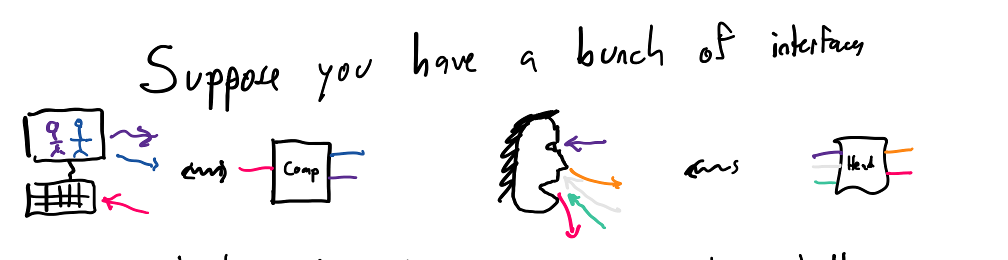
Learners' languages

David Spivak
Topos Institute

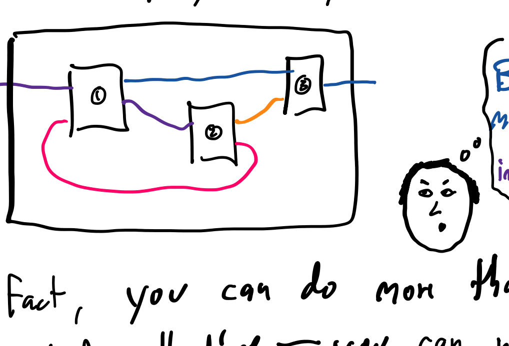
4th International Conference on Applied Category Theory
Cambridge UK, 2021 July 15

① Introduction

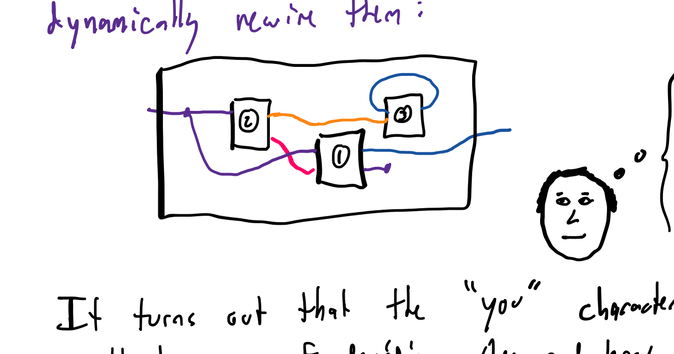
Suppose you have a bunch of interfaces



with outputs and inputs, and you can interconnect them



But in fact, you can do more than interconnect them once and for all time — you can watch what flows along all the input & output wires, and dynamically rewire them.



It turns out that the "you" character above — your particular way of deciding who and how to rewire your machines — can be modeled as an object in a topos. This means that mathematicians have pre-defined a great type theory and logic for describing "your" character traits.

It also turns out that each "neuron" and each "population of neurons" in a deep learning architecture is such a character. Its particular way of rewiring is the adjustment of "weights & biases" via gradient descent & backpropagation.

② Background on Lens and Poly

The ACT community is talking a lot about lenses.

$$Ob(Lens) := \{ \binom{A}{B} \mid A, B \in Set \}$$

$$Lens\left(\binom{A}{B}, \binom{A'}{B'}\right) := Set(A, A') \times Set(A \times B', B)$$

get: $A \to A'$
put: $A \times B' \to B$

They form a full subcategory of my good friend Poly

$$Lens \subseteq Poly \simeq Fun(Set, Set)$$

$$\binom{A}{B} \mapsto Ay^B := (X \mapsto A \times X^B)$$

Lens is the full subcategory of Poly spanned by the monomials.

Poly has four interesting monoidal products $+, \times, \otimes, \triangleleft$, of which Lens inherits three:

$Lens \overset{full}{\subseteq} Poly$
$\times \mapsto \times$
$\otimes \mapsto \otimes$
$\triangleleft \mapsto \triangleleft$
} strong monoidal

So Lens is like "Poly without coproducts". From the context of dynamical systems, it only precludes "interfaces that can change in time", where "allowable input can vary, based on position."

Poly also has two monoidal closures:

- one for $\times$, i.e. Cartesian closure $Poly(p \times q, r) \cong Poly(p, r^q)$
- one for $\otimes$, "the pretty one" $Poly(p \otimes q, r) \cong Poly(p, [q, r])$

$$[p, q] \cong \sum_{\varphi: p \to q} y^{\sum_{i \in p(1)} q[\varphi_1 i]}$$

And lens inherits one, the pretty one.

So if you don't need $+$ or Cartesian closure, you can work in Lens

$$Lens \overset{full}{\subseteq} Poly$$

$\binom{A}{B} \longmapsto Ay^B$
$\times \longmapsto \times$
$\otimes \longmapsto \otimes$
$\triangleleft \longmapsto \triangleleft$
$[\ ,\ ] \longmapsto [\ ,\ ]$

But there's another subtle difference. Polynomials $p \in Poly$ are naturally thought of as functors

$$p: Set \to Set$$

so it's natural to talk about coalgebras

$$S \to p(S), \quad S \in Set$$

which have a nice interpretation as dynamical systems with state set S. For example, if $p = By^A$, a function $S \to B \times S^A$ can be rewritten as a pair of functions

$S \to B$ "output B's"
$A \times S \to S$ "update using input A's"

Objects $\binom{A}{B}$ in Lens aren't typically thought of as functors, so a coalgebra on $\binom{A}{B}$ sounds weird. However, it turns out that there's a bijection

$$Lens\left(\binom{S}{S}, \binom{A}{B}\right) \cong Set(S, B \times S^A) = By^A\text{-coalg}$$

so we can get around the weirdness and see dynamical systems totally within Lens. (Yay!)

But there's a little snag left to deal with. From the Poly / functor point of view, the natural sort of map to consider between interface-p dynamical systems is a map of coalgebras:

$$\begin{array}{ccc} S & \xrightarrow{\varphi} & p(S) \\ f \downarrow & & \downarrow p(f) \\ S' & \xrightarrow[\varphi']{} & p(S') \end{array} \qquad \begin{array}{c} S \\ \downarrow f \\ T \end{array} \in Set/(S, S')$$

But from the Lens point of view, the natural sort of map to consider between interface $\binom{A}{B}$-dynamical systems is a map in the slice category $Lens_{/\binom{A}{B}}\left(\binom{S}{S}, \binom{S'}{S'}\right)$

$$\binom{S}{S} \xrightarrow{\varphi} \binom{A}{B} \xleftarrow{\varphi'} \binom{S'}{S'}, \quad \binom{S}{S} \to \binom{S'}{S'}$$

These turn out to be very different!

$$Lens_{/\binom{A}{B}}(\varphi, \varphi') \not\cong By^A\text{-}Coalg(\varphi, \varphi')$$

SAME OBJECTS
TOTALLY DIFFERENT MORPHISMS!

And that's the difference between Bruno Gavranović et al.'s

"Para(Lens)" approach

and today's

"Coalgebras" approach.

(Or see David Jaz Myers' double category containing both.)

I'll let their group explain the merits of their approach.

The main merit of the approach in Learners' languages is that for any $p \in Poly$, the category

p-Coalg forms a topos

Thus, while you can talk about "learners", in the sense of Backprop as functor, in either setting (we'll see how soon), we get a ready-made language of dependent type theory and its associated higher-order logic if we use the coalgebra maps as morphisms.

③ Org, the "topos-enriched" monoidal category of organizations

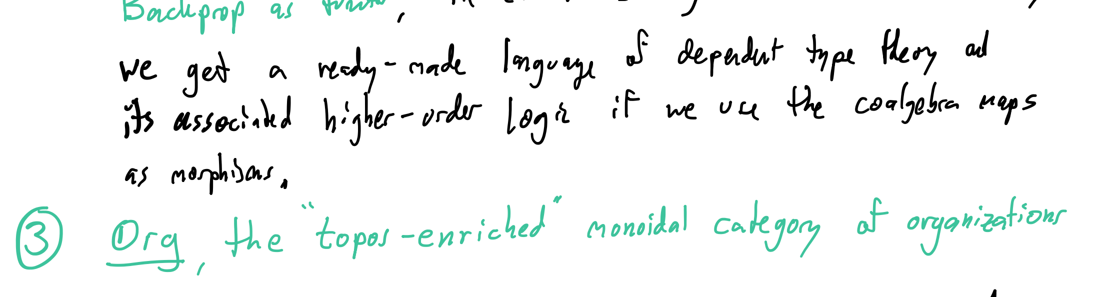
$$Ob(Org) := Ob(Poly) \qquad \text{"interfaces"} \qquad By^A$$

$$Org(p, q) := [p, q]\text{-coalg} \in Topos$$

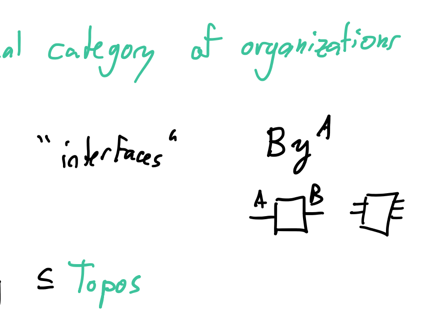
Monoidal structure: unit $= y$ $\binom{1}{1}$, product $= \otimes$ $\quad \binom{A}{B} \otimes \binom{A'}{B'} = \binom{A \times A'}{B \times B'}$.

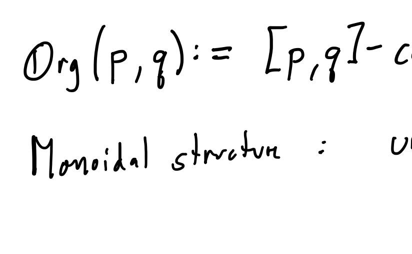
"Process theory" as Coecke-Kissinger would call it:

States	Processes	Effects	Scalars
$Org(y, p)$	$Org(p, q)$	$Org(p, y)$	$Org(y, y)$
$=$ p-coalg	$=$ $[p, q]$-coalg	$=$ $[p, y]$-coalg	$=$ y-coalg
$=$ Topos of p-dynamical systems	$=$ Topos of organizers	$=$ Topos of simulators	$=$ Topos of Lawvere DDS's
$S \to p(S)$	$S \to [p, q](S)$	$S \to [p, y](S)$	$S \to S$



Let's turn things sideways in 3D and collapse boxes into strings





Ten learners, five of which are input-output dynamical systems, and five of which are organizers, dynamically rewiring their systems,

Deep learning, the algorithm of updating "weights & biases" via gradient descent and backpropagation is fully described by a logical formula in the internal language of toposes of the form $Org(\mathbb{R}^n y^{\mathbb{R}^n}, \mathbb{R}^n y^{\mathbb{R}^n})$ $= Org\left(\binom{\mathbb{R}^m}{\mathbb{R}^m}, \binom{\mathbb{R}^n}{\mathbb{R}^n}\right)$

$$\text{Deep Learning} \subseteq Prop_{Org(\mathbb{R}^m y^{\mathbb{R}^m}, \mathbb{R}^n y^{\mathbb{R}^n})}.$$

But even the data scientists who adjust the architectures could be described in the same topos.

Org includes dynamical systems, deep learning, Martingale betting games, etc. It's quite general.

④ Conclusion

Deep learning algorithms can be described as objects in a topos. These same toposes describe "dynamic rewiring" characters, like me. They fit together in a monoidal 2-category Org

$$Org(p, q) = [p, q]\text{-Coalg}$$

If you like lenses, do the whole thing there, but think of $\binom{A}{B}$ as a functor

$$\binom{A}{B}: Set \to Set$$
$$X \longmapsto A \times X^B$$

and use its coalgebras.