ -term  $M_3$  such that  $M_1 \xrightarrow{*}_\beta M_3$  and  $M_2 \xrightarrow{*}_\beta M_3$ .



(2) The  $\lambda$ -calculus has the **Church–Rosser property**: for any two  $\lambda$ -terms  $M_1, M_2$ , if  $M_1 \xrightarrow{*} \beta M_2$ , then there is some  $\lambda$ -term  $M_3$  such that  $M_1 \xrightarrow{*} \beta M_3$  and  $M_2 \xrightarrow{*} \beta M_3$ .

Given



ref. Gallier and Quaintance, 2025

## fixed point combinators

- can be used to describe recursively-defined functions in  $\lambda$ -calculus
- **Curry  $Y$ -combinator:**  $Y = \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))$

then for any  $\lambda$ -term  $F$  we have

$$F(YF) \leftarrow^*_{\beta} YF.$$

*Proof.* Write  $W = \lambda x. F(xx)$ . We have

$$F(YF) = F\left((\lambda f. (\lambda x. f(xx))(\lambda x. f(xx)))F\right) \rightarrow_{\beta} F((\lambda x. F(xx))(\lambda x. F(xx))) = F(WW),$$

and

$$\begin{aligned} YF &= (\lambda f. (\lambda x. f(xx))(\lambda x. f(xx)))F \rightarrow_{\beta} (\lambda x. F(xx))(\lambda x. F(xx)) = (\lambda x. F(xx))W \\ &\rightarrow_{\beta} F(WW). \end{aligned}$$

- **Turing  $\Theta$  combinator:**  $\Theta = (\lambda xy. y(xxy))(\lambda xy. y(xxy))$

then for any  $\lambda$ -term  $F$  we have

$$\Theta F \xrightarrow{+}_{\beta} F(\Theta F).$$

*Proof.* If we write  $A = (\lambda xy. y(xxy))$ , then  $\Theta = AA$ . We have

$$\begin{aligned} \Theta F &= (AA)F = ((\lambda xy. y(xxy))A)F \\ &\rightarrow_{\beta} (\lambda y. y(AAy))F \\ &\rightarrow_{\beta} F(AAF) \\ &= F(\Theta F), \end{aligned}$$

## recursive functions via combinators

In general, if we want to define a function  $G$  recursively such that

$$GX \xrightarrow{+}_{\beta} M(X, G)$$

where  $M(X, G)$  is  $\lambda$ -term containing recursive occurrences of  $G$ , we let  $F = \lambda gx. M(x, g)$  and

$$G = \Theta F.$$

Then we have

$$G \xrightarrow{+}_{\beta} FG = (\lambda gx. M(x, g))G \longrightarrow_{\beta} \lambda x. M(x, g)[g := G] = \lambda x. M(x, G),$$

so

$$GX \xrightarrow{+}_{\beta} (\lambda x. M(x, G))X \longrightarrow_{\beta} M(x, G)[x := X] = M(X, G),$$

## recursive functions

- **primitive recursive functions**  $f : \mathbb{N}^n \rightarrow \mathbb{N}$
- **generators**: *constant*  $Z^n(x) = 0$  (on  $\mathbb{N}^n$ ), *successor*:  $S(x) = x + 1$  (on  $\mathbb{N}$ ), *projections*:  $\pi_i^n(x) = x_i$  (on  $\mathbb{N}^n$ )
- **operations**: *composition*, *bracketing*, *primitive recursion*
- **partial recursive functions**:  $f : \mathcal{D}(f) \subseteq \mathbb{N}^n \rightarrow \mathbb{N}$
- additional  $\mu$ -minimization of  $g : \mathcal{D}(g) \subseteq \mathbb{N}^{n+1} \rightarrow \mathbb{N}$

$$f(x_1, \dots, x_n) = \min\{x \mid g(x, x_1, \dots, x_n) = 0\}$$

- Church thesis: equivalent to computable functions in Turing machine ( $x \notin \mathcal{D}(f)$ : never halts)

also computable in Church's  $\lambda$ -calculus

## First Step: describing $\mathbb{N}$ in the $\lambda$ -calculus

- **idea**: integers  $n \in \mathbb{N}$  are captured in iterations of functions
- $\lambda$ -terms  $F, M$ : have  $F^0(M) = M$ ,  $F^{n+1}(M) = F(F^n(M))$
- so can define **Church numerals**

$$\mathbf{c}_n := \lambda fx. f^n(x)$$

$$\mathbf{c}_0 = \lambda fx. x, \mathbf{c}_1 = \lambda fx. fx, \mathbf{c}_2 = \lambda fx. f(fx), \text{ etc}$$

- for  $\lambda$ -term  $F$

$$\mathbf{c}_n Fz = (\lambda fx. f^n(x))Fz \xrightarrow{\dagger}_{\beta} F^n(z)$$

- get zero function  $Z^1(n) = 0$  by setting  $Z_c = \lambda x. \mathbf{c}_0$  so

$$Z_c \mathbf{c}_n = (\lambda x. \mathbf{c}_0) \mathbf{c}_n \rightarrow_{\beta} \mathbf{c}_0[x := \mathbf{c}_n] = \mathbf{c}_0$$

- successor function from  $\text{Succ}_c = \lambda nfx. f(nfx)$

$$\begin{aligned} \text{Succ}_c \mathbf{c}_n &= (\lambda nfx. f(nfx)) \mathbf{c}_n \\ &\rightarrow_{\beta} (\lambda fx. f(nfx))[n := \mathbf{c}_n] = \lambda fx. f(\mathbf{c}_n fx) \\ &\rightarrow_{\beta} \lambda fx. f(f^n(x)) \\ &= \lambda fx. f^{n+1}(x) = \mathbf{c}_{n+1}. \end{aligned}$$

- combinators for addition and multiplication on  $\mathbb{N}$

$$\mathbf{Add} = \lambda mnfx. mf(nfx)$$

$$\mathbf{Mult} = \lambda xyz. x(yz).$$

$$\mathbf{Add} \mathbf{c}_m \mathbf{c}_n \xrightarrow{+}_{\beta} \mathbf{c}_{m+n}$$

$$\mathbf{Mult} \mathbf{c}_m \mathbf{c}_n \xrightarrow{+}_{\beta} \mathbf{c}_{m*n}$$

- case of addition (multiplication similar with an inductive proof)

$$\begin{aligned} \mathbf{Add} \mathbf{c}_m \mathbf{c}_n &= (\lambda mnfx. mf(nfx)) \mathbf{c}_m \mathbf{c}_n \\ &\xrightarrow{+}_{\beta} (\lambda fx. \mathbf{c}_m f(\mathbf{c}_n f x)) \\ &\xrightarrow{+}_{\beta} \lambda fx. f^m(f^n(x)) \\ &= \lambda fx. f^{m+n}(x) = \mathbf{c}_{m+n}. \end{aligned}$$

