

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

\leadsto reduce to algebraic questions

- eg.
- refuting $\neg \exists y$ or $\stackrel{?}{=} \text{ctx}$
 - 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~~

- ~~eg.~~
- ~~• refuting $\exists y$ or $\forall x$~~
 - ~~• 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 $\llbracket \Gamma \vdash t : B \rrbracket$ to each program

program

$\Gamma \vdash t : B$

$\llbracket - \rrbracket$
~~~~~>

morphism

$\llbracket \Gamma \rrbracket \xrightarrow{\llbracket t \rrbracket} \llbracket B \rrbracket$

in a category

└ with enough structure  
to model the language

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

program

$\Gamma \vdash t : B$

$\llbracket - \rrbracket$   
~~~~~>

morphism

$\llbracket \Gamma \rrbracket \xrightarrow{\llbracket t \rrbracket} \llbracket B \rrbracket$

in a category

└ with enough structure
to model the language

soundness: $t = t' \Rightarrow \llbracket t \rrbracket = \llbracket t' \rrbracket$

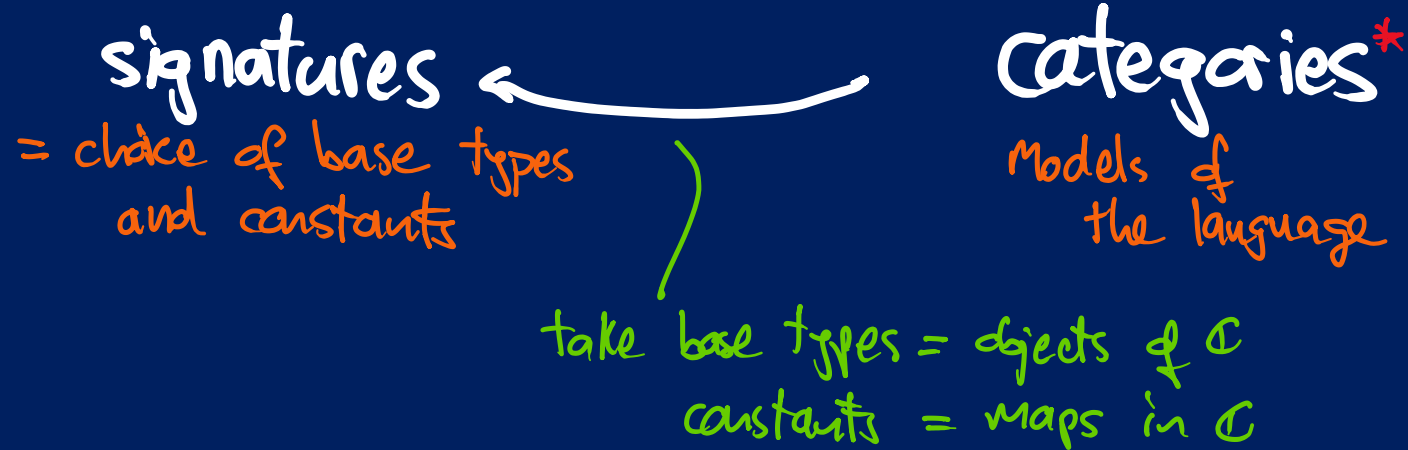
completeness: $\llbracket t \rrbracket = \llbracket t' \rrbracket$ in every model
 $\Rightarrow t = t'$

