Clones, closed categories, and combinatory logic

Philip Saville, University of Oxford

FoSSaCS '24

What's it all about?

1) Pedagogical: exposition of multi-ary models for simple type theories

$$\Lambda^{\times,\to} \rightleftarrows \text{cartesian closed categories}$$
$$\Lambda^{\times} \rightleftarrows \text{cartesian categories}$$
$$\Lambda^{\to} \rightleftarrows \text{??}$$

2) Categorical semantics for simply-typed λ-calculus without products (modulo βη)

3) Categorical semantics for SK-combinatory logic

A schema for categorical semantics

for a fixed simply-typed language :

A schema for categorical semantics

for a fixed simply-typed language:

signatures

= choices of base types and constants

(categories with structure)

= Models

eg/ products for modelling $\times$,
exponentials for modelling $\to$
$\vdots$

A schema for categorical semantics

for a fixed simply-typed language:

$$\text{signatures} \longleftarrow \left(\begin{array}{c}\text{categories} \\ \text{with structure}\end{array} \right)$$

signatures = choices of base types and constants

categories with structure = Models

eg/ products for modelling $\times$,
exponentials for modelling $\to$
⋮

$\mathbb{C} \longmapsto$ base types = objects
constants = morphisms

A schema for categorical semantics

for a fixed simply-typed language :

signature $S \longmapsto$ category $\mathbb{F}[S]$ with
- objects : types
- maps $A \to B$: terms $(x : A \vdash t : B)$

$\wr$

signatures $\rightleftarrows$ (categories with structure)

= choices of base types and constants

= Models

eg/ products for modelling $\times$, exponentials for modelling $\to$

$\vdots$

$\wr$

$\mathbb{C} \longmapsto$ base types = objects
constants = morphisms

A schema for categorical semantics

for a fixed simply-typed language:

FREE

signature $S \longmapsto$ category $\mathbb{F}[S]$ with
- objects : types
- maps $A \to B$: terms $(x : A \vdash t : B)$

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \left(\begin{array}{c}\text{categories} \\ \text{with structure}\end{array}\right) \arrow[l, bend left] \end{tikzcd}$$

= choices of base types and constants

= Models

eg/ products for modelling $\times$, exponentials for modelling $\to$

$\vdots$

$\mathbb{C} \longmapsto$ base types = objects
constants = morphisms

Soundness and completeness

for any model

$\mathcal{C}$

for any choice of interpretation
of base types and constants in $\mathcal{C}$

for any signature

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

Soundness and completeness

for any model

$$\begin{tikzcd}
\mathbb{F}[S] \arrow[rr, "S[\![-]\!]", "\exists!\ \text{structure-preserving functor}"'] & & \mathbb{C} \\
& S \arrow[ur, "s"'] &
\end{tikzcd}$$

for any choice of interpretation of base types and constants in $\mathbb{C}$

for any signature S

Soundness and completeness

for any model

$$\begin{tikzcd}
\mathbb{F}[S] \arrow[rr, "S[\![-]\!]"] & & \mathbb{C} \\
 & & \\
S \arrow[uu] \arrow[uurr, "s"'] & &
\end{tikzcd}$$

∃! structure-preserving functor

which extends the interpretation of S

for any choice of interpretation of base types and constants in $\mathbb{C}$

for any signature S

Soundness and completeness

for any model

$$\begin{tikzcd} \mathbb{F}[S] \arrow[r, "s[\![-]\!]"] & \mathbb{C} \\ S \arrow[u] \arrow[ur, "s"'] & \end{tikzcd}$$

∃! structure-preserving functor

which extends the interpretation of S

for any signature S

interpretation $s[\![\Gamma \vdash t : B]\!]$ of each term t;

structure-preserving ⇒ <u>soundness</u>

Soundness and completeness

$$
\begin{tikzcd}
\mathbb{F}[S] \arrow[r, "s[\![-]\!]"] & \mathcal{C} \\
S \arrow[u, hook] \arrow[ur, "s"'] &
\end{tikzcd}
$$

for any model $\mathcal{C}$

$\exists!$ structure-preserving functor

which extends the interpretation of S

for any signature S

interpretation $s[\![\Gamma \vdash t : B]\!]$ of each term t;
structure-preserving
$\Rightarrow$ **soundness**