## Second Step: some Combinators

**I**, **K**, **K\***, and **S** are the combinators defined by

$$\mathbf{I} = \lambda x. x$$

$$\mathbf{K} = \lambda xy. x$$

$$\mathbf{K}_* = \lambda xy. y$$

$$\mathbf{S} = \lambda xyz. (xz)(yz),$$

for any  $\lambda$ -terms  $M, N, P$  these satisfy:

$$\mathbf{I}M \xrightarrow{+}_{\beta} M$$

$$\mathbf{K}MN \xrightarrow{+}_{\beta} M$$

$$\mathbf{K}_*MN \xrightarrow{+}_{\beta} N$$

$$\mathbf{S}MNP \xrightarrow{+}_{\beta} (MP)(NP)$$

$$\mathbf{K}\mathbf{I} \xrightarrow{+}_{\beta} \mathbf{K}_*$$

$$\mathbf{S}\mathbf{K}\mathbf{K} \xrightarrow{+}_{\beta} \mathbf{I}.$$

- for example: in the case of  $SMNP \xrightarrow{+}_{\beta} (MP)(NP)$

$$\begin{aligned}
 SMNP &= (\lambda xyz. (xz)(yz))MNP \longrightarrow_{\beta} ((\lambda yz. (xz)(yz))[x := M])NP \\
 &= (\lambda yz. (Mz)(yz))NP \\
 &\longrightarrow_{\beta} ((\lambda z. (Mz)(yz))[y := N])P \\
 &= (\lambda z. (Mz)(Nz))P \\
 &\longrightarrow_{\beta} ((Mz)(Nz))[z := P] = (MP)(NP).
 \end{aligned}$$

## combinators for Boolean truth values

- need to encode  $\mathbf{T}, \mathbf{F}$  ( $\top, \perp$ ) as  $\lambda$ -terms (i.e. as functions)
- achieved through a ternary **if-then-else** combinator and  $\mathbf{T}, \mathbf{F}$  combinators that realize the correct behavior of conditional statements

## if-then-else combinator and Boolean values

- define **if-then-else combinator** as

$$\text{if then else} = \lambda bxy . bxy$$

- then define **Boolean values** as  $\mathbf{T} := \mathbf{K}$  and  $\mathbf{F} := \mathbf{K}_*$  so that for any  $\lambda$ -terms  $P, Q$  have

$$\text{if } \mathbf{T} \text{ then } P \text{ else } Q \xrightarrow{+}_{\beta} P$$

$$\text{if } \mathbf{F} \text{ then } P \text{ else } Q \xrightarrow{+}_{\beta} Q.$$

- for example the first case gives

$$\begin{aligned} \text{if } \mathbf{T} \text{ then } P \text{ else } Q &= (\text{if then else})\mathbf{T}PQ \\ &= (\lambda bxy . bxy)\mathbf{T}PQ \\ &\longrightarrow_{\beta} ((\lambda xy . bxy)[b := \mathbf{T}])PQ = (\lambda xy . \mathbf{T}xy)PQ \\ &\longrightarrow_{\beta} ((\lambda y . \mathbf{T}xy)[x := P])Q = (\lambda y . \mathbf{T}Py)Q \\ &\longrightarrow_{\beta} (\mathbf{T}Py)[y := Q] = \mathbf{T}PQ \\ &= \mathbf{K}PQ \xrightarrow{+}_{\beta} P, \end{aligned}$$

- so define  $\mathbf{T}, \mathbf{F}$  as functions of two arguments that ignore one and return the other ( $\mathbf{T}$  returns first argument,  $\mathbf{F}$  second)

## pair and projections combinator

- for any  $\lambda$ -terms  $M, N$  define combinators  $\langle M, N \rangle$  and  $\pi_1, \pi_2$  as

$$\langle M, N \rangle = \lambda z. zMN = \lambda z. \text{if } z \text{ then } M \text{ else } N$$

$$\pi_1 = \lambda z. z\mathbf{K}$$

$$\pi_2 = \lambda z. z\mathbf{K}_*.$$

- so that

$$\pi_1 \langle M, N \rangle \xrightarrow{+}_\beta M$$

$$\pi_2 \langle M, N \rangle \xrightarrow{+}_\beta N$$

$$\langle M, N \rangle \mathbf{T} \xrightarrow{+}_\beta M$$

$$\langle M, N \rangle \mathbf{F} \xrightarrow{+}_\beta N.$$

- for example:

$$\begin{aligned} \pi_1 \langle M, N \rangle &= (\lambda z. z\mathbf{K})(\lambda z. zMN) \\ &\longrightarrow_\beta (z\mathbf{K})[z := \lambda z. zMN] = (\lambda z. zMN)\mathbf{K} \\ &\longrightarrow_\beta (zMN)[z := \mathbf{K}] = \mathbf{K}MN \xrightarrow{+}_\beta M, \end{aligned}$$

## iteration combinator

- combinator **Iter** =  $\lambda nfx . nfx$  satisfies

$$\mathbf{Iterc}_nFX \xrightarrow{\dagger}_{\beta} F^nX$$

- note that **Succ<sub>c</sub>** =  $\lambda nfx . f(nfx)$
- also have a combinator **IsZero<sub>c</sub>** =  $\lambda x . x(\mathbf{KF})\mathbf{T}$  that tests if natural number is zero or not
- also a *predecessor*: **Pred<sub>c</sub>** =  $\lambda xyz . x(\lambda pq . q(py))(\mathbf{Kz})\mathbf{I}$

## $\lambda$ -definable functions

- function  $f : \mathbb{N}^n \rightarrow \mathbb{N}$  is  $\lambda$ -definable if  $\exists$  a closed  $\lambda$ -term  $F$  with
  - $(m_1, \dots, m_n) \in \mathcal{D}(f)$  iff  $F\mathbf{c}_{m_1} \cdots \mathbf{c}_{m_n}$  reduces to a  $\beta$ -normal form under  $\beta$ -reduction
  - $F\mathbf{c}_{m_1} \cdots \mathbf{c}_{m_n}$  satisfies

$$F\mathbf{c}_{m_1} \cdots \mathbf{c}_{m_n} \xrightarrow{\star} \beta \mathbf{c}_{f(m_1, \dots, m_n)}$$

**Note:** because of Church–Rosser theorem, since  $\mathbf{c}_{f(m_1, \dots, m_n)}$  is in  $\beta$ -normal form (because defined as  $\mathbf{c}_n = \lambda f x. f^n(x)$ ) we can replace

$$F\mathbf{c}_{m_1} \cdots \mathbf{c}_{m_n} \xrightarrow{\star} \beta \mathbf{c}_{f(m_1, \dots, m_n)}$$

with

$$F\mathbf{c}_{m_1} \cdots \mathbf{c}_{m_n} \xrightarrow{\star} \beta \mathbf{c}_{f(m_1, \dots, m_n)}$$

this says that  $\mathbf{c}_{f(m_1, \dots, m_n)}$  computed from  $F\mathbf{c}_{m_1} \cdots \mathbf{c}_{m_n}$  via a sequence of  $\beta$ -reductions that reaches a  $\beta$ -normal form

- so  $f(m_1, \dots, m_n)$  is defined when  $\mathbf{c}_{f(m_1, \dots, m_n)}$  is achieved, i.e. when  $F\mathbf{c}_{m_1} \cdots \mathbf{c}_{m_n}$  reduces to  $\beta$ -normal form (case  $(m_1, \dots, m_n) \in \mathcal{D}(f)$ )
- if no reduction sequence for  $F\mathbf{c}_{m_1} \cdots \mathbf{c}_{m_n}$  reaches a  $\beta$ -normal form then  $(m_1, \dots, m_n) \notin \mathcal{D}(f)$

a *total function*  $f : \mathbb{N}^n \rightarrow \mathbb{N}$  is  $\lambda$ -definable iff it is total computable and a *partial function* is  $\lambda$ -definable iff it is partial computable

Focus on the implication **computable**  $\Rightarrow$   **$\lambda$ -definable** (Kleene's theorem, 1936)

## Total recursive functions

- generators:

- 1 have  $\mathbf{Z}_c = \lambda x c_0$  computing *constant function*  $Z(n) = 0$ ; can extend to multivariable function by “currying” (multivariable function as chain of single variable:  $f : A \times B \rightarrow C$  as  $F : A \rightarrow (B \rightarrow C)$  with  $F_a(b) = f(a, b)$ )
- 2 also have *successor* from  $\mathbf{Succ}_c = \lambda n f x. f(nfx)$
- 3 need the *projections* (coordinate functions)  $\pi_i^n$ : these are computed by

$$\mathbf{U}_i^n = \lambda x_1 \cdots x_n. x_i$$

- operations:

- 1 *composition*: if function  $g$  is  $\lambda$ -definable with combinator  $G$  and  $h_1, \dots, h_m$   $\lambda$ -definable with combinators  $H_1, \dots, H_m$  then

$$F = \lambda x_1 \cdots x_n. G (H_1 x_1 \cdots x_n) \cdots (H_m x_1 \cdots x_n)$$

- 2 *bracketing*: multivariable to single variable via currying

## Primitive Recursion

- $f$  defined by primitive recursion from  $g$  and  $h$  if

$$f(0, x_1, \dots, x_m) = g(x_1, \dots, x_m)$$

$$f(n + 1, x_1, \dots, x_m) = h(f(n, x_1, \dots, x_m), n, x_1, \dots, x_m)$$

- if  $g$  is  $\lambda$ -definable with combinator  $G$  and  $h$  is  $\lambda$ -definable with combinator  $H$
- then  $f$  is also  $\lambda$ -definable with combinator

$$F = \lambda n x_1 \cdots x_m . \pi_1(\mathbf{Iter} \ n \ \lambda z . \langle H \pi_1 z \pi_2 z x_1 \cdots x_m, \mathbf{Succ}_c(\pi_2 z) \rangle \langle G x_1 \cdots x_m, \mathbf{c}_0 \rangle)$$

- proved showing by induction that

$$(\lambda z . \langle H \pi_1 z \pi_2 z \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{Succ}_c(\pi_2 z) \rangle)^n \langle G \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{c}_0 \rangle \xrightarrow{+}_{\beta} \langle \mathbf{c}_{f(n, n_1, \dots, n_m)}, \mathbf{c}_n \rangle.$$

- first step

$$\begin{aligned}
 & (\lambda z. \langle H \pi_1 z \pi_2 z \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{Succ}_c(\pi_2 z) \rangle)^0 \langle G \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{c}_0 \rangle \\
 & \xrightarrow{+}_\beta \langle G \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{c}_0 \rangle = \langle \mathbf{c}_{g(n_1, \dots, n_m)}, \mathbf{c}_0 \rangle = \langle \mathbf{c}_{f(0, n_1, \dots, n_m)}, \mathbf{c}_0 \rangle.
 \end{aligned}$$

- induction step

$$\begin{aligned}
 & (\lambda z. \langle H \pi_1 z \pi_2 z \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{Succ}_c(\pi_2 z) \rangle)^{n+1} \langle G \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{c}_0 \rangle \\
 & = (\lambda z. \langle H \pi_1 z \pi_2 z \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{Succ}_c(\pi_2 z) \rangle)^n \langle G \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{c}_0 \rangle \\
 & \quad \left( \lambda z. \langle H \pi_1 z \pi_2 z \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{Succ}_c(\pi_2 z) \rangle \right)^n \langle G \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{c}_0 \rangle \\
 & \xrightarrow{+}_\beta (\lambda z. \langle H \pi_1 z \pi_2 z \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{Succ}_c(\pi_2 z) \rangle) \langle \mathbf{c}_{f(n, n_1, \dots, n_m)}, \mathbf{c}_n \rangle \\
 & \quad \xrightarrow{+}_\beta \langle H \mathbf{c}_{f(n, n_1, \dots, n_m)} \mathbf{c}_n \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_m}, \mathbf{Succ}_c \mathbf{c}_n \rangle \\
 & \quad \quad \xrightarrow{+}_\beta \langle \mathbf{c}_{h(f(n, n_1, \dots, n_m), n, n_1, \dots, n_m)}, \mathbf{c}_{n+1} \rangle = \langle \mathbf{c}_{f(n+1, n_1, \dots, n_m)}, \mathbf{c}_{n+1} \rangle.
 \end{aligned}$$

## Primitive Recursion as fixed point combinator

- define combinator  $J$  as

$$J = \lambda f x x_1 \cdots x_m. \text{if IsZero}_c x \text{ then } G x_1 \cdots x_m \text{ else } H(f(\mathbf{Pred}_c x) x_1 \cdots x_m)(\mathbf{Pred}_c x) x_1 \cdots x_m$$

- then take

$$F = \Theta J$$

with  $\Theta$  the Turing combinator

- this combinator  $F$  defines  $f$  (primitive recursion from  $g$  and  $h$ )

## $\mu$ -Minimization

- assume  $f$  is **total** and defined by  $\mu$ -minimization from  $g$

$$f(x_1, \dots, x_m) = \min\{x \in \mathbb{N} \mid g(x, x_1, \dots, x_m) = 0\}$$

(i.e. assume there is for any  $(x_1, \dots, x_m)$  a solution to  $g(x, x_1, \dots, x_m) = 0$ )

- assume  $g$  is  $\lambda$ -definable with combinator  $G$
- define combinator  $J$  given by

$$J = \lambda f x x_1 \dots x_m. \text{ if } \text{IsZero}_c G x x_1 \dots x_m \text{ then } x \text{ else } f(\text{Succ}_c x) x_1 \dots x_m$$

- then take  $F = \Theta J$  with  $\Theta$  the Turing combinator
- this gives

$$F \mathbf{c}_n \mathbf{c}_{n_1} \dots \mathbf{c}_{n_n} \xrightarrow{+}_{\beta} \begin{cases} \mathbf{c}_n & \text{if } g(n, n_1, \dots, n_m) = 0 \\ F \mathbf{c}_{n+1} \mathbf{c}_{n_1} \dots \mathbf{c}_{n_n} & \text{otherwise,} \end{cases}$$

## Partial Recursive Functions

- adapt the  $\mu$ -minimization to partial recursive functions (where there need not be always a solution to  $g(x, x_1, \dots, x_m) = 0$  so points  $(x_1, \dots, x_m)$  for which no solution are  $(x_1, \dots, x_m) \notin \mathcal{D}(f)$ )
- **key idea Kleene normal form**: can always write any partial recursive function  $f : \mathcal{D}(f) \subset \mathbb{N}^n \rightarrow \mathbb{N}$  in the form

$$f = g \circ \mu h$$

where  $g, h$  are primitive recursive functions  $g : \mathbb{N} \rightarrow \mathbb{N}$  and  $h : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$  and  $\mu g$  is  $\mu$ -minimization on  $g$

- this description shows that

$$F \mathbf{c}_n \mathbf{c}_{n_1} \dots \mathbf{c}_{n_n} \xrightarrow{+}_{\beta} \begin{cases} \mathbf{c}_n & \text{if } g(n, n_1, \dots, n_m) = 0 \\ F \mathbf{c}_{n+1} \mathbf{c}_{n_1} \dots \mathbf{c}_{n_n} & \text{otherwise,} \end{cases}$$

still defines the function  $f$  where defined,  $(n_1, \dots, n_m) \in \mathcal{D}(f)$ , while the process never terminates where not defined,  $(n_1, \dots, n_m) \notin \mathcal{D}(f)$

## Lambda Calculus and Category Theory

**Problem:** in (untyped)  $\lambda$ -calculus, objects can be both functions and arguments of functions so a **semantics** would need to provide a set  $D$  that is also its own function space  $D^D$  (by Cantor's argument this cannot be done).