Soundness and completeness



* with extra structure
depending on the type theory

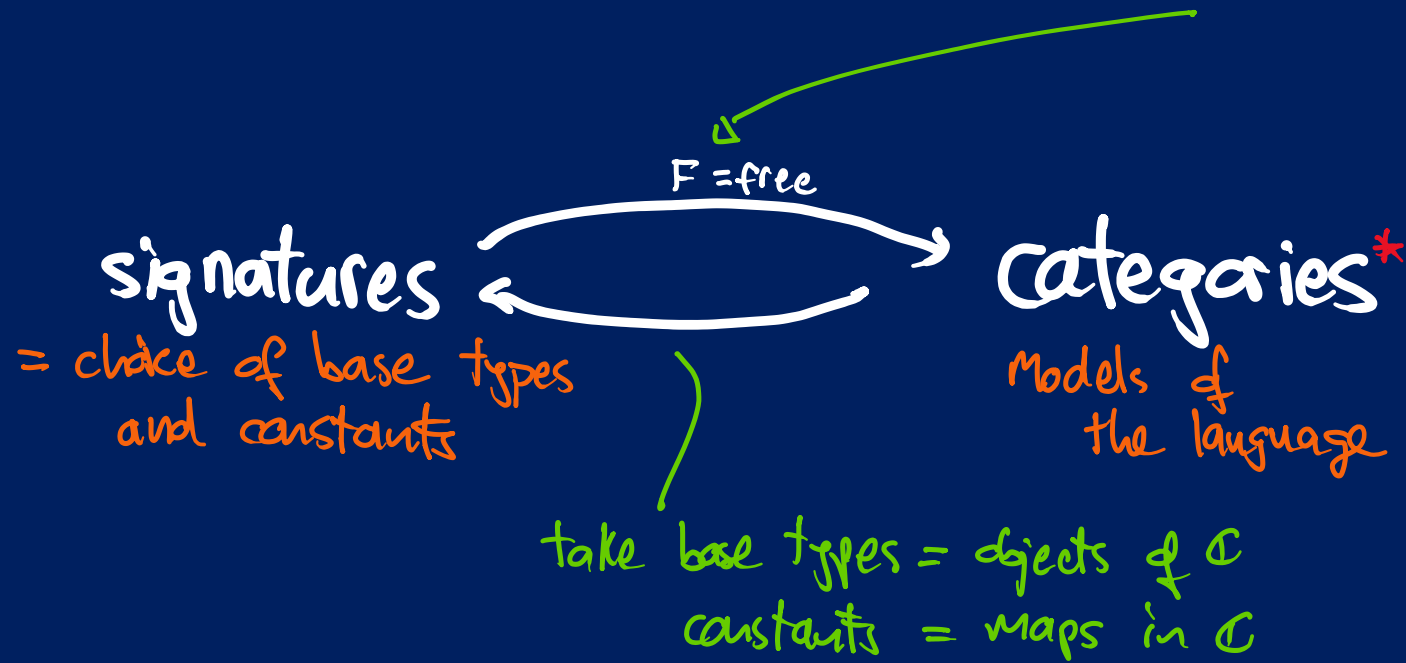
Soundness and completeness



* with extra structure
depending on the type theory

Soundness and completeness

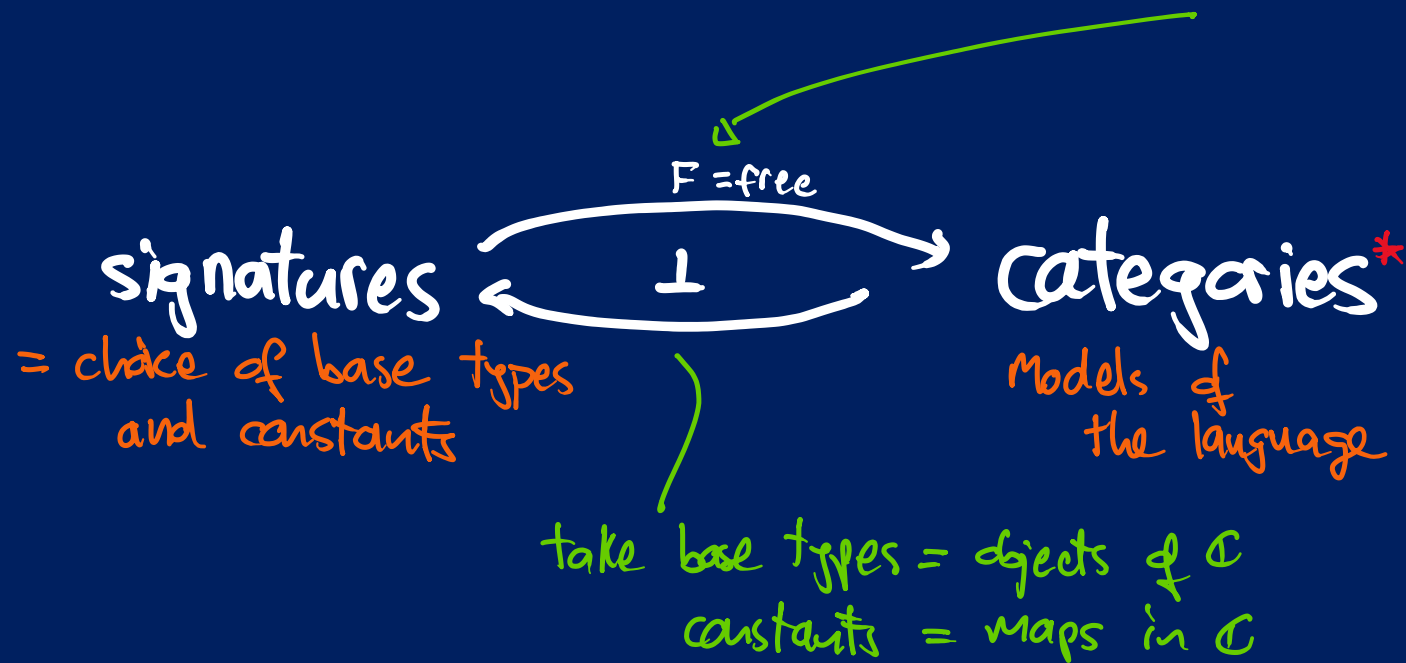
given a signature S
build a category $F[S]$
using the syntax of
the language



* with extra structure
depending on the type theory

Soundness and completeness

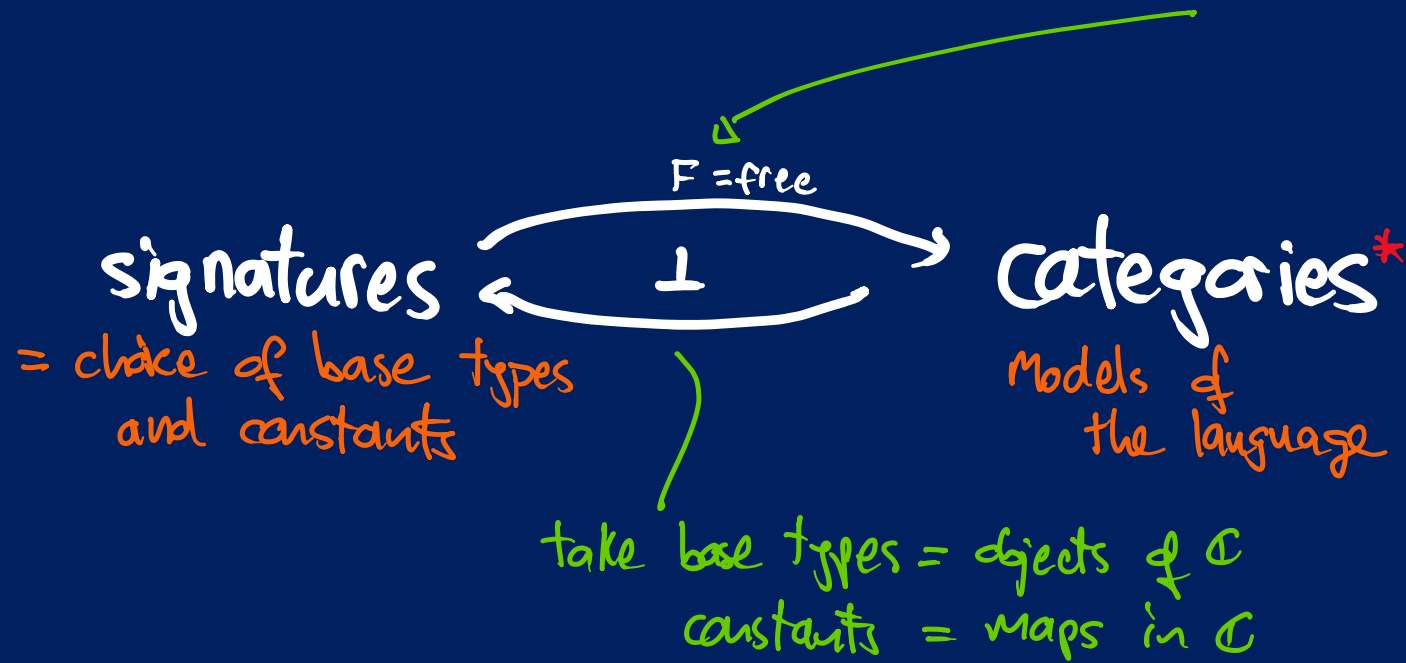
given a signature S
build a category $F[S]$
using the syntax of
the language



* 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 ...

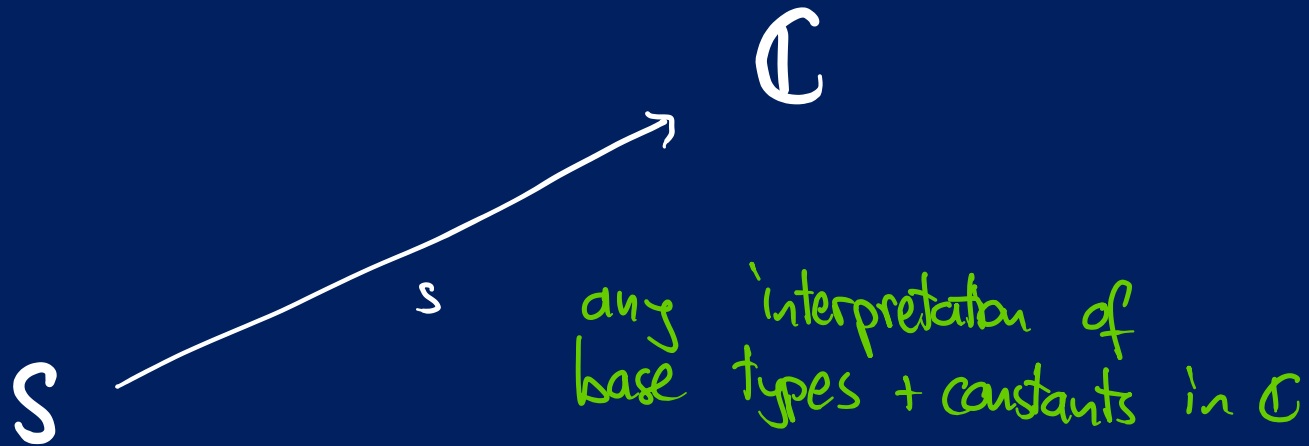
signatures $\overset{F}{\curvearrowright} \underset{\perp}{\curvearrowleft}$ categories

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

* with extra structure
depending on the type theory

signatures $\begin{matrix} \xrightarrow{F} \\ \perp \\ \xleftarrow{L} \end{matrix}$ categories

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



* with extra structure
depending on the type theory

signatures $\begin{matrix} \xrightarrow{F} \\ \perp \\ \xrightarrow{\quad} \end{matrix}$ categories

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

$F[S] \xrightarrow[\text{such that...}]{\exists! \text{ functor } s[-]}$ \mathbb{C}

