Multi-ary Models for Programming Languages*

Philip Saville , University of Oxford

Feb. 2024

What is denotational semantics?

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

2. Answer questions about program behaviour using the model

What is denotational semantics?

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

② Answer questions about program behaviour using the model

$\rightsquigarrow$ reduce to algebraic questions

eg.
- refuting $=_{\beta\eta}$ or $\simeq_{ctx}$
- proving program transformations are safe
- making precise what programs are

What is denotational semantics?

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

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

~~$\rightsquigarrow$ reduce to algebraic questions~~

~~eg. • refuting $=_{\beta\eta}$ or $\simeq_{ctx}$~~
~~• 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 $\qquad \xrightarrow{[\![-]\!]} \qquad$ morphism

$\Gamma \vdash t : B \qquad\qquad\qquad [\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![B]\!]$

in a category

with enough structure to model the language

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

program $\overset{[\![-]\!]}{\rightsquigarrow}$ morphism

$\Gamma \vdash t : B$ $\rightsquigarrow$ $[\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![B]\!]$

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

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \text{categories}^{*} \arrow[l, bend left] \end{tikzcd}$$

= choice of base types and constants

models of the language

* with extra structure depending on the type theory

Soundness and completeness

signatures ← categories*

= choice of base types and constants

models of the language

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

* 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

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "F = \text{free}"] & \text{categories}^{*} \arrow[l, bend left] \end{tikzcd}$$

= choice of base types
and constants

models of
the language

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

* 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

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "F = \text{free}"] \arrow[r, phantom, "\perp"] & \text{categories}^{*} \arrow[l, bend left] \end{tikzcd}$$

= choice of base types and constants

models of the language

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

* 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

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "F = \text{free}"] \arrow[r, phantom, "\perp"] & \text{categories}^{*} \arrow[l, bend left] \end{tikzcd}$$

= choice of base types
and constants

models of
the language

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

equivalently ...

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \text{categories} \arrow[l, bend left] \end{tikzcd}$$

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

* with extra structure depending on the type theory

$$\text{signatures} \underset{\longleftarrow}{\overset{F}{\underset{\perp}{\longrightarrow}}} \text{categories}$$

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

$$\begin{tikzcd} & \mathbb{C} \\ S \arrow[ur, "s"'] & \end{tikzcd}$$

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

* with extra structure depending on the type theory

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \text{categories} \arrow[l, bend left] \end{tikzcd}$$

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

$$\begin{tikzcd} F[S] \arrow[rr, dashed, "\exists!\ \text{functor}^*\ s[\![-]\!]", "\text{such that} \ldots"'] & & \mathbb{C} \\ S \arrow[urr, "s"'] & & \end{tikzcd}$$

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

* with extra structure depending on the type theory

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \text{categories} \arrow[l, bend left] \end{tikzcd}$$

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

$$\begin{tikzcd} F[S] \arrow[rr, dashed, "\exists!\ \text{functor}^{*}\ s[\![-]\!]", "\text{such that...}"'] & & \mathbb{C} \\ S \arrow[u] \arrow[urr, "s"'] & & \end{tikzcd}$$

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

* with extra structure depending on the type theory

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \text{categories} \arrow[l, bend left] \end{tikzcd}$$

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

$$\begin{tikzcd} F[S] \arrow[r, dashed, "\exists!\ \text{functor}^*\ s[\![-]\!]", "\text{such that...}"'] & \mathbb{C} \\ S \arrow[u] \arrow[ur, "s"'] & \end{tikzcd}$$

maps = terms in the type theory

$s[\![t]\!]$ = semantic interpretation of term t

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

* with extra structure depending on the type theory

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \text{categories} \arrow[l, bend left] \end{tikzcd}$$

Soundness = $s[\![-]\!]$ preserves all the structure

completness = $F[S]$ 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

$$\begin{tikzcd} F[S] \arrow[rr, dashed, "\exists!\ \text{functor}^*\ s[\![-]\!]", "\text{such that} \dots"'] & & \mathbb{C} \\ S \arrow[u] \arrow[urr, "s"'] & & \end{tikzcd}$$