**Solution:** don't use sets!

**first idea** (Dana Scott, 1969): restrict  $D^D$  from "all functions" to a space of continuous functions with a particular choice of topology on  $D$ ; then observed right kinds of topological spaces form a *Cartesian closed category*, and that is what matters

- $\lambda$ -algebras and  $\lambda$ -models
- categorical semantics for (untyped)  $\lambda$ -calculus
- setting: **Cartesian closed categories**
- **reflexive objects** in Cartesian closed categories

## $\lambda$ -algebras and $\lambda$ -models

- two approaches to models for  $\lambda$ -calculus
- **$\lambda$ -algebras**: defined by *equations* (designed to satisfy all provable equations of  $\lambda$ -calculus), formulated in terms of *combinators*  $\mathbf{K} = \lambda xy.x$  and  $\mathbf{S} = \lambda xyz.(xz)(yz)$  and axioms, closed under substructures and homomorphism image
- **$\lambda$ -models**: described by first order axioms, not closed under substructures or homomorphism images
- **categorical description** provides a framework for realization of these two concepts, with  $\lambda$ -models corresponding to  $\lambda$ -algebras in a category with a “reflexive object” that has “enough points”

## combinatory algebras

- $\mathfrak{M} = (X, \cdot)$  a set with a binary operation (think of  $X$  as some set of functions with  $\cdot$  as composition)
- *extensional* if for  $a, b \in X$  have  $(\forall x \in X \ ax = bx) \Rightarrow a = b$
- set of **terms over  $\mathfrak{M}$** :  $\mathcal{T}(\mathfrak{M})$  inductively defined by

variables  $x_i \in \mathcal{T}(\mathfrak{M})$ , constants  $a \in \mathfrak{M} \Rightarrow c_a \in \mathcal{T}(\mathfrak{M})$

$$A, B \in \mathcal{T}(\mathfrak{M}) \Rightarrow (AB) \in \mathcal{T}(\mathfrak{M})$$

- so elements of  $\mathcal{T}(\mathfrak{M})$  are also functions, their arguments either coming from  $\mathfrak{M}$  or free variables  $x_i$
- **two distinguished elements**:  $k, s$  with corresponding **K** and **S** in  $\mathcal{T}(\mathfrak{M})$  satisfying

$$\mathbf{K}_{xy} = x \quad \text{and} \quad \mathbf{S}_{xyz} = (xz)(yz)$$

- **combinatory algebra**:  $\mathfrak{M} = (X, \cdot, k, s)$  as above

## valuations in combinatory algebras

- $\mathfrak{M} = (X, \cdot)$  and  $\mathcal{T}(\mathfrak{M})$  as above
- $\mathcal{V} = \{x_1, x_2, \dots, x_n, \dots\}$  set of variables for terms in  $\mathcal{T}(\mathfrak{M})$
- **valuation in  $\mathfrak{M}$** : a function  $\rho : \mathcal{V} \rightarrow X$
- **interpretation** of  $A \in \mathcal{T}(\mathfrak{M})$  under  $\rho$  inductively defined as

variables:  $(x)_\rho^{\mathfrak{M}} := \rho(x)$ , constants:  $(c_a)_\rho^{\mathfrak{M}} := a$

$$(AB)_\rho^{\mathfrak{M}} := (A)_\rho^{\mathfrak{M}} \cdot (B)_\rho^{\mathfrak{M}}$$

- $A = B$  is true in  $\mathfrak{M}$  under  $\rho$ :

$$(\mathfrak{M}, \rho) \models (A = B) \quad \text{if} \quad (A)_\rho^{\mathfrak{M}} = (B)_\rho^{\mathfrak{M}}$$

## $\lambda$ -algebras

- combinatory algebra  $\mathfrak{M} = (X, \cdot, k, s)$  as above
- Curry axioms for the elements  $k, s$

$$1. k = s(s(ks)(s(kk)k))(k(skk))$$

$$2. s = s(s(ks)(s(k(s(ks))))(s(k(s(kk))))s))(k(k(skk)))$$

$$3. s(kk) = s(s(ks)(s(kk)(s(ks)k)))(kk)$$

$$4. s(ks)(s(kk)) = s(kk)(s(s(ks)(s(kk)(skk)))(k(skk)))$$

$$5. s(k(s(ks)))(s(ks)(s(ks))) = s(s(ks)(s(kk)(s(ks)(s(k(s(ks)))s))))(ks)$$

- this set of 5 Curry axioms denoted by  $A_\beta$
- a  $\lambda$ -algebra is a combinatory algebra satisfying the Curry axioms

## $\lambda$ -calculus vs combinatory algebra

- building  $\lambda$ -terms versus building combinatory terms

$$M, N ::= x \mid c \mid MN \mid \lambda x.M$$

$$A, B ::= x \mid c \mid AB \mid \mathbf{K} \mid \mathbf{S}$$

- rules and axioms of  $\lambda$ -calculus

$$(\alpha) \quad M =_{\alpha} N \Rightarrow M = N$$

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

$$(refl) \quad M = M$$

$$(symm) \quad M = N \Rightarrow N = M$$

$$(trans) \quad M = N, N = P \Rightarrow M = P$$

$$(cong) \quad M = M', N = N' \Rightarrow MN = M'N'$$

$$(\xi) \quad M = N \Rightarrow \lambda x.M = \lambda x.N$$

- rules and axioms of combinatory algebra

$$(k) \quad \mathbf{K}AB = A$$

$$(s) \quad \mathbf{S}ABC = AC(BC)$$

$$(refl) \quad A = A$$

$$(symm) \quad A = B \Rightarrow B = A$$

$$(trans) \quad A = B, B = C \Rightarrow A = C$$

$$(cong) \quad A = A', B = B' \Rightarrow AB = A'B'$$

$$(subst) \quad A = A' \Rightarrow A[B/x] = A'[B/x]$$

- For  $X$  = set of constants,  $\Lambda_X$  = all  $\lambda$ -terms obtained from rules and axioms of  $\lambda$ -calculus and  $\mathcal{T}(\mathfrak{M}_X)$  = all combinatory terms obtained from combinatory rules and axioms
- translators:**  $\lambda^* : \mathcal{T}(\mathfrak{M}_X) \rightarrow \Lambda_X$  and  $cl : \Lambda_X \rightarrow \mathcal{T}(\mathfrak{M}_X)$

## properties of translators

- $cl : \Lambda_X \rightarrow \mathcal{T}(\mathfrak{M}_X)$  does **not match  $\beta$ -reduction** to equivalence: for example  $(\lambda z. (\lambda x. x)z)_{cl} = \mathbf{S(KI)I}$  and  $(\lambda z. z)_{cl} = \mathbf{I}$  but  $\mathbf{S(KI)I}$  and  $\mathbf{I}$  are not equivalent according to combinatory rules and axioms

$$M \overset{\star}{\leftrightarrow}_{\beta} N \not\Rightarrow M_{cl} =_{CL} N_{cl}$$

- but have
  - 1 for any  $\lambda$ -term  $M$  have

$$M_{cl, \lambda^*} \overset{\star}{\leftrightarrow}_{\beta} M$$

- 2 for any  $\lambda$ -term  $M, N$

$$M_{cl} =_{CL} N_{cl} \Rightarrow M \overset{\star}{\leftrightarrow}_{\beta} N$$

