§0: Organisation

This talk: about work I did during (and a bit after) my PhD with Marcelo Fiore in Cambridge.

About: using ideas from categorical logic / semantics of PL to prove coherence theorems that make doing things in Esp (and other places) much easier.

I will **aim** to get across the big picture + main ideas, plus some sense of how you can use them.
Also aim to be accessible to this quite broad audience.
As a result: will be quite high level, spend quite a lot of time on background and motivation. People who know these aspects, please bear with me!

This work is presented more fully in the LICS paper in the abstract, and in painful levels of detail in my thesis.

§1: Motivation / Big Idea

Starting point: bicategory Esp of generalised species.

obj: categories $\mathbb{C}, \mathbb{D}, \ldots$

1-cells $\mathbb{C} \xrightarrow{P} \mathbb{D}$: generalised species, ie functors $!\mathbb{C} \longrightarrow \widehat{\mathbb{D}}$

where $!\mathbb{C}$ = free monoidal cat. on $\mathbb{C}$, equivalently $!\mathbb{C} \times \mathbb{D}^{op} \longrightarrow$ Set.

eg) 1-cells $\mathbb{1} \longrightarrow \mathbb{1}$
= functors $\mathbb{B}ij \times \mathbb{1} \longrightarrow$ Set
$\cong$ functors $\mathbb{B}ij \longrightarrow$ Set
= Joyal species.

2-cells $\tau : P \Rightarrow Q$: natural transformations

horizontal composition: $\mathbb{C} \xrightarrow{P} \mathbb{D} \xrightarrow{Q} \mathbb{E}$, with $Q \cdot P$: $\mathbb{C} \to \mathbb{D} \to \mathbb{E}$

$Q \cdot P : !\mathbb{C} \longrightarrow \widehat{\mathbb{E}}$

$$(Q \cdot P)(u)_e = \int^{v \in !\mathbb{D}} Q(v)_e \times P^{|v|}(u)$$

▷ defined as another coend!

So: $Esp(\mathbb{C}, \mathbb{D}) = Cat(!\mathbb{C}, \widehat{\mathbb{D}})$
$\cong Cat(!\mathbb{C} \times \mathbb{D}^{op}, \text{Set})$

We want to calculate things inside Esp, but constantly working with the coend for horizontal composition is quite a lot of work — on top of the usual tedium of calculating in a bicategory.

So we want: tools to make such calculations as easy as possible, using the fact Esp has a kind of cartesian closed structure.

What kind of thing are we looking for?

A simple CT exercise in CCCs:

Lemma 1: every $(X \Rightarrow X)$ has a canonical monoid structure.

Pf:
Construct the maps + prove some eq^ns hold. //

The corresponding statement in the bicategorical setting, e.g. for Esp, would be

Lemma 2?: every $(X \Rightarrow X)$ has a canonical pseudomonoid structure.
i.e. we have a monoid structure but the unit + assoc. laws are replaced by invertible 2-cells AND these are subject to coherence axioms.

Pf 2?:
Do essentially the 1-dim. proof to construct the invertible 2-cells, THEN do lots of calculating to prove the coherence axioms - extremely tedious ∵ all defined by the UP of the function space, which is only an equivalence... //

This talk: reducing the lemma 2? to lemma 1.
Hinges on two observations.

First is a categorification of ideas from semantics of LL.
Second is classical categorical logic.

OBSERVATION 1: (FGHW) Esp is not just any Licategory: it has (bicategorical) cartesian closed structure.

Reminder:

= CCC

def: a cartesian closed category is a category $\mathbb{C}$ equipped with objects

$$\prod_{i=1}^{n} A_i \quad (n \in \mathbb{N}) \quad , \quad A \Rightarrow B$$

for every $A_i, A, B \in \mathbb{C}$, together with isos

$$\mathbb{C}(\Gamma, \prod_{i=1}^{n} A_i) \cong \prod_{i=1}^{n} \mathbb{C}(\Gamma, A_i) \quad , \quad \mathbb{C}(\Gamma \times A, B) \cong \mathbb{C}(\Gamma, A \Rightarrow B)$$

natural in $\Gamma \in \mathbb{C}$. //

NOTE: usually defined in terms of binary + nullary product, but doing finite products directly avoids the case distinction, so simplifies exposition a bit.

eg//

Set, any presheaf category, any Heyting algebra, Cat,
- any coKleisli $\mathbb{C}_!$ for a suitable comonad $!$ on an smcc $\mathbb{C}$ —
- e.g. take $!$ = free commutative comonoid on Rel.

//

SUGGESTS

Esp has the bicategorical version of this structure. For categories $\mathbb{C}_i, \mathbb{C}, \mathbb{D}$, set

$$\bigotimes_{i=1}^{n} \mathbb{C}_i := \sum_{i=1}^{n} \mathbb{C}_i \quad , \quad \mathbb{C} \Rightarrow \mathbb{D} := (!\mathbb{C})^{op} \times \mathbb{D}$$

Then we get

$$\begin{aligned} Esp(\mathbb{B}, \otimes_i \mathbb{C}_i) &\equiv Esp(\mathbb{B}, \Sigma_i \mathbb{C}_i) \\ &\equiv Cat(!\mathbb{B}, \widehat{\Sigma_i \mathbb{C}_i}) \\ &\cong Cat(!\mathbb{B}, \Pi_i \widehat{\mathbb{C}_i}) \\ &\cong \Pi_{i=1}^n Cat(!\mathbb{B}, \widehat{\mathbb{C}_i}) \\ &= \Pi_{i=1}^n Esp(\mathbb{B}, \mathbb{C}_i) \end{aligned}$$

$$\begin{aligned} Esp(\mathbb{B}, \mathbb{C} \Rightarrow \mathbb{D}) &\equiv Esp(\mathbb{B}, (!\mathbb{C})^* \times \mathbb{D}) \\ &= Cat(!\mathbb{B}, \widehat{(!\mathbb{C})^* \times \mathbb{D}}) \\ &\cong Cat(!\mathbb{B} \times !\mathbb{C}, \mathbb{D}) \\ &\simeq Cat(!(\mathbb{B} + \mathbb{C}), \mathbb{D}) \\ &= Esp(\mathbb{B} \otimes \mathbb{C}, \mathbb{D}) \end{aligned}$$

comes from how ! interacts with presheaves. ONLY AN EQUIVALENCE

This leads to the following:

= cc-bicategory

def: a cartesian closed bicategory is a bicategory $\mathcal{C}$ equipped with objects

$$\Pi_{i=1}^n A_i \quad (n \in \mathbb{N}), \qquad A \Rightarrow B$$

for every $A_i, A, B \in \mathcal{C}$, together with equivalences

$$\mathcal{C}(\Gamma, \Pi_{i=1}^n A_i) \simeq \Pi_{i=1}^n \mathcal{C}(\Gamma, A_i), \qquad \mathcal{C}(\Gamma \times A, B) \simeq \mathcal{C}(\Gamma, A \Rightarrow B)$$

pseudo natural in $\Gamma \in \mathcal{C}$. //

eg// Esp, any 2-category $Hom(\mathbb{B}^{op}, Cat)$, any cc-2-category, bicategories of games, coKleisli bicategories arising from Sweedler + ! [Paquet] //

This definition is quite difficult to work with, because pseudonaturality is quite painful.

It turns out to be easier if you use a bicategorical version of the universal arrow formulation. Also we can WLOG ask for adjoint equivalences. (Always ask for as much structure as possible!)

This def: is equivalent to the one above:

= cc-bicategory

def: a cartesian closed bicategory is a bicategory $\mathcal{C}$ equipped with objects

$$\prod_{i=1}^{n} A_i \quad (n \in \mathbb{N}) \qquad , \qquad A \Rightarrow B$$

and 1-cells for every $A_i, A, B \in \mathcal{C}$,

$$\pi_i : \prod_{i=1}^{n} A_i \longrightarrow A_i \qquad , \qquad \mathrm{eval} : (A \Rightarrow B) \times A \longrightarrow B$$

which are <u>bipniversal</u> in the sense of T. Fiore: for each Γ we have chosen functors

$$(\star) \quad \mathcal{C}(\Gamma, \prod_{i=1}^{n} A_i) \underset{\mathrm{tup}(-)}{\overset{\langle \pi_1 \circ -, \ldots, \pi_n \circ - \rangle}{\simeq}} \prod_{i=1}^{n} \mathcal{C}(\Gamma, A_i) \qquad \mathcal{C}(\Gamma, A \Rightarrow B) \underset{\Lambda(-)}{\overset{\mathrm{eval} \circ (- \times A)}{\simeq}} \mathcal{C}(\Gamma \times A, B)$$

and invertible 2-cells

$$\varpi_i : \pi_i \circ (\mathrm{tup}(f_1, \ldots, f_n)) \overset{\cong}{\Longrightarrow} f_i \qquad , \qquad \varepsilon : \mathrm{eval} \circ (\Lambda f \times A) \overset{\cong}{\Longrightarrow} f$$

$(i = 1, \ldots, n)$

which are the counits of adjoint equivalences in $(\star)$. //

See: Fiore-Gambino-Hyland-Winskel "The cartesian closed bicat. of generalised species of structures" for the details.

So we've seen:

> Esp is a cartesian closed bicategory, in the sense you get by weakening the 1-dim. def: from isos to (adjoint) equivalences. It's a "2-dimensional ccc".

OBSERVATION 2: [Lambek, ...] cccs have a sound + complete "internal language" which is moreover normalising: there are canonical representatives for the equivalence classes of the equational theory.

NOTE: Federico also mentioned these ideas. Classic in the categorical logic / semantics-of-PL literature.

Might be lots of unfamiliar terms here! Let's break it down.

Idea: a ccc describes the behaviour of function spaces and products in categories like Set:

$$\frac{\Gamma \longrightarrow \prod_{i=1}^{n} A_i}{(\Gamma \longrightarrow A_i)_{i=1,\dots,n}} \qquad h \;\updownarrow\; (\pi_i \circ h)_{i=1,\dots,n} \qquad \gamma \longmapsto (f_1\gamma, \dots, f_n\gamma) \;\updownarrow\; (f_1, \dots, f_n)$$

$$\frac{\Gamma \longrightarrow (A \Rightarrow B)}{\Gamma \times A \longrightarrow B} \qquad g \;\updownarrow\; (\gamma, a) \longmapsto g(\gamma)(a) \qquad \gamma \longmapsto (a \mapsto f(\gamma, a)) \;\updownarrow\; f$$

= STLC

The simply-typed λ-calculus is a formal language for functions that axiomatises the same behaviours.

It's used in PL as the basis for studying functional PLs; in logic it's viewed as a term language for intuitionistic logic. [Curry-Howard-Lambek correspondence].

Idea: we write <u>judgements</u> like

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

to mean M is a program / function that, when you feed it arguments a_1 in A_1, a_n in A_n ... then it will give you back something in B.

We usually denote this <u>meaning</u> using double brackets:

$$[\![x_1 : A_1, \dots, x_n : A_n \vdash M : B]\!] = [\![A_1]\!] \times \cdots \times [\![A_n]\!] \xrightarrow{[\![M]\!]} [\![B]\!]$$

We only have 5 rules.