$s[\![t]\!]$ = semantic interpretation of term t

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

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \text{categories} \arrow[l, bend left] \end{tikzcd}$$

= specifications of type theories

⇝ generate terms

morphisms $f : A \to B$

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

Some awkwardness

① have to restrict to terms in contexts of length 1 when defining F[S]

② products get conflated with contexts

③ $[\![t]\!]$ in the free model is not t itself

④ need some fiddly inductions to show F[S] has the right structure

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \text{categories} \arrow[l, bend left] \end{tikzcd}$$

= specifications of type theories

~ generate terms

$\boxed{x_1 : A_1, \ldots, x_n : A_n \vdash t : B}$

morphisms $\boxed{f : A \to B}$

This talk:

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

$$\begin{tikzcd}
& \text{multi-ary structures} \arrow[dl, bend left] \arrow[dr, bend left] & \\
\text{signatures} \arrow[ur, bend left] \arrow[rr, bend left, "F"] \arrow[rr, phantom, "\perp"] & & \text{categories} \arrow[ll, bend left] \arrow[ul, bend left]
\end{tikzcd}$$

= specifications of type theories

⇝ generate terms

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

morphisms $f : A \to B$

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

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

$$\begin{tikzcd}
 & \text{multi-ary structures} \arrow[dl, shift left] \arrow[dr, shift left] & \\
\text{signatures} \arrow[ur, shift left, phantom, "\dashv"] \arrow[ur, shift left] \arrow[rr, bend left, "F"] \arrow[rr, phantom, "\perp"] & & \text{categories} \arrow[ul, shift left] \arrow[ll, bend left]
\end{tikzcd}$$

= specifications of type theories

~ generate terms

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

morphisms $f : A \to B$

This talk:

interpretation of t in free model is t itself

contexts separate from products

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

$$
\begin{tikzcd}
 & \text{multi-ary structures} \arrow[dl, bend left] \arrow[dr, bend left] & \\
\text{signatures} \arrow[ur, bend left] \arrow[rr, bend left, "F"] \arrow[rr, phantom, "\perp"] & & \text{categories} \arrow[ll, bend left] \arrow[ul, bend left]
\end{tikzcd}
$$

easy to prove soundness and completeness

= specifications of type theories

~ generate terms

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

morphisms $f : A \to B$

└ generalises easily via enrichment

e.g. 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

clones

cartesian simple type theories

muticategories, symmetric multicategories

clones

ordered / linear

~~cartesian~~ simple type theories

- a type theory has: (simple, inserted above "type theory")
    - types $A, B, C, \ldots$
    - terms $x_1 : A_1, \ldots, x_n : A_n \vdash f : B$, including $x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i$ for $i = 1, \ldots, n$
    - a substitution operation

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

- a type theory has:
  simple

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

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

- a type theory (simple) has:
  - types $A, B, C, \ldots$
  - terms $x_1 : A_1, \ldots, x_n : A_n \vdash f : B$,
    including $x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i$ for $i = 1, \ldots, n$
  - a substitution operation

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

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

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

simple

$x_i[{}^{u_1}/_{x_1},\dots,{}^{u_n}/_{x_n}] = u_i$

$t[{}^{x_1}/_{x_1},\dots,{}^{x_n}/_{x_n}] = t$

$t[{}^{u_1}/_{x_1},\dots,{}^{u_n}/_{x_n}][{}^{v_1}/_{y_1},\dots,{}^{v_m}/_{y_m}] = t[\dots,{}^{u_i[\dots,{}^{v_j}/_{y_j},\dots]}/_{x_i},\dots]$

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

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

multisorted, abstract

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

[Hall]

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

$$\frac{f : A_1, \ldots, A_n \longrightarrow B \qquad (g_i : \Delta \longrightarrow A_i)_{i=1,\ldots,n}}{f[g_1, \ldots, g_n] : \Delta \longrightarrow B}$$

multisorted, abstract

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

[Hall]

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

$$\frac{f : A_1, \ldots, A_n \longrightarrow B \qquad (g_i : \Delta \longrightarrow A_i)_{i=1\ldots n}}{f[g_1, \ldots, g_n] : \Delta \longrightarrow B}$$

multisorted, abstract

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

[Hall]

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

$$\frac{f : A_1, \ldots, A_n \longrightarrow B \qquad (g_i : \Delta \longrightarrow A_i)_{i=1,\ldots,n}}{f[g_1, \ldots, g_n] : \Delta \longrightarrow B}$$

multisorted, abstract

def: a clone $\mathbb{C}$ has: [Hall]

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

$$\frac{f : A_1, \ldots, A_n \to B \qquad (g_i : \Delta \to A_i)_{i=1,\ldots,n}}{f[g_1, \ldots, g_n] : \Delta \to B}$$

multisorted, abstract

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

[Hall]

$$p_i[f_1,\dots,f_n] = f_i$$
$$f[p_1,\dots,p_n] = f$$
$$(f[g_1,\dots,g_n])[h_1,\dots,h_m] = f[\dots,g_i[h_\cdot],\dots]$$

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

$$\frac{f : A_1, \dots, A_n \longrightarrow B \qquad (g_i : \Delta \longrightarrow A_i)_{i=1\dots n}}{f[g_1, \dots, g_n] : \Delta \longrightarrow B}$$

clones $\simeq$

- 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
- ...

cartesian simple type theories

A GENERAL THEORY

$\boxed{\begin{array}{c}\text{type theories with structure} \\ \simeq \\ \text{monads on the category of clones}\end{array}}$

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

Syntax from semantics

simply-typed λ-calculus

for any signature of base types and constants

types : $A, B ::= \underbrace{\beta}_{\text{base types}} \mid 1 \mid A \times B \mid A \to B$

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

equations : (β) $\pi_i(\langle t_1, t_2 \rangle) = t_i$ $\qquad (\lambda x.t)\,u = t[u/x]$

(η) $\langle \pi_1(t), \pi_2(t) \rangle = t$ $\qquad \lambda x.(t\,x) = t$

$\Lambda^{\times}$ = just products

$\Lambda^{\to}$ = just exponentials

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

Modelling products

cartesian product in category $\mathbb{C}$ $=$ universal arrow from $\Delta^{(n)} : \mathbb{C}^{\times n} \longrightarrow \mathbb{C}$ to $(A_1, \ldots, A_n) \in \mathbb{C}^{\times n}$

Modelling products

cartesian product in clone $\mathbb{C}$ $\overset{\text{def}}{=}$ universal arrow from $\Delta^{(n)} : \mathbb{C}^{\times n} \longrightarrow \mathbb{C}$ to $(A_1, \ldots, A_n) \in \mathbb{C}^{\times n}$

Modelling products

a <u>cartesian product</u> in a clone $\mathbb{C}$ consists of:

1) an object $A_1 \times A_2$

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

s.t.

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

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

is an iso.

Modelling products

a <u>cartesian product</u> in a clone $\mathbb{C}$ consists of:

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

1) an object $A_1 \times A_2$

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

s.t.

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

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

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 $\pi_i : A_1 \times A_2 \to A_i$

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

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

s.t.

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

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

is an iso.

Modelling products

a <u>cartesian product</u> in a clone $\mathbb{C}$ consists of:

1) an object $A_1 \times A_2$

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

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

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

s.t.

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

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

is an iso.

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

Modelling products

a <u>cartesian product</u> in a clone $\mathbb{C}$ consists of:

1) an object $A_1 \times A_2$

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

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

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

s.t.

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

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

is an iso.

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

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

Modelling products

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

1) an object $A_1 \times A_2$

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

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

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

s.t.

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

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

$\pi_i(p)[{}^{\langle u_1, u_2 \rangle}/_{p}] = u_i$

$\langle \pi_1(p), \pi_2(p) \rangle = p$

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

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left, "\text{take syntax of } \Lambda^{\times}"] \arrow[r, phantom, "\perp"] & \text{clones with products} \arrow[l, bend left] \end{tikzcd}$$

= base types + constants

free clone with products = syntax of $\Lambda^{\times}$

Modelling products

defined using natural
generalisation from categories

clones with products

take syntax of $\Lambda^{\times}$

$\dashv$

signatures

= base types + constants

restrict to unary maps

categories with products

easy algebraic proof ?