- 3 for any  $A, B$  combinatory terms have

$$A =_{CL} B \Rightarrow A_{\lambda^*} \overset{\star}{\leftrightarrow}_{\beta} B_{\lambda^*}$$

## interpreting $\lambda$ -terms in combinatory algebra

- $\mathfrak{M} = (X, \cdot)$  combinatory algebra,  $\mathcal{V} = \{x_1, x_2, \dots, x_n, \dots\}$  set of variables
- given a  $\lambda$ -term  $M \in \Lambda_X$  with free variables  $x_1, \dots, x_n$  a **local interpretation** is determined by a **valuation of the variables**  
 $\rho : \mathcal{V} \rightarrow X$

$$\llbracket x \rrbracket_\rho := \rho(x), \quad \llbracket c \rrbracket_\rho := c, \quad \llbracket AB \rrbracket_\rho := \llbracket A \rrbracket_\rho \cdot \llbracket B \rrbracket_\rho$$

$$\llbracket \mathbf{K} \rrbracket_\rho := k, \quad \llbracket \mathbf{S} \rrbracket_\rho := s$$

- $A = B$  **holds locally** if  $\mathfrak{M} \models (A = B)$  which means for all valuations  $\rho : \mathcal{V} \rightarrow X$  have  $\llbracket A \rrbracket_\rho = \llbracket B \rrbracket_\rho$  in  $X$

## $\lambda$ -terms in $\lambda$ -algebras

- seen that  $\lambda$ -calculus and derivation according to axioms and rules of combinatory algebra are related by the translators  $\lambda^* : \mathcal{T}(\mathfrak{M}_X) \rightarrow \Lambda_X$  and  $\text{cl} : \Lambda_X \rightarrow \mathcal{T}(\mathfrak{M}_X)$  but are **not** equivalent (because  $\beta$ -reduction is not captured by equivalence in combinatory algebra)
- write  $\lambda \vdash (M = N)$  for obtained in  $\lambda$ -calculus (including e.g.  $\lambda \vdash (\lambda z. (\lambda x. x)z = \lambda z. z)$  involving  $\overset{*}{\leftrightarrow}_{\beta}$ )
- write  $\text{CL} \vdash (A = B)$  for derived with rules and axioms of combinatory algebra
- $A_{\beta}$  = set of 5 Curry axioms and  $\text{CL} + A_{\beta} \vdash (A = B)$  for derived with rules and axioms of  $\lambda$ -algebras
- main result:

$\lambda$  and  $\text{CL} + A_{\beta}$  are equivalent

$\lambda$  and  $\text{CL} + A_\beta$  are equivalent

- $\lambda \vdash (M_{cl, \lambda^*} = M)$  more precisely  $M_{cl, \lambda^*} \rightarrow_\beta M$
  - $\text{CL} + A_\beta \vdash (A_{\lambda^*, cl} = A)$
  - $\lambda \vdash (M = N) \iff \text{CL} + A_\beta \vdash (M_{cl} = N_{cl})$
  - $\text{CL} + A_\beta \vdash (A = B) \iff \lambda \vdash (A_{\lambda^*} = B_{\lambda^*})$
- 
- Henk Barendregt, *The Lambda Calculus: Its Syntax and Semantics*, 1984

## $\lambda$ -algebras and local interpretation

### other properties of $\lambda$ -algebras:

- 1 combinatory algebra  $\mathfrak{M} = (X, \cdot)$  such that for all combinatory terms  $A, B \in \mathcal{T}(\mathfrak{M}_X)$  have

$$A_{\lambda^*} \overset{\star}{\leftrightarrow}_{\beta} B_{\lambda^*} \Rightarrow \mathfrak{M} \models (A = B)$$

- 2 combinatory algebra  $\mathfrak{M} = (X, \cdot)$  such that for all  $M, N \in \Lambda_X$

$$M \overset{\star}{\leftrightarrow}_{\beta} N \Rightarrow \mathfrak{M} \models (M = N)$$

## $\lambda$ -models

- given a  $\lambda$ -algebra  $\mathfrak{M}$  as above
- also define combinators  $\mathbf{I} = \mathbf{SKK}$  and  $\mathbf{1} = \mathbf{SKI}$
- $\mathfrak{M}$  is a  $\lambda$ -model if it also satisfies Meyer-Scott axiom

$$(\forall x \ ax = bx) \Rightarrow \mathbf{1}a = \mathbf{1}b$$

## Cartesian Closed Categories (CCC)

- category  $\mathcal{C}$  is a Cartesian closed if
  - it has a **terminal object**  $\mathbb{1}$
  - it has **finite products**  $A \times B$  for objects  $A, B$  characterized by universal property (commutative diagram)

$$\begin{array}{ccccc} & & Y & & \\ & f_1 \swarrow & | & \searrow f_2 & \\ X_1 & & \downarrow f & & X_2 \\ & \xleftarrow{\pi_1} & X_1 \times X_2 & \xrightarrow{\pi_2} & \end{array}$$

- it has **exponentials**  $B^A$  for objects  $A, B$  characterized by

$$\text{Hom}_{\mathcal{C}}(X \times Y, Z) \cong \text{Hom}_{\mathcal{C}}(X, Z^Y)$$

and endowed with a morphism (evaluation)  $\varepsilon : B^A \times A \rightarrow B$

- for every  $f : C \times A \rightarrow B$  there is a unique morphism  $\lambda(f) : C \rightarrow B^A$  such that

$$f = \varepsilon \circ (\lambda(f) \times 1_A)$$

- diagram for evaluation  $\varepsilon$

$$\begin{array}{ccc}
 B^A \times A & \xrightarrow{\varepsilon} & B \\
 \lambda(f) \times A \uparrow & & \nearrow \\
 C \times A & & 
 \end{array}$$

- product  $f \times g = \langle f \circ \pi_1, g \circ \pi_2 \rangle$  for  $f : A \rightarrow B$  and  $g : C \rightarrow D$ : unique morphism  $h : A \times C \rightarrow B \times D$  making diagram commute

$$\begin{array}{ccccc}
 A & \xleftarrow{\pi_1} & A \times C & \xrightarrow{\pi_2} & C \\
 f \downarrow & & \vdots h & & \downarrow g \\
 B & \xleftarrow{\pi_1} & B \times D & \xrightarrow{\pi_2} & D
 \end{array}$$

- $\mathcal{C}$  has all finite products if products over any (finite) family  $X = \prod_i X_i$  with universal property (comm diagrams)

$$\begin{array}{ccc}
 & & X \\
 & \nearrow f & \downarrow \pi_i \\
 Y & \xrightarrow{f_i} & X_i
 \end{array}$$

- an object  $X$  in a Cartesian closed category  $\mathcal{C}$  **has enough points** if for any morphisms  $f, g : X \rightarrow X$  with  $f \neq g$  there is a point  $x : \mathbb{1} \rightarrow X$  such that  $f \circ x \neq g \circ x$
- the same then holds for morphisms  $f, g : X \rightarrow Y$
- in a *well pointed* category this holds for all objects

