

Multi-ary Models for Programming Languages*

Philip Saville , University of Oxford

Feb. 2024

What is denotational semantics?

- ① Assign a meaning $\llbracket \Gamma \vdash t : B \rrbracket$ to each program
- ② Answer questions about program behaviour using the model

What is denotational semantics?

- ① Assign a meaning $\llbracket \Gamma \vdash t : B \rrbracket$ to each program
- ② Answer questions about program behaviour using the model
 - reduce to algebraic questions
 - eg.
 - refuting $\models_{\mathcal{B}} \phi$ or $\vdash_{\mathcal{C}tx} \phi$
 - proving program transformations are safe
 - making precise what programs are

What is denotational semantics?

① Assign a meaning $\llbracket \Gamma \vdash t : B \rrbracket$ to each program

② Answer ~~questions about program behaviour using the model~~

~~→ reduce~~

~~to algebraic questions~~

- ~~e.g.~~
- refuting $\neg P_1$ or $\neg Q_1$
 - proving program transformations are safe
 - making precise what programs are

① Assign a meaning $[\Gamma \vdash t : B]$ to each program

① Assign a meaning $[\Gamma \vdash t : B]$ to each program

program
 $\Gamma \vdash t : B$

$C - g$
maps

morphism
 $[\Gamma] \xrightarrow{[t]} [B]$
in a category

[with enough structure
to model the language]