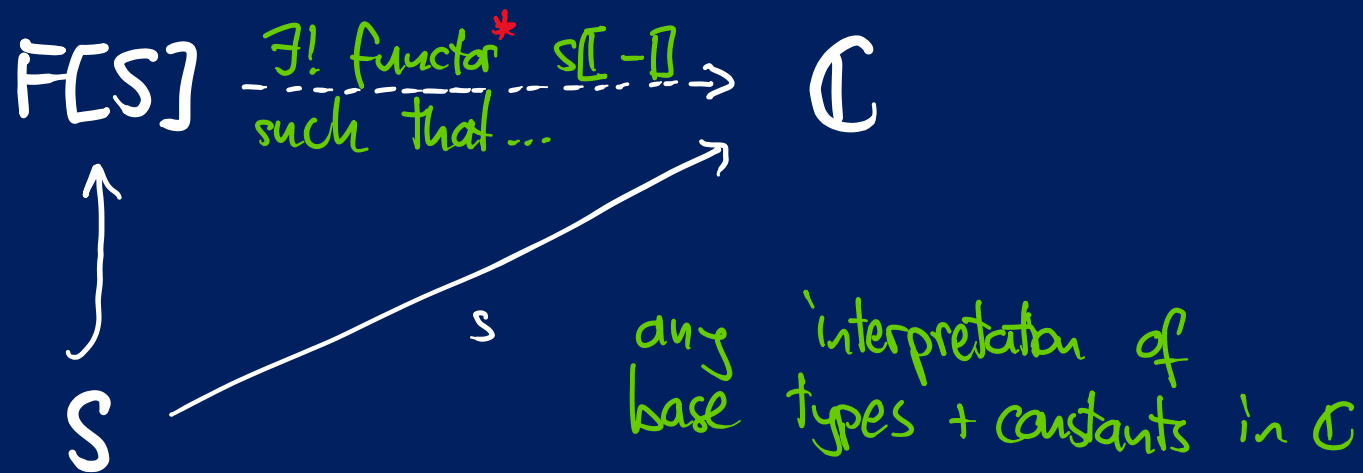
$S \xrightarrow{s}$

any interpretation of
base types + constants in \mathbb{C}

* with extra structure
depending on the type theory

signatures $\begin{matrix} \xrightarrow{F} \\ \perp \\ \xrightarrow{L} \end{matrix}$ categories

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

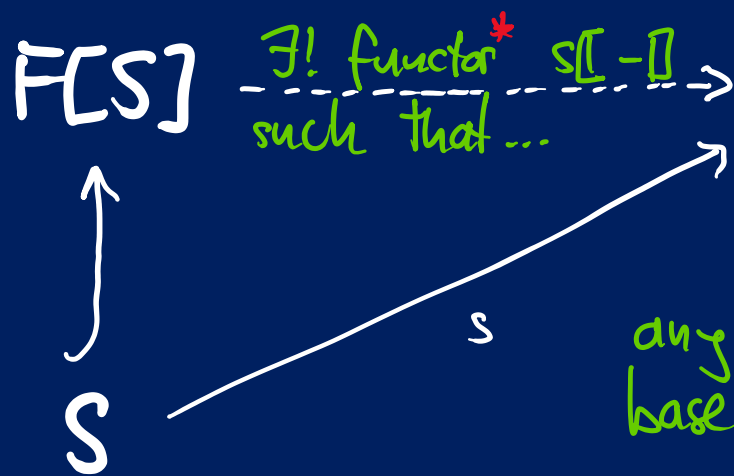


* with extra structure
depending on the type theory

signatures $\overset{F}{\curvearrowright} \underset{\perp}{\curvearrowleft}$ categories

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

maps
= terms in the
type theory



} $s[t] =$ semantic interpretation
of term t

any interpretation of
base types + constants in C

* with extra structure
depending on the type theory

signatures $\begin{matrix} \xrightarrow{F} \\ \perp \\ \xrightarrow{\quad} \end{matrix}$ categories

soundness = $\llbracket - \rrbracket$ preserves all the structure

completeness = $F[S]$ exists

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

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

maps = terms in the type theory

$F[S] \xrightarrow{\exists! \text{ functor } \llbracket - \rrbracket \text{ such that } \dots} \mathbb{C}$

$\llbracket t \rrbracket =$ semantic interpretation of term t

S

$S \xrightarrow{s} \mathbb{C}$

any interpretation of base types + constants in \mathbb{C}

signatures $\overset{F}{\curvearrowright} \underset{\perp}{\curvearrowleft}$ categories

= specifications of type theories
~ generate terms

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

morphism $f: A \rightarrow B$

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

signatures $\begin{matrix} \xrightarrow{F} \\ \perp \\ \xrightarrow{\quad} \end{matrix}$ categories

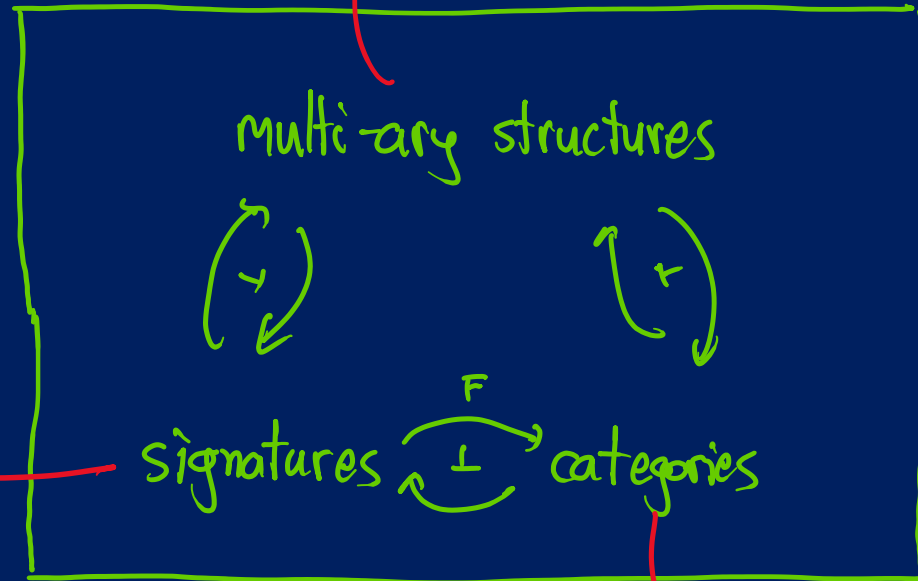
= specifications of type theories
~ generate terms

morphisms $f: A \rightarrow B$

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

This talk:

multimaps $A_1, \dots, A_n \xrightarrow{f} B$



= specifications of type theories
→ generate terms

morphisms $f: A \rightarrow B$

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

This talk:

~ known to experts

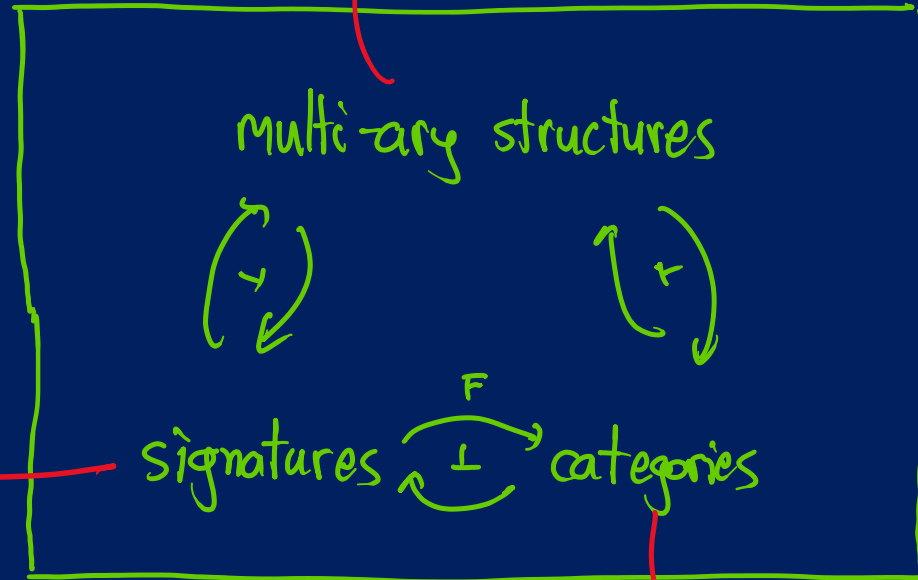
eg, Lambek '69; Hyland '14, '15;
Fiore et al '99, '10, '21, ...;
Shulman '23; ... and others?