## Reflexive Object

- $\mathcal{C}$  Cartesian closed category
- **reflexive object**: triple  $\mathcal{U} = (U, \mathfrak{A}, \mathfrak{L})$  with  $U$  object of  $\mathcal{C}$  and morphisms  $\mathfrak{A} : U \rightarrow U^U$  and  $\mathfrak{L} : U^U \rightarrow U$  satisfying  $\mathfrak{A} \circ \mathfrak{L} = 1_{U^U}$
- i.e.  $U^U$  is a **retract** of  $U$
- **extensional** (or **strict**) reflexive object if also  $\mathfrak{L} \circ \mathfrak{A} = 1_U$ , so **isomorphism**  $U \simeq U^U$
- for  $f, g : U^n \rightarrow U$  have  $f \bullet g := \varepsilon \circ (\mathfrak{A} \times 1_U) \circ \langle f, g \rangle$  with  $\langle f, g \rangle : U^n \rightarrow U \times U$  and  $\varepsilon$  the evaluation

$$f \bullet g : U^n \xrightarrow{\langle f, g \rangle} U \times U \xrightarrow{\mathfrak{A} \times 1_U} U^U \times U \xrightarrow{\varepsilon} U$$

- $\mathfrak{M} = (X, \bullet)$  given by  $X = |U| = \{x : \mathbb{1} \rightarrow U\}$  set of “elements” (points) of the object  $U$  and  $\cdot$  operation given by

$$f \bullet g := \varepsilon \circ (\mathfrak{A} \times 1_U) \circ \langle f, g \rangle$$

- to verify this gives a  $\lambda$ -algebra
  - 1 representing  $\lambda$ -terms as morphisms  $U^n \rightarrow U$  produces syntactic description of a  $\lambda$ -algebra
  - 2 if the object  $U$  has enough points then it is also a  $\lambda$ -model

Sketch of the argument (following Barendregt)

## Representation of $\lambda$ -terms

- associate to a  $\lambda$  term  $M$  with a (finite) set  $\mathcal{I}$  of free variables a morphism

$$\llbracket M \rrbracket_{\mathcal{I}} : U^{\mathcal{I}} \rightarrow U$$

with product  $U^{\mathcal{I}} := \prod_{i \in \mathcal{I}} U_i$  of copies  $U_i$  of  $U$

- For  $\mathcal{I} = \{x_1, \dots, x_n\}$  a set of variables containing the set  $FV(M)$  of free variables of  $M$  define  $\llbracket M \rrbracket_{\mathcal{I}} : U^{\mathcal{I}} \rightarrow U$  inductively

## inductive procedure for representation of $\lambda$ -terms

- for a single variable  $\llbracket x_i \rrbracket_{\mathcal{I}} = \pi_{x_i}^{\mathcal{I}}$  with  $\pi_{x_i}^{\mathcal{I}} \langle t_1, \dots, t_n \rangle = t_i$
- for a constant  $c_a$ , with  $a : \mathbb{1} \rightarrow U$  a point, take  $\llbracket c_a \rrbracket_{\mathcal{I}} = a \circ !_{U^{\mathcal{I}}}$  with  $!_{U^{\mathcal{I}}} : U^{\mathcal{I}} \rightarrow \mathbb{1}$  the unique morphism to the terminal object
- for  $M = PQ$  take  $\llbracket PQ \rrbracket_{\mathcal{I}} = \llbracket P \rrbracket_{\mathcal{I}} \bullet \llbracket Q \rrbracket_{\mathcal{I}}$
- for  $M = \lambda x. P$  take  $\llbracket \lambda x. P \rrbracket_{\mathcal{I}} = \mathfrak{L} \circ \lambda(\llbracket P \rrbracket_{\mathcal{I}, x})$  where  $\lambda(f)$  is the unique morphism with  $f = \varepsilon \circ (\lambda(f) \times 1)$  and  $\mathfrak{L} : U^U \rightarrow U$
- so

$$U^{\mathcal{I}, x} \xrightarrow{\llbracket P \rrbracket_{\mathcal{I}, x}} U \quad \text{gives} \quad U^{\mathcal{I}} \xrightarrow{\lambda(\llbracket P \rrbracket_{\mathcal{I}, x})} U^U$$

so that

$$\llbracket \lambda x. P \rrbracket_{\mathcal{I}} : U^{\mathcal{I}} \xrightarrow{\lambda(\llbracket P \rrbracket_{\mathcal{I}, x})} U^U \xrightarrow{\mathfrak{L}} U$$

- given this representation  $M \mapsto \llbracket M \rrbracket_{\mathcal{I}} : U^{\mathcal{I}} \rightarrow U$  of  $\lambda$ -terms
- get valuations: take  $\rho : \mathbb{1} \rightarrow U$  a point  $\rho \in |U|$
- this determines  $\rho^{\mathcal{I}} : \mathbb{1} \rightarrow U^{\mathcal{I}}$ , written with notation  $\rho^{\mathcal{I}} = \langle \rho(x_1), \dots, \rho(x_n) \rangle$
- use these to obtain valuations (with  $\mathcal{I} = FV(M)$  set of free variables)

$$\llbracket M \rrbracket_{\rho} = \llbracket M \rrbracket_{\mathcal{I}} \circ \rho^{\mathcal{I}}$$

so  $\llbracket M \rrbracket_{\rho} \in |U|$

- can then show (Barendregt) that

$$\lambda \vdash (M = N) \quad \Rightarrow \quad \llbracket M \rrbracket = \llbracket N \rrbracket$$

- so every Cartesian closed category  $\mathcal{C}$  with a reflexive object  $U$  determines a  $\lambda$ -algebra  $(|U|, \bullet, \llbracket \rrbracket)$

## enough points and $\lambda$ -model

- combinator  $\mathbf{1} = \mathbf{S}(\mathbf{KI})$  satisfies (Barendregt)

$$\llbracket \mathbf{1}M \rrbracket_{\mathcal{I}} = \mathcal{L} \circ \mathcal{A} \llbracket M \rrbracket_{\mathcal{I}}$$

- if  $U \simeq U^U$  (so  $\mathcal{L} \circ \mathcal{A} = 1_U$ ) just  $\llbracket \mathbf{1}M \rrbracket_{\mathcal{I}} = \llbracket M \rrbracket_{\mathcal{I}}$
- if  $U$  has enough points: for all  $f, g : U \rightarrow U$  with  $f \neq g$  there is some  $x : \mathbb{1} \rightarrow U$  with  $f \circ x \neq g \circ x$
- so if for all  $x \in |U|$  have  $a \bullet x = b \bullet x$  in  $(|U|, \bullet)$

$$\begin{aligned} a \bullet x &= \varepsilon \circ (\mathcal{A} \times 1_U) \circ \langle a, x \rangle = \varepsilon \circ (\mathcal{A} \times 1_U) \circ \langle b, x \rangle = b \bullet x \\ &\Rightarrow \varepsilon \circ ((\mathcal{A} \circ a) \times 1_U) \circ \langle 1_{\mathbb{1}}, x \rangle = \varepsilon \circ ((\mathcal{A} \circ b) \times 1_U) \circ \langle 1_{\mathbb{1}}, x \rangle \end{aligned}$$

since  $U$  has enough points:

$$\varepsilon \circ ((\mathcal{A} \circ a) \times 1_U) = \varepsilon \circ ((\mathcal{A} \circ b) \times 1_U)$$

also show (Barendregt) that

$$\mathbf{1}c = \mathcal{L} \circ \mathcal{A} \circ c = \mathcal{L} \circ \lambda(\varepsilon \circ (\mathcal{A} \circ c) \times 1_U)$$

so get  $\mathbf{1}a = \mathbf{1}b$ :  $(|U|, \bullet, \llbracket \ \rrbracket)$  is a  $\lambda$ -model

## Diagonal arguments and Lawvere's fixed point theorem in categories

- a single fixed point theorem in Cartesian closed categories
- it accounts for all the “diagonal arguments” in logic
- Cantor diagonal and power sets cardinality, Russell paradox, Curry fixed point combinator of  $\lambda$ -calculus, Gödel's incompleteness
- **all of them are the same theorem** in different forms