① Assign a meaning $[\Gamma \vdash t : B]$ to each program

program

$\Gamma \vdash t : B$

$t \rightarrow$
~~~  
 $\rightsquigarrow$

morphism

$[\Gamma] \xrightarrow{[t]} [B]$

in a category

[with enough structure  
to model the language]

soundness:  $t = t' \Rightarrow [t] = [t']$

completeness:  $[t] = [t']$  in every model  
 $\Rightarrow t = t'$

# Soundness and completeness



\* with extra structure  
depending on the type theory

# Soundness and completeness

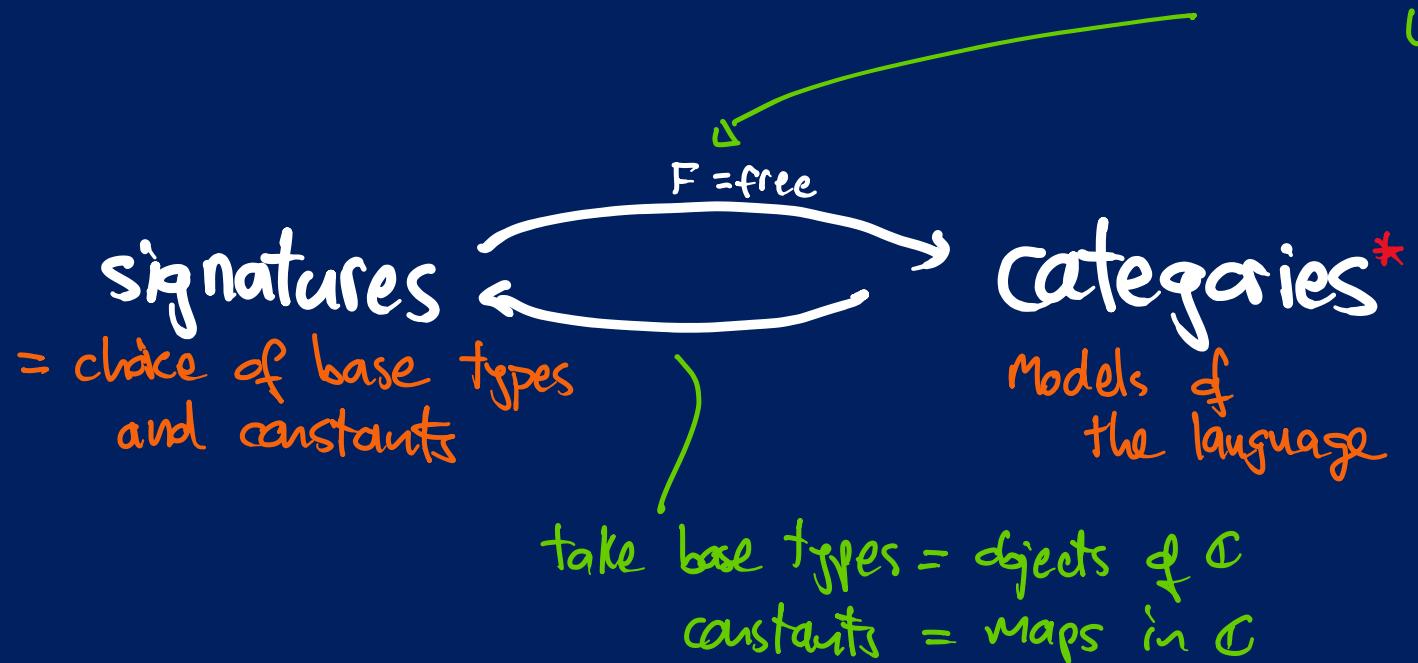
signatures ←  
= choice of base types  
and constants )

categories\*  
Models of  
the language

take base types = objects of  $\mathcal{C}$   
constants = maps in  $\mathcal{C}$

\* with extra structure  
depending on the type theory

# Soundness and completeness

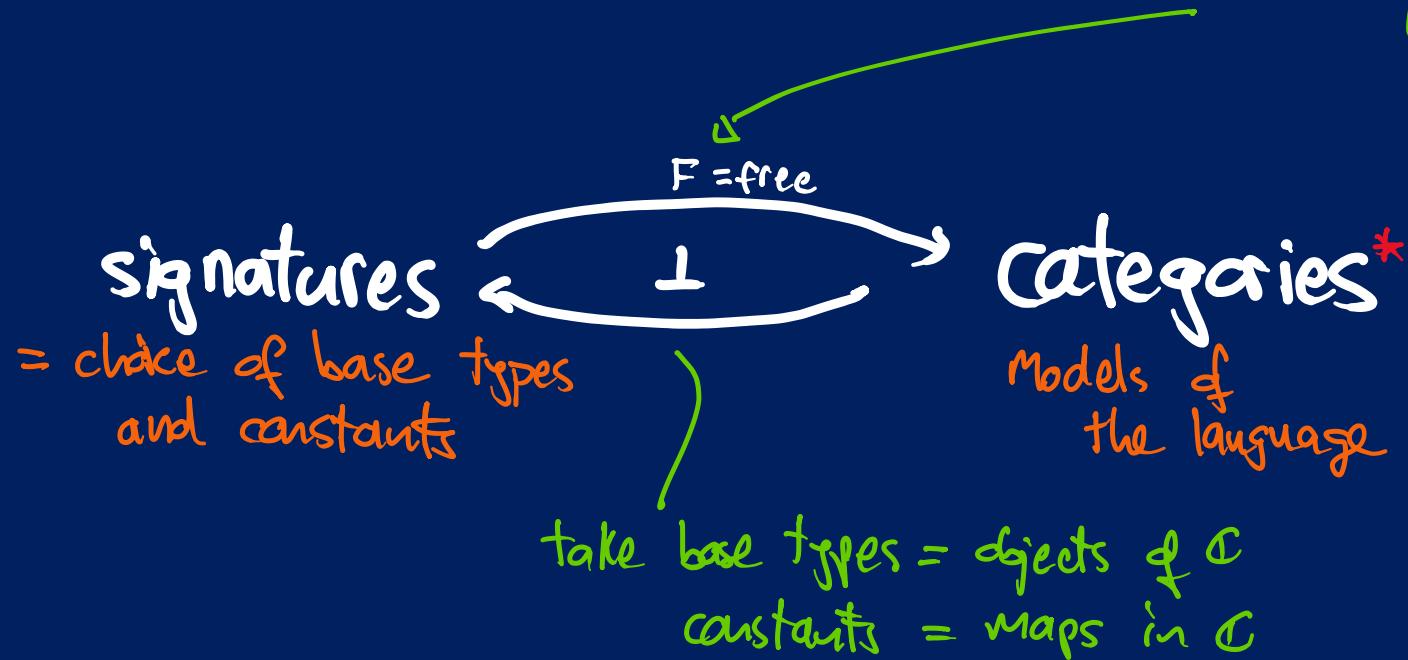


given a  
build a  
using  
the  
language

signature  $S$   
category  $F[S]$   
the syntax of  
language

\* with extra structure  
depending on the type theory

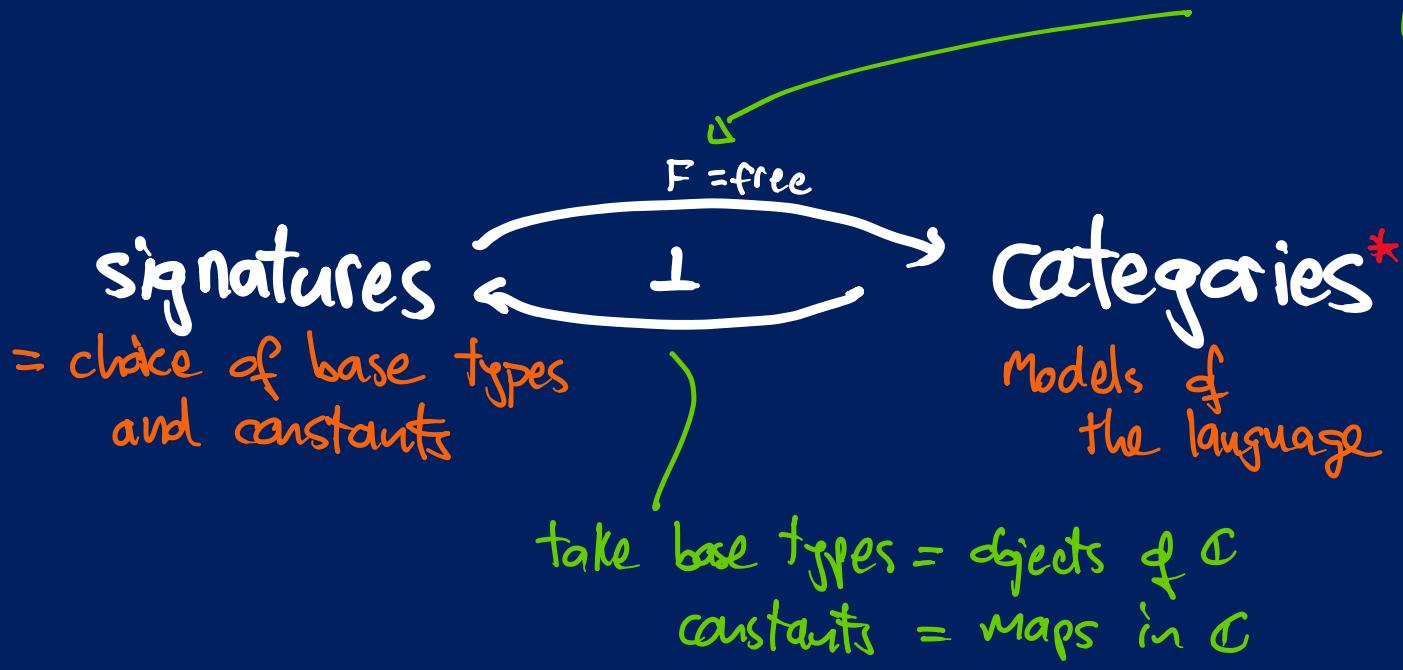
# Soundness and completeness



\* with extra structure  
depending on the type theory

# Soundness and completeness

given a signature  $S$   
build a category  $F[S]$   
using the syntax of the language



equivalently ...

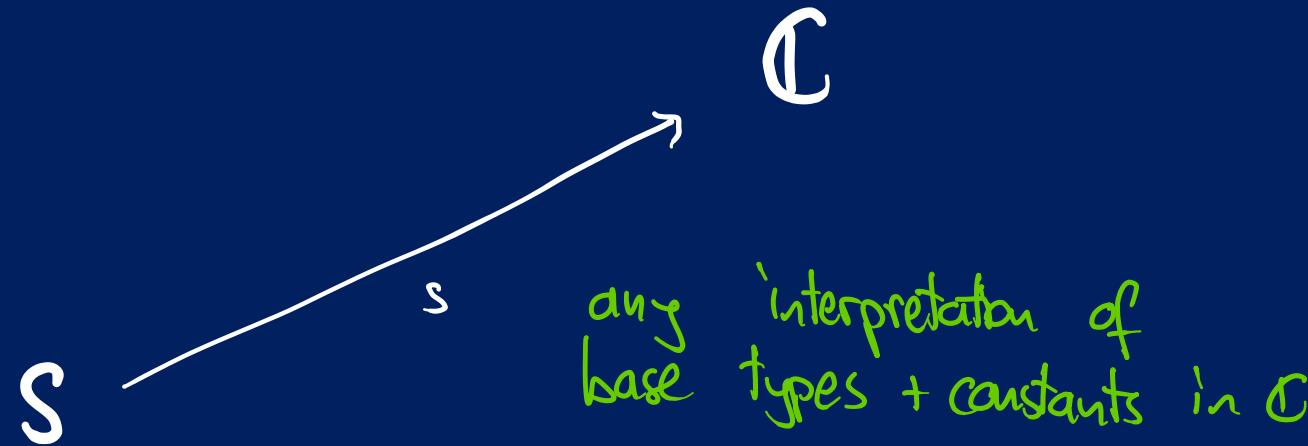
signatures  $\xrightarrow{\perp} \text{categories}$

for every signature  $S$  there is a free category\*  $F[S]$  such that:

\* with extra structure  
depending on the type theory

signatures  $\xrightarrow{F}$  categories

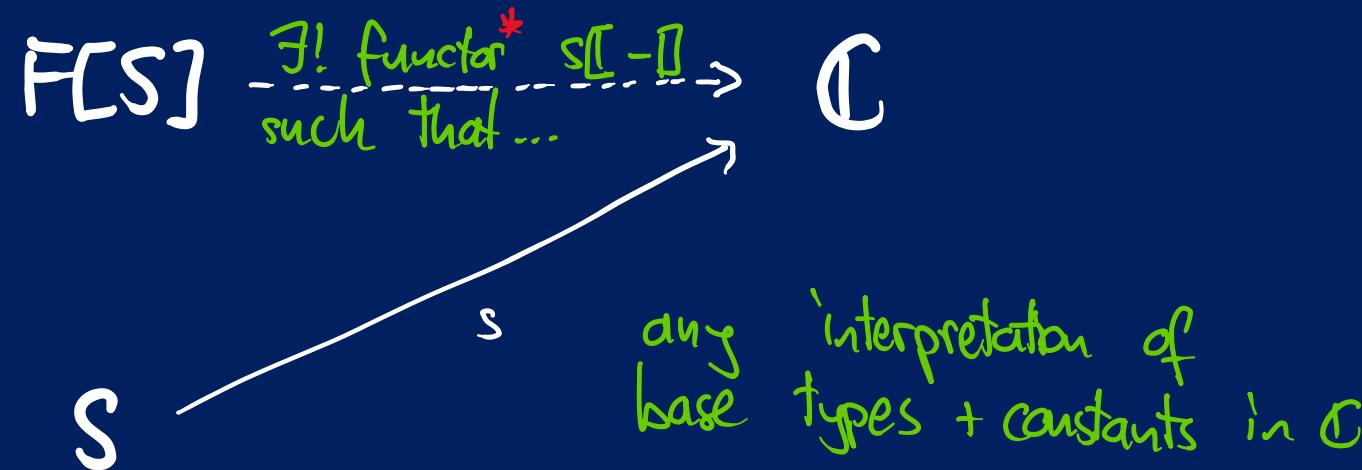
for every signature  $S$  there is a free category\*  $F[S]$  such that:



\* with extra structure  
depending on the type theory

signatures  $\xrightarrow{F}$  categories

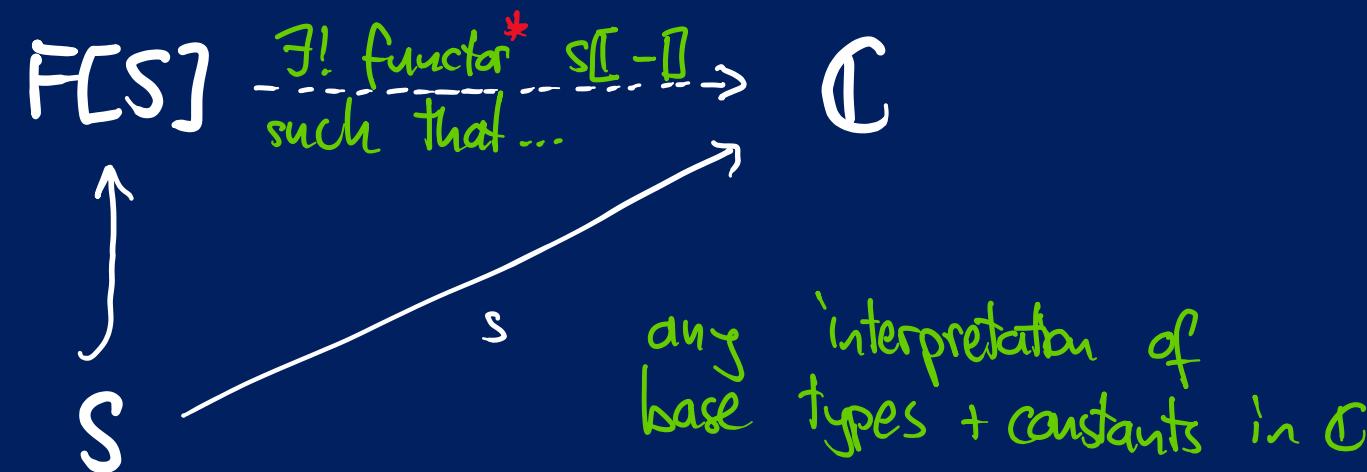
for every signature  $S$  there is a free category\*  $F[S]$  such that:



\* with extra structure  
depending on the type theory

signatures  $\xrightarrow{F}$  categories

for every signature  $S$  there is a free category\*  $F[S]$  such that:

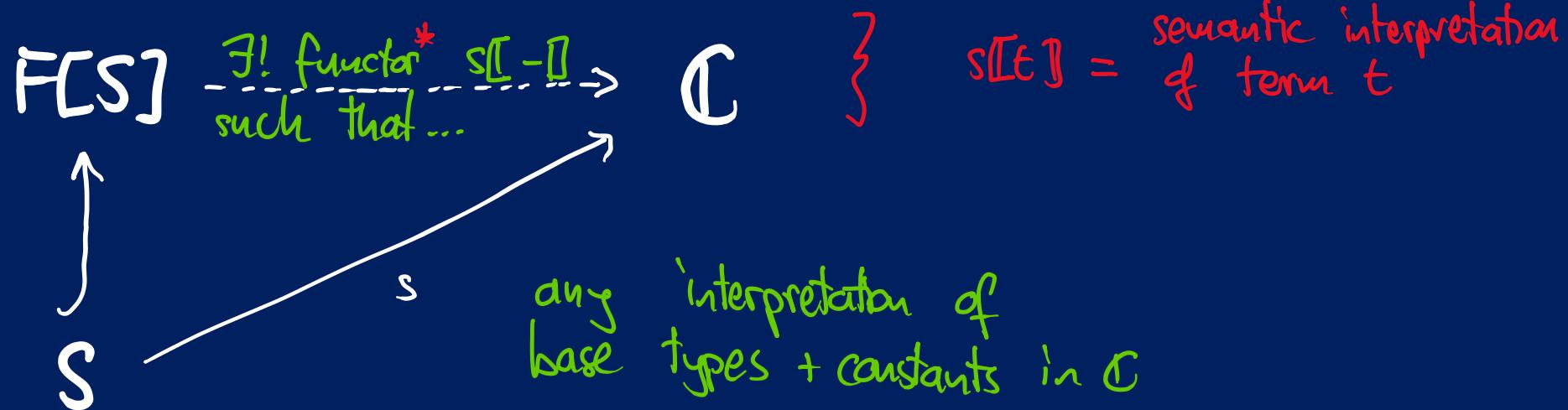


\* with extra structure  
depending on the type theory

signatures  $\xrightarrow{F}$  categories

for every signature  $S$  there is a free category\*  $F[S]$  such that:

maps  
= terms in the type theory



\* with extra structure  
depending on the type theory

signatures  $\xrightarrow{F} \perp$  categories

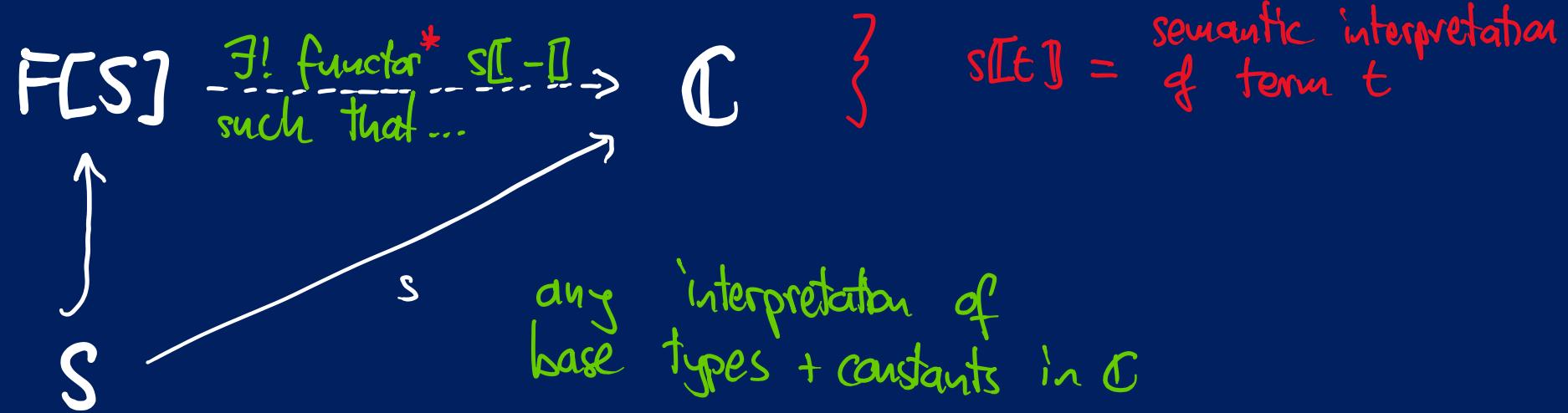
soundness =  $S[-]$  preserves all the structure

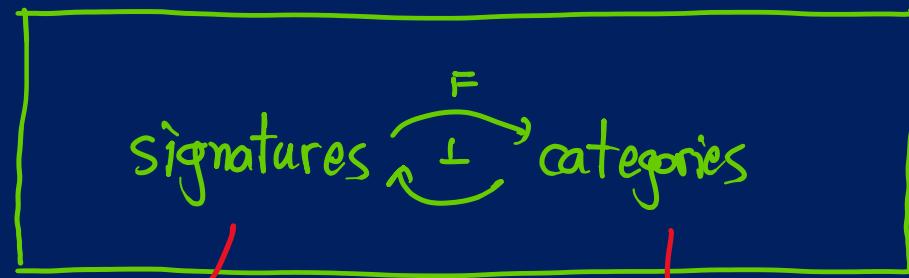
completeness =  $F[S]$  exists

(since the equations in  $F[S]$  are exactly the syntax)

for every signature  $S$  there is a category\*  $F[S]$  such that:

= terms in the type theory  
maps





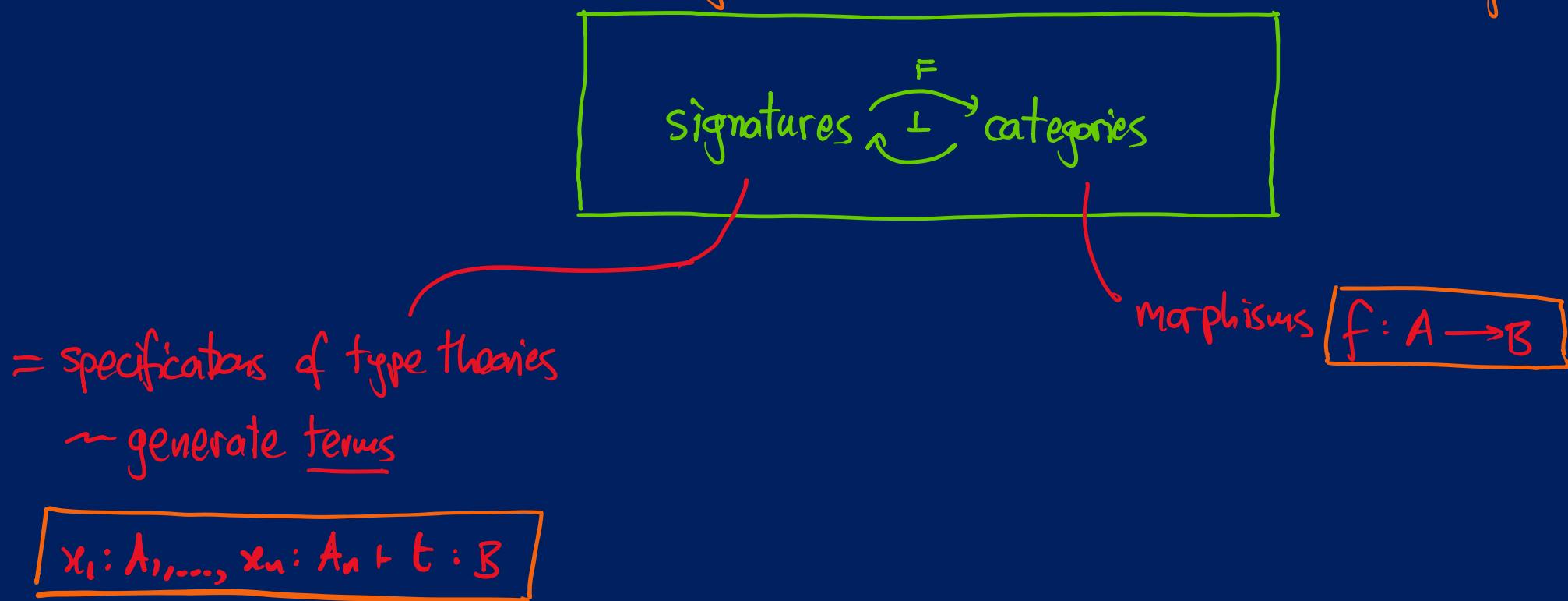
= specifications of type theories  
~ generate terms

$x_1 : A_1, \dots, x_n : A_n \vdash t : B$

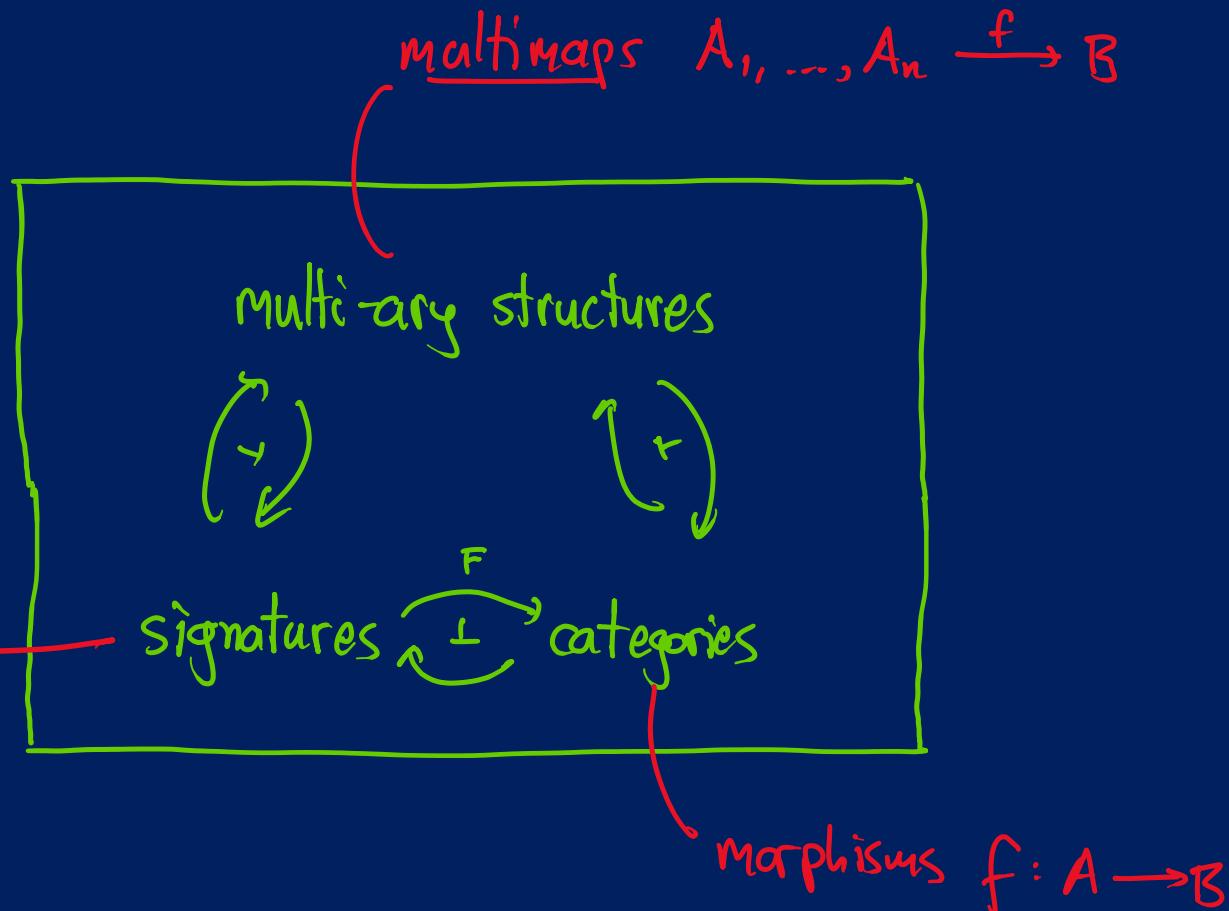
Morphisms  $f : A \rightarrow B$

## Some awkwardness

- ① have to restrict to terms in contexts of length 1 when defining  $F[S]$
- ② products get conflated with contexts
- ③  $\llbracket t \rrbracket$  in the free model is not  $t$  itself
- ④ need some fiddly inductions to show  $F[S]$  has the right structure



# This talk:



= specifications of type theories  
~ generate terms

$$x_1: A_1, \dots, x_n: A_n \vdash t : B$$

# This talk:

~ known to experts

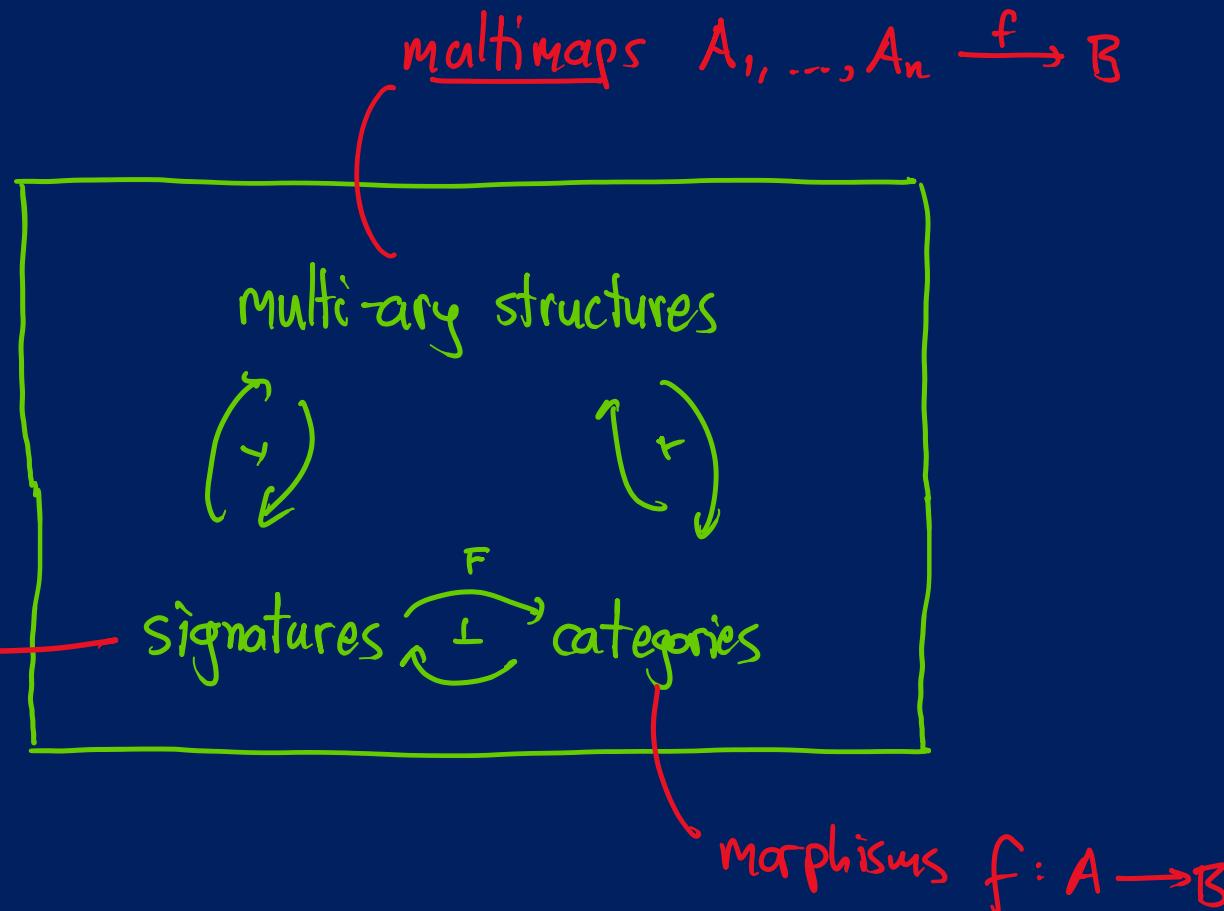
e.g., Lambek '69; Hyland '14, '15;  
Fiore et al '99, '10, '21, ...;  
Shulman '23; ... and others?

~ based on my Fossacs '24

= specifications of type theories

~ generate terms

$x_1 : A_1, \dots, x_n : A_n \vdash t : B$



# This talk:

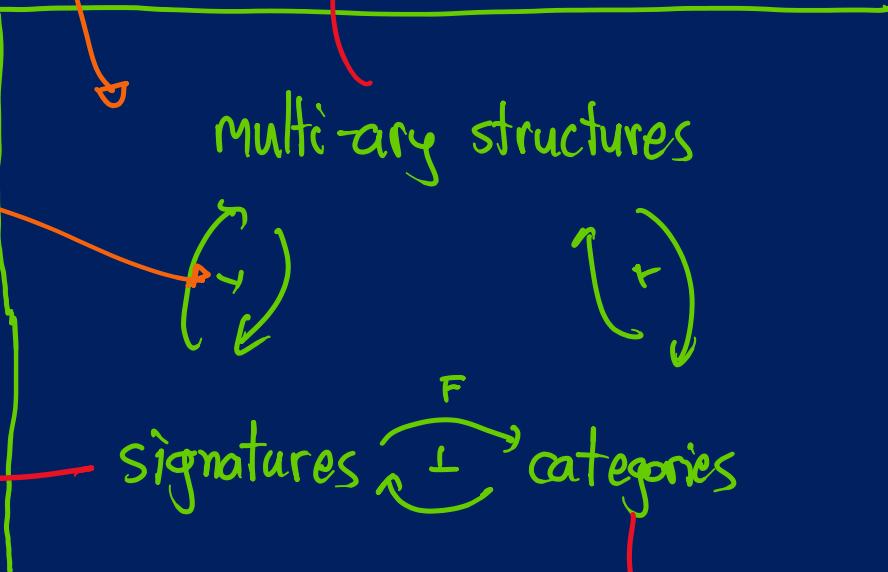
interpretation of  $t$   
in free model is  
 $t$ . itself

contexts separate from products

multimaps

$$\overbrace{A_1, \dots, A_n}^f \rightarrow B$$

easy to  
prove soundness  
and completeness



= specifications of type theories  
~ generate terms

$$x_1 : A_1, \dots, x_n : A_n \vdash t : B$$

↳ generalises easily e.g. gradings, refinements,  
via enrichment 2-dimensional, effectful,...

## This talk

- ① { easy to extract syntax from semantics  
easy to prove soundness and completeness  
distinguishes contexts and products
- ② tells us why strong monads appear in Moggi's work
- ③ relates  $\lambda$ -calculus models with models of CL

# Multi-ary structures

= has contraction, permutation,  
weakening

cartesian simple  
type theories

= has contraction, permutation,  
weakening

cartesian simple  
type theories



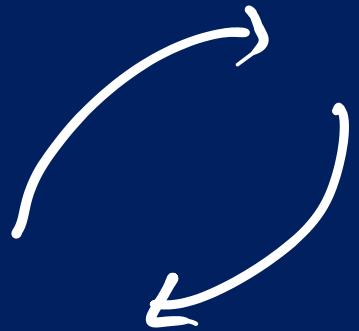
A white curved arrow originates from the word 'clones' at the top right and points downwards and to the left towards the text 'cartesian simple type theories'.

ordered / linear

~~cartesian~~ simple  
type theories

multicategories, symmetric  
multicategories

clones



- simple
- a type theory has:
- types  $A, B, C, \dots$
  - terms  $x_1 : A, \dots, x_n : A_n \vdash f : B$ ,  
including  $x_1 : A, \dots, x_n : A_n \vdash x_i : A_i$  for  $i = 1, \dots, n$
  - a substitution operation

$$\frac{x_1 : A, \dots, x_n : A \vdash f : B \quad (\Delta \vdash g_i : A_i)_{i=1 \dots n}}{\Delta \vdash f[g_{1/x_1}, \dots, g_{n/x_n}] : B}$$

- simple
- a type theory has:
- types  $A, B, C, \dots$