$[\![t]\!] = [\![t']\!]$ in every model
$\Rightarrow$ equation holds in $\mathbb{F}[S]$
$\Rightarrow$ $t \doteq t'$ as terms
$\rightsquigarrow$ **completeness**

A schema for categorical semantics

└ Soundness and completeness, categorically

signature $S \longmapsto$ category $\mathbb{F}[S]$ with
- objects : types
- maps $A \to B$: terms $(x : A \vdash t : B)$

$$\begin{tikzcd} \text{signatures} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \left(\begin{array}{c} \text{categories} \\ \text{with structure} \end{array} \right) \arrow[l, bend left] \end{tikzcd}$$

= choices of base types and constants

= Models

eg/ products for modelling $\times$,
exponentials for modelling $\to$
⋮

$\mathbb{C} \longmapsto$ base types = objects
constants = morphisms

This is very powerful, but...

type theories $\rightleftarrows$ categories

terms $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ morphisms $f : A \to B$

This is very powerful, but...

type theories $\overset{??}{\rightleftarrows}$ categories

terms $\underbrace{x_1 : A_1, \dots, x_n : A_n}_{\text{arbitrary length}} \vdash t : B$

morphisms $f : \underbrace{A}_{\text{length } 1} \to B$

This is very powerful, but...

type theories $\rightleftarrows$ categories

terms $\underbrace{x_1 : A_1, \dots, x_n : A_n}_{\text{arbitrary length}} \vdash t : B$

morphisms $f : \underbrace{A}_{\text{length } 1} \to B$

standard solution:

encode contexts as products

$$x_1 : A_1, \dots, x_n : A_n \vdash t : B \longmapsto \prod_{i=1}^{n} [\![A_i]\!] \xrightarrow{[\![t]\!]} [\![B]\!]$$

standard solution:

encode contexts as products

$$x_1 : A_1, \ldots, x_n : A_n \vdash t : B \longmapsto \prod_{i=1}^{n} [\![A_i]\!] \xrightarrow{[\![t]\!]} [\![B]\!]$$

standard solution:

encode contexts as products

$$x_1 : A_1, \ldots, x_n : A_n \vdash t : B \longmapsto \prod_{i=1}^{n} [\![A_i]\!] \xrightarrow{[\![t]\!]} [\![B]\!]$$

some downsides:

① conflates product types with contexts

standard solution:

encode contexts as products

$$x_1 : A_1, \ldots, x_n : A_n \vdash t : B \longmapsto \prod_{i=1}^{n} [\![A_i]\!] \xrightarrow{[\![t]\!]} [\![B]\!]$$

if $x : A, y : B \vdash t : C$, $[\![t]\!]$ has context $p : A \times B$

some downsides:

① conflates product types with contexts

② in $\mathbb{F}[S]$ we don't get $[\![t]\!] = t$

standard solution:

encode contexts as products

$$x_1 : A_1, \ldots, x_n : A_n \vdash t : B \longmapsto \prod_{i=1}^{n} [\![A_i]\!] \xrightarrow{[\![t]\!]} [\![B]\!]$$

if $x : A, y : B \vdash t : C$,
$[\![t]\!]$ has context $p : A \times B$

some downsides:

① conflates product types with contexts

② in $\mathbb{F}[S]$ we don't get $[\![t]\!] = t$

③ not clear how to model languages without products $\rightsquigarrow$ can't construct $\mathbb{F}[S]$!

some downsides:

① conflates product types with contexts

② in $\mathbb{F}[S]$ we don't get $[\![t]\!] = t$

③ not clear how to model languages without products $\rightsquigarrow$ can't construct $\mathbb{F}[S]$!

$\hookrightarrow$ we know:

λ-calculus with $\times, 1, \to$ $\quad \overset{\simeq}{\rightleftarrows} \quad$ cartesian closed categories

λ-calculus with just $\times$ $\quad \overset{\simeq}{\rightleftarrows} \quad$ cartesian categories

λ-calculus with just $\to$ $\quad \overset{\simeq}{\rightleftarrows} \quad$??