① I can always give back one of the things you give me:

$$[\![\frac{\text{NO ASSUMPTIONS AT ALL}}{x_1 : A_1, \_, x_n : A_n \vdash x_i : A_i} (i=1,\dots,n)]\!] = \prod_{i=1}^n [\![A_i]\!] \xrightarrow{\pi_i} [\![A_i]\!]$$
$$(a_1, \dots, a_n) \longmapsto a_i$$

② If you give me n functions, I can pair them up:

new writing Γ for $(x_1 : A_1, \dots, x_n : A_n)$

$$[\![\frac{(\Gamma \vdash M_i : A_i)_{i=1,\dots,n}}{\Gamma \vdash \langle M_1, \dots, M_n \rangle : \prod_{i=1}^n A_i}]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![M_1]\!], \dots, [\![M_n]\!] \rangle} \prod_{i=1}^n [\![A_i]\!]$$
$$(a_1, \dots, a_n) \longmapsto ([\![M_1]\!](\vec{a}), \dots, [\![M_n]\!](\vec{a}))$$

③ If you give me a function into a product I can extract a component:

$$[\![\frac{\Gamma \vdash M : \prod_{i=1}^n A_i}{\Gamma \vdash \pi_k(M) : A_k} (k=1,\dots,n)]\!] = [\![\Gamma]\!] \xrightarrow{[\![M]\!]} \prod_{i=1}^n [\![A_i]\!] \xrightarrow{\pi_k} [\![A_k]\!]$$
$$(a_1, \dots, a_n) \longmapsto \text{tuple of } n \text{ things} \longmapsto \text{extract } k\text{th component}$$

④ If you give me a function with a spare argument, I can turn that into a map into the function space: (cf. actions)

$$\left[\!\left[\frac{\Gamma, y:B \vdash M : C}{\Gamma \vdash \lambda y.\, M : B \to C} \right]\!\right] = \begin{array}{ccc} [\![\Gamma]\!] & \longrightarrow & ([\![B]\!] \Rightarrow [\![C]\!]) \\ \gamma & \longmapsto & (b \mapsto [\![M]\!](\gamma, b)) \end{array}$$

Might look odd but it's very familiar: if we have $x+1$ we can turn that into a function by writing

$$x \longmapsto x+1 \qquad \equiv \qquad \lambda x.\, x+1$$

⑤ I can pass arguments to functions:

$$\left[\!\left[\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \right]\!\right] = [\![\Gamma]\!] \xrightarrow{\langle [\![M]\!], [\![N]\!] \rangle} ([\![A]\!] \Rightarrow [\![B]\!]) \times [\![A]\!] \xrightarrow{\text{eval}} [\![B]\!]$$

$$\gamma \longmapsto ([\![M]\!](\gamma), [\![N]\!](\gamma)) \mapsto ([\![M]\!](\gamma))([\![N]\!](\gamma))$$

evaluate the fn $[\![M]\!]\gamma$ at the argument $[\![N]\!]\gamma$

To get the bijections for cc-structure we need some eqns:

① if I tuple up some maps then project out, I get the component:

$$\pi_i(\langle M_1, \ldots, M_n \rangle) =_\beta M_i \qquad (i = 1, \ldots, n)$$

② if I pair up a bunch of projections out of a function, I do nothing:

$$\langle \pi_1(M), \ldots, \pi_n(M) \rangle =_\eta M$$

↑ look at each component, then pair them all up

③ to apply a function to an argument, we replace all the occurrences of the variable by the argument:

$$(x \mapsto x+1)(3) = 3+1 = 4$$

in terms of STLC:

find each occurrence of x, replace it by N

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

④ extensionality: the fn f is the same as $x \mapsto f(x)$. So we get:

$$\lambda x.\, (M\, x) =_\eta M. \quad //$$

With a little algebra you can prove you get bijections corresponding to the ones above:

CCC		STLC
$\dfrac{\Gamma \longrightarrow \prod_{i=1}^n A_i}{(\Gamma \longrightarrow A_i)_{i=1,\dots,n}}$	$\leftrightsquigarrow$	$\dfrac{\Gamma \vdash M : \prod_{i=1}^n A_i}{(\Gamma \vdash N_i : A_i)_{i=1,\dots,n}}$
$\dfrac{\Gamma \longrightarrow (A \Rightarrow B)}{\Gamma \times A \longrightarrow B}$	$\leftrightsquigarrow$	$\dfrac{\Gamma \vdash M : A \longrightarrow B}{\Gamma, y:B \vdash N : B}$

All of this to say:

We commonly ∴ say STLC is an INTERNAL LANGUAGE for CCCs.

① STLC is sound and complete for reasoning in CCCs: an eqn holds in the free CCC on some data iff there's a corresponding eqn in STLC.

You can phrase this as follows:

Thm: for a certain category of "signatures" there is an adjunction between Sig and CCCat, and the free CCC F[S] on a signature S is given by the syntax of STLC:

= basic data

obj: types

= equivalently, terms $\vdash M : A \to B$ in an empty context

maps $A \to B$: terms $x:A \vdash M:B$ in contexts of length 1, modulo $\beta\eta$.

identity: $x:A \vdash x:A$

composition: substitution //

→ since this is associative and unital via $x[t/x] = t$, $t[x/x] = t$

Completeness follows ∴ if an eqn holds in every model, it holds in the free model, so the syntax.

The free property says that if you choose how to interpret the signature, you get a map $F[S] \xrightarrow{[\![-]\!]} \mathbb{C}$ which respects CC-structure. Hence, if $M =_{\beta\eta} N$ then $[\![M]\!] = [\![N]\!]$.

Why is this useful for us? Look at an example.

Lemma 1: if $\mathcal{C}$ any ccc, $(X \Rightarrow X)$ has canonical monoid structure.

Pf: can do a long diagram chase by hand... or use STLC!
What is this monoid structure, intuitively? Composition of fns.
So define $id := \lambda y.\, y$ and $g \circ f := \lambda x.\, g(f\, x)$. Then showing monoid structure is just like set, e.g.:

$$\begin{aligned} g \circ id &= \lambda x.\, g((\lambda y.\, y)\, x) \\ &\overset{\beta}{=} \lambda x.\, (g\, x) \\ &\overset{\eta}{=} g \end{aligned}$$

MUCH SHORTER

and likewise for the other laws. //

So now we know:

> ① STLC is sound and complete for reasoning in cccs: an eqn holds in the free ccc on some data iff there's a corresponding eqn in STLC.
>
> ② ... And this makes some calculations easier.

This is kind of good, but we can do even better.

As a formal language, STLC has very nice properties.
Suppose we direct the rules, and then think of these rewrites as a program running:

$$\pi_i(\langle M_1, \ldots, M_n \rangle) \overset{\beta}{\Longrightarrow} M_i \qquad M \overset{\eta}{\Longrightarrow} \langle \pi_1(M), \ldots, \pi_n(M) \rangle$$

$$(\lambda x.\, M)\, N \overset{\beta}{\Longrightarrow} M[N/x] \qquad M \overset{\eta}{\Longrightarrow} \lambda x.\, (M\, x)$$

Nb: these directions might remind you of counit and unit.

Then we can prove, if we set things up right, that every program will run – do some work – until it terminates. We call this program the **normal form**.

This picks a canonical representative of each $\beta\eta$-equivalence class. And it gives an alg: for checking equality!

$$P \underset{\text{reduce to nf}}{\Longrightarrow} nf(P) \underset{\text{uniqueness of nfs}}{=\!=} nf(Q) \underset{\text{reduce to nf}}{\Longleftarrow} Q$$

We even know, from **standardisation**, that the rewrites can also be put in a kind of normal form.

So, in total, our second observation is:

> ① STLC is sound and complete for reasoning in CCCs: an eqn holds in the free CCC on some data iff there's a corresponding eqn in STLC.
>
> ② ... And this makes some calculations easier.
>
> ③ But even better! We can find "normal forms" / canonical representatives for both terms and rewrites.

PUTTING TOGETHER OUR TWO OBSERVATIONS:

We have:

OBSERVATION 1

> Esp is a cartesian closed bicategory, in the sense you get by weakening the 1-dim. defn from iscs to (adjoint) equivalences. It's a "2-dimensional ccc".

OBSERVATION 2

> ① STLC is sound and complete for reasoning in cccs: an eqn holds in the free ccc on some data iff there's a corresponding eqn in STLC.
>
> ② ... And this makes some calculations easier.
>
> ③ But even better! We can find "normal forms" / canonical representatives for both terms and rewrites.

This all suggests the following IDEA:

① There should be a 2-dimensional version of STLC that's sound and complete for cc-bicats.

As well as judgements $\Gamma \vdash M : A$ for terms / 1-cells, we should have judgements $\Gamma \vdash \tau : M \Rightarrow N : A$ for 2-cells — we can naturally think of these as rewrites.

② We should be able to adapt proofs of normalisation to show coherence: there is at most one (structural) 2-cell $\sigma : f \Rightarrow g$ in the free bi-ccc, since there's at most one rewrite $\sigma : M \Rightarrow g$ between any two terms.

All this works out! We then get:

- ✱ coherence: all diagrams commute
- ✱ reduction of difficulty of calculating in free cc-bicat to calculating in free ccc / STLC. Since every rewrite in our 2-dim. lang. corresponds to a βη-equality and is unique, we can:
  - ① define the 1-dim structure using eqns in STLC — corresponding
    then each βη-equality has a corresponding rewrite, so we get the req'd isos.
  - ② any eqns you need on the isos hold by coherence.

In particular we get, as promised:

Lemma 2: in any cc-bicat, $(X \Rightarrow X)$ has a canonical pseudomonoid structure.

Pf: by the proof for cccs and the two points above. //

So our STRATEGY is:

1. define an STLC-like language that's sound + complete for cc-bicats, with types, terms, and rewrites
2. adapt a semantic argument for normalisation of STLC to show ∃! rewrite between any two terms

In the rest of the talk I'll briefly sketch how you do ① + ②. For more details see my thesis & papers cited in abstract.

NOTES: ① will echo some of what Federico said — I think it's fair to say we agree on quite a lot of the aspects here

② I'll be talking about a semantic approach; Federico has some nice work using syntactic ideas from standardisation to prove coherence by putting rewrites in normal forms.

---

§2: Designing an internal language for cc-bicats.

We want a formal language that:

① is sound + complete for cc-bicats — and for which the proof is tractable!

important! These objects are subtle and a lot can go wrong with definitions

② looks like STLC, so we can adapt arguments from STLC to this lang.

Not obvious how to build this! But if we do something ad hoc that could make our proofs very hard.

Solution: let the algebra do the work for us.

We do this by using similar ideas to what Federico talked about. (Except his language was for monoidal structure and mine is for cartesian.)

We:
- ⓐ write down an algebraic description of syntax of STLC
- ⓑ bicategorify the def'n ...
- ⓒ take the free such thing as our def'n of the syntax. Then points ① + ② above come almost for free!

ⓐ An algebraic description of STLC syntax

(Jacobs, Hyland, ...)

Can't be a category: syntax has $\Gamma \vdash M : B$ where Γ has any finite length. But in a category $A \longrightarrow B$ we have inputs of length 1.

Instead must have multimaps $A_1 \dots A_n \longrightarrow B$, with any finite number of inputs.

I'll give the def^n and the corresponding syntax — should be very clear the free model is the syntax of (fragments of) STLC.

We start with the very basic part of the language:

def: a (multisorted, abstract) clone $\mathbb{C}$ consists of:

- objects $A, B, \dots$
- multimaps $A_1 \dots A_n \xrightarrow{M} B$ $\sim$ $x_1 : A_1, \dots, x_n : A_n \vdash M : B$
  - we write $\mathbb{C}(A_1 \dots A_n ; B)$ for the hom-set
- instead of identities, projections

$$(p_i : A_1 \dots A_n \longrightarrow A_i)_{i=1\dots n} \sim (x_1 : A_1 \dots x_n : A_n \vdash x_i : A_i)_i$$

- instead of composition, substitution

$$\begin{array}{ccc} \mathbb{C}(A_1 \dots A_n ; B) \times \prod_{i=1}^n \mathbb{C}(\Delta ; A_i) & & \\ \downarrow \qquad\qquad (M, (V_1 \dots V_n)) & \sim & \dfrac{x_1 : A_1 \dots x_n : A_n \vdash M : B \qquad (\Delta \vdash V_i : A_i)_{i=1\dots n}}{\Delta \vdash M[V_1/x_1 \dots V_n/x_n] : B} \\ \mathbb{C}(\Delta ; B) \qquad M[V_1 \dots V_n] & & \end{array}$$

subject to two unit laws + an assoc. law.

eg//

$$p_i[a_1 \dots a_n] = a_i \sim x_i[V_1/x_1 \dots V_n/x_n] = V_i.$$

NB: a clone is equivalently a cartesian multicategory. //

Also observe: if you restrict to multimaps $A \longrightarrow B$ you get a category. In fact there's an adjunction

$$\text{free} \dashv \text{restrict to unary maps} : \mathbf{Clone} \to \mathbf{Cat}$$

//

Now we need to add in products and function types. Recall that we had bijections

$$\frac{\Gamma \vdash M : \prod_{i=1}^n A_i}{(\Gamma \vdash N_i : A_i)_{i=1,\dots,n}} \qquad \frac{\Gamma \vdash M : A \longrightarrow B}{\Gamma, y:B \vdash N : B}$$

We just ask for these!

def: a cc-clone is a clone $\mathbb{C}$ equipped with:

- for $A_1, \dots, A_n \in \mathbb{C}$, an object $\prod_{i=1}^n A_i$ and maps $(\pi_i : \prod_{i=1}^n A_i \longrightarrow A_i)_{i=1,\dots,n}$.

  (why I didn't use π_i for projections as variable maps)

$$\sim \quad \frac{}{p : \prod_{i=1}^n A_i \vdash \pi_i(p) : A_i} \; i=1,\dots,n$$

- for $B, C \in \mathbb{C}$ an object $B \Rightarrow C \in \mathbb{C}$ and a map $eval : (B \Rightarrow C), B \longrightarrow C$.

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

- chosen isomorphisms making (π_1, π_2) and $eval$ universal arrows:

$$\mathbb{C}(\Gamma ; \pi_{i=1}^{n} A_i) \underset{\text{tupling}}{\overset{(\pi_1 -, \ldots, \pi_n -)}{\cong}} \pi_{i=1}^{n} \mathbb{C}(\Gamma ; A_i)$$

$$f \longmapsto (\Gamma . A \xrightarrow{f . A} (A \Rightarrow B), A \xrightarrow{eval} B)$$

$$\mathbb{C}(\Gamma ; A \Rightarrow B) \underset{\text{currying}}{\overset{\Lambda(-)}{\cong}} \mathbb{C}(\Gamma, A ; B)$$

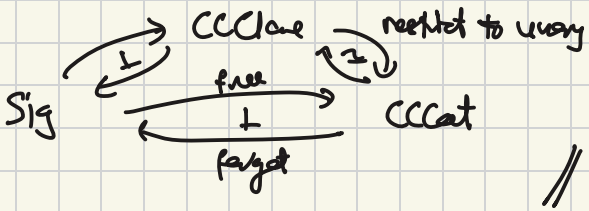
It's then quite easy to show:

Prop:

① the free cc-clone on a signature of base types and constants is given by exactly the syntax of STLC, with multimaps $A_1, \ldots A_n \longrightarrow B$ the terms $x_1 : A_1, \ldots, x_n : A_n \vdash M : B$.

② restricting a cc-clone to the unary maps gives a ccc.

③ the adjunction between Sig and CCCat factors through CC clone as shown:



|| here the products matter!

Hence: the reason STLC is the internal language of CCCs is that it's (clearly) the internal language for cc-clones, and the two are equivalent.

(b) Bicategorifying the def^n

We now write down the obvious bicategorical counterpart.

def: a (multisorted, abstract) **biclone** $\mathbb{C}$ consists of:

- objects $A, B, \dots$
- multimaps $A_1, \dots, A_n \xrightarrow{m} B$ + rewrites $A_1, \dots, A_n \Downarrow \tau : m \to m' : B$

with a vertical composition operation and identities:

$$A_1, \dots, A_n \Downarrow \tau : m \to m' : B \quad \rightsquigarrow \quad A_1, \dots, A_n \Downarrow \tau' \circ \tau : m \to m'' : B$$

$$A_1, \dots, A_n \Downarrow id : m \to m : B .$$

We write $\mathbb{C}(A_1, \dots, A_n; B)$ for the hom-category.

- instead of identities, projections

$$(p_i : A_1, \dots, A_n \longrightarrow A_i)_{i=1,\dots,n}$$

- instead of composition, substitution

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

$$(m, (v_1, \dots, v_n)) \longmapsto M[v_1, \dots, v_n]$$

- invertible rewrites witnessing assoc. and unit laws of substitution, subject to Δ and $\pentagon$ laws of a bicat. //

Note: restricting to unary maps now yields a bicategory.

It's easy to write down a free model — you just follow the rules. In a clone substitution was substitution of terms. For a biclone we have an explicit substitution operation:

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash M : B \qquad (\Delta \vdash V_i \cdot A_i)_{i=1..n}}{\Delta \vdash M\{x_1 \mapsto V_1, \ldots, x_n \mapsto V_n\} : B}$$

Then we have rewrites like

$$\lambda : x_i\{x_1 \mapsto V_1, \ldots, x_n \mapsto V_n\} \Rightarrow V_i$$

$$\varrho : M\{x_1 \mapsto x_1, \ldots, x_n \mapsto x_n\} \Rightarrow M$$

This gives us our core language. For adding cc-structure we again just bicategorify:

def: a **cc-biclone** is a biclone $\mathcal{C}$ equipped with:

- for $A_1, \ldots, A_n \in \mathcal{C}$, an object $\prod_{i=1}^n A_i$ and maps $(\pi_i : \prod_{i=1}^n A_i \longrightarrow A_i)_{i=1,\ldots,n}$.

- for $B, C \in \mathcal{C}$ an object $B \Rightarrow C \in \mathcal{C}$ and a map $\mathrm{eval} : (B \Rightarrow C), B \longrightarrow C$.

- chosen adjoint equivalences making $(\pi_1, \ldots, \pi_n)$ and eval biuniversal arrows:

$$\mathcal{C}(\Gamma; \textstyle\prod_{i=1}^n A_i) \underset{\text{tupling}}{\overset{(\pi_1 \circ -, \ldots, \pi_n \circ -)}{\rightleftarrows}} \textstyle\prod_{i=1}^n \mathcal{C}(\Gamma; A_i) \qquad (\simeq \perp)$$

$$f \longmapsto (\Gamma, A \xrightarrow{f, A} (A \Rightarrow B), A \xrightarrow{\mathrm{eval}} B)$$

$$\mathcal{C}(\Gamma; A \Rightarrow B) \underset{\Lambda(-) \text{ currying}}{\rightleftarrows} \mathcal{C}(\Gamma, A; B) \qquad (\simeq \perp)$$

Because the UP is written down with universal arrows it's very easy to write down a corresponding syntax, which I called $\Lambda^{\times,\to}_{ps}$. For example, for products you get

$$\frac{(\Gamma \vdash M_i : A_i)_{i=1,\ldots,n}}{\Gamma \vdash \varpi_i : \pi_i\{\langle M_1, \ldots, M_n \rangle\} \Rightarrow M_i : A_i}$$

+ eqns so that we get

$$\frac{(\Gamma \vdash \alpha_i : \pi_i\{M\} \Rightarrow N_i : A_i)_{i=1,\ldots,n}}{\Gamma \vdash \mathrm{p}^\dagger(\alpha_1, \ldots, \alpha_n) : M \Rightarrow \langle N_1, \ldots, N_n \rangle : \prod_{i=1}^n A_i}$$

$$\frac{\text{rewrites } (\pi_i\{M\} \Rightarrow N : A_i)_{i=1,\ldots,n}}{\text{rewrites } M \Rightarrow \langle N_1, \ldots, N_n \rangle}$$

$\mathrm{p}^\dagger(-, \ldots, =)$ (down); compose with ϖ_i (up)

It's then easy, essentially construction, to show:

Prop: ① there is an adjunction $\mathrm{Sig} \rightleftarrows$ CC-clone, with the free model given by the syntax of $\Lambda_{ps}^{\times,1,\to}$;

② there is an equivalence CC-clone $\overset{\text{restrict to cwos}}{\rightleftarrows}$ cc-bicat.

Hence, $\Lambda_{ps}^{\times,1,\to}$ is sound and complete – an internal language – for cc-bicats. //

Moreover, because we built Λ_{ps} by replacing the isos in STLC by equivalences, it's easy to show:

if $M =_{\beta\eta} M'$ in STLC ... there's an invertible rewrite between corresponding STLC terms

if $\tau : M \overset{\cong}{\Rightarrow} M'$ in $\Lambda_{ps}^{\times,1,\to}$ ... the corresponding STLC terms are $\beta\eta$-equal.

So, summing up this section:

- We can define a language $\Lambda_{ps}^{\times,1,\to}$ that's sound and complete for cc-bicats by writing down the free cc-biclone – a categorified version of the algebraic structure of STLC
- $\Lambda_{ps}^{\times,1,\to}$ really is "STLC up to iso". In particular... every $\beta\eta$-equality in STLC corresponds to an invertible rewrite

Also: since STLC + $\Lambda_{ps}^{\times,1,\to}$ are internal languages, this can be read as a thm relating the free cc-bicat to the free ccc.

Stepping back... remember our strategy:

① We WTS that there's at most one (structural) 2-cell between any 1-cells in a cc-bicat, e.g. Esp. So we care about the free model.

② We now know the free model is given by the syntax of our language $\Lambda^{\times,\to}_{\beta\eta}$. So it STS that there's at most one rewrite $\Gamma \vdash \tau : M \Rightarrow N : A$ in $\Lambda^{\times,\to}_{\beta\eta}$ between any two terms.

③ This is a kind of normalisation result: we WTS there's a unique, canonical representative in each equivalence class of rewrites.

We prove this using an argument called normalisation-by-evaluation. This is originally a syntactic argument, but we use a wholly semantic version, due to Marcelo.

§3: Proving coherence via normalisation-by-evaluation

We give the high-level idea. In STLC, if you direct the β- and η-reductions right, you can show every term has a normal form: a term from which no further reductions are possible.

Normal forms are unique, so we get a procedure for checking if two terms M, N are $\beta\eta$-equal:

① reduce M and N to their normal forms.

$$M \Rightarrow \mathrm{nf}(M) \qquad \mathrm{nf}(N) \Leftarrow N$$

Since all reductions are β or η, $M =_{\beta\eta} \mathrm{nf}(M)$ and $N =_{\beta\eta} \mathrm{nf}(N)$.

② check if $\mathrm{nf}(M) \equiv \mathrm{nf}(N)$, i.e. if they're the same term. If so, we get reductions

$$M \Rightarrow \mathrm{nf}(M) = \mathrm{nf}(N) \Leftarrow N$$

so, since all reductions are β or η, $M =_{\beta\eta} N$.

Note: this says we can get from M to N as

$$M \overset{\text{normalise}}{\Rightarrow} \mathrm{nf}(M) = \mathrm{nf}(N) \overset{\text{normalise}^{-1}}{\Rightarrow} N \quad ?$$

Our argument takes essentially the same approach. We show:

① for any $\Lambda^{\times}_{\beta\eta}$ term M there's an invertible rewrite $v_M : M \Rightarrow nf(M)$.

② we show that if $\tau : M \Rightarrow N$ is any rewrite then the following diagram commutes:

$$\begin{array}{ccc} M & \xrightarrow[\cong]{v_M} & nf(M) \\ {\scriptstyle\tau}\downarrow & & \| \\ N & \xrightarrow[\cong]{v_N} & nf(N) \end{array} \qquad (*)$$

Hence, $\tau = v_N^{-1} \bullet v_M$ so must be unique.

In fact what we prove is more general: we show that for any cc-bicategory we get

$$\begin{array}{ccc} [\![M]\!] & \xrightarrow{[\![v_M]\!]} & [\![nf(M)]\!] \\ {\scriptstyle[\![\tau]\!]}\downarrow & & \| \\ [\![N]\!] & \xrightarrow[[\![v_N]\!]]{} & [\![nf(N)]\!] \end{array} \qquad (**)$$

By then taking this in the syntactic model, we get back (*).

To show (**) we build a cc-bicategory that lets us talk about both the syntax and its semantic interpretation, together.

To do this, we think as follows: for each type A in $\Lambda^{\times,\to}_{ps}$ we have a family of sets

$$T_A(\Gamma) := \{\text{terms } M \text{ s.t. } \Gamma \vdash M : A\}$$

By defining a category $\mathbf{Ctx}$ whose objects are exactly the contexts $\Gamma, \Delta, \ldots$ of $\Lambda^{\times,\to}_{ps}$, we can see this as a presheaf:

$$T_A := \Gamma \longmapsto \{\text{terms } M \text{ s.t. } \Gamma \vdash M : A\}$$

Now suppose we have a cc-bicat $\mathcal{C}$ with an interpretation $[\![-]\!]$ of $\Lambda^{\times,\to}_{ps}$. We then get a semantic interpretation pseudonat. trans with components

$$\begin{array}{ccc} T_A(\Gamma) & & M \\ \downarrow{\scriptstyle [\![-]\!]} & & \downarrow \\ \mathcal{C}([\![\Gamma]\!], [\![A]\!]) & & [\![M]\!] \end{array}$$

So we get $T_A \longrightarrow \mathcal{C}([\![-]\!], [\![A]\!])$ for each type A.

We define a bicategory that describes exactly these objects.

def: let $\mathcal{C}$ be a cc-bicat and $[\![-]\!]$ an interpretation of $\Lambda^{\times,\to}_{ps}$ in $\mathcal{C}$. The bicategory $Gl(\mathcal{C})$ has:

- objects: triples consisting of

$P : Ctx \longrightarrow Cat$ ~ "terms of a fixed type"

$A \in \mathcal{C}$

a pseudonatural trans $P \longrightarrow \mathcal{C}([\![-]\!], A) : Ctx \longrightarrow Cat.$

"semantic interpretation"

$M \in P\Gamma \longmapsto [\![\Gamma \vdash M : A]\!]$

- maps: squares that commute up to iso:

$$\begin{array}{ccc} P & \xrightarrow{\tau} & Q \\ \downarrow & \cong \phi & \downarrow \\ \mathcal{C}([\![-]\!], A) & \xrightarrow[f_-]{} & \mathcal{C}([\![-]\!], B) \end{array}$$

- 2-cells: pairs of 2-cells + a compat condition. //

It's a general fact this is a cc-bicat because it's obtained by gluing: it's a comma-like construction for bicategories:

$$\begin{array}{ccccc} Gl(\mathcal{C}) & \longrightarrow & Hom(Ctx, Cat) \\ \downarrow & \swarrow\!\!\swarrow \lambda & \| \\ \mathcal{C} & \xrightarrow[A \mapsto \mathcal{C}([\![-]\!], A)]{} & Hom(Ctx, Cat) \end{array}$$

cartesian closed

strictly preserves cc-structure

Then we can isolate particular objects. For each type A we have, by the semantic interpretation,

neutrals of type A (includes variables)

$$\mathrm{Neut}_A ::= \mathrm{Neut}_A(-) \xrightarrow[[\![-]\!]]{} \mathcal{C}([\![-]\!], [\![A]\!])$$

> semantic interp in $\mathcal{C}$

normals of type A

$$\mathrm{Norm}_A ::= \mathrm{Norm}_A(-) \xrightarrow[[\![-]\!]]{} \mathcal{C}([\![-]\!], [\![A]\!])$$

natural from a syntax POV!

Semantic interp. in $\mathrm{GI}(\mathcal{C})$

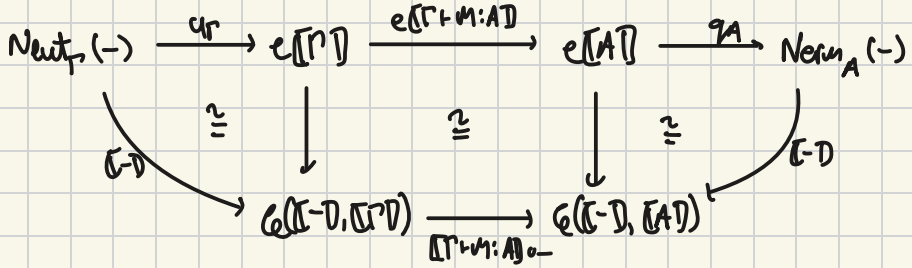
$$\mathrm{INT}_A ::= e[\![A]\!] \longrightarrow \mathcal{C}([\![-]\!], [\![A]\!])$$

We then inductively define a family of maps

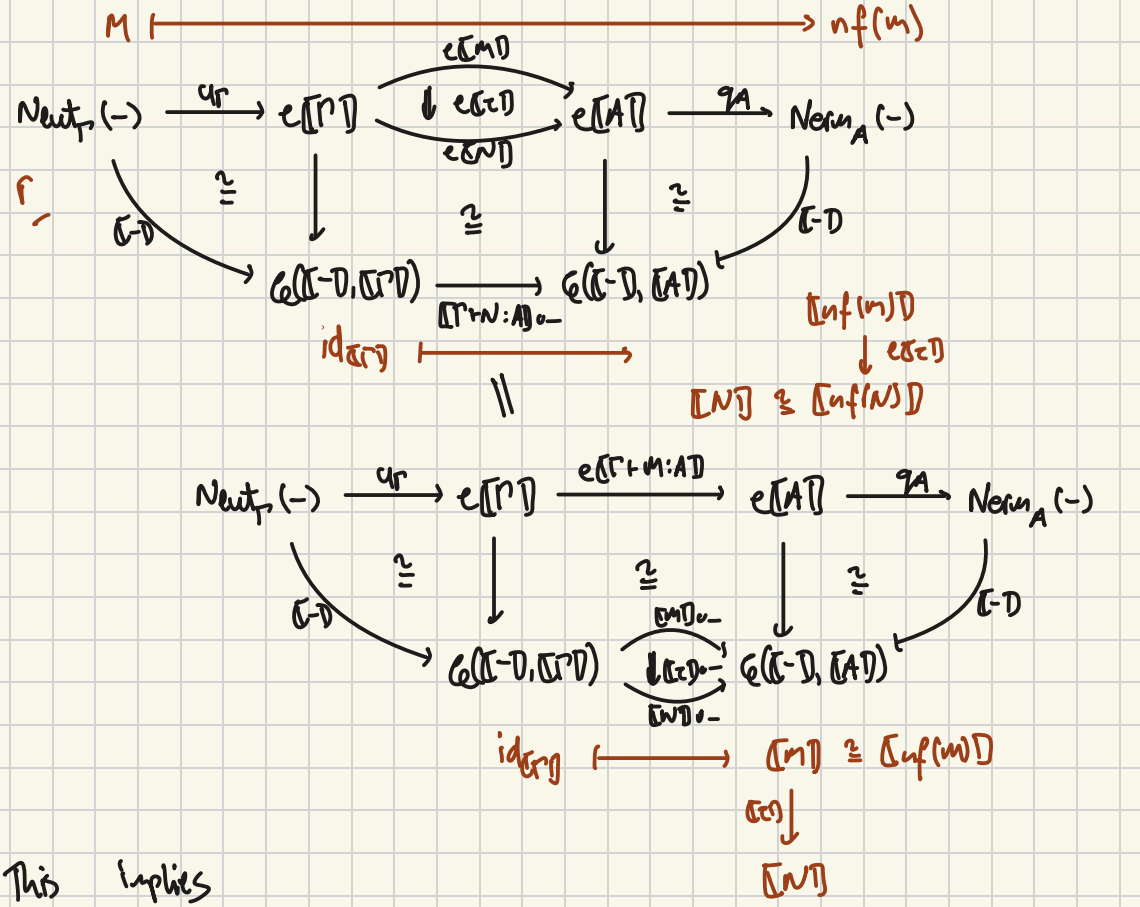
$$\mathrm{NEUT}_A \xrightarrow{u_A} \mathrm{INT}_A \xrightarrow{q_A} \mathrm{Norm}_A$$

} follows the proof for STLC

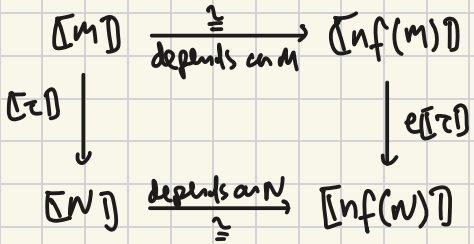
in $\mathrm{GI}(\mathcal{C})$. So for any $\Gamma \vdash M : A$ in $\Lambda^{\times,\to}_{\beta\eta}$ we get



and for $\Gamma \vdash \tau : M \Rightarrow N : A$ we get

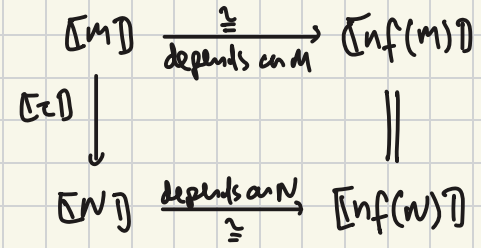


This implies



But: inspecting the setup you see $e[\![\tau]\!] = id$ since every $e[\![A]\!](\Gamma)$ not in Cat but in Set, by construction.

Hence we get



as we wanted.

From this we get the following:

Thm: for any $\Lambda^{\pi,\to}_{ps}$ terms M and N there is at most one rewrite τ s.t. $\tau : M \Rightarrow N$. Hence, in the free cc-bicat. there is at most one 2-cell between any two 1-cells. //

§4: Summary

We've seen:

- Esp is a special case of a cc-bicategory — a ccc "up to iso"
- by writing down the wire algebraic structure, we get an "internal language" for cc-bicats that's sound and complete.
- in this language, there's at most one rewrite (2-cell) between any two terms (1-cells).
- Hence: much easier calculations in a cc-bicat!

⊛ coherence: all diagrams commute

⊛ reduction of difficulty of calculating in free cc-bicat to calculating in free ccc / STLC. Since every rewrite in our 2-dim. lang. corresponds to a βη-equality and is unique, we can:

① define the 1-dim structure using eqns in STLC — then each βη-equality has a corresponding rewrite, so we get the req'd isos.

② any eqns you need on the isos hold by coherence.

In particular we get, as promised:

Lem 2: in any cc-bicat, $(X \Rightarrow X)$ has a canonical pseudomonoid structure.

Pf: by the proof for cccs and the two points above. //