  - terms  $x_1 : A_1, \dots, x_n : A_n \vdash f : B$ ,  
including  $x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i$  for  $i = 1, \dots, n$
  - a substitution operation
- 
- $$\frac{x_1 : A_1, \dots, x_n : A_n \vdash f : B \quad (\Delta \vdash g_i : A_i)_{i=1 \dots n}}{\Delta \vdash f[g_1/x_1, \dots, g_n/x_n] : B}$$

- simple
- a type theory has:
- types  $A, B, C, \dots$
  - terms  $x_1 : A_1, \dots, x_n : A_n \vdash f : B$ ,  
including  $x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i$  for  $i = 1, \dots, n$
  - a substitution operation
- 
- $$\frac{x_1 : A_1, \dots, x_n : A_n \vdash f : B \quad (\Delta \vdash g_i : A_i)_{i=1 \dots n}}{\Delta \vdash f[g_1/x_1, \dots, g_n/x_n] : B}$$

simple  
A

a type theory has:

- types  $A, B, C, \dots$

- terms  $x_1 : A_1, \dots, x_n : A_n \vdash f : B$ ,

including  $x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i$  for  $i = 1, \dots, n$

- a substitution operation

$$\frac{x_1 : A_1, \dots, x_n : A_n \vdash f : B \quad (\Delta \vdash g_i : A_i)_{i=1 \dots n}}{\Delta \vdash f[g_1/x_1, \dots, g_n/x_n] : B}$$

simple  
A  
a type theory has:

- types  $A, B, C, \dots$
- terms  $x_1 : A_1, \dots, x_n : A_n \vdash f : B$ ,  
including  $x_i : A_i, \dots, x_n : A_n \vdash x_i : A_i$  for  $i = 1, \dots, n$
- a substitution operation

$$\frac{x_1 : A_1, \dots, x_n : A_n \vdash f : B \quad (\Delta \vdash g_i : A_i)_{i=1 \dots n}}{\Delta \vdash f[g_1/x_1, \dots, g_n/x_n] : B}$$

$$x_i[u_1/x_1, \dots, u_n/x_n] = u_i$$

$$t[x_1, \dots, x_n] = t$$

$$t[u_1/x_1, \dots, u_n/x_n][v_1/y_1, \dots, v_m/y_m] = t[\dots, u_i[v_i/y_i, \dots] / x_i, \dots]$$

multisorted, abstract

def: a  $\wedge$  clone  $C$  has:

[Hall]

- objects  $A, B, C, \dots$
- multimaps  $f, g, \dots : A_1, \dots, A_n \rightarrow B$ ,  
including  $p_i^{A_1, \dots, A_n} : A_1, \dots, A_n \rightarrow A_i$  for  $i = 1 \dots n$
- a substitution operation

$$f : A_1, \dots, A_n \rightarrow B \quad (g_i : \Delta \rightarrow A_i)_{i=1 \dots n}$$

---

$$f[g_1, \dots, g_n] : \Delta \rightarrow B$$

multisorted, abstract

def: a  $\wedge$  clone  $\mathbb{C}$  has:

[Hall]

- objects  $A, B, C, \dots$
- multimaps  $f, g, \dots : A_1, \dots, A_n \rightarrow B$ ,  
including  $p_i^{A_1, \dots, A_n} : A_1, \dots, A_n \rightarrow A_i$  for  $i = 1, \dots, n$
- a substitution operation

$$f : A_1, \dots, A_n \rightarrow B \quad (g_i : \Delta \rightarrow A_i)_{i=1 \dots n}$$

---

$$f[g_1, \dots, g_n] : \Delta \rightarrow B$$

multisorted, abstract

def: a  $\wedge$  clone  $C$  has:

[Hall]

- objects  $A, B, C, \dots$
- multimaps  $f, g, \dots : A_1, \dots, A_n \rightarrow B$ ,  
including  $p_i^{A_1, \dots, A_n} : A_1, \dots, A_n \rightarrow A_i$  for  $i = 1, \dots, n$
- a substitution operation

$$f : A_1, \dots, A_n \rightarrow B \quad (g_i : \Delta \rightarrow A_i)_{i=1 \dots n}$$

---

$$f[g_1, \dots, g_n] : \Delta \rightarrow B$$

multisorted, abstract

def: a  $\wedge$  clone  $C$  has:

[Hall]

- objects  $A, B, C, \dots$
- multimaps  $f, g, \dots : A_1, \dots, A_n \rightarrow B$ ,  
including  $p_i^{A_1, \dots, A_n} : A_1, \dots, A_n \rightarrow A_i$  for  $i = 1, \dots, n$
- a substitution operation

$$f : A_1, \dots, A_n \rightarrow B \quad (g_i : \Delta \rightarrow A_i)_{i=1, \dots, n}$$

---

$$f[g_1, \dots, g_n] : \Delta \rightarrow B$$

multisorted, abstract

def: a  $\wedge$  clone  $C$  has:

[Hall]

- objects  $A, B, C, \dots$
- multimaps  $f, g, \dots : A_1, \dots, A_n \rightarrow B$ ,  
including  $p_i^{A_1, \dots, A_n} : A_1, \dots, A_n \rightarrow A_i$  for  $i = 1, \dots, n$
- a substitution operation

$$f : A_1, \dots, A_n \rightarrow B \quad (g_i : \Delta \rightarrow A_i)_{i=1, \dots, n}$$

---

$$f[g_1, \dots, g_n] : \Delta \rightarrow B$$

$$p_i[f_1, \dots, f_n] = f_i$$

$$f[p_1, \dots, p_n] = f$$

$$(f[g_1, \dots, g_n])[h_1, \dots, h_m] = f[\dots, g_i[h_i], \dots]$$

( $n > 0$ )

cartesian simple  
type theories

clones  $\simeq$

- algebraic theories
- substitution algebras  
[Turi, Plotkin, Felleisen '99]
- cartesian multicategories
- ...

cartesian simple  
type theories

clones  $\simeq$

- algebraic theories
- substitution algebras  
[Turi, Plotkin, Fiole '99]
- cartesian multicategories
- ...

A GENERAL THEORY

type theories with structure

s1  
monads on the category  
of clones

(see Arkor-McDermott 2021,  
Arkor-Fiore 2020,...)

# Syntax from semantics

# simply-typed $\lambda$ -calculus

for any signature of  
base types and constants

base  
types

types :  $A, B ::= \beta \mid \perp \mid A \times B \mid A \rightarrow B$

terms :  $t, u ::= x \mid c \mid () \mid \langle t, u \rangle \mid \pi_i(t) \mid \lambda x. t \mid t \ u$   
 $\{ i = 1, 2 \}$   
constants

equations:

|             |                                          |                             |
|-------------|------------------------------------------|-----------------------------|
| ( $\beta$ ) | $\pi_i(\langle t_1, t_2 \rangle) = t_i$  | $(\lambda x. t) u = t[x/x]$ |
| ( $\eta$ )  | $\langle \pi_1(t), \pi_2(t) \rangle = t$ | $\lambda x(t x) = t$        |

$\wedge^x$  = just products

$\wedge^\rightarrow$  = just exponentials

$\wedge^{x,\rightarrow}$  = products + exponentials