Another old solution: multi-ary models

$$\begin{tikzcd}
\text{signatures} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \left(\begin{array}{c}\text{categories} \\ \text{with structure}\end{array}\right) \arrow[l, bend left]
\end{tikzcd}$$

= choices of base types and constants

terms

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

morphisms

$f : A \to B$

Another old solution: multi-ary models

(multi-ary models)

multi-maps

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

$$\text{signatures} \underset{\longleftarrow}{\overset{\longrightarrow}{\perp}} \left(\begin{array}{c}\text{categories}\\ \text{with structure}\end{array}\right)$$

= choices of base types and constants

terms

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

morphisms

$f : A \to B$

Another old solution: multi-ary models

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

multi-maps

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

signatures = choices of base types and constants

terms

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

morphisms

$f : A \to B$

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

multi-maps

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

= choices of base types and constants

① contexts separate from types

② in the free model, $[\![t]\!] = t$ ⇝ no encoding needed!

③ can easily model eg $\Lambda^{\to}$ = simply-typed λ-calculus without product types easy to define free models

multi-maps

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

restrict to maps $A \to B$ ✓

use products to construct contexts

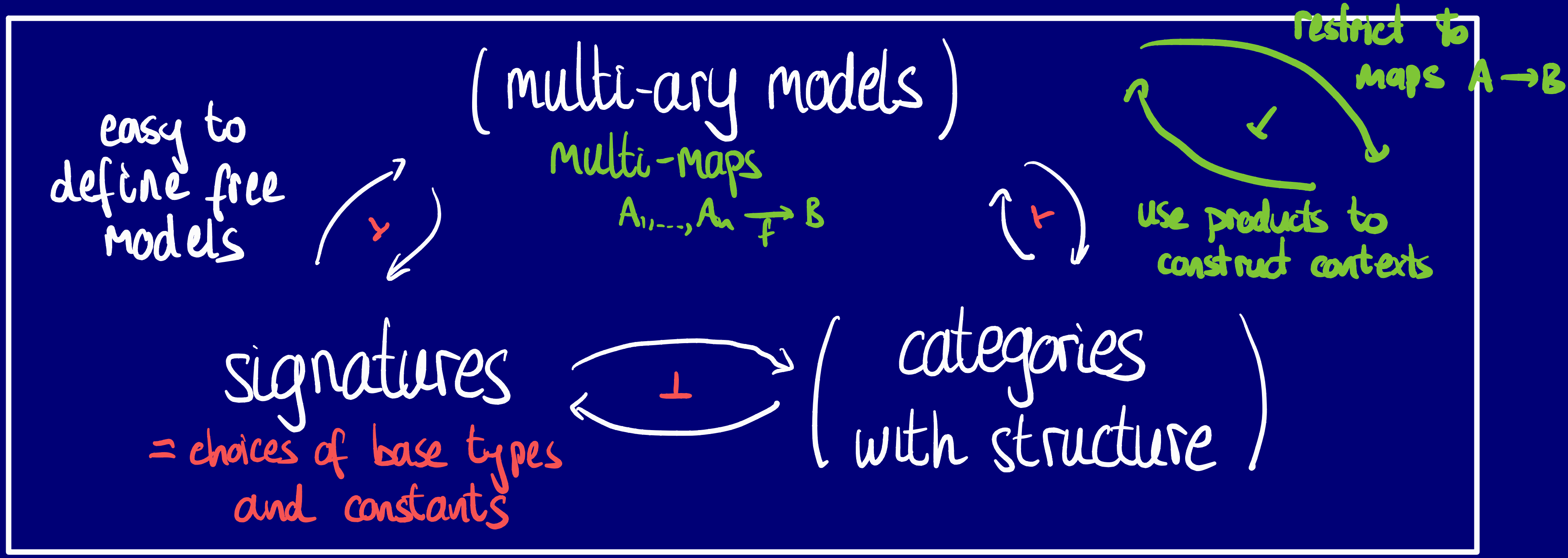
$$\begin{tikzcd}
 & \text{(multi-ary models)} \arrow[dl, bend left] \arrow[dr, bend left] & \\
\text{signatures} \arrow[ur, bend left] \arrow[ur, phantom, "\perp"] \arrow[rr, bend left] \arrow[rr, phantom, "\perp"] & & \text{(categories with structure)} \arrow[ll, bend left] \arrow[ul, bend left] \arrow[ul,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