~ based of my FOSSACS '24

= specifications of type theories
~ generate terms

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

multimaps $A_1, \dots, A_n \xrightarrow{f} B$



morphisms $f: A \rightarrow B$

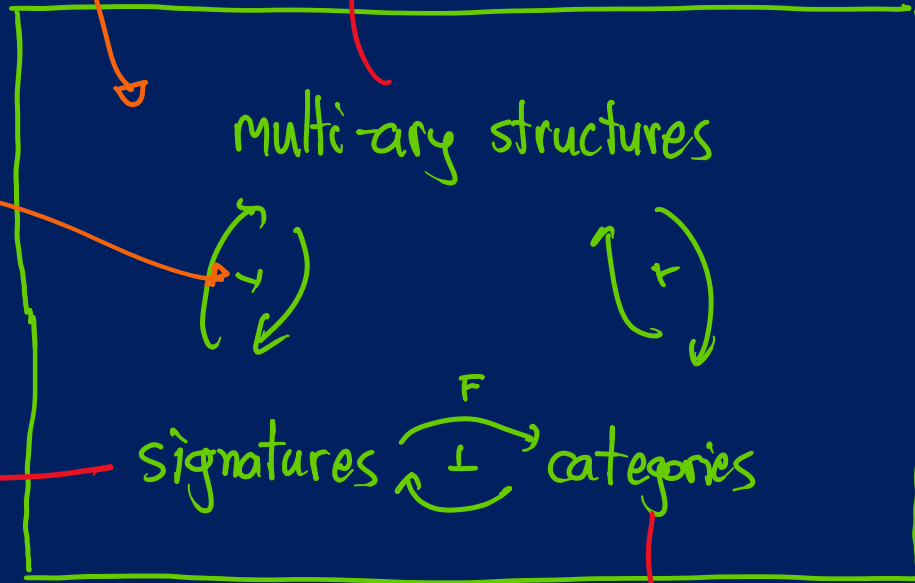
This talk:

interpretation of t
in free model is
 t . itself

contexts separate from products

multimaps $A_1, \dots, A_n \xrightarrow{f} B$

easy to
prove soundness
and completeness



= specifications of type theories
~ generate terms

morphisms $f: A \rightarrow B$

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

↳ generalises easily via enrichment eg gradings, refinements, 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 λ -calculus models with models of CL

Multi-ary structures

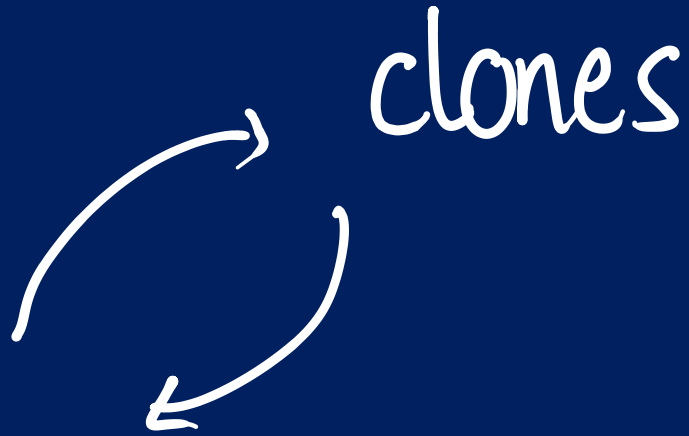
= has contraction, permutation,
weakening

cartesian simple
type theories

= has contraction, permutation,
weakening



cartesian simple
type theories



clones

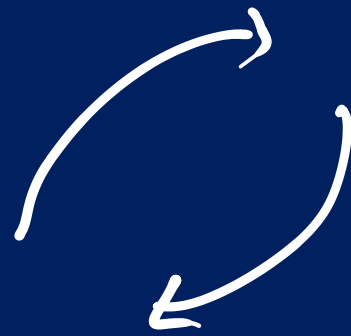
multicategories,

symmetric
multicategories

clones

ordered/linear

~~cartesian~~ simple
type theories



a ^{simple} 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 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 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

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

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

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

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

multisorted, abstract

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

[Hall]

- objects A, B, C, \dots
- multimaps $f_1, \dots : A_1, \dots, A_n \rightarrow B$, $(n \geq 0)$
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 $f_{i, \dots} : A_1, \dots, A_n \rightarrow B$, $(n \geq 0)$
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 $f_1, \dots : A_1, \dots, A_n \rightarrow B$, $(n \geq 0)$
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 $f, g, \dots : A_1, \dots, A_n \rightarrow B$, $(n \geq 0)$
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

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

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

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

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

[Hall]


- objects A, B, C, \dots
- multimaps $f, g, \dots : A_1, \dots, A_n \rightarrow B$, $(n \geq 0)$
including $p_i^{A_1, \dots, A_n} : A_1, \dots, A_n \rightarrow A_i$ for $i = 1, \dots, n$
- a substitution operation

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

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

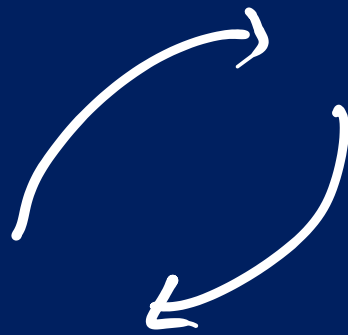
cartesian simple
type theories

clones



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

cartesian simple
type theories



clones \simeq

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

A GENERAL THEORY

type theories with structure
SI
monads on the category
of clones

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

Syntax from semantics

simply-typed λ -calculus

for any signature of
base types and constants

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

base types

terms : $t, u ::= x \mid c \mid () \mid \langle t, u \rangle \mid \pi_i(t) \mid \lambda x. t \mid t u$

constants *(i=1,2)*

equations : $(\beta) \quad \pi_i(\langle t_1, t_2 \rangle) = t_i$ $(\lambda x. t) u = t[u/x]$
 $(\gamma) \quad \langle \pi_1(t), \pi_2(t) \rangle = t$ $\lambda x(t x) = t$

Λ^\times = just products

Λ^\rightarrow = just exponentials

$\Lambda^{\times, \rightarrow}$ = products + exponentials

Modelling products

Cartesian
product in
category \mathcal{C}

$=$

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

Cartesian
product in
clone \mathbb{C}

def
 \equiv

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

a cartesian product in a clone \mathbb{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.

$$\mathbb{C}(T; A_1 \times A_2) \xrightarrow{t \mapsto (\pi_1[t], \pi_2[t])} \mathbb{C}(T; A_1) \times \mathbb{C}(T; A_2)$$

is an iso.

Modelling products

a cartesian product in a clone \mathbb{C} consists of:

1) an object $A_1 \times A_2$

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

A_1 type A_2 type

 $A_1 \times A_2$ type

s.t.

$$\mathbb{C}(T; A_1 \times A_2) \xrightarrow{t \mapsto (\pi_1[t], \pi_2[t])} \mathbb{C}(T; A_1) \times \mathbb{C}(T; A_2)$$

is an iso.

Modelling products

a cartesian product in a clone \mathbb{C} consists of:

1) an object $A_1 \times A_2$

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

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

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

s.t.

$$\mathbb{C}(T; A_1 \times A_2) \xrightarrow{t \mapsto (\pi_1[t], \pi_2[t])} \mathbb{C}(T; A_1) \times \mathbb{C}(T; A_2)$$

is an iso.

Modelling products

a cartesian product in a clone \mathbb{C} consists of:

1) an object $A_1 \times A_2$

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

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

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

s.t.

$$\mathbb{C}(\Gamma; A_1 \times A_2) \xrightarrow{t \mapsto (\pi_1[t], \pi_2[t])} \mathbb{C}(\Gamma; A_1) \times \mathbb{C}(\Gamma; A_2)$$

is an iso.

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

Modelling products

a cartesian product in a cbe \mathbb{C} consists of:

1) an object $A_1 \times A_2$

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

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

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

s.t.

$$\mathbb{C}(\Gamma; A_1 \times A_2) \xrightarrow{t \mapsto (\pi_1[t], \pi_2[t])} \mathbb{C}(\Gamma; A_1) \times \mathbb{C}(\Gamma; A_2)$$

is an iso.

$$\begin{aligned} (\Gamma \vdash t : A_1 \times A_2) &\longmapsto (\Gamma \vdash \pi_i(p)[t/p] : A_i)_{i=1,2} \\ (\Gamma \vdash \langle u_1, u_2 \rangle : A_1 \times A_2) &\longleftarrow (\Gamma \vdash u_i : A_i)_{i=1,2} \end{aligned}$$

Modelling products

a cartesian product in a clone \mathbb{C} consists of:

- 1) an object $A_1 \times A_2$
- 2) multi-maps $\pi_i : A_1 \times A_2 \rightarrow A_i$

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

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

s.t.

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

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

$\pi_i(p) \llbracket \langle u_1, u_2 \rangle / p \rrbracket = u_i$

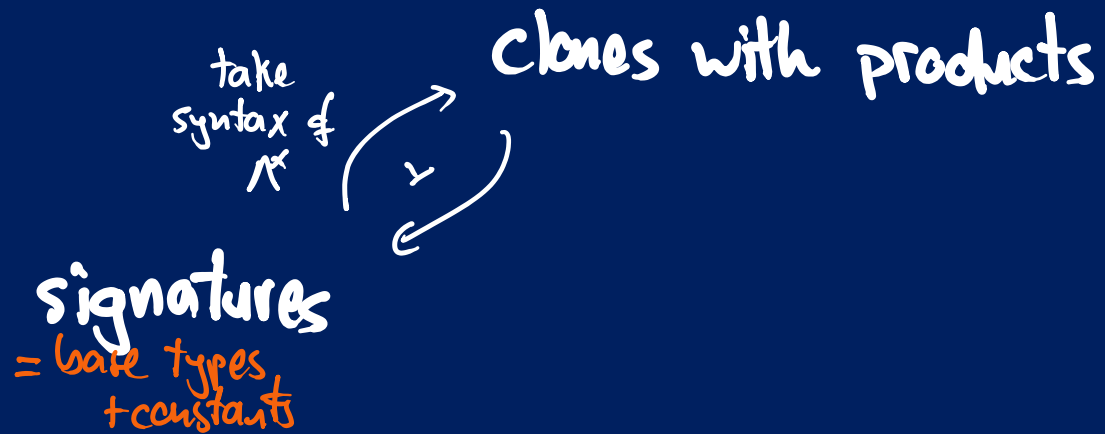
$\langle \pi_1(p), \pi_2(p) \rangle = p$ is an iso.

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

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

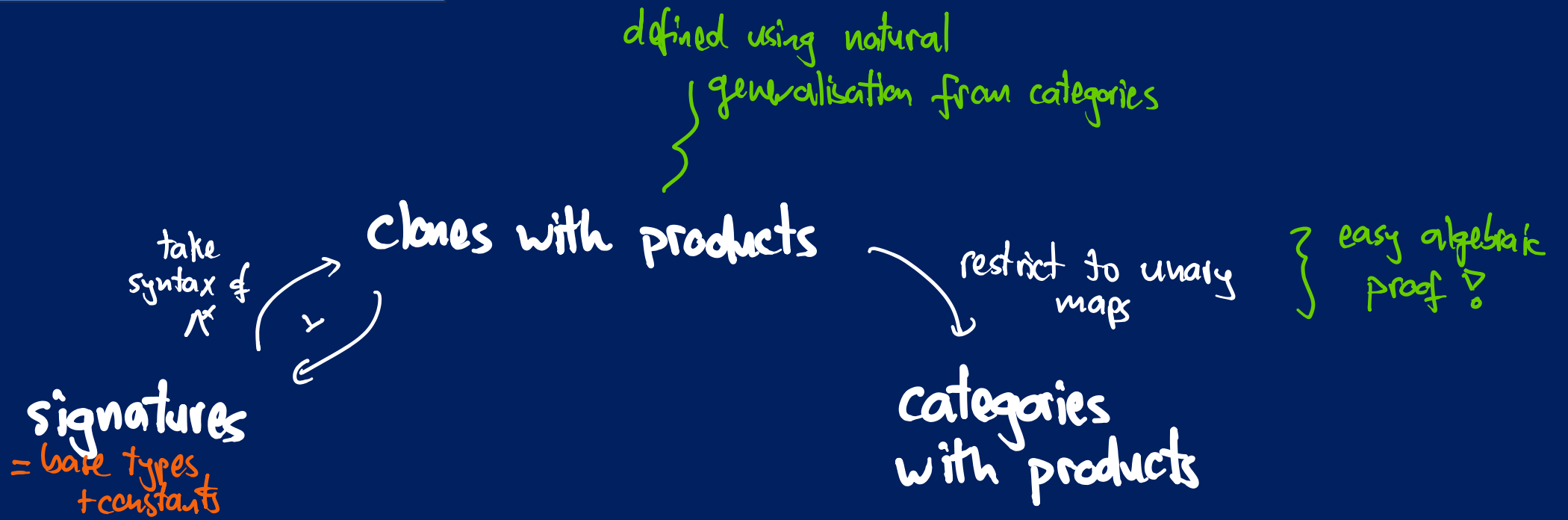
Modelling products

defined using natural
} generalisation from categories



free clone with products = syntax of Λ^x

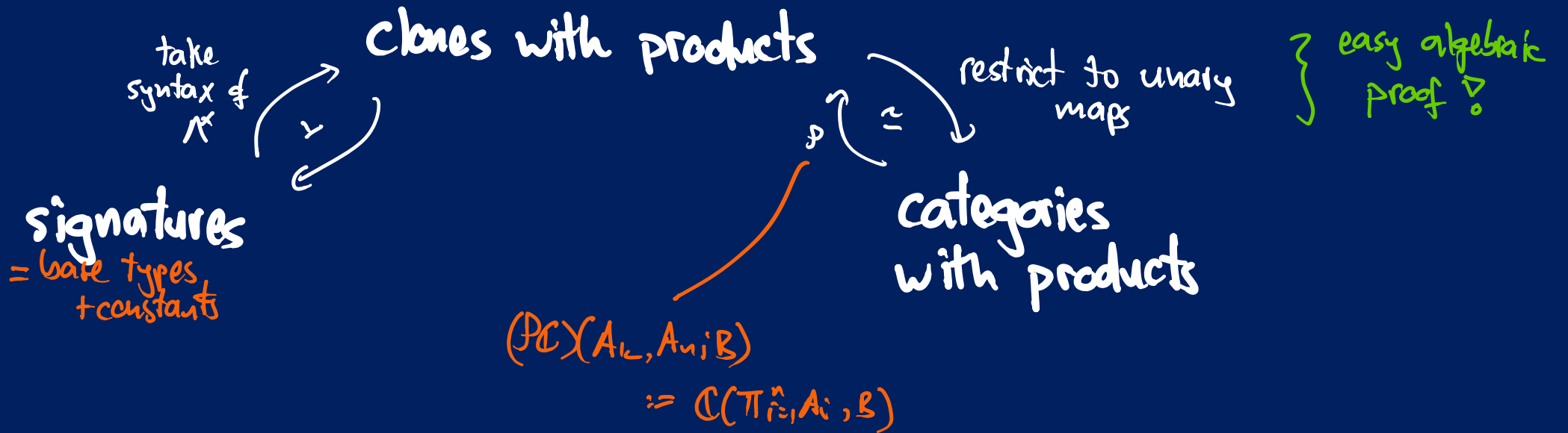
Modelling products



free clone with products = syntax of Λ^x

Modelling products

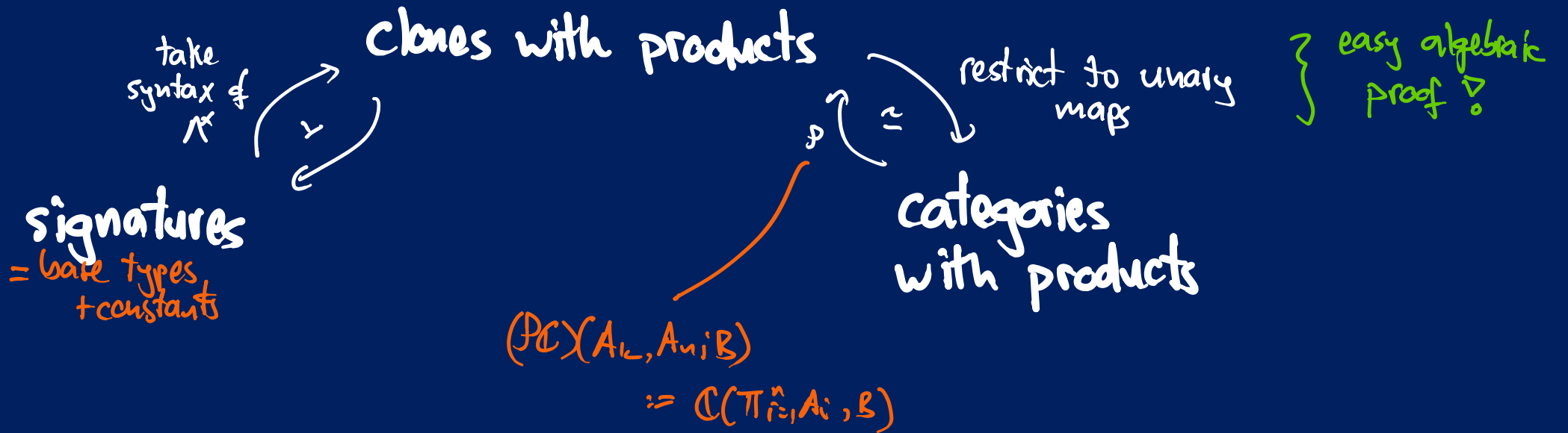
defined using natural
} generalisation from categories



free clone with products = syntax of Λ^x

Modelling products

defined using natural
} generalisation from categories



free clone with products = syntax of Λ^x

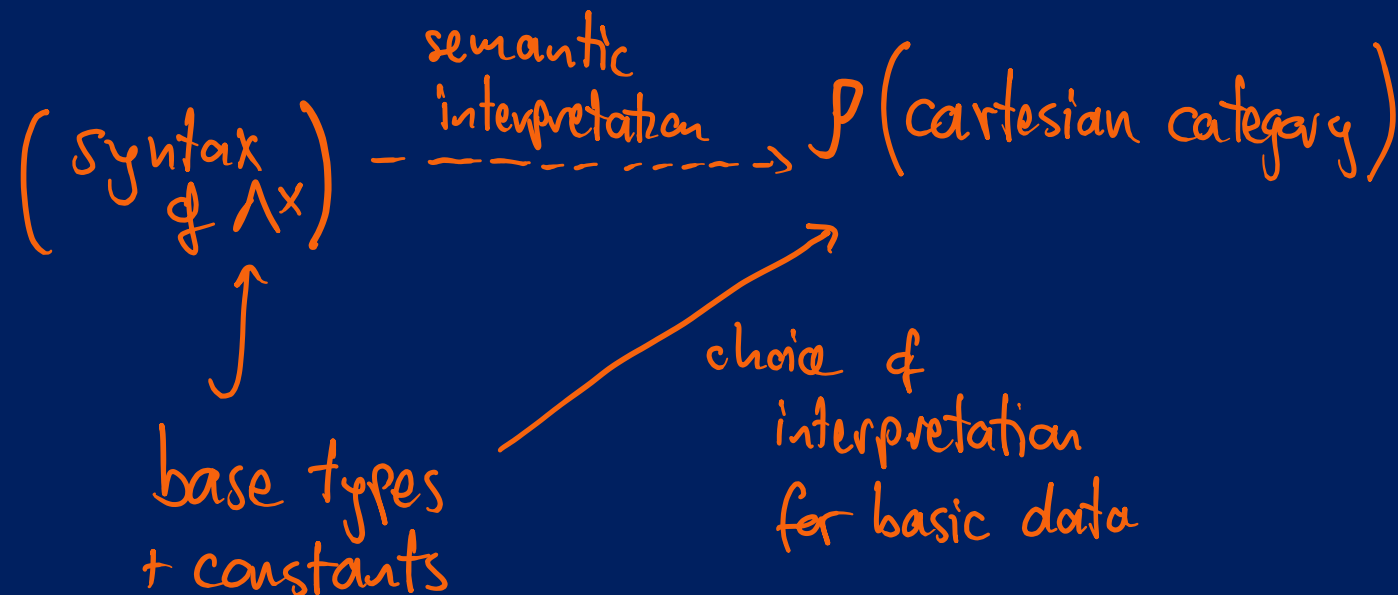
free cartesian category = Λ^x -terms $x:A \vdash t:B$

Modelling products: soundness and completeness

free clone with
all products

= syntax of λ^x

= typed λ -calculus
with just products



Modelling products: soundness and completeness

free clone with
all products

= syntax of λ^x

= typed λ -calculus
with just products

sound :: cartesian clone homomorphism



base types
+ constants

choice of
interpretation
for basic data

where products
= contexts
comes from!

Modelling products

natural definition of products for clones

\Rightarrow {
syntax for \wedge^x
soundness + completeness
equivalence with categorical models

Modelling products

in the linear setting, this is
how you derive &

natural definition of products for clones

\Rightarrow

syntax for \wedge^x

soundness + completeness

equivalence with categorical models

Modelling effects

Modelling effects

natural definition of *monads* for clones

⇒ { syntax for *Moggi's λ_{ml}*
soundness + completeness
equivalence with categorical models

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

A type

TA type

the structure of effectful programming is captured by a strong monad

9//
 $TA = A + I$
 $TA = List(A)$
 $TA = S^* \times A$

① mark types as effectful

A type
TA type

the structure of effectful programming is captured by a strong monad

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

① mark types as effectful

② every pure program is trivially effectful

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

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

the structure of effectful programming is captured by a strong monad

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

$$\begin{aligned} \text{eg// } A &\xrightarrow{\text{inl}} A + I \\ A &\longrightarrow \text{List } A \\ a &\longmapsto [a] \\ A &\longrightarrow S^* \times A \\ a &\longmapsto (\varepsilon, a) \end{aligned}$$

① mark types as effectful

② every pure program is trivially effectful

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

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

the structure of effectful programming is captured by a strong monad

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

$$\begin{aligned} \text{eg// } A &\xrightarrow{\text{inl}} A + I \\ A &\longrightarrow \text{List } A \\ a &\longmapsto [a] \\ A &\longrightarrow S^* \times A \\ a &\longmapsto (\varepsilon, a) \end{aligned}$$

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

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

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

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

the structure of effectful programming is captured by a strong monad

eg//

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

eg//

$$\begin{aligned} A &\xrightarrow{\text{inl}} A+1 \\ A &\longrightarrow \text{List } A \\ a &\longmapsto [a] \\ A &\longrightarrow S^* \times A \\ a &\longmapsto (\varepsilon, a) \end{aligned}$$

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

eg//

$$\begin{aligned} & \llbracket \text{let } x = u \text{ in } t \rrbracket (\gamma) \\ &= \text{let } \llbracket u \rrbracket (\gamma) = (w, a) \text{ in} \\ & \quad \text{let } \llbracket t \rrbracket (\gamma, a) = (v, b) \text{ in} \\ & \quad (w \# v, b) \end{aligned}$$

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

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

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

the structure of effectful programming is captured by a strong monad

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 $\mathcal{C} =$

- $T: \mathcal{C} \rightarrow \mathcal{C}$ 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

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 $f^*: TA \rightarrow TB$

A type

 TA type

$\Gamma \vdash t: A$

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

??

$\Gamma \vdash u: TA \quad \Gamma, x: A \vdash t: TB$

 $\Gamma \vdash \text{let } x = u \text{ in } t: TB$

$[\Gamma] \xrightarrow{[\eta]} T[A]$

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

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

the structure of effectful programming is captured by a strong monad

to give a ^{strong} 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 $f^{\#}: TA \rightarrow TB$
- $t: A \times TB \rightarrow T(A \times B)$
such that axioms hold

A type

 TA type

$\Gamma \vdash t: A$

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

$\Gamma \vdash u: TA \quad \Gamma, x:A \vdash t: TB$

 $\Gamma \vdash \text{let } x = u \text{ in } t: TB$

$[\Gamma] \xrightarrow{[\eta]} T[A]$

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

$[\Gamma] \xrightarrow{[\text{id}, [\eta]]} [\Gamma] \times T[A] \xrightarrow{[t]} T([\Gamma] \times [A]) \xrightarrow{[t]^{\#}} T[B]$

Where does the strength

$$t: A \times TB \longrightarrow T(A \times B)$$

come from?

Modelling effects

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

a monad in a 2-category \mathcal{C}
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

monad in
2-category of
categories,
functors,
nat. trans.
= usual defⁿ
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 $\text{return} : A \rightarrow TA$,
a bind operation
($\gg=$)

Modelling effects

[jww. Nayan Rajesh]

monad
on \mathcal{C}

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

$$\frac{\begin{array}{l} \Gamma, x:A \vdash t : TB \\ \Gamma \vdash u : TA \end{array}}{\Gamma \vdash \text{let } x = u \text{ in } t : TB}$$

...

Modelling effects

[jww. Nayan Rajesh]

monad
on \mathcal{C}

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

$$\frac{\begin{array}{l} \Gamma, x:A \vdash t : TB \\ \Gamma \vdash u : TA \end{array}}{\Gamma \vdash \text{let } x = u \text{ in } t : TB}$$

...

free clone
equipped with a
monad

= syntax of Moggi's
monadic
metalanguage

Modelling effects

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

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

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 Λ^{\rightarrow} and CL

[FOSSACS 24]

$\lambda \rightarrow$

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

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

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

$(\lambda x. t) u \equiv_{\beta} t[u/x]$

$\lambda x. t^x x \equiv_{\eta} t$

CL

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

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

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

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

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

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

λ →

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

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

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

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

BINDING

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

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

SUBSTITUTION

CL

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

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

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

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

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

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

extensionality

λ \rightarrow

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

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

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

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

\downarrow BINDING

$(\lambda x. t) u \equiv_{\beta} t[u/x]$

$\lambda x. t^x x \equiv_{\eta} t$

\downarrow SUBSTITUTION

CL

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

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

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

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

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

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

$\Gamma, x:A \vdash s^x x = t^x x : B$

extensionality

$\lambda \rightarrow$

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

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

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

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

BINDING

$$(\lambda x. t) u \equiv_{\beta} t[u/x]$$

$$\lambda x. t^x x \equiv_{\eta} t$$

SUBSTITUTION

CL

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

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

$$\Gamma \vdash S:(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$$

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

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

$$\Gamma \vdash s:A \rightarrow B \quad \Gamma \vdash t:A \rightarrow B$$

$$\Gamma, x:A \vdash s^x x = t^x x : B$$

$$\Rightarrow s = t$$

extensionality

$\lambda \rightarrow$

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

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

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

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

BINDING

$(\lambda x. t) u \equiv_{\beta} t[u/x]$

$\lambda x. t^x x \equiv_{\eta} t$

SUBSTITUTION

NO SUBSTITUTION

ALGEBRAIC (no binding)

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

CL

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

$\Gamma \vdash S : (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

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

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

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

extensionality

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

$$\left(\lambda^{\rightarrow}\text{-terms} \right) \text{ modulo } \beta\eta \cong \left(\text{CL-terms modulo weak extensionality} \right)$$

~ 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 $A \rightarrow B \in \mathbb{C}$ $\frac{A \text{ type} \quad B \text{ type}}{A \rightarrow B \text{ type}}$
- multimaps $A \Rightarrow B, A \xrightarrow{\text{eval}_{A \Rightarrow B}} B$ $\frac{f: A \rightarrow B, x: A \vdash fx: B}{}$

[essentially Hyland '14, '15]

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

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

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

- multimaps $A \Rightarrow B, A \xrightarrow{\text{eval}_{A,B}} B$

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

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

$$\begin{array}{ccc}
 \mathbb{C}(\Gamma, A; B) & \xleftrightarrow{\cong} & \mathbb{C}(\Gamma; A \Rightarrow B) \\
 \Gamma, A \xrightarrow{u, A} (A \Rightarrow B), A \xrightarrow{\text{eval}} B & \longleftarrow u & \\
 & & \frac{f: A \rightarrow B, x: A \vdash fx: B}{\Gamma, x: A \vdash (fx)[\frac{u^x}{f}, \frac{x}{x}]: B}
 \end{array}$$

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

[essentially Hyland '14, '15]

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

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

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

- multimaps $A \Rightarrow B, A \xrightarrow{\text{eval}_{A \Rightarrow B}} B$

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

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

$$\begin{array}{ccc}
 \frac{\Gamma, x: A \vdash t: B}{\Gamma \vdash \lambda x. t: A \rightarrow B} & t \longmapsto \lambda t & \\
 \mathbb{C}(\Gamma, A; B) & \xrightarrow{\cong} & \mathbb{C}(\Gamma; A \Rightarrow B) \\
 \Gamma, A \xrightarrow{u, A} (A \Rightarrow B), A \xrightarrow{\text{eval}} B & \longleftarrow u & \\
 & & \frac{\Gamma \vdash u: A \rightarrow B}{\Gamma, x: A \vdash u^x: A \rightarrow B} \\
 & & \frac{f: A \rightarrow B, x: A \vdash fx: B \quad \Gamma, x: A \vdash xc: A}{\Gamma, x: A \vdash (fx)[\frac{u^x}{f}, \frac{x}{c}]: B}
 \end{array}$$

[essentially Hyland '14, '15]

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

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

- multimaps $A \Rightarrow B, A \xrightarrow{\text{eval}_{A \Rightarrow B}} B$ $\frac{f: A \rightarrow B, x: A \vdash fx: B}{}$

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

$$\frac{\Gamma, x: A \vdash t: B}{\Gamma \vdash \lambda x. t: A \rightarrow B} \quad \begin{array}{c} t \mapsto \Lambda t \\ \mathbb{C}(\Gamma, A; B) \xrightarrow{\cong} \mathbb{C}(\Gamma; A \Rightarrow B) \end{array}$$

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

$$\frac{\Gamma \vdash u: A \rightarrow B}{\Gamma, x: A \vdash u^x: A \rightarrow B} \quad \frac{f: A \rightarrow B, x: A \vdash fx: B}{\Gamma, x: A \vdash \gamma x: A}$$

$$(\lambda x. t)^x \gamma x \approx_{\beta} t \quad ; \quad \lambda x. u^x \gamma x =_{\eta} \gamma x$$

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

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

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

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

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

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

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

A type B type

 $A \rightarrow B$ type

$\Gamma \vdash t: A \rightarrow B$ $\Gamma \vdash u: A$
 $\Gamma \vdash t \cdot u: B$

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

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

A type B type

 $A \rightarrow B$ type

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

$\Gamma \vdash t: A \rightarrow B$ $\Gamma \vdash u: A$
 $\Gamma \vdash t \cdot u: B$

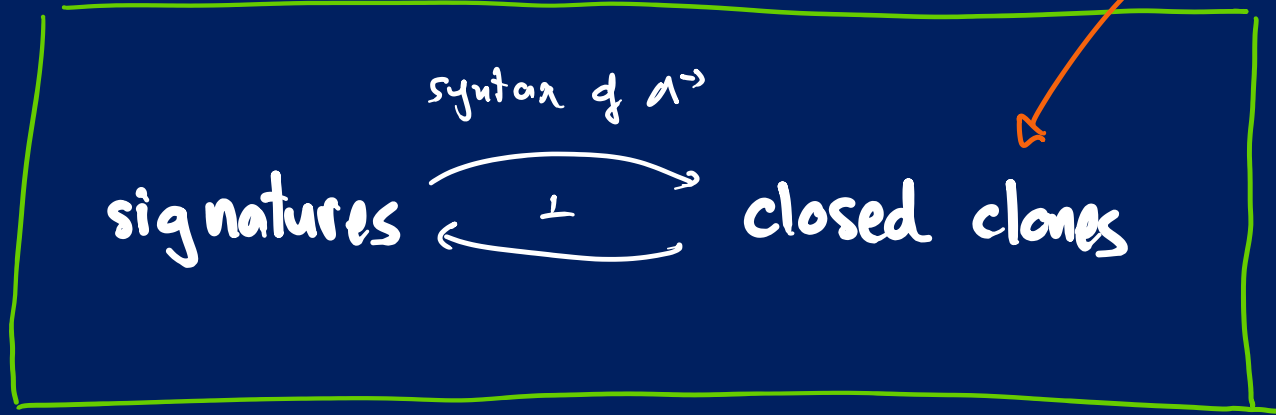
- distinguished nullary multimaps S and K
s.t. the CL equations hold

$\overline{\vdash S: \dots}$ $\overline{\vdash K: \dots}$
+weakening

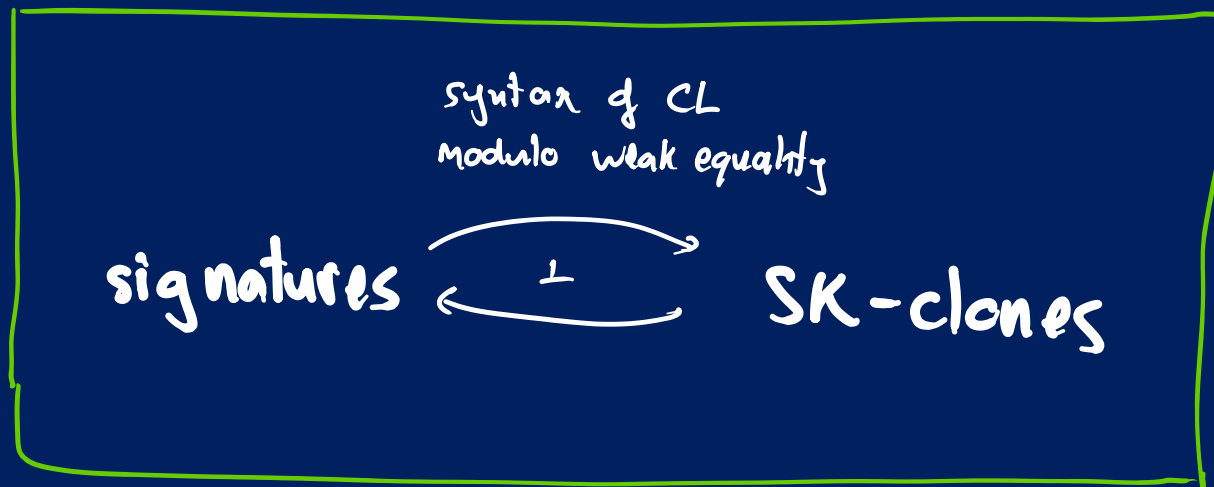
$$S \ p_1 \ p_2 \ p_3 = (p_1 \ p_3) (p_2 \ p_3) \quad \parallel$$

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

these are good models...



NB have exponentials
but no products!



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" $\left\{ \begin{array}{l} \text{for every } \Gamma \vdash t : A \rightarrow B \text{ define } \vdash t^c : \Gamma \rightarrow (A \rightarrow B) \\ \text{such that } \Gamma \vdash (t^c)^\Gamma x_1 \dots x_n = t : A \rightarrow B \end{array} \right.$

+ 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" $\left\{ \begin{array}{l} \text{for every } \Gamma \vdash t : A \rightarrow B \text{ define } \vdash t^c : \Gamma \rightarrow (A \rightarrow B) \\ \text{such that } \Gamma \vdash (t^c)^\Gamma x_1 \dots x_n = t : A \rightarrow B \end{array} \right.$

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

ie $\mathbb{C}(\emptyset; \Gamma \Rightarrow (A \Rightarrow B)) \xrightarrow{q^{\Gamma; A, B}} \mathbb{C}(\Gamma; A \Rightarrow B)$ is an iso
 $t \longmapsto (t^\Gamma) x_1 \dots x_n$

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-class that admit extensional bracket abstraction

"bracket abstraction" $\left\{ \begin{array}{l} \text{for every } \Gamma \vdash t : A \rightarrow B \text{ define } \vdash t^c : \Gamma \rightarrow (A \rightarrow B) \\ \text{such that } \Gamma \vdash (t^c)^{\Gamma} x_1 \dots x_n = t : A \rightarrow B \end{array} \right.$

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

ie $\mathbb{C}(\Delta; \Gamma \Rightarrow (A \Rightarrow B)) \xrightarrow{q^{\Gamma; A, B}} \mathbb{C}(\Gamma; A \Rightarrow B)$ 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 \cong category of
closed clones

CL with extensional
weak equality

\wedge^{\rightarrow}

syntax of CL
with extensional
weak equality = free extensional
SK - clone

category
of
"SK-categories"

\cong

category of
extensional
SK - clones = category of
closed clones

CL with extensional
weak equality

$\wedge \rightarrow$

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

- eg//
- correspond to categorical models
 - soundness + completeness
 - contexts \neq products
 - ...

on-going work: ① do this in an enriched setting

② what other languages have a "combinator" version?