# Modelling products

Cartesian product in category  $\mathbb{C}$  = Universal arrow from  
 $\Delta^{(n)} : \mathbb{C}^{x^n} \longrightarrow \mathbb{C}$   
to  $(A_1, \dots, A_n) \in \mathbb{C}^{x^n}$

# Modelling products

Cartesian product in  $\mathcal{C}$   $\stackrel{\text{def}}{=}$  Universal arrow from  
 $\Delta^{(n)} : \mathcal{C}^{x^n} \longrightarrow \mathcal{C}$   
to  $(A_1, \dots, A_n) \in \mathcal{C}^{x^n}$

# Modelling products

a cartesian product in a cbne  $\mathcal{C}$  consists of:

1) an object  $A_1 \times A_2$

2) multimaps  $\pi_i : A_1 \times A_2 \rightarrow A_i$

s.t.

$$t \longmapsto (\pi_1[t], \pi_2[t])$$

$$\mathcal{C}(\Gamma; A_1 \times A_2) \longrightarrow \mathcal{C}(\Gamma; A_1) \times \mathcal{C}(\Gamma; A_2)$$

is an iso.

# Modelling products

a cartesian product in a cbne  $\mathcal{C}$  consists of:

1) an object  $A_1 \times A_2$

$$\frac{A_1 \text{ type } A_2 \text{ type}}{A_1 \times A_2 \text{ type}}$$

2) multimaps  $\pi_i : A_1 \times A_2 \rightarrow A_i$

s.t.

$$t \mapsto (\pi_1[t], \pi_2[t])$$

$$\mathcal{C}(\Gamma; A_1 \times A_2) \longrightarrow \mathcal{C}(\Gamma; A_1) \times \mathcal{C}(\Gamma; A_2)$$

is an iso.

# Modelling products

a cartesian product in a cbne  $\mathcal{C}$  consists of:

1) an object  $A_1 \times A_2$

2) multimaps  $\pi_i : A_1 \times A_2 \rightarrow A_i$

s.t.

$$t \mapsto (\pi_1[t], \pi_2[t])$$

$$\mathcal{C}(\Gamma; A_1 \times A_2) \longrightarrow \mathcal{C}(\Gamma; A_1) \times \mathcal{C}(\Gamma; A_2)$$

is an iso.

$$A_1 \text{ type } A_2 \text{ type}$$

$$A_1 \times A_2 \text{ type}$$

$$p : A_1 \times A_2 \vdash \pi_i(p) : A_i$$

# Modelling products

a cartesian product in a cbne  $\mathcal{C}$  consists of:

1) an object  $A_1 \times A_2$

$A_1, \text{ type } A_2 \text{ type}$

$A_1 \times A_2 \text{ type}$

2) multimaps  $\pi_i : A_1 \times A_2 \rightarrow A_i$

$p : A_1 \times A_2 \vdash \pi_i(p) : A_i$

s.t.

$$t \mapsto (\pi_1(t), \pi_2(t))$$

$$\mathcal{C}(\Gamma; A_1 \times A_2) \longrightarrow \mathcal{C}(\Gamma; A_1) \times \mathcal{C}(\Gamma; A_2)$$

is an iso.

$$(\Gamma \vdash t : A_1 \times A_2) \longmapsto (\Gamma \vdash \pi_i(p)[t] : A_i)_{i=1,2}$$

# Modelling products

a cartesian product in a cbne  $\mathcal{C}$  consists of:

1) an object  $A_1 \times A_2$

2) multimaps  $\pi_i : A_1 \times A_2 \rightarrow A_i$

s.t.

$$t \mapsto (\pi_1[t], \pi_2[t])$$

$$\mathcal{C}(\Gamma; A_1 \times A_2) \xrightarrow{\quad} \mathcal{C}(\Gamma; A_1) \times \mathcal{C}(\Gamma; A_2)$$

is an iso.

$$(\Gamma \vdash t : A_1 \times A_2) \mapsto (\Gamma \vdash \pi_i(p)[t] : A_i)_{i=1,2}$$

$$(\Gamma \vdash \langle u_1, u_2 \rangle : A_1 \times A_2) \leftarrow (\Gamma \vdash u_i : A_i)_{i=1,2}$$

$$A_1 \text{ type } A_2 \text{ type}$$

$$A_1 \times A_2 \text{ type}$$

$$p : A_1 \times A_2 \vdash \pi_i(p) : A_i$$

# Modelling products

a cartesian product in a cbne  $\mathcal{C}$  consists of:

1) an object  $A_1 \times A_2$

2) multimaps  $\pi_i : A_1 \times A_2 \rightarrow A_i$

s.t.

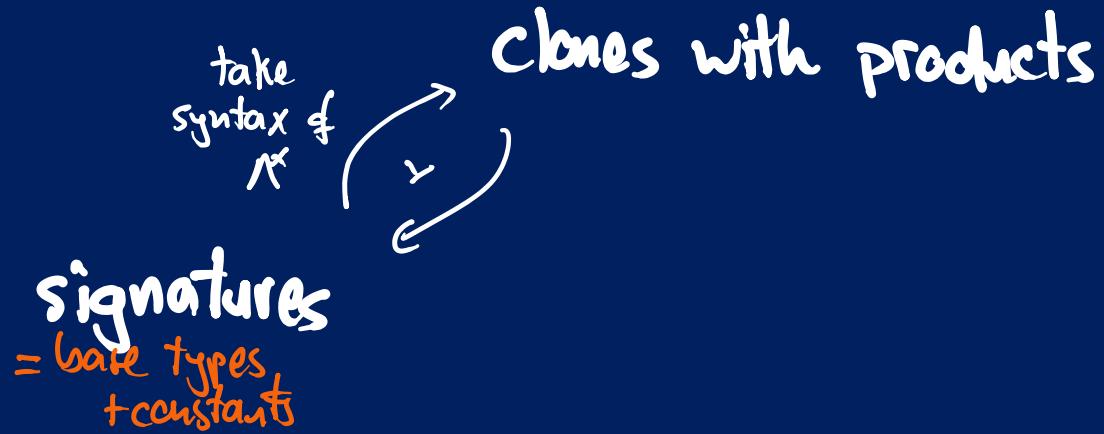
$$\begin{array}{c} t \mapsto (\pi_1(t), \pi_2(t)) \\ \mathcal{C}(\Gamma; A_1 \times A_2) \xrightarrow{\quad} \mathcal{C}(\Gamma; A_1) \times \mathcal{C}(\Gamma; A_2) \\ \pi_i(p)[\langle u_1, u_2 \rangle / p] = u_i \\ \langle \pi_1(p), \pi_2(p) \rangle = p \text{ is an iso.} \end{array}$$
$$\begin{array}{c} (\Gamma \vdash t : A_1 \times A_2) \longleftarrow (\Gamma \vdash \pi_i(p)[t/p] : A_i)_{i=1,2} \\ (\Gamma \vdash \langle u_1, u_2 \rangle : A_1 \times A_2) \longleftarrow (\Gamma \vdash u_i : A_i)_{i=1,2} \end{array}$$

$$\frac{A_1 \text{ type } A_2 \text{ type}}{A_1 \times A_2 \text{ type}}$$

$$p : A_1 \times A_2 \vdash \pi_i(p) : A_i$$

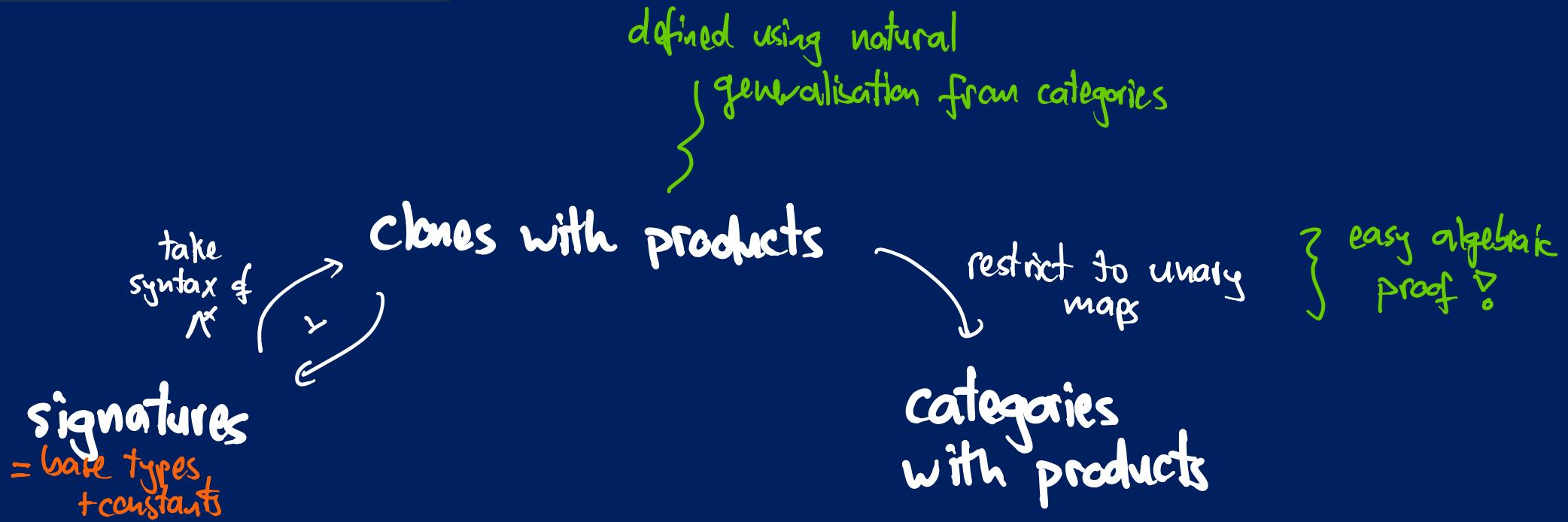
# Modelling products

defined using natural  
generalisation from categories



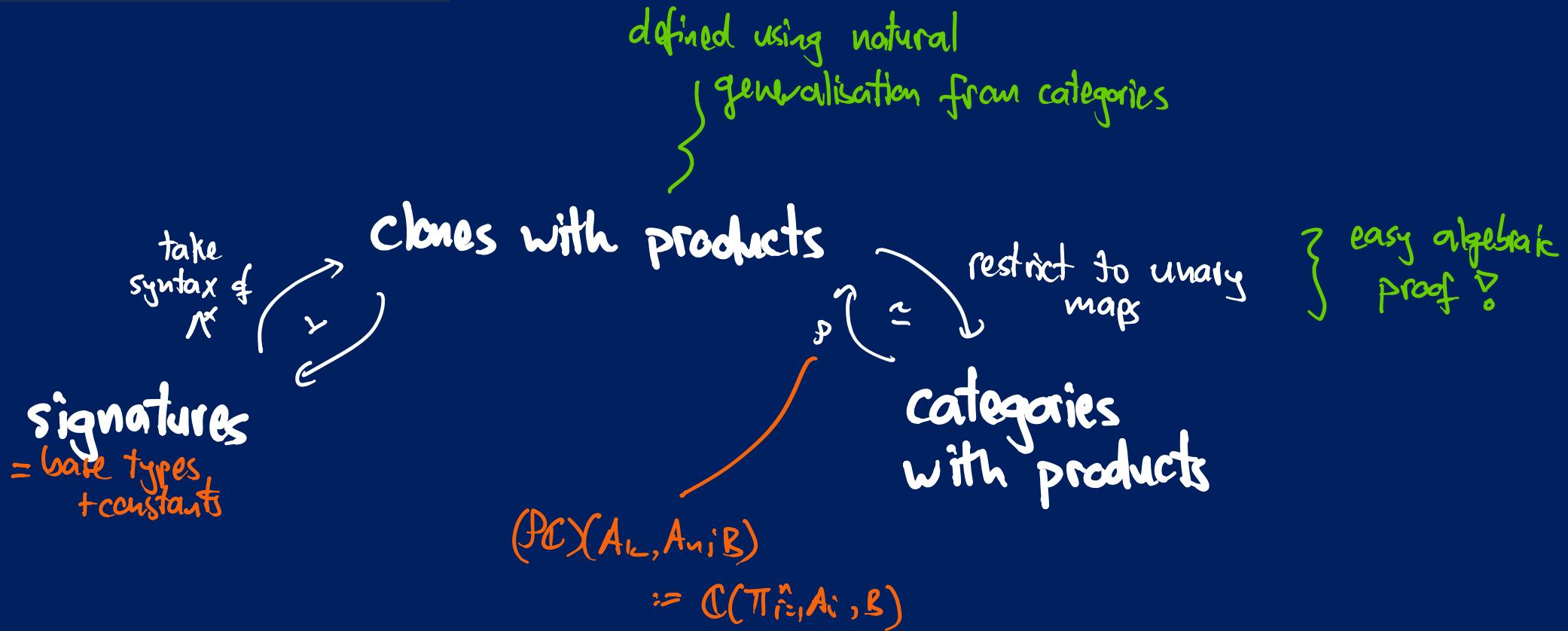
free clone with products = syntax of  $\wedge^x$

# Modelling products



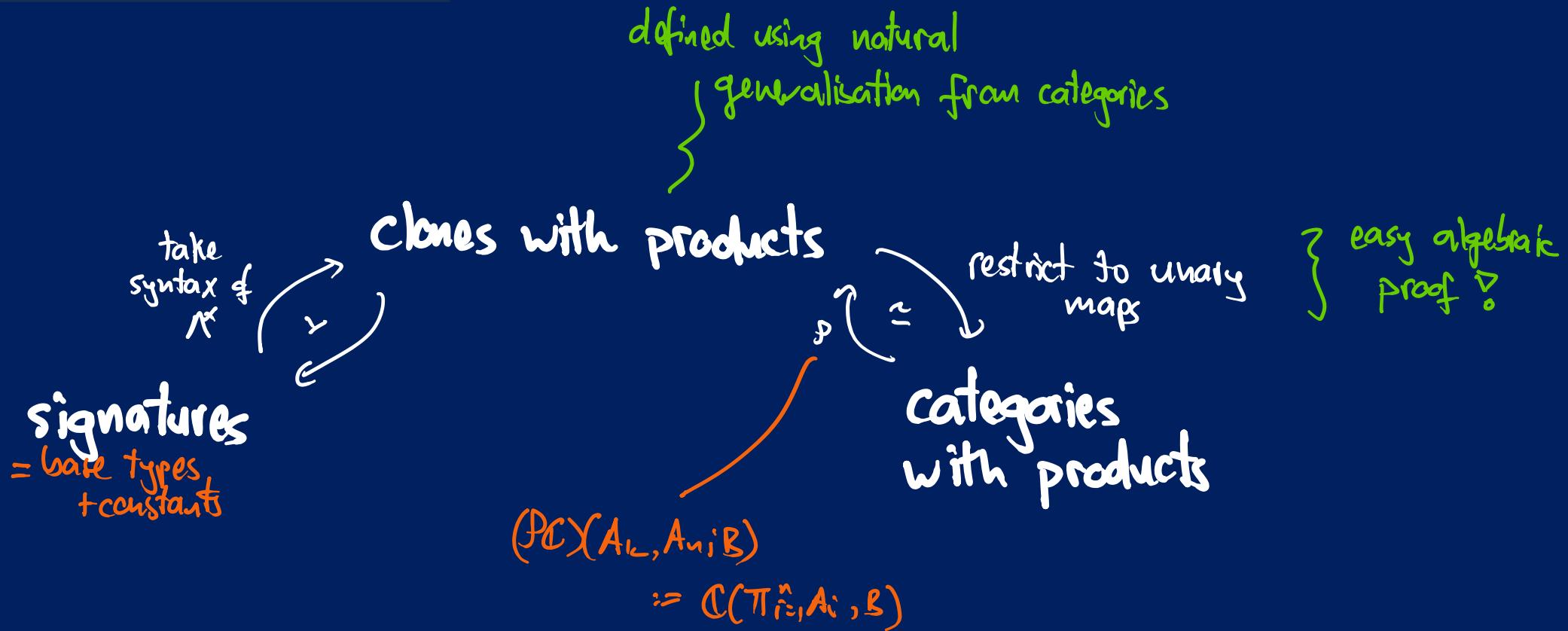
free clone with products = syntax of  $\wedge^X$

# Modelling products



free clone with products = syntax of  $\wedge^x$

# Modelling products

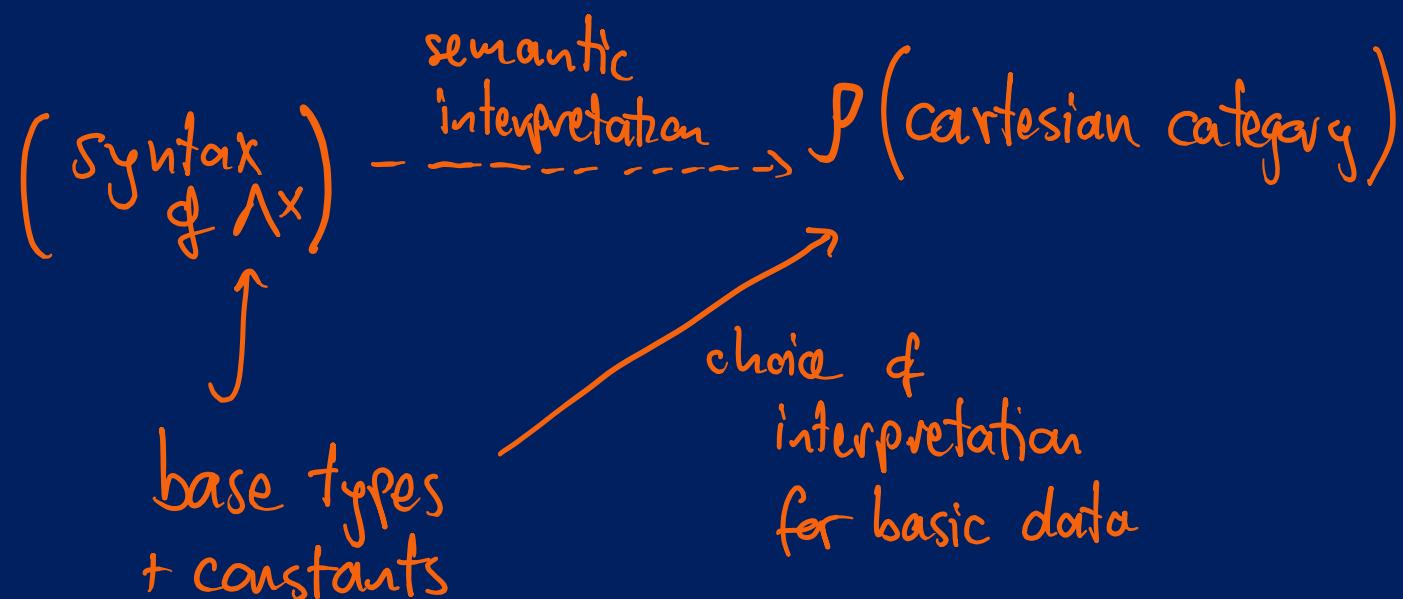


free clone with products = syntax of  $\Lambda^x$

free cartesian category =  $\Lambda^x$ -terms  $x:A + t:B$

# Modelling products: soundness and completeness

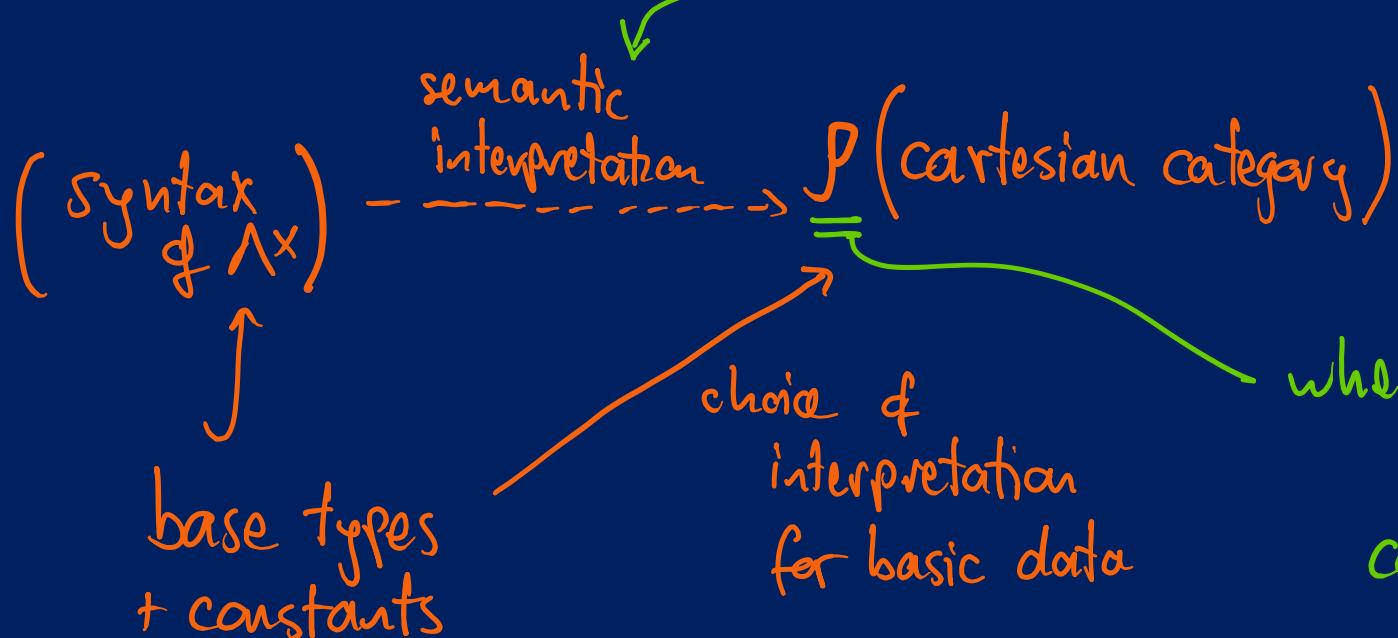
free clone with  
all products      =      syntax of  $\lambda x$       =  
typed  $\lambda$ -calculus  
with just products



# Modelling products: soundness and completeness

free clone with  
all products

= syntax of  $\Lambda^x$  = typed  $\lambda$ -calculus  
with just products



where products  
= contexts  
comes from!

# Modelling products

natural definition of products for clones

$\Rightarrow \left\{ \begin{array}{l} \text{syntax for } \wedge^x \\ \text{soundness + completeness} \\ \text{equivalence with categorical models} \end{array} \right.$

# Modelling products

in the linear setting, this is  
has you derive &

natural definition of products for clones

$\Rightarrow \left\{ \begin{array}{l} \text{syntax for } \wedge^x \\ \text{soundness + completeness} \\ \text{equivalence with categorical models} \end{array} \right.$

# Modelling effects

# Modelling effects

natural definition of monads for clones