= choices of base types and constants

① contexts separate from types

② in the free model, $[\![t]\!] = t$ ⇝ no encoding needed!

③ can easily model eg/ $\Lambda^{\to}$ = simply-typed λ-calculus without product types



① contexts separate from types

② in the free model, $[\![t]\!] = t$ ~ no encoding needed!

③ can easily model eg/ $\Lambda^{\to}$ = simply-typed λ-calculus without product types

+ works for ordered / linear / cartesian calculi

+ has natural extensions for refining languages

- more general frameworks encompassing this idea: Shulman, Fiore et al, ...
- particular instances: Hyland, Hyland + de Paiva, Blanco + Zeilberger, Melliès, ...

This insight is

old [Lambek] + known to experts

but deserves to be better-known!

- more general frameworks encompassing this idea: Shulman, Fiore et al, ...
- particular instances: Hyland, Hyland + de Paiva, Blanco + Zeilberger, Melliès, ...

This insight is

old [Lambek] + known to experts

but deserves to be better-known!

⇝ alternatives are available

Jacobs' fibrational models, CwFs, ...

Modelling $\Lambda^{\to}$ in clones

simply-typed λ-calculus without product types modulo $\beta\eta$

also in the paper:

- ordered / linear multi-ary models for $\to$, $\otimes$, $\&$
- multi-ary models of $\to, \times$

plus how these arise naturally and give the right syntax

def

$\simeq$ cartesian multicategory

An abstract clone $\mathbb{C}$ has

def

$\simeq$ cartesian multicategory

An abstract clone $\mathcal{C}$ has

- objects $A, B, \ldots$

types $A, B, \ldots$

def

$\simeq$ cartesian multicategory

An abstract clone $\mathbb{C}$ has

- objects $A, B, \ldots$ — types $A, B, \ldots$
- multimaps $A_1, \ldots, A_n \xrightarrow{t} B$ — terms $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$

def

$\simeq$ cartesian multicategory

An abstract clone $\mathbb{C}$ has

- objects $A, B, \ldots$ — types $A, B, \ldots$
- multimaps $A_1, \ldots, A_n \xrightarrow{t} B$ — terms $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$
- projections $p_i^A : A_1, \ldots, A_n \longrightarrow A_i \quad (1 \leq i \leq n)$

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

def

≃ cartesian multicategory

An abstract clone $\mathbb{C}$ has

- objects $A, B, \ldots$ — types $A, B, \ldots$
- multimaps $A_1, \ldots, A_n \xrightarrow{t} B$ — terms $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$
- projections $p_i^A : A_1, \ldots, A_n \longrightarrow A_i$ $(1 \leq i \leq n)$

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

- a substitution operation

$$\mathbb{C}(A_1, \ldots, A_n; B) \times \prod_{i=1}^{n} \mathbb{C}(\Delta; A_i) \longrightarrow \mathbb{C}(\Delta; B)$$

$$t, (u_1, \ldots, u_n) \longmapsto t[u_1, \ldots, u_n]$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash t : B \qquad (\Delta \vdash u_i : A_i)_{i=1,\ldots,n}}{\Delta \vdash t[u_1/x_1, \ldots, u_n/x_n] : B}$$

def

≃cartesian multicategory

An abstract clone $\mathbb{C}$ has

- objects $A, B, \ldots$ — types $A, B, \ldots$
- multimaps $A_1, \ldots, A_n \xrightarrow{t} B$ — terms $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$
- projections $p_i^A : A_1, \ldots, A_n \longrightarrow A_i$ $(1 \leq i \leq n)$ — $$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i}\ \text{var}$$
- a substitution operation

$$\mathbb{C}(A_1, \ldots, A_n ; B) \times \prod_{i=1}^{n} \mathbb{C}(\Delta ; A_i) \longrightarrow \mathbb{C}(\Delta ; B)$$

$$t, (u_1, \ldots, u_n) \longmapsto t[u_1, \ldots, u_n]$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash t : B \qquad (\Delta \vdash u_i : A_i)_{i=1, \ldots, n}}{\Delta \vdash t[u_1/x_1, \ldots, u_n/x_n] : B}$$

s.t.