free clone with products = syntax of $\Lambda^{\times}$

Modelling products

defined using natural generalisation from categories

$$
\begin{tikzcd}
 & \text{clones with products} \arrow[dl, bend left] \arrow[dr, bend left, "\text{restrict to unary maps}"] & \\
\text{signatures} \arrow[ur, bend left, "\text{take syntax of } \Lambda^{\times}"] \arrow[ur, phantom, "\perp"] & & \text{categories with products} \arrow[ul, bend left, "\mathcal{P}"] \arrow[ul, phantom, "\simeq"]
\end{tikzcd}
$$

easy algebraic proof!

signatures = base types + constants

$(\mathcal{P}\mathbb{C})(A_1, \ldots, A_n; B) := \mathbb{C}(\prod_{i=1}^{n} A_i, B)$

free clone with products = syntax of $\Lambda^{\times}$

Modelling products

defined using natural generalisation from categories

easy algebraic proof ?

$$
\begin{tikzcd}
 & \text{clones with products} \arrow[dl, bend left, ""] \arrow[dr, bend left, "\text{restrict to unary maps}"] & \\
\text{signatures} \arrow[ur, bend left, "\text{take syntax of } \Lambda^{\times}"] \arrow[ur, phantom, "\perp"] & & \text{categories with products} \arrow[ul, bend left, "\mathcal{P}"] \arrow[ul, phantom, "\simeq"]
\end{tikzcd}
$$

signatures = base types + constants

$(\mathcal{P}\mathbb{C})(A_1, \ldots, A_n; B) := \mathbb{C}(\prod_{i=1}^{n} A_i, B)$

free clone with products = syntax of $\Lambda^{\times}$

free cartesian category = $\Lambda^{\times}$-terms $x : A \vdash t : B$

Modelling products: soundness and completeness

free clone with all products $=$ syntax of $\Lambda^{\times}$ $=$ typed λ-calculus with just products

$$\begin{tikzcd}
\left(\text{syntax of } \Lambda^{\times}\right) \arrow[r, dashed, "\text{semantic interpretation}"] & P\left(\text{cartesian category}\right) \\
\text{base types + constants} \arrow[u] \arrow[ur, "\text{choice of interpretation for basic data}"'] &
\end{tikzcd}$$

Modelling products: soundness and completeness

free clone with all products = syntax of $\Lambda^\times$ = typed λ-calculus with just products

sound :: cartesian clone homomorphism

semantic interpretation

$$\left(\begin{array}{c}\text{syntax}\\ \text{of } \Lambda^\times\end{array}\right) \dashrightarrow \underset{=}{P}\,(\text{cartesian category})$$

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 $\Lambda^{\times}$
- 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 $\Lambda^{\times}$

soundness + completeness

equivalence with categorical models

Modelling effects

Modelling effects

natural definition of monads for clones

$\Longrightarrow$ 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

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

the structure of effectful programming is captured by a strong monad

eg/

$TA = A + 1$

$TA = List(A)$

$TA = S^* \times A$

① mark types as effectful

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

the structure of effectful programming is captured by a strong monad

eg/ $TA = A+1$

$TA = \mathrm{List}(A)$

$TA = S^* \times A$

① mark types as effectful

② every pure program is trivially effectful

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

the structure of effectful programming is captured by a strong monad

eg// $TA = A+1$

$TA = List(A)$

$TA = S^* \times A$

eg// $A \xrightarrow{inl} A+1$

$A \longrightarrow List A$

$a \longmapsto [a]$

$A \longrightarrow S^* \times A$

$a \longmapsto (\varepsilon, a)$

① mark types as effectful

② every pure program is trivially effectful

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

the structure of effectful programming is captured by a strong monad

eg//

$TA = A + 1$

$TA = List(A)$

$TA = S^* \times A$

eg//

$A \xrightarrow{inl} A+1$

$A \longrightarrow List\,A$

$a \longmapsto [a]$

$A \longrightarrow S^* \times A$

$a \longmapsto (\varepsilon, a)$

① mark types as effectful

② every pure program is trivially effectful

③ explicit sequencing by let-binding

$$\frac{A \text{ type}}{TA \text{ type}} \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash \mathrm{return}(t) : TA} \qquad \frac{\Gamma \vdash u : TA \qquad \Gamma, x : A \vdash t : TB}{\Gamma \vdash \mathrm{let}\ x = u\ \mathrm{in}\ t : TB}$$