$\Rightarrow \left\{ \begin{array}{l} \text{syntax for Maggi's } \lambda\text{ml} \\ \text{soundness + completeness} \\ \text{equivalence with categorical models} \end{array} \right.$

# Modelling effects

Moggi's insight:

the structure of effectful programs

is captured by a strong monad

the structure of effectful  
programming is captured  
by a strong monad

---

the structure of effectful  
programming is captured  
by a strong monad

- 
- ① mark types as effectful

$$\frac{\text{A type}}{\text{TA type}}$$

the structure of effectful  
programming is captured  
by a strong monad

↳  
TA = A + I  
TA = List(A)  
TA = S<sup>\*</sup> × A

① mark types as effectful

$$\frac{\text{A type}}{\text{TA type}}$$

the structure of effectful  
programming is captured  
by a strong monad

$$\text{eg/} \quad \begin{aligned} TA &= A + 1 \\ TA &= \text{List}(A) \\ TA &= S^* \times A \end{aligned}$$

- ① mark types as effectful
- ② every pure program is trivially effectful

$$\frac{A \text{ type}}{TA \text{ type}}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \text{return}(t) : TA}$$

the structure of effectful  
programming is captured  
by a strong monad

$$\text{eg/} \quad \begin{aligned} TA &= A + 1 \\ TA &= \text{List}(A) \\ TA &= S^* \times A \end{aligned}$$

$$\text{eg/} \quad \begin{aligned} A &\xrightarrow{\text{inl}} A + 1 \\ A &\longrightarrow \text{List } A \\ a &\longmapsto [a] \end{aligned}$$

$$\begin{aligned} A &\longrightarrow S^* \times A \\ a &\longmapsto (\varepsilon, a) \end{aligned}$$

- ① mark types as effectful
- ② every pure program is trivially effectful

$$\frac{A \text{ type}}{TA \text{ type}}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \text{return}(t) : TA}$$

the structure of effectful  
programming is captured  
by a strong monad

$$\text{eg/} \quad \begin{aligned} TA &= A + I \\ TA &= \text{List}(A) \\ TA &= S^* \times A \end{aligned}$$

$$\text{eg/} \quad \begin{aligned} A &\xrightarrow{\text{inl}} A + I \\ A &\longrightarrow \text{List } A \\ a &\longmapsto [a] \end{aligned}$$

$$\begin{aligned} A &\longrightarrow S^* \times A \\ a &\longmapsto (\varepsilon, a) \end{aligned}$$

- ① mark types as effectful
- ② every pure program is trivially effectful
- ③ explicit sequencing by let-binding

$$\frac{A \text{ type}}{TA \text{ type}}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \text{return}(t) : TA}$$

$$\frac{\Gamma \vdash u : TA \quad \Gamma, x : A \vdash t : TB}{\Gamma \vdash \text{let } x = u \text{ in } t : TB}$$

the structure of effectful  
programming is captured  
by a strong monad

eg/  
 $\text{TA} = A + I$   
 $\text{TA} = \text{List}(A)$   
 $\text{TA} = S^* \times A$

eg/  
 $A \xrightarrow{\text{inl}} A + I$   
 $A \longrightarrow \text{List } A$   
 $a \longmapsto [a]$

$A \longrightarrow S^* \times A$   
 $a \longmapsto (\varepsilon, a)$

① mark types as effectful

eg/

$$\begin{aligned} & [\![\text{let } x = u \text{ in } t]\!](\gamma) \\ &= \text{let } [\![u]\!](\gamma) = (w, a) \text{ in} \\ & \quad \text{let } [\![t]\!](\gamma, a) = (v, b) \text{ in} \\ & \quad (w + v, b) \end{aligned}$$

② every pure program is trivially effectful

③ explicit sequencing by let-binding

$$\frac{A \text{ type}}{\text{TA} \text{ type}}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \text{return}(t) : \text{TA}}$$

$$\frac{\Gamma \vdash u : \text{TA} \quad \Gamma, x : A \vdash t : \text{TB}}{\Gamma \vdash \text{let } x = u \text{ in } t : \text{TB}}$$

the structure of effectful  
programming is captured  
by a strong monad

to give a monad on a category  $\mathcal{C} =$

the structure of effectful  
programming is captured  
by a strong monad

to give a monad on a category  $\mathbb{C} =$   
•  $T : \mathbb{C} \rightarrow \mathbb{C}$  a functor

A type  
TA type

the structure of effectful  
programming is captured  
by a strong monad

to give a monad on a category  $\mathcal{C} =$

- $T : \mathcal{C} \rightarrow \mathcal{C}$  a functor
- $\eta_A : A \rightarrow TA$  for each  $A$

$$\frac{A \text{ type}}{TA \text{ type}}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \text{return}(t) : TA}$$

the structure of effectful  
programming is captured  
by a strong monad

$$\frac{A \text{ type}}{TA \text{ type}}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \text{return}(t) : TA}$$

$$\frac{\Gamma \vdash u : TA \quad \Gamma, x:A \vdash t : TB}{\Gamma \vdash \text{let } x = u \text{ in } t : TB}$$

to give a monad on a category  $\mathcal{C} =$

- $T : \mathcal{C} \rightarrow \mathcal{C}$  a functor
- $\eta_A : A \rightarrow TA$  for each  $A$
- for each  $f : A \rightarrow TB$ , a map  $f^* : TA \rightarrow TB$

??

$$[\Gamma] \xrightarrow{\text{Env}} T[\Gamma]$$

$$[\Gamma] \times [A] \xrightarrow{\text{Ext}} T[B]$$

$$[\Gamma] \xrightarrow{\langle \text{id}, \text{Env} \rangle} [\Gamma] \times T[\Gamma] \xrightarrow{??} T([\Gamma] \times T[\Gamma]) \xrightarrow{\langle \text{Ext} \rangle^*} T[B]$$

the structure of effectful  
programming is captured  
by a strong monad

$$\frac{A \text{ type}}{TA \text{ type}}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \text{return}(t) : TA}$$

$$\frac{\Gamma \vdash u : TA \quad \Gamma, x : A \vdash t : TB}{\Gamma \vdash \text{let } x = u \text{ in } t : TB}$$