## Preliminary: Yoneda embedding

### Yoneda Lemma:

- (small) category  $\mathcal{D}$  and functor  $K : \mathcal{D} \rightarrow \text{Sets}$ , object  $X \in \mathcal{D}$
- $\exists$  bijection  $\text{Nat}(\text{Hom}_{\mathcal{D}}(X, \cdot), K) \cong K(X)$  that maps natural transformation  $\alpha \in \text{Hom}_{\mathcal{D}}(X, \cdot) \rightarrow K$  to its image on  $\alpha_X(1_X)$
- in fact  $\alpha$  determined by  $\alpha_X(1_X)$ : for  $f : X \rightarrow Y$  have  $\alpha_Y(f) = K(f)(\alpha_X(1_X))$

$$\begin{array}{ccc} \text{Hom}_{\mathcal{D}}(X, X) & \xrightarrow{\alpha_X} & K(X) \\ f \circ - \downarrow & & \downarrow K(f) \\ \text{Hom}_{\mathcal{D}}(X, Y) & \xrightarrow{\alpha_Y} & K(Y) \end{array}$$

## characterization of natural transformations of representable functors:

- **representable functor**  $K : \mathcal{D} \rightarrow \text{Sets}$  if natural isomorphism  $\text{Hom}_{\mathcal{D}}(X, \cdot) \simeq K$
- for  $X, Y \in \mathcal{D}$  and natural transformation  $\alpha : \text{Hom}_{\mathcal{D}}(X, \cdot) \rightarrow \text{Hom}_{\mathcal{D}}(Y, \cdot) \cong \text{Hom}_{\mathcal{D}}(Y, X)$  (Yoneda Lemma) take  $\alpha(1_X) =: h : Y \rightarrow X$  then  $\alpha$  determined by this as  $\alpha(f) = f \circ h$

**Yoneda functor** (Yoneda embedding)  $\Upsilon : \mathcal{D} \rightarrow \text{Func}(\mathcal{D}^{op}, \text{Sets})$

$$\Upsilon : (h : A \rightarrow B) \mapsto (\text{Hom}_{\mathcal{D}}(h, \cdot) : \text{Hom}_{\mathcal{D}}(B, \cdot) \rightarrow \text{Hom}_{\mathcal{D}}(A, \cdot))$$

- by Yoneda Lemma bijection, get that this functor  $\Upsilon$  is full (surjective on morphisms) and faithful (injective on morphisms) so notion of **embedding** of categories

## Algebraic theories

- usually thought of as a set of  $k$ -ary operations (with specified axioms) that can be constructed inductively
- categorical formulation (due to Lawvere, 1963)
- an **algebraic theory** consists of
  - 1 a category  $\mathbb{A}$  with all finite products
  - 2 objects of  $\mathbb{A}$  form a sequence  $A^0, A^1, \dots, A^n, \dots$  with (for all  $n, m$ )

$$A^n \times A^m = A^{n+m}$$

- 3  $A^0 = \mathbb{1}$  is the **terminal object**
- **interpretation**: can think of objects  $A^n$  as the natural numbers  $n \in \mathbb{N}$
  - morphisms  $A^n \rightarrow A^m$  are given by a tuple  $(t_1, \dots, t_m)$  of **terms** depending on  $n$  free variables  $(x_1, \dots, x_n)$  (inputs)
  - write these as

$$(x_1 \dots x_n \vdash \langle t_1, \dots, t_m \rangle) : A^n \rightarrow A^m$$

- **composition as substitution:** morphisms  $(x_1 \dots x_k \vdash \langle t_1, \dots, t_m \rangle) : A^k \rightarrow A^m$  and  $(x_1 \dots x_m \vdash \langle u_1, \dots, u_n \rangle) : A^m \rightarrow A^n$  compose to

$$(x_1 \dots x_k \vdash \langle s_1, \dots, s_n \rangle) : A^k \rightarrow A^n$$

$$s_i = u_i[(x_1, \dots, x_m) := (t_1, \dots, t_m)]$$

substituting the terms  $t_j$  in place of the variables  $x_j$

- **identity morphism:**  $(x_1 \dots x_k \vdash \langle x_1, \dots, x_k \rangle)$
- **projections**  $\pi_i : A^n \rightarrow A$  of products:  $(x_1 \dots x_k \vdash x_i)$
- **axioms** of an algebraic theory are equations expressing relations between morphisms
- two morphisms equal if componentwise matching of terms, with equality of terms following from the axioms of the theory.
- **elements:** morphisms  $A^0 \rightarrow A^1$  where  $A^0 = \mathbb{1}$

## Models of algebraic theories

- **model** of a category  $\mathbb{A}$  in a category  $\mathcal{C}$  is a **functor**  $M : \mathbb{A} \rightarrow \mathcal{C}$  preserving all finite products
- **category of models**  $\text{Mod}_{\mathcal{C}}(\mathbb{A}) =$  category of functors  $M : \mathbb{A} \rightarrow \mathcal{C}$  preserving finite products, with natural transformations as morphisms
- **algebraic category**: equivalent to a category  $\text{Mod}_{\mathcal{C}}(\mathbb{A})$

## Example: algebraic theory of groups $\mathbb{G}$

- terms inductively built from

$$(x, y \vdash x \cdot y): G^2 \rightarrow G, \quad (x \vdash x^{-1}): G \rightarrow G, \quad (\vdash e): G,$$

- projections  $(x_1 \dots x_k \vdash x_i): G^n \rightarrow G$
- **models** in categories  $\mathcal{C}$ :
  - $\mathcal{C} = \text{Sets}$ : a group
  - $\mathcal{C} = \text{TopSp}$  topological spaces: a topological group
  - $\mathcal{C} = \text{Mfld}$  smooth manifolds: a Lie group
  - $\mathcal{C} = \text{CRings}$  commutative rings: a Hopf algebra
- $\text{Mod}_{\text{Sets}}(\mathbb{G}) = \text{Grps}$  the category of groups, etc

## universal models

- $\mathbb{A}$  an algebraic theory
- a **universal model** for  $\mathbb{A}$ :
  - $\mathcal{C}$  category with corresponding category of models  $\text{Mod}_{\mathcal{C}}(\mathbb{A})$
  - an object  $U \in \text{Mod}_{\mathcal{C}}(\mathbb{A})$  with property:
  - for any *terms*  $u, v$ :  
equality  $u = v$  holds in  $U$  iff  $\mathbb{A}$  proves that  $u = v$

## Yoneda embedding and universal models

the Yoneda embedding  $\mathbb{A} \rightarrow \text{Func}(\mathbb{A}^{op}, \text{Sets}) = [\mathbb{A}^{op}, \text{Sets}]$  in presheaves on  $\mathbb{A}$  is a universal model

- Yoneda functor preserves limits (in particular finite products), so is a model
- Yoneda functor is faithful, so any equation relating morphisms that is satisfied in the model is also satisfied in  $\mathbb{A}$

## Example: free group as universal group

- algebraic theory of groups  $\mathbb{G}$
- Yoneda embedding  $\mathbb{G} \rightarrow \text{Func}(\mathbb{G}^{op}, \text{Sets}) = [\mathbb{G}^{op}, \text{Sets}]$
- “group object”: the functor  $U = \text{Hom}_{\mathbb{G}}(\cdot, A^1)$
- this determines a *family of sets*  $U_n := \text{Hom}_{\mathbb{G}}(A^n, A^1)$
- set of terms  $t(x_1, \dots, x_n)$  on  $n$ -variables (with axioms of a group)
- so can identify  $U_n$  with the free group  $F_n$  on  $n$  generators
- universal model for the theory of groups is free group

## Lawvere fixed point theorem (1969)

- morphism  $g : A \rightarrow B$  in a Cartesian closed category is **point-surjective** if for any elements  $b : \mathbb{1} \rightarrow B$  there is an element  $a : \mathbb{1} \rightarrow A$  such that  $g \circ a = b$
- a **fixed point** of a morphism  $f : A \rightarrow A$  is an element  $a : \mathbb{1} \rightarrow A$  such that  $f \circ a = a$
- $\mathcal{C}$  a Cartesian closed category

if there exists in  $\mathcal{C}$  a point-surjective morphism  $d : A \rightarrow B^A$  then every morphism  $f : B \rightarrow B$  has a fixed point.

## proof

- **notation:** just write  $f(x)$  for  $f \circ x$  when  $x : \mathbb{1} \rightarrow X$  and  $f : X \rightarrow Y$
- $d$  is point-surjective: for every point  $h : \mathbb{1} \rightarrow B^A$  (i.e.  $h : A \rightarrow B$ )  $\exists x : \mathbb{1} \rightarrow A$  with  $d(x) = h : A \rightarrow B$
- take  $h(a) = f(d(a)(a))$ : there is an  $x : \mathbb{1} \rightarrow A$  with

$$d(x)(a) = f(d(a)(a))$$

- so if apply  $d(x)$  to  $x$  itself get

$$d(x)(x) = f(d(x)(x))$$

- so the element  $b : \mathbb{1} \rightarrow B$  given by  $b = d(x)(x)$  is a fixed point  $f(b) = b$

## Power set cardinality (Cantor, 1878)

for any set  $A$ , the power set  $\mathcal{P}(A)$  has cardinality strictly greater than  $A$

- Category Sets is Cartesian closed
- Subsets of  $A$  same as functions  $\chi : A \rightarrow \{0, 1\}$
- $\exists$  non-trivial permutation of the two-point set  $\{0, 1\}$  (i.e. negation)
- so there cannot be a point-surjective (i.e. surjective map of sets)  $A \rightarrow \{0, 1\}^A = \mathcal{P}(A)$  (otherwise it would imply all self maps of  $\{0, 1\}$  have a fixed point by Lawvere's theorem)

## Russell's paradox (1901)

- **property of sets** as  $P : \text{Sets} \rightarrow \{0, 1\}$

An inconsistency follows if any property of sets  $P : \text{Sets} \rightarrow \{0, 1\}$  defines a **set**  $\{Y \in \text{Sets} \mid P(Y) = 1\}$

- consider the membership relation  $\epsilon : \text{Sets} \rightarrow \{0, 1\}^{\text{Sets}}$

$$\epsilon(A)(B) = \begin{cases} 1 & B \in A \\ 0 & B \notin A \end{cases}$$

- this morphism would be point-surjective (surjective map of sets) is for all  $P \in \{0, 1\}^{\text{Sets}}$  there is an  $X_P \in \text{Sets}$  with  $\epsilon(X_P) = P : \text{Sets} \rightarrow \{0, 1\}$  i.e. if

$$P(Y) = \epsilon(X_P)(Y) = \begin{cases} 1 & Y \in X_P \\ 0 & Y \notin X_P \end{cases}$$

- i.e. if for every morphism  $P : \text{Sets} \rightarrow \{0, 1\}$  there is a set  $X_P = \{Y \in \text{Sets} \mid P(Y) = 1\}$

## Fixed points in $\lambda$ -calculus

every term in  $\lambda$ -calculus has a fixed point

- categorical model of  $\lambda$ -calculus provided by a **reflexive object**  $U$  in a Cartesian closed category: isomorphism  $U \rightarrow U^U$
- in particular this implies  $U \rightarrow U^U$  point-surjection
- $\lambda$ -terms seen as morphisms  $f : U \rightarrow U$  so by Lawvere's theorem any such morphism has a fixed point
- this is the same as the fixed point of a  $\lambda$ -term obtained via the Curry combinator

$$d(x) = \lambda u.f(d(u)(u))$$

with the fixed point

$$d(x)(x) = \lambda u.f(d(u)(u))x = f(d(x)(x))$$

## Gödel incompleteness (1935)

No consistent theory  $\mathbb{A}$  can express its own truth

- consider a category  $\mathbb{A}$  that is an algebraic theory with objects  $A^0, A^1, \dots, A^n, \dots$  (with  $A = A^1$ )
- $\tilde{\mathbb{A}}$  including an additional object  $\mathbf{2}$
- unary predicates: morphisms  $\varphi : A \rightarrow \mathbf{2}$  (“properties”)
- $\mathbb{A}$  is **consistent** if there is a morphism  $\sigma : \mathbf{2} \rightarrow \mathbf{2}$  (negation) such that  $\sigma \circ \varphi \neq \varphi$  for all predicates  $\varphi : A \rightarrow \mathbf{2}$
- **truth is definable** in  $\mathbb{A}$  if  $\exists$  truth morphism  $T : A \times A \rightarrow \mathbf{2}$  such that for any  $\varphi : A \rightarrow \mathbf{2}$  there is a “Gödel number”  $c : \mathbb{1} \rightarrow A$  with  $T(c, a) = \varphi(a)$  for  $a : \mathbb{1} \rightarrow A$
- **idea**: binary predicate  $T(c, a)$ : “is  $c$  a proof that  $a$  has property  $\varphi$ ”?  $\varphi$  provable if  $\varphi = T(c)$  for some  $c$
- but  $T : A \rightarrow \mathbf{2}^A$  cannot be point-surjective, otherwise  $\sigma : \mathbf{2} \rightarrow \mathbf{2}$  would have a fixed point ( $\mathbb{A}$  would be inconsistent)
- so there are predicates  $\varphi : A \rightarrow \mathbf{2}$  with  $\varphi \neq T(c)$

- to match Gödel incompleteness still need to show this applies to theories  $\mathbb{A}$  that extend Peano arithmetic
- take an algebraic theory  $\mathbb{A}$  where object  $A$  has  $\text{Hom}(A, A) = \{S^n\}_{n \in \mathbb{N}}$  with generator  $S : A \rightarrow A$ , and such that there exists a point  $0 : \mathbb{1} \rightarrow A$  with  $S^n(0) \neq S^m(0)$  for  $n \neq m$
- then for any object  $B = A^m$  of  $\mathbb{A}$ , any point  $b : \mathbb{1} \rightarrow B$ , and any morphism  $f : B \rightarrow B$  there is a morphism  $h : A \rightarrow B$  with commutative diagram

$$\begin{array}{ccccc}
 \mathbb{1} & \xrightarrow{0} & A & \xrightarrow{S} & A \\
 & \searrow b & \downarrow h & & \downarrow h \\
 & & B & \xrightarrow{f} & B
 \end{array}$$

so  $(A, 0, S)$  is an  $\mathbb{N}$ -object in the theory  $\mathbb{A}$  and this property formalizes Peano arithmetic

**Credits:** these slides incorporate material from

- Jean Gallier and Jocelyn Quaintance, *Proofs, Computability, Undecidability, Complexity, And the Lambda Calculus An Introduction*, 2025
- Mario Román García, *Category Theory and Lambda Calculus*, 2018
- Henk Barendregt, *The Lambda Calculus: Its Syntax and Semantics*, 1984
- Peter Selinger, *The Lambda Calculus is Algebraic*