the structure of effectful programming is captured by a strong monad

eg// $TA = A+1$

$TA = List(A)$

$TA = S^* \times A$

eg// $A \xrightarrow{inl} A+1$

$A \longrightarrow List A$

$a \longmapsto [a]$

$A \longrightarrow S^* \times A$

$a \longmapsto (\varepsilon, a)$

① mark types as effectful

② every pure program is trivially effectful

③ explicit sequencing by let-binding

eg//

$[\![\text{let } x = u \text{ in } t]\!](\gamma)$

$= \text{let } [\![u]\!](\gamma) = (w, a) \text{ in}$

$\quad \text{let } [\![t]\!](\gamma, a) = (v, b) \text{ in}$

$\quad (w + v, b)$

$$\frac{A \text{ type}}{TA \text{ type}} \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash \text{return}(t) : TA} \qquad \frac{\Gamma \vdash u : TA \qquad \Gamma, x : A \vdash t : TB}{\Gamma \vdash \text{let } x = u \text{ in } t : TB}$$

the structure of effectful programming is captured by a strong monad

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

the structure of effectful programming is captured by a strong monad

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

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

- $T : \mathbb{C} \to \mathbb{C}$ a functor

the structure of effectful programming is captured by a strong monad

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

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

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

- $T : \mathbb{C} \to \mathbb{C}$ a functor
- $\eta_A : A \to TA$ for each A

the structure of effectful programming is captured by a strong monad

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

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

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

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

??

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

$$\frac{[\![\Gamma]\!] \xrightarrow{[\![u]\!]} T[\![A]\!] \qquad [\![\Gamma]\!] \times [\![A]\!] \xrightarrow{[\![t]\!]} T[\![B]\!]}{[\![\Gamma]\!] \xrightarrow{\langle id, [\![u]\!] \rangle} [\![\Gamma]\!] \times T[\![A]\!] \xrightarrow{??} T([\![\Gamma]\!] \times [\![A]\!]) \xrightarrow{[\![t]\!]^*} T[\![B]\!]}$$

the structure of effectful programming is captured by a strong monad

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

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

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

to give a (strong) monad on a category $\mathbb{C}$ =

- $T : \mathbb{C} \longrightarrow \mathbb{C}$ a functor
- $\eta_A : A \longrightarrow TA$ for each A
- for each $f : A \longrightarrow TB$, a map $f^* : TA \longrightarrow TB$
- $t : A \times TB \longrightarrow T(A \times B)$

such that axioms hold

$$[\![\Gamma]\!] \xrightarrow{[\![u]\!]} T[\![A]\!]$$

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

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

Where does the strength

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

come from?

Modelling effects

objects $A, B, \ldots$
1-cells $f, g : A \to B$
2-cells $\tau : f \Rightarrow g$

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

- an object c
- a 1-cell $T : c \longrightarrow c$
- 2-cells $\eta : id_c \Longrightarrow T$

  $\mu : T \circ T \Longrightarrow T$

+ axioms

monad in
2-category of
categories,
functors,
nat. trans.
= usual def^n
of monad!

Modelling effects

[jww. Nayan Rajesh]

instantiating in the 2-category of clones:

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

Modelling effects

[jww. Nayan Rajesh]

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

...

Modelling effects

[jww. Nayan Rajesh]

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

...

free clone equipped with a monad = syntax of Moggi's monadic metalanguage

Modelling effects

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

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

if $\mathbb{C}$ has products, T becomes a strong monad on the cartesian category $\bar{\mathbb{C}}$ = restrict $\mathbb{C}$ to unary maps

(linear version: monoidal)

Modelling effects

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

$$
\begin{tikzcd}[column sep=huge]
\text{signatures} \arrow[r, bend left, "\text{syntax } \& \ \lambda_{ml}"] \arrow[r, phantom, "\perp"] & \begin{array}{c}\text{clones with products} \\ + \text{ a monad}\end{array} \arrow[l, bend left] \arrow[r, bend left, "\begin{array}{c}\text{restrict to} \\ \text{unary contexts}\end{array}"] \arrow[r, phantom, "\perp"] & \begin{array}{c}\text{cartesian categories} \\ \text{with a strong monad}\end{array} \arrow[l, bend left]
\end{tikzcd}
$$

$\rightsquigarrow$ 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 $\Lambda^{\rightarrow}$ and CL

[FoSSaCS 24]

$\Lambda^{\to}$

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

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

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}$$

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

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

CL

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

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t \cdot u : B}$$

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

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

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

$$\underbrace{\frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A \to B \qquad \Gamma, x : A \vdash s^x\,x = t^x\,x : B}{}}_{\text{extensionality}} \Longrightarrow s = t$$

$\Lambda^{\to}$

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

$$\frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \underbrace{\lambda x}_{\text{BINDING}} . t : A \to B}$$

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

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

CL

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

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

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

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

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

$$\underbrace{\frac{\Gamma \vdash s : A \to B \quad \Gamma \vdash t : A \to B}{\Gamma, x : A \vdash s^x\, x = t^x\, x : B}}_{\text{extensionality}} \implies s = t$$

$\Lambda^{\to}$

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

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}$$

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

BINDING

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

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

SUBSTITUTION

CL

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

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t \cdot u : B}$$

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

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

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

$$\underbrace{\frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A \to B}{\Gamma, x : A \vdash s^x\, x = t^x\, x : B} \implies s = t}_{\text{extensionality}}$$

$\Lambda^{\to}$

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

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}$$

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

BINDING

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

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

SUBSTITUTION

CL

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

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t \cdot u : B}$$

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

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

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

$$\underbrace{\frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A \to B}{\Gamma, x : A \vdash s^x\, x = t^x\, x : B} \implies s = t}_{\text{extensionality}}$$

$\Lambda^{\to}$

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

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}$$

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

BINDING

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

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

SUBSTITUTION

NO SUBSTITUTION

CL

ALGEBRAIC (no binding)

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

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t \cdot u : B}$$

$$\overline{\Gamma \vdash S : (A \to (B \to C)) \longrightarrow ((A \to B) \to (A \to C))}$$

$$\overline{\Gamma \vdash K : A \to (B \to A)}$$

$$((S \cdot f) \cdot g) \cdot x = (f \cdot x) \cdot (g \cdot x)$$

$$(K \cdot x) \cdot y = x$$

weak equality

$$\frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A \to B}{\Gamma, x : A \vdash s^x\, x = t^x\, x : B} \implies s = t$$

extensionality

a classical result:

[Schönfinkel, Curry, ...]

$$\begin{pmatrix} \lambda^{\to}\text{-terms} \\ \text{modulo } \beta\eta \end{pmatrix} \cong \begin{pmatrix} \text{CL-terms modulo} \\ \text{weak extensionality} \end{pmatrix}$$

$\rightsquigarrow$ quite easy to prove

$\rightsquigarrow$ 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 $A \to B \in \mathbb{C}$

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

[essentially Hyland '14, '15]

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

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

$$\frac{A \text{ type} \qquad B \text{ type}}{A \to B \text{ type}}$$

- multimaps $A \to B, A \xrightarrow{\mathrm{eval}_{A,B}} B$

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

[essentially Hyland '14, '15]

<u>def</u>: a <u>closed clone</u> is a clone $\mathbb{C}$ with

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

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

- multimaps $A \Rightarrow B, A \xrightarrow{eval_{A,B}} B$

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

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

$$\mathbb{C}(\Gamma, A; B) \underset{\cong}{\rightleftarrows} \mathbb{C}(\Gamma; A \Rightarrow B)$$

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

$$\cfrac{f : A \to B, x : A \vdash fx : B \qquad \cfrac{\cfrac{\Gamma \vdash u : A \to B}{\Gamma, x : A \vdash u^x : A \to B} \quad \Gamma, x : A \vdash x : A}{}}{\Gamma, x : A \vdash (fx)[u^x/f, x/x] : B}$$

[essentially Hyland '14, '15]

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

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

$$\frac{A \text{ type} \qquad B \text{ type}}{A \to B \text{ type}}$$

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

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

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

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

$$t \longmapsto \Lambda t$$

$$\mathbb{C}(\Gamma, A; B) \underset{\longleftarrow}{\overset{\longrightarrow}{\cong}} \mathbb{C}(\Gamma; A \Rightarrow B)$$

$$\Gamma, A \xrightarrow{u, A} (A \Rightarrow B), A \xrightarrow{\mathrm{eval}} B \longleftarrow\!\!| \ u$$

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

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

[essentially Hyland '14, '15]

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

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

$$\frac{A \text{ type} \qquad B \text{ type}}{A \to B \text{ type}}$$

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

$$\overline{f : A \to B,\ x : A \vdash f x : B}$$

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

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

$$t \longmapsto \Lambda t$$

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

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

$$\frac{\dfrac{\Gamma \vdash u : A \to B}{\Gamma, x : A \vdash u^x : A \to B} \qquad f : A \to B, x : A \vdash f x : B \qquad \Gamma, x : A \vdash x : A}{\Gamma, x : A \vdash (f x)[u^x/f, x/x] : B}$$

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

def: an <u>SK-clone</u> is a clone $\mathbb{C}$ with

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

$$\frac{A \text{ type} \qquad B \text{ type}}{A \to B \text{ type}}$$

def: an <u>SK-clone</u> is a clone $\mathbb{C}$ with

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

$$\frac{A \text{ type} \qquad B \text{ type}}{A \to B \text{ type}}$$

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

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t \cdot u : B}$$

def: an <u>SK-clone</u> is a clone $\mathbb{C}$ with

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

$$\frac{A \text{ type} \qquad B \text{ type}}{A \to B \text{ type}}$$

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

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t \cdot u : B}$$

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

$$\frac{}{\vdash S : \ldots} \qquad \frac{}{\vdash K : \ldots}$$

+ weakening

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

$$K\, p_1\, p_2 = p_1 \qquad \ldots$$

these are good models...

NB have exponentials but no products?

syntax of $\lambda^{\to}$

$$\text{signatures} \underset{\perp}{\rightleftarrows} \text{closed clones}$$

syntax of CL modulo weak equality

$$\text{signatures} \underset{\perp}{\rightleftarrows} \text{SK-clones}$$

How to add extensionality?

Notation for weakening:

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

basic idea for syntax:

"bracket abstraction"

$$\begin{cases} \text{for every } \Gamma \vdash t : A \to B \text{ define } \vdash t^c : \Gamma \to (A \to B) \\ \text{such that } \Gamma \vdash (t^c)^{\Gamma}\, x_1 \dots x_n = t : A \to B \end{cases}$$

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

How to add extensionality?

Notation for weakening:

$$\frac{\Delta \vdash t : B}{\Delta, \Gamma \vdash t^{\Gamma} : B}\,\mathrm{wkn}$$

basic idea for syntax:

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

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

ie $\mathbb{C}(\diamond ; \Gamma \Rightarrow (A \Rightarrow B)) \xrightarrow{q^{\Gamma;A,B}} \mathbb{C}(\Gamma ; A \Rightarrow B)$ is an iso

$t \longmapsto (t^{\Gamma})\, x_1 \cdots x_n$

How to add extensionality?

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

$\not\parallel$

$\approx$ exactly the SK-clones that admit extensional bracket abstraction

basic idea for syntax:

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

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

ie

$$\mathbb{C}(\diamond ; \Gamma \Rightarrow (A \Rightarrow B)) \xrightarrow{q^{\Gamma;A,B}} \mathbb{C}(\Gamma ; A \Rightarrow B) \text{ is an iso}$$

$$t \longmapsto (t^\Gamma)\, x_1 \ldots x_n$$

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

$\simeq$

category of
closed clones

CL with extensional
weak equality

$\lambda^{\to}$

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

category
of
"SK-categories"
=
category of extensional SK-clones $\simeq$ category of closed clones

CL with extensional weak equality

$\lambda^{\rightarrow}$

Summary

1) multi-ary models clarify the usual categorical models & 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 & 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?