to give a <sup>strong</sup>  $\wedge$  monad on a category  $\mathcal{C} =$

- $T : \mathcal{C} \rightarrow \mathcal{C}$  a functor
- $\eta_A : A \rightarrow TA$  for each  $A$
- for each  $f : A \rightarrow TB$ , a map  $f^* : TA \rightarrow TB$
- $t : A \times TB \rightarrow TCA \times B)$   
such that axioms hold

$$[\Gamma] \xrightarrow{\text{Env}} T[\Gamma]$$

$$[\Gamma] \times [A] \xrightarrow{\text{Ext}} T[B]$$

$$[\Gamma] \xrightarrow{\langle \text{id}, \text{Env} \rangle} [\Gamma] \times T[\Gamma] \xrightarrow[t]{??} T([\Gamma] \times [A]) \xrightarrow{\langle t \rangle^*} T[B]$$

Where does the strength

$$t: A \times T B \rightarrow T(A \times B)$$

come from?

# Modelling effects

a monad in a 2-category  $\mathcal{E}$   
consists of:

- an object  $C$
- a 1-cell  $T : C \rightarrow C$
- 2-cells  $\eta : \text{id}_C \Rightarrow T$   
 $\mu : T \circ T \Rightarrow T$

+ axioms

objects  $A, B, \dots$   
1-cells  $f, g : A \rightarrow B$   
2-cells  $\tau : f \Rightarrow g$

monad in  
2-category of  
categories,  
functors,  
nat. trans.  
= usual def<sup>n</sup>  
of monad!

# Modelling effects

[jww, Nayan Rajesh]

instantiating in the 2-category of clones:

monad on  $\mathcal{C}$  = a type  $TA$  for each type  $A$ ,  
a unit  $return : A \rightarrow TA$ ,  
a bind operation  
 $(\gg=)$

# Modelling effects

[jww. Nayan Rajesh]

$$\text{monad on } \mathbb{C} = \frac{\Gamma \vdash t : A}{\Gamma \vdash \text{return}(t) : TA}, \frac{\Gamma, x : A \vdash t : TB}{\Gamma \vdash u : TA}, \frac{\Gamma \vdash u : TA}{\Gamma \vdash \text{let } x = u \text{ in } t : TB}$$

...

# Modelling effects

[jww. Nayan Rajesh]

$$\text{monad on } \mathcal{C} = \frac{\Gamma \vdash t : A}{\Gamma \vdash \text{return}(t) : TA}, \frac{\Gamma, x:A \vdash t : TB}{\Gamma \vdash \text{let } x = u \text{ in } t : TB}$$

...

$$\text{free clone equipped with a monad} = \text{syntax of monadic Moggie's metalanguage}$$

# Modelling effects

[jww. Nayan Rajesh]  
[see also: Kock, Slattery]

free clone  $\mathcal{C}$   
equipped with a monad  $T$  = syntax of monadic Moggi's metalinguage

if  $\mathcal{C}$  has products,  $T$  becomes a strong monad  
on the cartesian category  $\bar{\mathcal{C}} =$  restrict  $\mathcal{C}$  to unary maps  
(linear version: monoidal)

# Modelling effects

[jww. Nayan Rajesh]  
[see also: Kock, Slattery]



~ get soundness and completeness  
as above

# Modelling effects

[jww. Nayan Rajesh]  
[see also: Kock, Slattery]

the strength requirement is  
a shadow of a multi-ary structure

# Modelling $\wedge^\rightarrow$ and CL

[FoSSaCS 24]

$\wedge$

$$\frac{\Gamma, x:A \vdash t:B}{\Gamma \vdash \lambda x.t:A \rightarrow B}$$

$$(\lambda x.t) u =_\beta t[u/x]$$

$$\lambda x.t^x x =_\eta t$$

$$\frac{}{x_1:A_1, \dots, x_n:A_n \vdash x_i:A_i}$$

$$\frac{\Gamma \vdash t:A \rightarrow B \quad \Gamma \vdash u:A}{\Gamma \vdash t \cdot u:B}$$

CL

$$\frac{}{x_1:A_1, \dots, x_n:A_n \vdash x_i:A_i}$$

$$\frac{\Gamma \vdash t:A \rightarrow B \quad \Gamma \vdash u:A}{\Gamma \vdash t \cdot u:B}$$

$$\Gamma \vdash s : (A \rightarrow (B \rightarrow C)) \longrightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$$

$$\frac{}{\Gamma \vdash K:A \rightarrow (B \rightarrow A)}$$

$$\left. \begin{array}{l} ((S \cdot f) \cdot g) \cdot x = (f \cdot x) \cdot (g \cdot x) \\ (K \cdot x) \cdot y = x \end{array} \right\} \begin{array}{l} \text{weak} \\ \text{equality} \end{array}$$

$$\frac{\Gamma \vdash s:A \rightarrow B \quad \Gamma \vdash t:A \rightarrow B}{\Gamma, x:A \vdash s^x x = t^x x : B} \Rightarrow s = t$$

extensionality

$\wedge$

$$\frac{}{x_1:A_1, \dots, x_n:A_n \vdash x_i:A_i}$$

$$\frac{\Gamma, x:A \vdash t:B}{\Gamma \vdash \lambda x.t:A \rightarrow B}$$

T BINDING

$$\boxed{(\lambda x.t)u =_\beta t[u/x]}$$

$$\lambda x.t^x x =_\eta t$$

SUBSTITUTION

CL

$$\frac{}{x_1:A_1, \dots, x_n:A_n \vdash x_i:A_i}$$

$$\frac{\Gamma \vdash t:A \rightarrow B \quad \Gamma \vdash u:A}{\Gamma \vdash t \cdot u:B}$$

$$\Gamma \vdash s : (A \rightarrow (B \rightarrow C)) \longrightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$$

$$\Gamma \vdash K : A \rightarrow (B \rightarrow A)$$

$$\left. \begin{array}{l} ((S \cdot f) \cdot g) \cdot x = (f \cdot x) \cdot (g \cdot x) \\ (K \cdot x) \cdot y = x \end{array} \right\} \begin{array}{l} \text{weak} \\ \text{equality} \end{array}$$

$$\frac{\Gamma \vdash s:A \rightarrow B \quad \Gamma \vdash t:A \rightarrow B}{\Gamma, x:A \vdash s^x x = t^x x : B} \Rightarrow s = t$$

extensionality

$\wedge$

$$\frac{}{x_1:A_1, \dots, x_n:A_n \vdash x_i:A_i}$$

$$\frac{\Gamma, x:A \vdash t:B}{\Gamma \vdash \lambda x.t:A \rightarrow B}$$

T BINDING

$$\frac{\Gamma \vdash t:A \rightarrow B \quad \Gamma \vdash u:A}{\Gamma \vdash t \cdot u:B}$$

$$\begin{aligned} (\lambda x.t)u &=_{\beta} t[u/x] \\ \lambda x.t^x x &=_{\eta} t \end{aligned}$$

SUBSTITUTION

CL

$$\frac{}{x_1:A_1, \dots, x_n:A_n \vdash x_i:A_i}$$

$$\frac{\Gamma \vdash t:A \rightarrow B \quad \Gamma \vdash u:A}{\Gamma \vdash t \cdot u:B}$$

$$\Gamma \vdash S^{\Gamma}: (A \rightarrow (B \rightarrow C)) \longrightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$$

$$\Gamma \vdash K: A \rightarrow (B \rightarrow A)$$