1. $p_i[u_1, \ldots, u_n] = u_i$
2. $t[p_1, \ldots, p_n] = t$
3. $(t[u_\bullet])[v_\bullet] = t[\ldots, u_i[v_\bullet], \ldots]$

1. $x_i[u_1/x_1, \ldots, u_n/x_n] = u_i$
2. $t[x_1/x_1, \ldots, x_n/x_n] = t$
3. assoc. of subst.

def An abstract clone $\mathbb{C}$ has ≃ cartesian multicategory — an axiomatisation of a simple type theory

- objects $A, B, \ldots$ — types $A, B, \ldots$
- multimaps $A_1, \ldots, A_n \xrightarrow{t} B$ — terms $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$
- projections $p_i^A : A_1, \ldots, A_n \longrightarrow A_i$ $(1 \leq i \leq n)$ — $$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i}\,\text{var}$$
- a substitution operation

$$\mathbb{C}(A_1, \ldots, A_n; B) \times \prod_{i=1}^{n} \mathbb{C}(\Delta; A_i) \longrightarrow \mathbb{C}(\Delta; B)$$

$$t, (u_1, \ldots, u_n) \longmapsto t[u_1, \ldots, u_n]$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash t : B \qquad (\Delta \vdash u_i : A_i)_{i=1,\ldots,n}}{\Delta \vdash t[u_1/x_1, \ldots, u_n/x_n] : B}$$

s.t.

① $p_i[u_1, \ldots, u_n] = u_i$

② $t[p_1, \ldots, p_n] = t$

③ $(t[u_\bullet])[v_\bullet] = t[\ldots, u_i[v_\bullet], \ldots]$

① $x_i[u_1/x_1, \ldots, u_n/x_n] = u_i$

② $t[x_1/x_1, \ldots, x_n/x_n] = t$

③ assoc. of subst.

[Jacobs, Hyland, ...]

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

follows naturally from categorical definition

[Jacobs, Hyland, ...]

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

follows naturally from categorical definition

- for every $A, B \in \mathbb{C}$ an object $A \Rightarrow B$

[Jacobs, Hyland, ...]

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

follows naturally from categorical definition

- for every $A, B \in \mathbb{C}$ an object $A \Rightarrow B$
- a multimap $A \Rightarrow B, A \xrightarrow{\text{eval}} B$ inducing

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

$$\Gamma, A \xrightarrow{t, A} (A \Rightarrow B), A \xrightarrow{\text{eval}} B \longleftarrow\!\shortmid \Gamma \xrightarrow{t} (A \Rightarrow B)$$

[Jacobs, Hyland, ...]

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

follows naturally from categorical definition

- for every $A, B \in \mathbb{C}$ an object $A \Rightarrow B$
- a multimap $A \Rightarrow B, A \xrightarrow{eval} B$ inducing

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

$$\Gamma, A \xrightarrow{t, A} (A \Rightarrow B), A \xrightarrow{eval} B \longleftarrow\!\shortmid \Gamma \xrightarrow{t} (A \Rightarrow B)$$

$\rightsquigarrow$ free closed clone =
- objects = $\Lambda^{\to}$-types
- multimaps = $\Lambda^{\to}$-terms

[Jacobs, Hyland, ...]

def

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

follows naturally from categorical definition

- for every $A, B \in \mathbb{C}$ an object $A \Rightarrow B$
- a multimap $A \Rightarrow B, A \xrightarrow{eval} B$ inducing

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

$$\Gamma, A \xrightarrow{t, A} (A \Rightarrow B), A \xrightarrow{eval} B \longleftarrow\!\shortmid \Gamma \xrightarrow{t} (A \Rightarrow B)$$

$\rightsquigarrow \underbrace{\text{free closed clone}}_{\text{sound + complete for } \Lambda^{\rightarrow}}$ =

- objects = $\Lambda^{\rightarrow}$-types
- multimaps = $\Lambda^{\rightarrow}$-terms

We have a good multi-ary model,

but is there a good categorical one?

ie. a category with operations on it
that's sound and complete for $\lambda^{\to}$

Modelling $\Lambda^{\to}$ in categories

$$\begin{tikzcd}
 & \text{closed clones} \arrow[dl, bend left] \arrow[dr, bend left] & \\
\text{signatures} \arrow[ur, bend left] \arrow[ur, phantom, "\perp"] \arrow[rr, bend left] \arrow[rr, phantom, "\perp"] & & \boxed{??} \arrow[ul, bend left] \arrow[ul, phantom, "\perp"] \arrow[ll, bend left]
\end{tikzcd}$$

Proof goes via correspondence

between $\Lambda^{\rightarrow}$ and combinatory logic (typed, extensional)

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

$K\,x\,y = x$

$S\,x\,y\,z = (x\,z)(y\,z)$

Main ideas : adapt "closed categories"

cf. Eilenberg-Kelly,
Day-Laplaza,
Uustalu-Veltri-Zeilberger

cf. Eilenberg-Kelly, Day-Laplaza, Uustalu-Veltri-Zeilberger

Main ideas : adapt "closed categories"

① introduce SK-clones and extensional SK-clones

— sound + complete for (extensional) SK-combinatory logic

cf. Eilenberg-Kelly,
Day-Laplaza
Uustalu-Veltri-Zeilberger

Main ideas : adapt "closed categories"

① introduce SK-clones and extensional SK-clones

— sound + complete for (extensional) SK-combinatory logic

② adapt classical correspondence to show (extensional SK-clones) $\simeq$ (closed clones)

cf. Eilenberg-Kelly,
Day-Laplaza
Uustalu-Veltri-Zeilberger

Main ideas : adapt "closed categories"

① introduce SK-clones and extensional SK-clones

— sound + complete for (extensional) SK-combinatory logic

② adapt classical correspondence to show (extensional SK-clones) ≃ (closed clones)

③ Introduce SK-categories and show (SK-categories) ≃ (extensional SK-clones)

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\text{closed clones} \arrow[r, phantom, "\simeq"] \arrow[d, bend left] \arrow[d, phantom, "\dashv"] & \text{extensional SK-clones} \arrow[d, bend left] \arrow[d, phantom, "\simeq"] \\
\text{signatures} \arrow[u, bend left] \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \text{SK-categories} \arrow[l, bend left] \arrow[u, bend left]
\end{tikzcd}
$$

def

an SK-category is a category $\mathbb{C}$ with

1. an "exponential" $[-,=] : \mathbb{C}^{op} \times \mathbb{C} \longrightarrow \mathbb{C}$
2. a "closed terms" functor $U : \mathbb{C} \to \underline{Set}$
3. "application" maps $U[C,D] \times UC \longrightarrow UD$
4. "S" maps $S_{C,D,E} : [C,[D,E]] \to [[C,D],[C,E]]$
5. "K" maps $K_{C,D} : D \longrightarrow [C,D]$

... subject to axioms. //

cf: Eilenberg-Kelly, Day-LaPlaza "closed categories"
Uustalu, Veltri, Zeilberger "prounital closed categories"

Where next?

Where next?

⊛ richer notions of multi-ary model

⇝ canonical syntax + semantics for refined languages

eg// grading, effects, fuzziness, ...

⊛ versions of closed categories ⇄ ? clones / multi-ary models ( ) combinatory logics

Summing up + still lots to do!

1. multi-ary models significantly streamline some semantic arguments
2. SK-categories are sound and complete for $\Lambda^{\to}$
3. (extensional) SK-clones are sound and complete for (extensional) CL