$$\left. \begin{aligned} ((S \cdot f) \cdot g) \cdot x &= (f \cdot x) \cdot (g \cdot x) \\ (K \cdot x) \cdot y &= x \end{aligned} \right\} \begin{matrix} \text{weak} \\ \text{equality} \end{matrix}$$

$$\frac{\Gamma \vdash s:A \rightarrow B \quad \Gamma \vdash t:A \rightarrow B}{\Gamma, x:A \vdash s^x x = t^x x : B} \Rightarrow s = t$$

extensionality

$\wedge$

$$\frac{}{x_1:A_1, \dots, x_n:A_n \vdash x_i:A_i}$$

$$\frac{\Gamma, x:A \vdash t:B}{\Gamma \vdash \lambda x.t:A \rightarrow B}$$

T BINDING

$$\frac{\Gamma \vdash t:A \rightarrow B \quad \Gamma \vdash u:A}{\Gamma \vdash t \cdot u:B}$$

$$\begin{aligned} (\lambda x.t)u &=_{\beta} t[u/x] \\ \lambda x.t^x x &=_{\eta} t \end{aligned}$$

) SUBSTITUTION

CL

$$\frac{}{x_1:A_1, \dots, x_n:A_n \vdash x_i:A_i}$$

$$\frac{\Gamma \vdash t:A \rightarrow B \quad \Gamma \vdash u:A}{\Gamma \vdash t \cdot u:B}$$

$$\Gamma \vdash s:(A \rightarrow (B \rightarrow C)) \longrightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$$

$$\Gamma \vdash K:A \rightarrow (B \rightarrow A)$$

$$\left. \begin{aligned} ((S \cdot f) \cdot g) \cdot x &= (f \cdot x) \cdot (g \cdot x) \\ (K \cdot x) \cdot y &= x \end{aligned} \right\} \begin{matrix} \text{weak} \\ \text{equality} \end{matrix}$$

$$\frac{\Gamma \vdash s:A \rightarrow B \quad \Gamma \vdash t:A \rightarrow B}{\Gamma, x:A \vdash s^x x = t^x x : B} \Rightarrow s = t$$

extensionality

$\wedge$

$$\frac{}{\Gamma, x:A \vdash t:B}$$

$$\Gamma \vdash \lambda x.t : A \rightarrow B$$

$\overline{\text{BINDING}}$

$$\frac{x_1:A_1, \dots, x_n:A_n \vdash x_i:A_i}{}$$

$$\frac{\Gamma \vdash t:A \rightarrow B \quad \Gamma \vdash u:A}{\Gamma \vdash t \cdot u:B}$$

$$(\lambda x.t) u =_\beta t[u/x]$$

$$\lambda x.t^x x =_\eta t$$

SUBSTITUTION

NO SUBSTITUTION

ALGEBRAIC ("no binding")

CL

$$x_1:A_1, \dots, x_n:A_n \vdash x_i:A_i$$

$$\frac{\Gamma \vdash t:A \rightarrow B \quad \Gamma \vdash u:A}{\Gamma \vdash t \cdot u:B}$$

$$\Gamma \vdash s : (A \rightarrow (B \rightarrow C)) \longrightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$$

$$\Gamma \vdash K : A \rightarrow (B \rightarrow A)$$

$$\left. \begin{array}{l} ((S \cdot f) \cdot g) \cdot x = (f \cdot x) \cdot (g \cdot x) \\ (K \cdot x) \cdot y = x \end{array} \right\} \begin{array}{l} \text{weak} \\ \text{equality} \end{array}$$

$$\frac{\Gamma \vdash s:A \rightarrow B \quad \Gamma \vdash t:A \rightarrow B}{\Gamma, x:A \vdash s^x x = t^x x : B} \Rightarrow s = t$$

extensionality

a classical result : [Schönfinkel, Curry,...]

$$\left( \lambda\text{-terms} \atop \text{modulo } \beta\eta \right) \equiv \left( \text{CL-terms modulo} \atop \text{weak extensionality} \right)$$

~ quite easy to prove

~ harder : relate the models

[essentially Hyland '14, '15]

def: a closed clone is a clone  $\mathbb{C}$  with

- for every  $A, B \in \mathbb{C}$  a type  $A \rightarrow B \in \mathbb{C}$

$$\frac{A \text{ type} \quad B \text{ type}}{A \rightarrow B \text{ type}}$$

[essentially Hyland '14, '15]

def: a closed clone is a clone  $\mathbb{C}$  with

- for every  $A, B \in \mathbb{C}$  a type  $A \rightarrow B \in \mathbb{C}$

$$\frac{A \text{ type} \quad B \text{ type}}{A \rightarrow B \text{ type}}$$

- multimap  $A \rightrightarrows B, A \xrightarrow{\text{eval}_A, b} B$

$$\frac{}{f : A \rightarrow B, x : A \vdash f x : B}$$

[essentially Hyland '14, '15]

def: a closed clone is a clone  $\mathbb{C}$  with

- for every  $A, B \in \mathbb{C}$  a type  $A \multimap B \in \mathbb{C}$

$$\frac{A \text{ type} \quad B \text{ type}}{A \rightarrow B \text{ type}}$$

- multimap  $A \multimap B, A \xrightarrow{\text{eval}, \mathbb{C}} B$

$$\frac{f: A \rightarrow B, x: A \vdash f x : B}{}$$

... such that the map below is an iso:

$$\mathbb{C}(\Gamma, A; B) \xrightleftharpoons{\cong} \mathbb{C}(\Gamma; A \multimap B)$$

$$\Gamma, A \xrightarrow{u, A} (A \multimap B), A \xrightarrow{\text{eval}} B$$

$$\xleftarrow{u}$$

$$f: A \rightarrow B, x: A \vdash f x : B$$

$$\frac{}{\Gamma \vdash u: A \rightarrow B}$$

$$\frac{\Gamma, x: A \vdash u^x: A \rightarrow B}{\Gamma, x: A \vdash u: A \rightarrow B}$$

$$\frac{}{\Gamma, x: A \vdash x: A}$$

$$\Gamma, x: A \vdash (fx)[\frac{u^x}{f}, \frac{x}{f}] : B$$

[essentially Hyland '14, '15]

def: a closed clone is a clone  $\mathbb{C}$  with

- for every  $A, B \in \mathbb{C}$  a type  $A \multimap B \in \mathbb{C}$

$$\frac{A \text{ type} \quad B \text{ type}}{A \rightarrow B \text{ type}}$$

- multimap  $A \multimap B, A \xrightarrow{\text{eval}, \mathbb{C}} B$

$$\frac{f: A \rightarrow B, x: A \vdash f x : B}{}$$

... such that the map below is an iso:

$$\frac{\Gamma, x: A \vdash t : B}{\Gamma \vdash \lambda x. t : A \rightarrow B}$$

$$\mathbb{C}(\Gamma, A; B) \xrightleftharpoons[\sim]{t \mapsto \lambda t} \mathbb{C}(\Gamma; A \Rightarrow B)$$

$$\Gamma, A \xrightarrow{u, A} (A \Rightarrow B), A \xrightarrow{\text{eval}} B \quad \longleftarrow u$$

$$f: A \rightarrow B, x: A \vdash f x : B$$

$$\frac{\Gamma \vdash u : A \rightarrow B}{}$$

$$\frac{\Gamma, x: A \vdash u^x : A \rightarrow B}{}$$

$$\frac{\Gamma, x: A \vdash u : A}{\Gamma, x: A \vdash x : A}$$

$$\Gamma, x: A \vdash (fx)[u^x_f, x^x_f] : B$$

[Essentially Hyland '14, '15]

def: a closed clone is a clone  $\mathbb{C}$  with

- for every  $A, B \in \mathbb{C}$  a type  $A \multimap B \in \mathbb{C}$

- multimap  $A \multimap B, A \xrightarrow{\text{eval}, t} B$

$$\frac{A \text{ type} \quad B \text{ type}}{A \rightarrow B \text{ type}}$$

$$\frac{f: A \rightarrow B, x: A \vdash f x : B}{}$$

... such that the map below is an iso:

$$\frac{\Gamma, x: A \vdash t : B}{\Gamma \vdash \lambda x. t : A \rightarrow B}$$

$$\mathbb{C}(\Gamma, A; B) \xrightleftharpoons[\sim]{t \mapsto \lambda t} \mathbb{C}(\Gamma; A \Rightarrow B)$$

$$\Gamma, A \xrightarrow{u, A} (A \Rightarrow B), A \xrightarrow{\text{eval}} B \quad \longleftarrow u$$

$$(\lambda x. t)^* x =_B t \quad ; \quad \lambda x. u^x x =_T x$$

$$f: A \rightarrow B, x: A \vdash f x : B$$

$$\frac{\Gamma \vdash u : A \rightarrow B}{}$$

$$\frac{\Gamma, x: A \vdash u^x : A \rightarrow B}{}$$

$$\frac{\Gamma, x: A \vdash x : A}{}$$

$$\Gamma, x: A \vdash (fx)[u^x_f, x^x_f] : B$$

def: an SK-clone is a clone  $\mathcal{C}$  with

- for every  $A, B \in \mathcal{C}$  a type  $A \rightarrow B \in \mathcal{C}$

$$\frac{A \text{ type} \quad B \text{ type}}{A \rightarrow B \text{ type}}$$

def: an SK-clone is a clone  $\mathbb{C}$  with

- for every  $A, B \in \mathbb{C}$  a type  $A \rightarrow B \in \mathbb{C}$

$$\frac{A \text{ type} \quad B \text{ type}}{A \rightarrow B \text{ type}}$$

- for every  $t: \Gamma \rightarrow (A \Rightarrow B)$  and  $u: \Gamma \rightarrow A$   
a multimap  $t \cdot u: \Gamma \rightarrow B$

$$\frac{\Gamma \vdash t: A \rightarrow B \quad \Gamma \vdash u: A}{\Gamma \vdash t \cdot u: B}$$

def: an SK-clone is a clone  $\mathbb{C}$  with

- for every  $A, B \in \mathbb{C}$  a type  $A \rightarrow B \in \mathbb{C}$
- for every  $t: \Gamma \rightarrow (A \Rightarrow B)$  and  $u: \Gamma \rightarrow A$   
a multimap  $t \cdot u: \Gamma \rightarrow B$
- distinguished nullary multimaps  $S$  and  $K$   
s.t. the CL equations hold

$$\frac{A \text{ type} \quad B \text{ type}}{A \rightarrow B \text{ type}}$$

$$\frac{\Gamma \vdash t: A \rightarrow B \quad \Gamma \vdash u: A}{\Gamma \vdash t \cdot u: B}$$

$$\frac{}{\vdash S: \dots} \quad \frac{}{\vdash K: \dots}$$

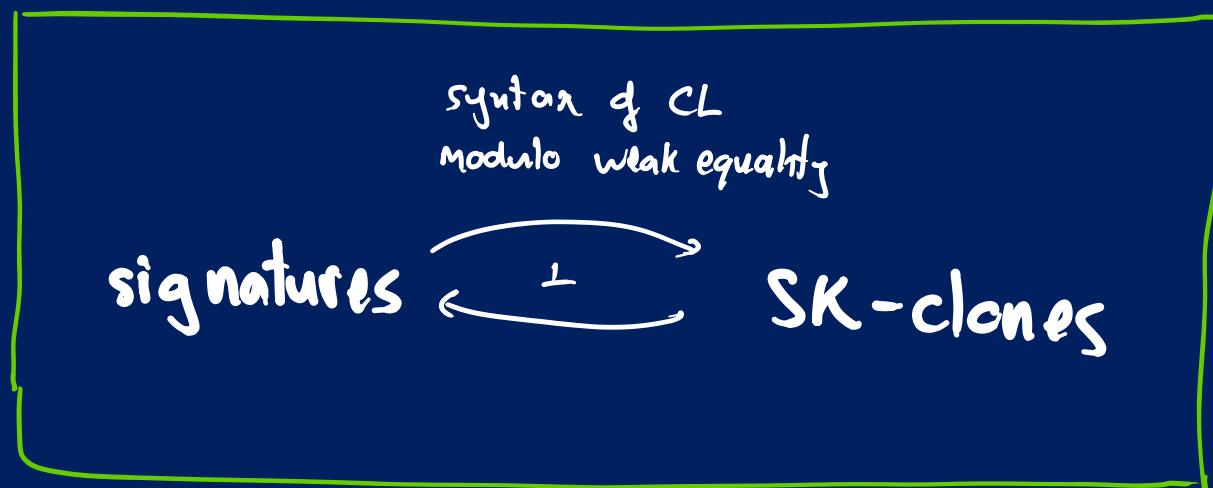
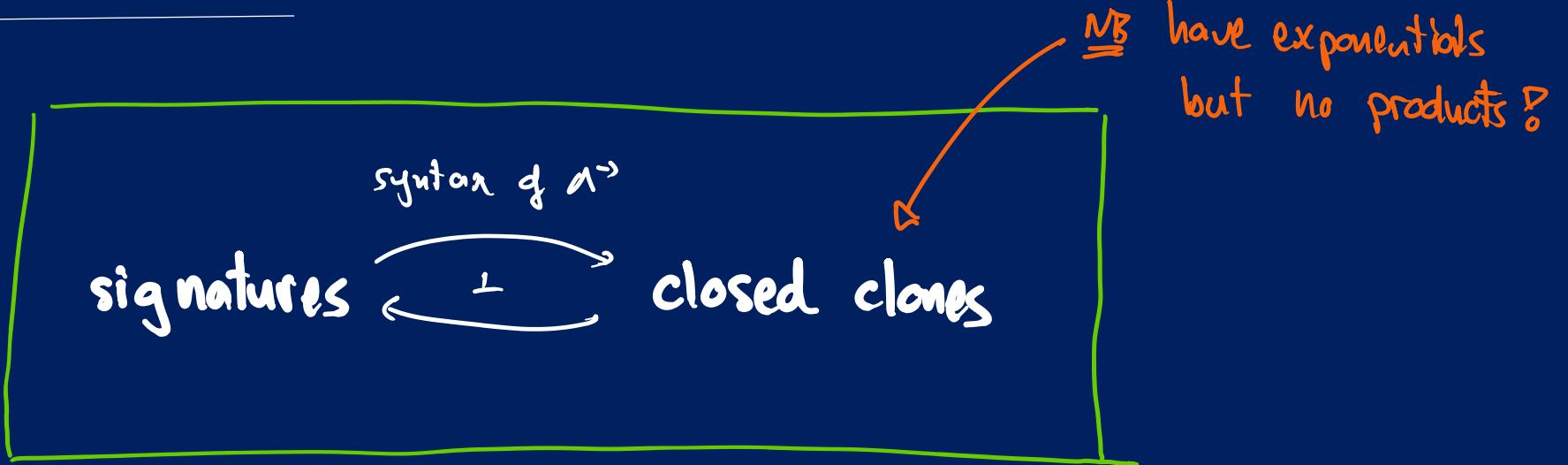
+ weakening

$$S \ p_1 \ p_2 \ p_3 = (p_1 \ p_3) (p_2 \ p_3)$$

$$\not\models$$

$$K \ p_1 \ p_2 = p_1 \quad \dots$$

these are good models...



# How to add extensionality?

notation for weakening:

$$\frac{\Delta \vdash t : B}{\Delta, \Gamma \vdash t^\Gamma : B} \text{ when}$$

basic idea for syntax:

"bracket abstraction" { for every  $\Gamma \vdash t : A \rightarrow B$  define  $\vdash t^c : \Gamma \rightarrow (A \rightarrow B)$   
such that  $\Gamma \vdash (t^c)^{\Gamma} x_1 \dots x_n = t : A \rightarrow B$

+ in the presence of extensionality,  $t^c$  is unique

# How to add extensionality?

notation for weakening:

$$\frac{\Delta \vdash t : B}{\Delta, \Gamma \vdash t^\Gamma : B} \text{ when}$$

basic idea for syntax:

"bracket abstraction" { for every  $\Gamma \vdash t : A \rightarrow B$  define  $\vdash t^c : \Gamma \rightarrow (A \rightarrow B)$   
such that  $\Gamma \vdash (t^c)^\Gamma x_1 \dots x_n = t : A \rightarrow B$

+ in the presence of extensionality,  $t^c$  is unique

i.e.  $\mathbb{C}(\Delta ; \Gamma \Rightarrow (A \Rightarrow B)) \xrightarrow{q^{\Gamma; A \Rightarrow B}} \mathbb{C}(\Gamma ; A \Rightarrow B)$  is an iso  
 $t \longmapsto (t^\Gamma) x_1 \dots x_n$

# How to add extensionality?

def: an SK-class is  
extensional if every  $q^{\Gamma; A, B}$   
is an ISO.

basic idea for syntax:

= exactly the SK-classes that admit  
extensional bracket abstraction

"bracket abstraction" { for every  $\Gamma \vdash t : A \rightarrow B$  define  $\vdash t^c : \Gamma \rightarrow (A \rightarrow B)$   
such that  $\Gamma \vdash (t^c)^{\Gamma} x_1 \dots x_n = t : A \rightarrow B$

+ in the presence of extensionality,  $t^c$  is unique

i.e.  $\mathbb{C}(\Delta ; \Gamma \Rightarrow (A \Rightarrow B)) \xrightarrow{q^{\Gamma; A, B}} \mathbb{C}(\Gamma ; A \Rightarrow B)$  is an ISO  
 $t \longmapsto (t^{\Gamma}) x_1 \dots x_n$

syntax of CL  
with extensional  
weak equality

=

free extensional  
SK - clone

syntax of CL  
with extensional  
weak equality

= free extensional  
SK - clone

category of  
extensional  
SK - clones

{  
CL with extensional  
weak equality

↪ category of  
closed clones



↪

syntax of CL  
with extensional  
weak equality

= free extensional  
SK - clone

category  
of  
"SK-categories"

category of  
extensional  
SK - clones

CL with extensional  
weak equality

↪ category of  
closed clones



↪

# Summary

- 1) multi-ary models clarify the usual categorical models of programs
- 2) this is a good way to derive canonical syntax for particular models

eg://

- correspond to categorical models
- soundness + completeness
- contexts  $\neq$  products

:

# Summary

- 1) multi-ary models clarify the usual categorical models of programs
- 2) this is a good way to derive canonical syntax for particular models

On-going work:

- ① do this in an enriched setting
- ② what other languages have a "combinator" version?

eg://

- correspond to categorical models
- soundness + completeness
- contexts  $\neq$  products

: