What is a good process semantics?

Robin Cockett

robin@cpsc.ucalgary.ca

University of Calgary

What is a good process semantics?

1. Where are we? Where should we be?
2. Communication on a channel.
3. Polycategories and representability.
4. Communication on many channels.
5. Communication protocols.

1.

Where are we?

Where should we be?

Distributed computing: the reality ...

In the 1970's networks, parallel, and distributed computing was going to solve everything!
Practitioners pushed back with "the fallacies" (Joy, Lyon, Deutsch, Gosling):

- The network is reliable.
- Latency is zero.
- Bandwidth is infinite
- The network is secure.
- Topology doesn't change.
- Transport cost is zero.
- The network is homogeneous.

Computing had blindly entered a new world of expectation and connectedness!

There was no turning back ...

Practice ahead of theory ...

Where was the theory?

- Where was the mathematics of processes, concurrency, communication?
- Was the theory only develop in response to practice?
- Was theory simply modeling practice? Should it?
- Was there a need to develop new theory ...
  .... or was it just taking time to link existing theory and practice?

Does mathematics have anything insightful to say about communicating processes?

A brief history of process semantics ...

- Petri nets, C. A. Petri (1962).
- Communicating Sequential Processes (CSP), C. A. R. Hoare (1978).
- Calculus of Communicating Processes (CCS), R. Milner (1979) [book (1989)].
- Algebra of Communicating Processes (ACP), J. Bergstra and J. W. Klop (1982).
- Robin Milner's quest to find the "λ-calculus of concurrency" produced the π-calculus with J. Parrow (1992) [book (1999)].
- Others: ambient calculus L. Cardelli, A.D. Gordon), PEPA (J. Hillston), the fusion calculus (J. Parrow and B. Victor), the spy calculus (M. Abadi and A. Gordon), ...
- "What are the fundamental structures of concurrency? We still don't know!"
  "Is this profusion a scandal of our subject: I used to think so ... now I am not so sure."
  Samson Abramsky (2005)

The complaint ...

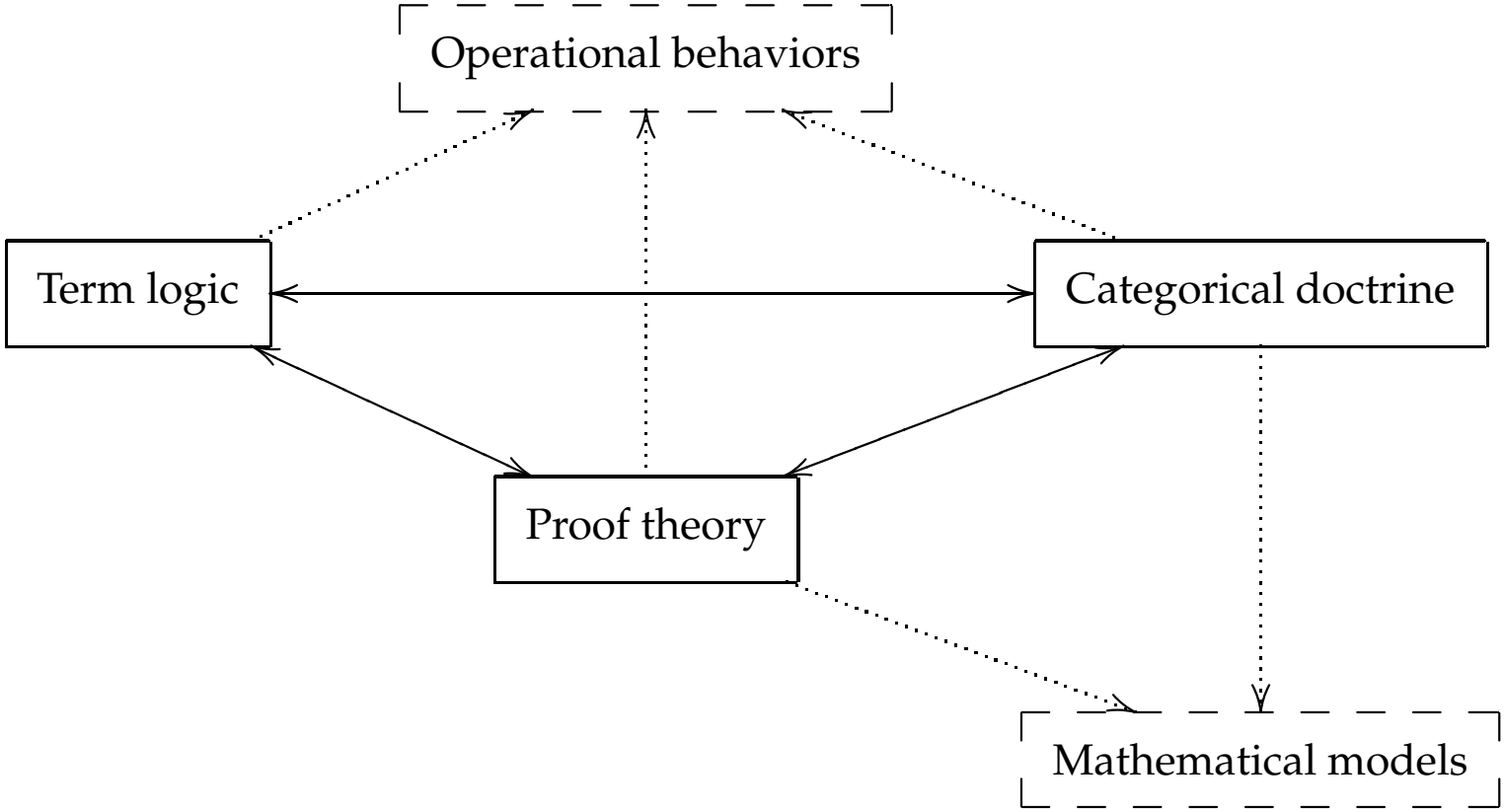
Abramsky's complaints:

- No Church's thesis for concurrency ...
- A tool kit: no unified theory ...
- Plasticity of definition, carvings in snow: no bedrock ...
- A profusion of syntax but no semantics ...
- Physics (quantum computing) and biology (biological modeling) are at our gates: what do we have to show them?

Abramsky's resolution:

- Should we expect more than a tool kit?
- The tool kit is not so bad: bisimulation, hiding, scope extrusion, ...
- The subject covers a wide range of phenomena ...

So what is a good process semantics?



What is missing?

Term logic: A convenient programming language: the focus of recent work.

Categorical doctrine: The algebraic semantics: why is it important?

- Universally given constructs (properties versus structure).
- Basic equational rules of equality: allowing (localized) program transformations.
- Compositional semantics: allowing program construction.
- Modular description: allows "feature" addition with control over their interactions.
- Interface to mathematics: models of these settings with different properties.

Proof theory: Detailed behavior of the free term model:

- Term construction
- Type inference and checking.
- Compositional behavior from cut elimination.

... in other words

BISIMULATION

SUCKS

So where are we?

Still developing the mathematics!!

... but we are further along than you might think!

2.

Communication on a channel

(A surprising bit of bedrock!)

Products and coproducts

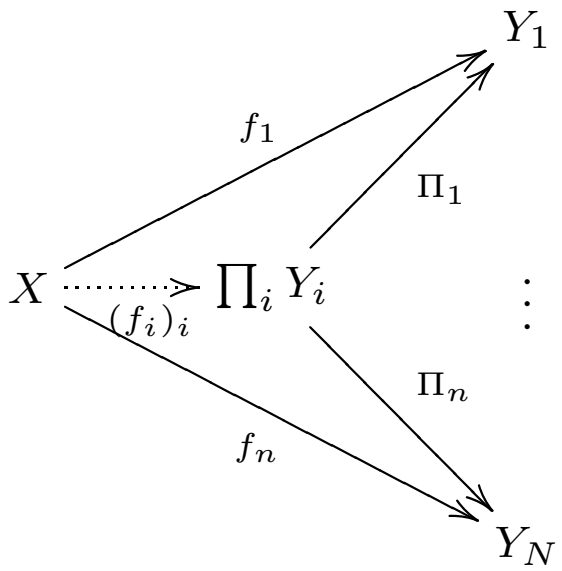
- Abelian groups, suplattices, relations: $A + B = A \times B$ (biproducts)
- Sets, topoi, cartesian closed categories, extensive and distributive categories
  - $A + B = A \sqcup B$ (disjoint union)
  - $A \times B$ (cartesian product)
  - $A \times (B + C) \cong (A \times B) + (A \times C)$

In all these settings the product and coproducts satisfy some very special properties!

What does $\Sigma\Pi(\mathbf{A})$ the category with free products and coproducts generated by the category $\mathbf{A}$ look like?

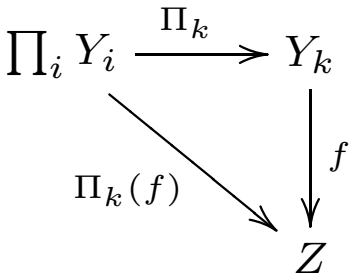
Andre Joyal: *Free bicomplete categories*.

Products



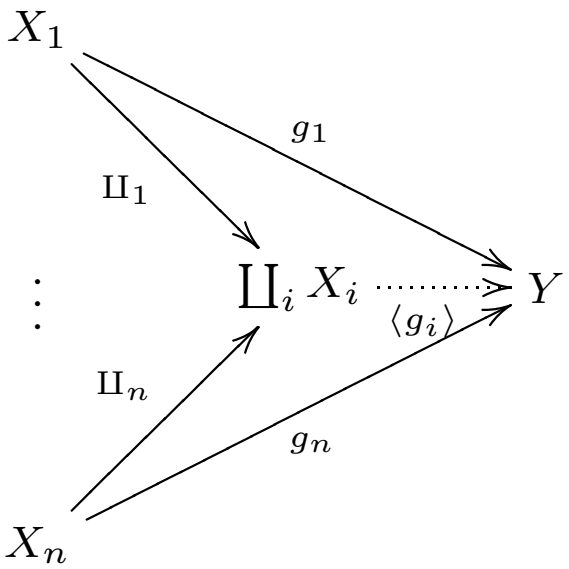
$$\begin{aligned} \Pi_k(f);g &= \Pi_k(f;g) \\ f;(g_i)_{i\in I} &= (f;g_i)_{i\in I} \\ (f_i)_{i\in I};\Pi_k(g) &= f_k;g \end{aligned}$$

where



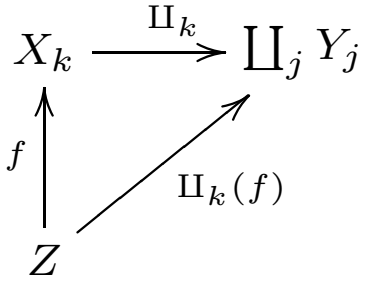
$\Pi_k;f = \Pi_k(f)$ and $\Pi_k = \Pi_k(1_{Y_k})$

Coproducts



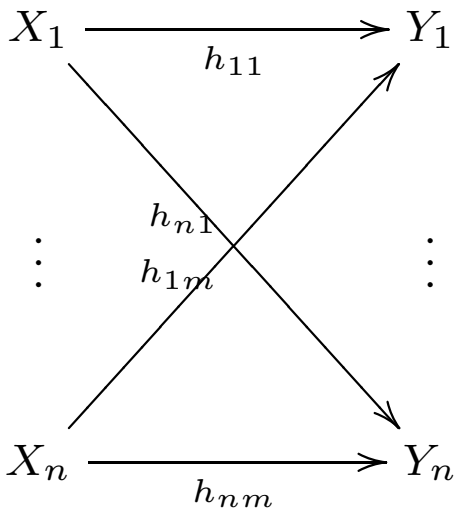
$$
\begin{aligned}
f; \amalg_k(g) &= \amalg_k(f; g) \\
\langle f_j \rangle_{j \in J}; g &= \langle f_j; g \rangle_{j \in J} \\
\amalg_k(f); \langle g_j \rangle_{j \in J} &= f; g_k
\end{aligned}
$$

where



$$f; \amalg_k = \amalg_k(f) \text{ and } \amalg_k = \amalg_k(1_{X_k})$$

Interactions



gives $\coprod_i X_i \xrightarrow{\langle (h_{ij})_i \rangle_j = (\langle h_{ij} \rangle_j)_i} \prod_j Y_j$

and some other basic equalities:

$$\Pi_i(\amalg_j(f)) = \amalg_j(\Pi_i(f)) \qquad \Pi_k((g_i)_i) = (\Pi_k(g_i))_i \qquad \amalg_k(\langle f_j \rangle_j) = \langle \amalg_k(f_j) \rangle_j$$

Logic of products and coproducts

$$\frac{}{A \vdash_{1_A} A}\ \text{id}$$

$$\frac{\{X_j \vdash_{f_j} Y\}_{j \in J}}{\coprod_j X_j \vdash_{\langle f_j \rangle_{j \in J}} Y}\ \text{cotuple} \qquad \frac{\{X \vdash_{g_i} Y_i\}_{i \in I}}{X \vdash_{(g_i)_{i \in I}} \prod_i Y_i}\ \text{tuple}$$

$$\frac{X \vdash_f Y_k}{X \vdash_{\amalg_k(f)} \coprod_{i \in I} Y_i}\ \text{coproj} \qquad \frac{X_k \vdash_f Y}{\prod_{i \in I} X_i \vdash_{\Pi_k(f)} Y}\ \text{proj}$$

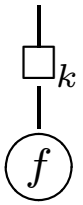
$$\frac{X \vdash_f Y \quad Y \vdash_g Z}{X \vdash_{f;g} Z}\ \text{cut}$$

Cut elimination

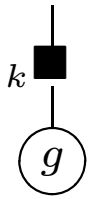
... is rewriting modulo equations:

$$
\begin{aligned}
f;1 &\Longrightarrow f \\
1;f &\Longrightarrow f \\
f;\Pi_k(g) &\Longrightarrow \Pi_k(f;g) \\
\Pi_k(f);g &\Longrightarrow \Pi_k(f;g) \\
\langle f_i\rangle_i;g &\Longrightarrow \langle f_i;g\rangle_i \\
f;(g_i)_i &\Longrightarrow (f;g_i)_i \\
\Pi_k(f);\langle g_i\rangle_i &\Longrightarrow f;g_k \\
(f_i)_i;\Pi_k(g) &\Longrightarrow f_k;g
\end{aligned}
\qquad
\begin{aligned}
\Pi_k(\langle f_j\rangle_j) &\Longmapsto \langle \Pi_k(f_j)\rangle_j \\
\Pi_k((f_i)_i) &\Longmapsto (\Pi_k(f_i))_i \\
\Pi_i(\Pi_j(f)) &\Longmapsto \Pi_j(\Pi_i(f)) \\
(\langle f_{ij}\rangle_i)_j &\Longmapsto \langle (f_{ij})_j\rangle_i
\end{aligned}
$$

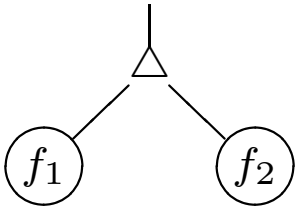
Process reading ...



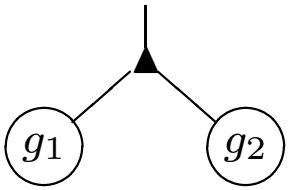
Output "k" on the right $= \Pi_k(f)$



Output "k" on the left $= \Pi_k(g)$



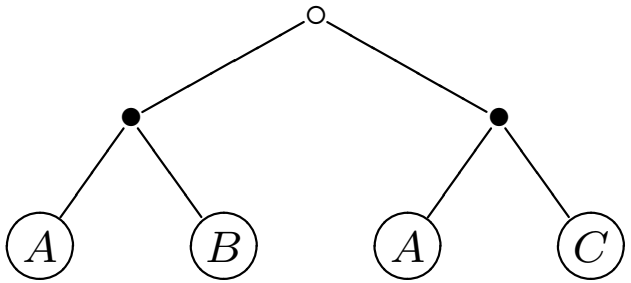
Listen for input on the left $= \langle f_1, f_2 \rangle$

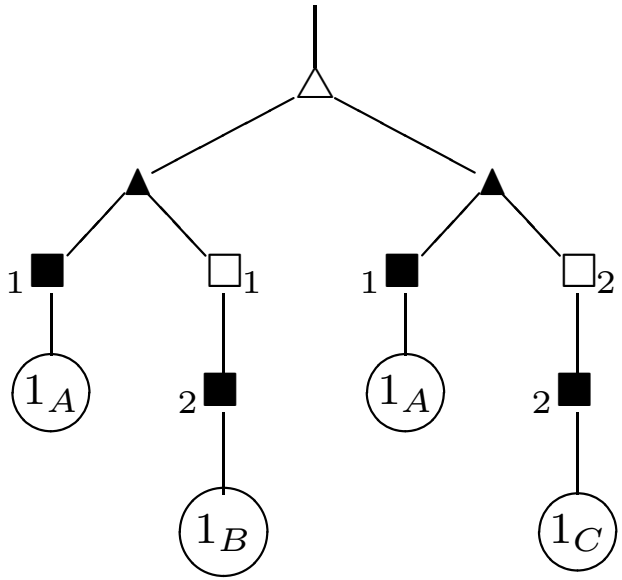


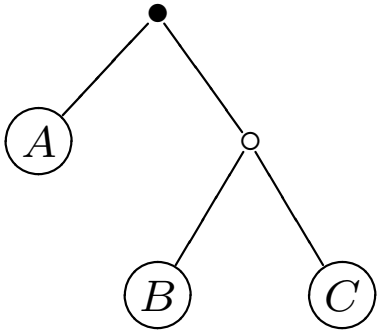
Listen for input on the right $= (g_1, g_2)$

Process reading of a map ...

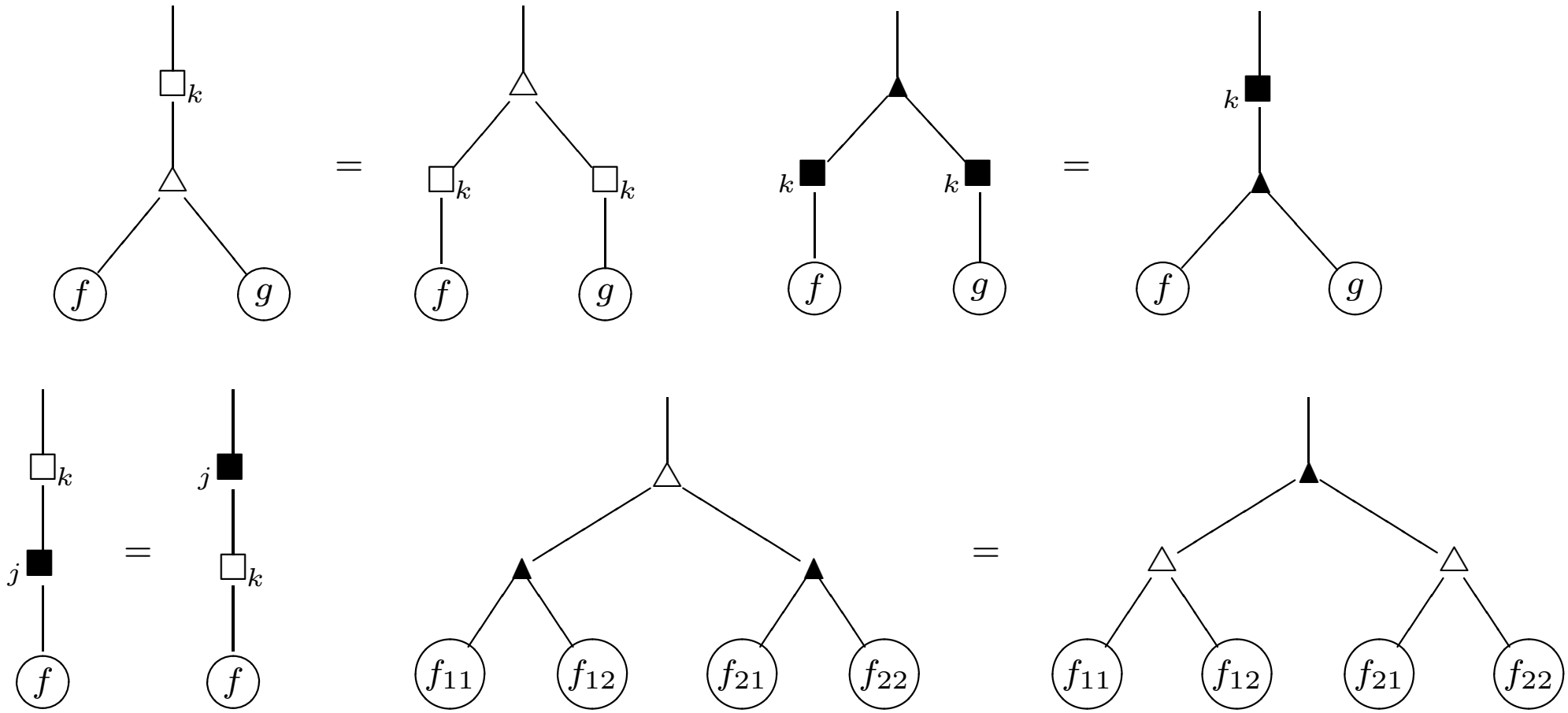
$$(A \times B) + (A \times C) \xrightarrow{\langle(\Pi_1(1_A),\, \amalg_1(\Pi_2(1_B)))\ ,\ (\Pi_1(1_A),\, \amalg_2(\Pi_2(1_C)))\rangle} A \times (B + C)$$







Process reading of the identities ...



Slogan

The logic of products and coproducts is the logic of communication along a channel.

Joyal: "... the mathematics is saying something."

Different readings ...

TYPE THEORY	CATEGORIES	PROOF THEORY	PROCESSES	GAMES
Type	Object	Proposition	Protocol	Game
Terms	Map	Proof	Process	Mediator
Substitution	Composition	Cut	Communication	Composition
Variables	Identities	(Identity) axioms	Relay	Copy cat

Joyal and Santocanale used the reading of games and mediators (Blass) ...
Cockett and Seely used the reading of propositions and proofs ...
Pastro used the reading as protocols and processes ...

... just products and coproducts.

Homework:

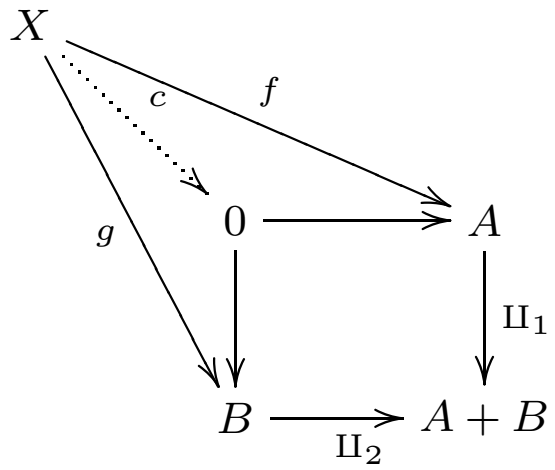
J.R.B. Cockett and R.A.G. Seely, *Finite sum-product logic*, TAC 8, 63–99.

Not everything is sorted out!!

The initial and final maps (the additive units) make things more complicated.

The additive units ...

- In $\Sigma\Pi(\mathbf{A})$ coprojections are not necessarily monic but they are weakly disjoint (Cockett and Santocanale):

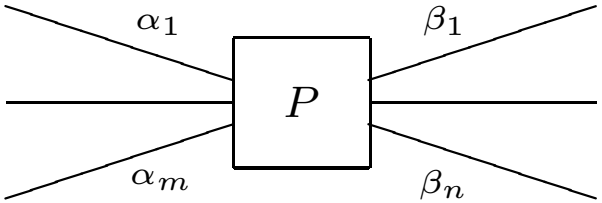


- When there are no additive units equality is easy (algorithm in above paper).
- In the presence of additive units the complexity of equality is still not known (Luigi Santocanale)!
- What is their meaning in communication? ... what happens when one party stops listening!

3.

Polycategories and representation

Processes connected to many channels



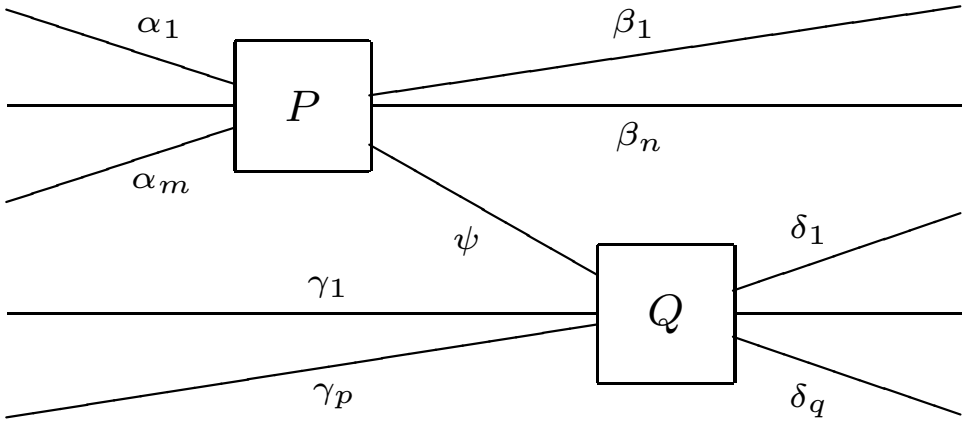
$$\alpha_1 : X_1, \ldots, \alpha_m : X_m \vdash_P \beta_1 : Y_1, \ldots, \beta_n : Y_n$$

$X_1, .., X_m, Y_1, ..., Y_n$ are *protocols* ...
These are types determine which events can happen next on each channel (e.g. given by products and coproduct types).

A process can listen or output to any channel to which it is attached. The process is the *system* and it communicates with its *environment*.

Communication

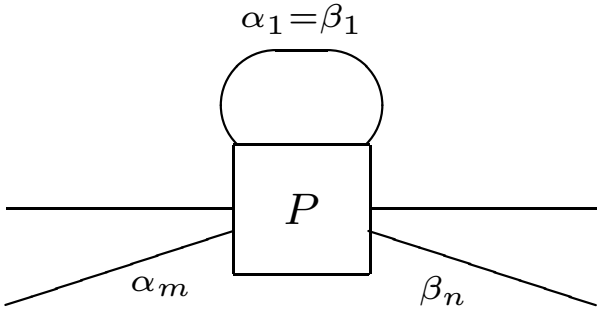
Plugging processes together ...



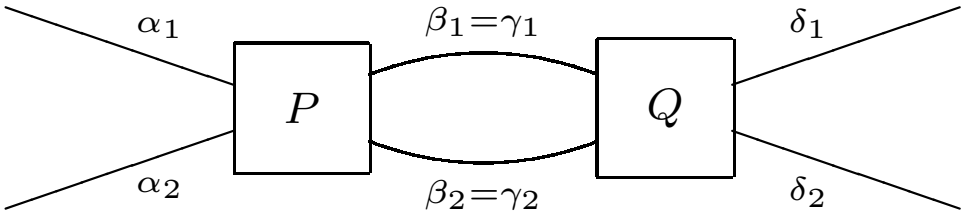
The combined processes become a composite process with the communication on ψ hidden.

Miscommunications ...

Plugging processes together in the wrong way can cause deadlock or livelock ...



Don't plug a process to itself..



Don't connect two processes in two different ways ..

Polycategories

A polycategory $\mathbf{P}$ consists of the data

- objects: $X_1, \ldots, Y_1, \ldots \in \mathbf{P}_0$
- polymaps: $\forall m, n \in \mathbb{N}$ a set

$$\mathbf{P}(X_1, \ldots, X_m \; ; \; Y_1, \ldots, Y_n)$$

- identities: for each $X \in \mathbf{P}_0$ a polymap $1_X \in \mathbf{P}(X; X)$.
- composition (cut): A map

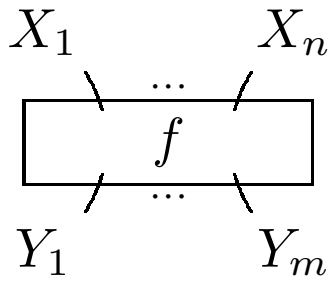
$$\mathbf{P}(\Gamma; \Delta_1, X, \Delta_2) \times \mathbf{P}(\Gamma_1, X, \Gamma_2; \Delta) \longrightarrow \mathbf{P}(\Gamma_1, \Gamma, \Gamma_2; \Delta_1, \Delta, \Delta_2)$$

  where Γ_1 or Δ_1 is empty and Γ_2 or Δ_2 is empty.

such that identities *are* identities and cut satisfies associativity and interchange.
A polycategory is symmetric in case $\mathbf{P}(\sigma\Gamma; \tau\Delta) = \mathbf{P}(\Gamma; \Delta)$ for permutations σ and τ, and certain obvious coherence conditions hold.

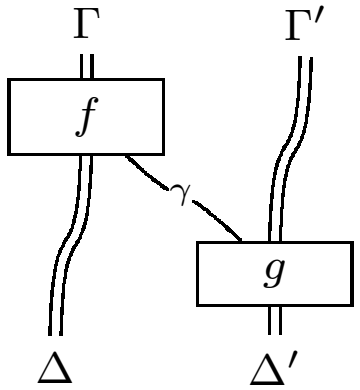
Polycategories

$$X_1, \ldots, X_n \vdash_f Y_1, \ldots, Y_m$$



Composition is modeled by the cut rule

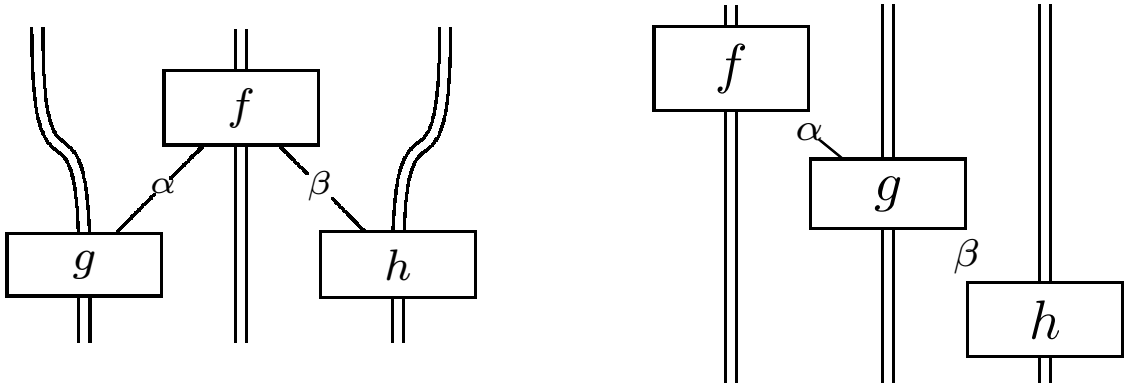
$$\frac{\Gamma \vdash_f \Delta, \gamma : Z \quad \gamma : Z, \Gamma' \vdash_g \Delta}{\Gamma, \Gamma' \vdash_{f;_\gamma g} \Delta, \Delta'}$$



Composition must have identities (these are wires) ..

Polycategories

Composition must satisfy the interchange and associative laws



When polycategories are symmetric crossing wires are allowed.

Pure proof theory of cut-elimination

Symmetric polycategories are the categorical proof theory for cut-elimination.

$$\frac{}{A \vdash_{1_A} A}\ \text{id}$$

$$\frac{\Gamma_1, X_1, X_2, \Gamma_2 \vdash \Gamma}{\Gamma_1, X_2, X_1, \Gamma_2 \vdash \Gamma}\ \text{exchange} \qquad \frac{\Gamma \vdash \Gamma_1, X_1, X_2, \Gamma_2}{\Gamma \vdash \Gamma_1, X_2, X_1, \Gamma_2}\ \text{exchange}$$

$$\frac{\Gamma_1 \vdash \Gamma_2, X \quad X, \Delta_1 \vdash \Delta_2}{\Gamma_1, \Delta_1 \vdash \Gamma_2, \Delta_2}\ \text{cut} \qquad \frac{\Gamma_1 \vdash X, \Gamma_2 \quad \Delta_1, X \vdash \Delta_2}{\Delta_1, \Gamma_1 \vdash \Delta_2, \Gamma_2}\ \text{cut}$$

$$\frac{\Gamma \vdash X \quad \Delta_1, X, \Delta_2 \vdash \Delta}{\Delta_1, \Gamma, \Delta_2 \vdash \Delta}\ \text{cut} \qquad \frac{\Gamma \vdash \Gamma_1, X, \Gamma_2 \quad X \vdash \Delta}{\Gamma \vdash \Gamma_1, \Delta, \Gamma_2}\ \text{cut}$$

Representability

A polycategory is *representable* in case there are *polynatural* equivalences

$$\mathbf{P}(\Gamma_1, X, Y, \Gamma_2; \Delta) \xrightarrow[\sim]{r_\otimes} \mathbf{P}(\Gamma_1, X \otimes Y, \Gamma_2; \Delta)$$

$$\mathbf{P}(\Gamma_1, \Gamma_2; \Delta) \xrightarrow[\sim]{r_\top} \mathbf{P}(\Gamma_1, \top, \Gamma_2; \Delta)$$

$$\mathbf{P}(\Gamma; \Delta_1, X, Y, \Delta_2) \xrightarrow[\sim]{r_\oplus} \mathbf{P}(\Gamma; \Delta_1, X \oplus Y, \Delta_2)$$

$$\mathbf{P}(\Gamma; \Delta_1, \Delta_2) \xrightarrow[\sim]{r_\bot} \mathbf{P}(\Gamma; \Delta_1, \bot, \Delta_2)$$

Replace the commas with composite types ...

Polynatural means that the transformation is invariant under cutting into the non-active position ...

Representability was introduced by Burroni (and used by Hermida) to simplify coherence for bicategories (and tricategories). In any polycategory having tensors is a *property* (representability) rather than extra *structure*.

The multiplicatives

Representability can be presented by sequent calculus rules of inference:

$$\frac{\Gamma_1, \Gamma_2 \vdash \Delta}{\Gamma_1, \top, \Gamma_2 \vdash \Delta}\ \text{split}\ \top \qquad \frac{\Gamma \vdash \Delta_1, \Delta_2}{\Gamma \vdash \Delta_1, \bot, \Delta_2}\ \text{split}\ \bot$$

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, A \otimes B, \Gamma_2 \vdash \Delta}\ \text{split}\ \otimes \qquad \frac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta_1, A \oplus B, \Delta_2}\ \text{split}\ \oplus$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \top}\ \text{fork}\ \top \qquad \frac{\vdash \Delta}{\bot \vdash \Delta}\ \text{fork}\ \bot$$

$$\frac{\Gamma_1 \vdash \Delta_1, A \quad \Gamma_2 \vdash B, \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, A \otimes B, \Delta_2}\ \text{fork}\ \otimes \qquad \frac{\Gamma_1, A \vdash \Delta_1 \quad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, A \oplus B, \Gamma_2 \vdash \Delta_1, \Delta_2}\ \text{fork}\ \oplus$$

Simple derivations

Here is a derivation of the associativity isomorphism:

$$\dfrac{\dfrac{\dfrac{\dfrac{A \vdash A \quad \dfrac{B \vdash B \quad C \vdash C}{B, C \vdash B \otimes C}\ \text{fork}}{A, B, C \vdash A \otimes (B \otimes C)}\ \text{fork}}{(A \otimes B), C \vdash A \otimes (B \otimes C)}\ \text{split}}{(A \otimes B) \otimes C \vdash A \otimes (B \otimes C)}\ \text{split}$$

Linearly distributive categories

Theorem 1. *Representable polycategories correspond precisely to linearly distributive categories.*

That is in a representable polycategory:

- $\otimes, \top$ and $\oplus, \bot$ become monoidal structure.
- There are coherent natural transformations

$$\delta_L : A \otimes (B \oplus C) \longrightarrow (A \otimes B) \oplus C$$

$$\delta_R : (B \oplus C) \otimes A \longrightarrow B \oplus (C \otimes A)$$

called the *linear distributions*.

Linear distributivity

Cockett and Seely *Weakly distributive categories* (now known as *linearly* distributive categories to emphasize the link to Girard's *linear logic*).
Here is the derivation of one of the linear distributions:

$$\dfrac{\dfrac{\dfrac{A \vdash A \quad B \vdash B}{A, B \vdash A \otimes B}\ \text{fork}\otimes \qquad C \vdash C}{A, B \oplus C \vdash A \otimes B, C}\ \text{fork}\oplus}{A \otimes (B \oplus C) \vdash (A \otimes B) \oplus C}\ \text{split}$$

There are many coherence requirements but they are all very natural. A typical coherence requirement is:

$$\begin{array}{ccc}
A \otimes (B \otimes (C \oplus D)) & \xrightarrow[a_\otimes]{} & (A \otimes B) \otimes (C \oplus D)) \\
\downarrow{\scriptstyle 1\otimes\delta_L} & & \\
A \otimes ((B \otimes C) \oplus D) & & \downarrow{\scriptstyle \delta_L} \\
\downarrow{\scriptstyle \delta_L} & & \\
(A \otimes (B \otimes C)) \oplus D & \xrightarrow[a_\otimes \oplus 1]{} & ((A \otimes B) \otimes C) \oplus D
\end{array}$$

Examples

- A distributive lattice $\wedge = \otimes$, $\vee = \oplus$ (not *-autonomous).
- A distributive category is a linearly distributive category (with respect to the product and coproduct and the obvious linear distribution) if and only if it is a poset.
- Any monoidal category is a degenerately a linear distributive category.
- Any $*$-autonomous category is a linearly distributive category.
- Any compact closed category is $*$-autonomous.
- (Joyal) Bicompletions of monoidal / linearly distributive categories are linearly distributive (generally not *-autonomous).
- (Koslowski) *Sets* with cartesian product and the following "par"

$$A \oplus B = \begin{cases} A & \text{if } B = 0 \\ B & \text{if } A = 0 \\ 1 & \text{otherwise} \end{cases}$$

is a linearly distributive category (but is *not* *-autonomous).

Units again ..

Units cause big problems:

- If you are French you pretend they don't exist! This is not wise
  ... because even if you don't mention them they are implicitly present.
- If you are Canadian you think they are the main interest of these settings!!
- A decision procedure for equality of maps *in the symmetric case* is known.
- It is open as to what the complexity of this decision problem is ... (conjecture: it is coNP?).
- A decision procedure for equality of maps in *special cases* of the non-symmetric case is known.

Blute, Cockett, Seely, Trimble *Natural deduction and coherence for weakly distributive categories.* JPAA 1996.
Schneck *Natural deduction and coherence for non-symmetric linearly distributive categories.* TAC 1999

4.

Communication on many channels

Polyproducts and polycoproducts

There are polynatural equivalences

$$\mathbf{P}(\Gamma_1, X + Y, \Gamma_2; \Delta) \xrightarrow[\sim]{r_+} \mathbf{P}(\Gamma_1, X, \Gamma_2; \Delta) \times \mathbf{P}(\Gamma_1, Y, \Gamma_2; \Delta)$$

$$\mathbf{P}(\Gamma_1, 0, \Gamma_2; \Delta) \xrightarrow[\sim]{r_0} 1$$

$$\mathbf{P}(\Gamma; \Delta_1, X \times Y, \Delta_2) \xrightarrow[\sim]{r_\times} \mathbf{P}(\Gamma; \Delta_1, X, \Delta_2) \times \mathbf{P}(\Gamma; \Delta_1, Y, \Delta_2)$$

$$\mathbf{P}(\Gamma; \Delta_1, \Delta_2) \xrightarrow[\sim]{r_1} 1$$

When $\mathbf{P}$ is representable we have

$$X \otimes (A + B) \cong (X \otimes A) + (X \otimes B) \quad \text{and} \quad (A + B) \oplus Y \cong (A \oplus Y) + (B \oplus Y).$$

e.g.

$$\dfrac{\dfrac{\mathbf{P}(\Gamma, X \otimes A, \Gamma'; \Delta) \quad \mathbf{P}(\Gamma, X \otimes B, \Gamma'; \Delta)}{\mathbf{P}(\Gamma, X, A, \Gamma'; \Delta) \quad \mathbf{P}(\Gamma, X, B, \Gamma'; \Delta)}}{\dfrac{\mathbf{P}(\Gamma, X, (A + B), \Gamma'; \Delta)}{\mathbf{P}(\Gamma, X \otimes (A + B), \Gamma'; \Delta)}}$$

Polylogic of products and coproducts

$$\frac{}{A \vdash_{1_A} A}\ \text{id}$$

$$\frac{\{\Gamma_1, \alpha : X_j, \Gamma_2 \vdash_{P_j} \Gamma_3\}_j}{\Gamma_1, \alpha : \coprod_j X_j, \Gamma_2 \vdash_{\alpha\langle P_j\rangle_j} \Gamma_3}\ \text{cotuple} \qquad \frac{\{\Gamma_1 \vdash_{Q_i} \Gamma_2, \alpha : Y_i, \Gamma_3\}_i}{\Gamma_1 \vdash_{\alpha\langle Q_i\rangle_i} \Gamma_2, \alpha : \prod_i Y_i, \Gamma_3}\ \text{tuple}$$

$$\frac{\Gamma_1 \vdash_P \Gamma_2, \alpha : Y_k, \Gamma_3}{\Gamma_1 \vdash_{\alpha[k]\cdot P} \Gamma_2, \alpha : \coprod_i Y_i, \Gamma_3}\ \text{coproj} \qquad \frac{\Gamma_1, \alpha : X_k, \Gamma_2 \vdash_Q \Gamma_3}{\Gamma_1, \alpha : \prod_i X_i, \Gamma_2 \vdash_{\alpha[k]\cdot Q} \Gamma_3}\ \text{proj}$$

$$\frac{\Gamma_1 \vdash_P \Gamma_2, \alpha : X, \Gamma_3 \quad \Delta_1, \alpha : X, \Delta_2 \vdash_Q \Delta_3}{\Delta_1, \Gamma_1, \Delta_2 \vdash_{P;_\alpha Q} \Gamma_2, \Delta_3, \Gamma_3}\ \text{cut}$$

Cut elimination ...

... is rewriting modulo equations:

$\alpha \neq \beta$

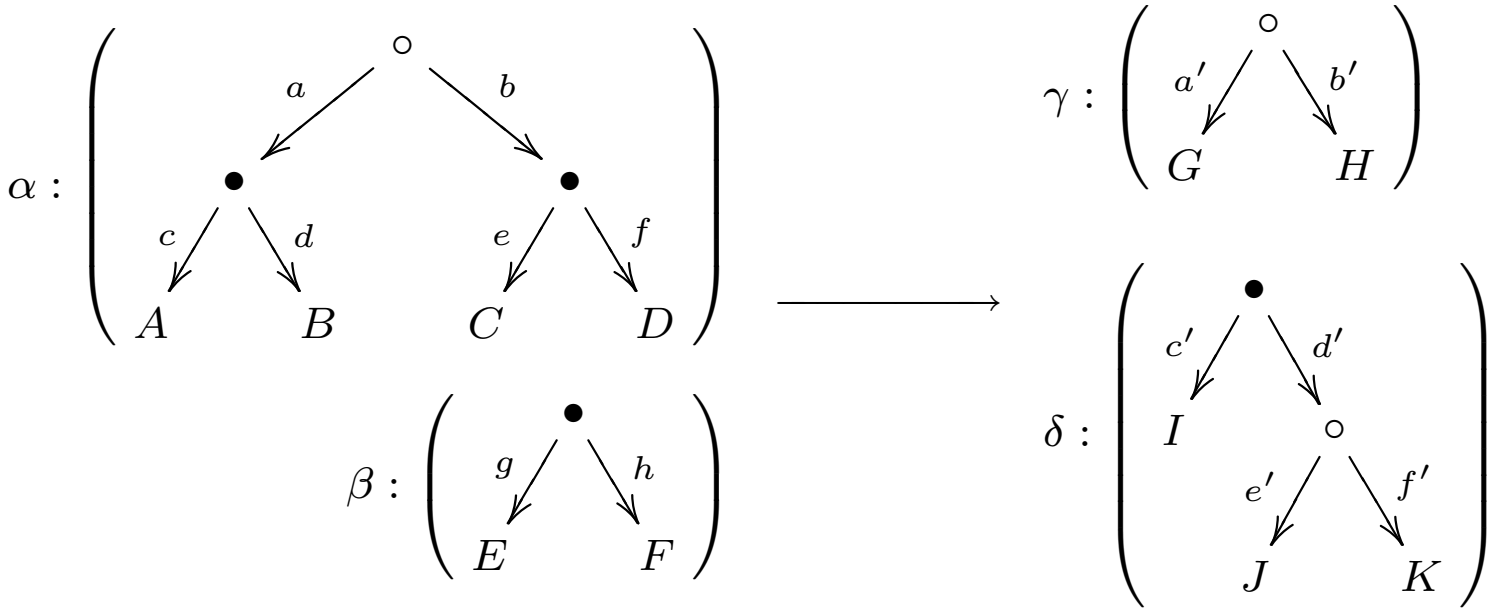
$$
\begin{aligned}
(\alpha[k] \cdot P) \;;_\gamma\; Q &\Longrightarrow \alpha[k] \cdot (P \;;_\gamma\; Q) \\
P \;;_\gamma\; (\beta[k] \cdot Q) &\Longrightarrow \beta[k] \cdot (P \;;_\gamma\; Q) \\
\alpha\langle P_i \rangle_i \;;_\gamma\; Q &\Longrightarrow \alpha\langle P_i \;;_\gamma\; Q \rangle_i \\
P \;;_\gamma\; \beta\langle Q_j \rangle_j &\Longrightarrow \beta\langle P \;;_\gamma\; Q_j \rangle_j \\
\gamma[k] \cdot P \;;_\gamma\; \gamma\langle Q_j \rangle_j &\Longrightarrow P \;;_\gamma\; Q_k \\
\gamma\langle P_i \rangle_i \;;_\gamma\; \gamma[k] \cdot Q &\Longrightarrow P_k \;;_\gamma\; Q
\end{aligned}
$$

$$
\begin{aligned}
\alpha\langle\beta\langle P_{ij}\rangle_j\rangle_i &\Longmapsto \beta\langle\alpha\langle P_{ij}\rangle_i\rangle_j \\
\alpha[k] \cdot \beta\langle P_j\rangle_j &\Longmapsto \beta\langle\alpha[k] \cdot P_j\rangle_j \\
\alpha[k] \cdot \beta[l] \cdot P &\Longmapsto \beta[l] \cdot \alpha[k] \cdot P
\end{aligned}
$$

A multi-channel process example

$$f : A, E \longrightarrow G, I \qquad g : B, E \longrightarrow G, J$$

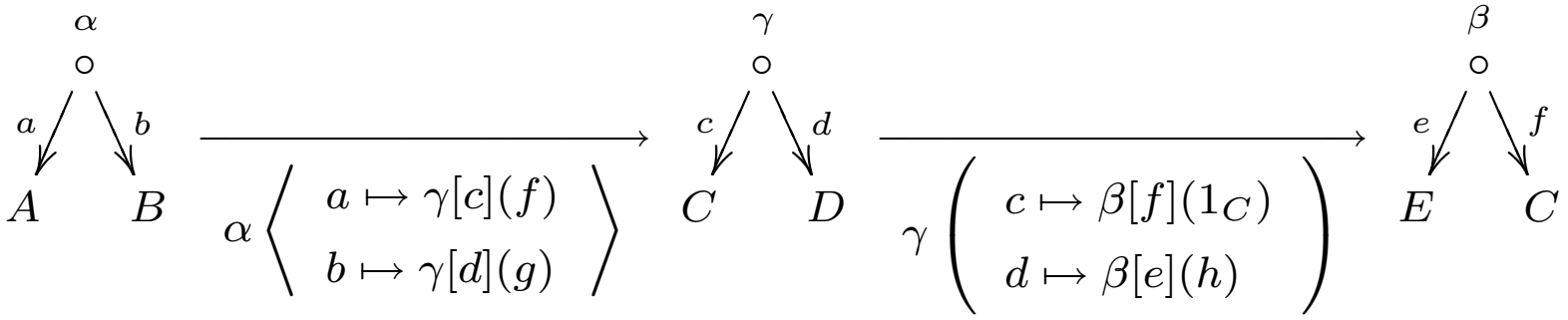
$$h : C, F \longrightarrow H, I \qquad k : D, F \longrightarrow H, K$$



$$\alpha \left\langle \begin{array}{l} a \mapsto \beta[g](\gamma[a'](\delta \left(\begin{array}{l} c' \mapsto \alpha[c](f) \\ d' \mapsto \alpha[d](\delta[e'](g)) \end{array} \right))) \\ b \mapsto \beta[h](\gamma[b'](\delta \left(\begin{array}{l} c' \mapsto \alpha[e](h) \\ d' \mapsto \alpha[f](\delta[f'](k)) \end{array} \right))) \end{array} \right\rangle$$

An example of cut-elimination

Suppose $f : A \longrightarrow D$, $g : B \longrightarrow C$, and $h : D \longrightarrow E$ are atomic axioms.



An example cont.

$$\alpha \left\langle \begin{array}{l} a \mapsto \gamma[c](f) \\ b \mapsto \gamma[d](g) \end{array} \right\rangle \quad ;_\gamma \quad \gamma \left(\begin{array}{l} c \mapsto \beta[f](h) \\ d \mapsto \beta[e](1_C) \end{array} \right)$$

$$\Rightarrow \alpha \left\langle \begin{array}{lll} a \mapsto \gamma[c](f) & ;_\gamma & \gamma \left(\begin{array}{l} c \mapsto \beta[f](h) \\ d \mapsto \beta[e](1_C) \end{array} \right) \\ b \mapsto \gamma[d](g) & ;_\gamma & \gamma \left(\begin{array}{l} c \mapsto \beta[f](h) \\ d \mapsto \beta[e](1_C) \end{array} \right) \end{array} \right\rangle$$

$$\Rightarrow \quad \alpha \left\langle \begin{array}{l} a \mapsto f ;_\gamma \beta[f](h) \\ b \mapsto g ;_\gamma \beta[e](1_C) \end{array} \right\rangle \quad \Rightarrow \quad \alpha \left\langle \begin{array}{l} a \mapsto \beta[f](f ;_\gamma h) \\ b \mapsto \beta[e](g ;_\gamma 1_C) \end{array} \right\rangle \quad \Rightarrow \quad \alpha \left\langle \begin{array}{l} a \mapsto \beta[f](f ;_\gamma h) \\ b \mapsto \beta[e](g) \end{array} \right\rangle$$

Forking and splitting

$$\frac{\Gamma_1, \alpha_1 : X, \alpha_2 : Y, \Gamma_2 \vdash_P \Gamma_3}{\Gamma_1, \alpha : X \otimes Y, \Gamma_2 \vdash_{\alpha\langle \alpha_1, \alpha_2 \mapsto P\rangle} \Gamma_3} \qquad \frac{\Gamma_1 \vdash_Q \Gamma_2, \beta_1 : X, \beta_2 : Y, \Gamma_3}{\Gamma_1 \vdash_{\beta\langle \beta_1, \beta_2 \mapsto Q\rangle} \Gamma_2, \beta : X \otimes Y, \Gamma_3}$$

$$\frac{\gamma_1 : \Gamma_1 \vdash_P \gamma_2 : \Gamma_2, \alpha_1 : X \quad \delta_1 : \Delta_1 \vdash_Q \alpha_2 : Y, \delta_2 : \Delta_2}{\gamma_1 : \Gamma_1, \delta_1 : \Delta_1 \vdash \alpha\left\langle \begin{array}{l} \alpha_1 \mid \gamma_1, \gamma_2 \mapsto P \\ \alpha_2 \mid \delta_1, \delta_2 \mapsto Q \end{array} \right\rangle \gamma_2 : \Gamma_2, \alpha : X \otimes Y, \delta_2 : \Delta_2}$$

$$\beta\langle \beta_1, \beta_2 \mapsto Q\rangle \equiv \texttt{split}\ \beta\ \texttt{as}\ \beta_1, \beta_2\ \texttt{in}\ Q$$

$$\alpha\left\langle \begin{array}{l} \alpha_1 \mid \gamma_1, \gamma_2 \mapsto P \\ \alpha_2 \mid \delta_1, \delta_2 \mapsto Q \end{array} \right\rangle \equiv \texttt{fork}\ \alpha\ \texttt{as}\ \begin{array}{l} \alpha_1 \mid \gamma_1, \gamma_2 \mapsto P \\ \alpha_2 \mid \delta_1, \delta_2 \mapsto Q \end{array}$$

More rewrites and identities

- $$\gamma \left\langle \begin{array}{l} \alpha \mid \Lambda \mapsto f \\ \beta \mid \Phi \mapsto g \end{array} \right\rangle \;;_\gamma\; \gamma\langle (\alpha, \beta) \mapsto h \rangle \Longrightarrow g;_\beta (f;_\alpha h))$$

- $$\alpha \left\langle \begin{array}{l} \alpha_1 \mid \Lambda_1 \mapsto f \\ \alpha_2 \mid \Lambda_2 \mapsto \beta \left(\begin{array}{l} a_1 \mapsto g_1 \\ a_2 \mapsto g_2 \end{array} \right) \end{array} \right\rangle \quad \Longrightarrow\!\!\!\!\!\!= \quad \beta \left(\begin{array}{l} a_1 \mapsto \alpha \left\langle \begin{array}{l} \alpha_1 \mid \Lambda_1 \mapsto f \\ \alpha_2 \mid \Lambda_2 \mapsto g_1 \end{array} \right\rangle \\ a_2 \mapsto \alpha \left\langle \begin{array}{l} \alpha_1 \mid \Lambda_1 \mapsto f \\ \alpha_2 \mid \Lambda_2 \mapsto g_2 \end{array} \right\rangle \end{array} \right)$$

There are many more identities.
See Cockett and Pastro, *A language for multiplicative-additive linear logic*, ENTCS, 2005.

The multiplicative-additive fragment has map equality decidable. However, the complexity of deciding equality is unknown. The unit-free case was handled very neatly by:
Hughes and Glabbeek, *Proof nets for unit-free multiplicative-additive linear logic*, LICS 2003.

Work in progress: Craig Pastro and I have a detailed *process reading* of the initial additive $*$-autonomous category. The system uses rules which respond to histories of interaction.

5.

Communication Protocols

Protocols

So far we have only handled *simple* protocols (using products and coproducts) which are finite. More sophisticated protocols can be introduced through datatypes.
There are two formulations:

- Lambek style datatypes: the fixed point formulation of inductive and coinductive datatypes.
- Mendler style datatypes (Vene, Uustalu) and circular style datatypes (Santocanale).

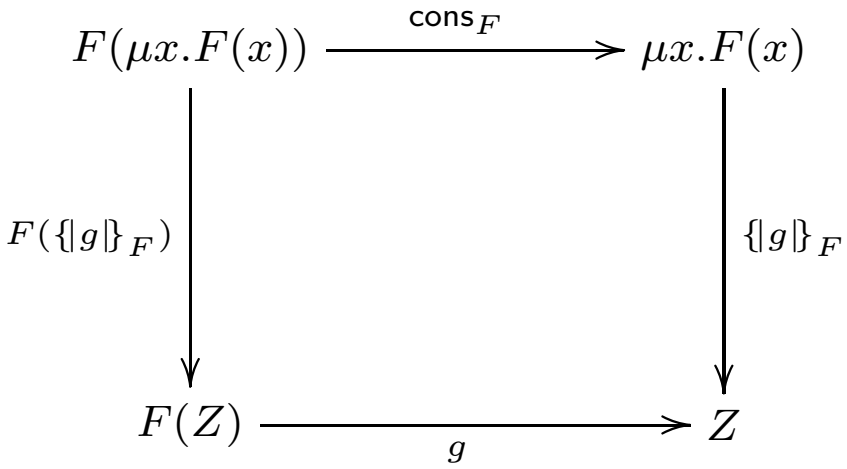
... for polycategories only the second formulation works.

Inductive datatypes

$\mathbf{X}$ a category and $F : \mathbf{X} \longrightarrow \mathbf{X}$ an endofunctor. An *inductive datatype* for F is an object $\mu x.F(x) \in \mathbf{X}$ together with a map

$$\mathsf{cons}_F : F(\mu x.F(x)) \longrightarrow \mu x.F(x)$$

such that the *inductive axiom* holds: Given $Z \in \mathbf{X}$ and a map $g : F(Z) \longrightarrow Z$ then there is a unique map $\{\!|g|\!\}_F$, such that



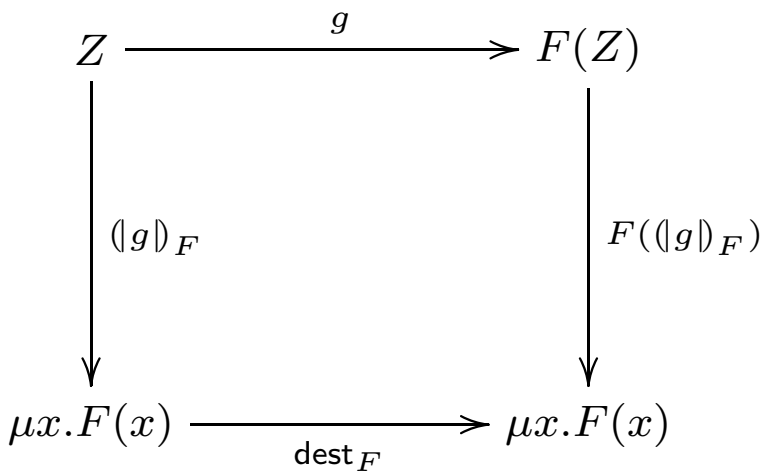
commutes.

Coinductive datatypes

$\mathbf{X}$ a category and $F : \mathbf{X} \longrightarrow \mathbf{X}$ an endofunctor. An *coinductive datatype* for F is an object $\nu x.F(x) \in \mathbf{X}$ together with a map

$$\mathsf{dest}_F : \nu x.F(x) \longrightarrow F(\mu x.F(x))$$

such that the *coinductive axiom* holds: Given $Z \in \mathbf{X}$ and a map $g : Z \longrightarrow F(Z)$ then there is a unique map $(\!|g|\!)_F$, such that



commutes.

Circular datatypes: combinators

A *combinator* provides a manner of taking arrows $f : F(X) \longrightarrow G(X)$ defined parametrically over a type variable $X \in \mathbf{X}$ and producing a map $c[f] : H(X) \longrightarrow K(X)$ written as an inference:

$$\frac{F(X) \xrightarrow{f} G(X)}{H(X) \xrightarrow[c[f]]{} K(X)}\,.$$

The sense in which this is parametric is that we have the following extension of this inference for commuting diagrams:

$$\begin{array}{ccc} H(X) & \xrightarrow{c[f]} & K(X) \\ {\scriptstyle H(x)}\downarrow & & \downarrow{\scriptstyle K(x)} \\ H(X') & \xrightarrow[c[h]]{} & K(X') \end{array} \quad \text{implies} \quad \begin{array}{ccc} F(X) & \xrightarrow{f} & G(X) \\ {\scriptstyle F(x)}\downarrow & & \downarrow{\scriptstyle G(x)} \\ F(X') & \xrightarrow[h]{} & G(X') \end{array}$$

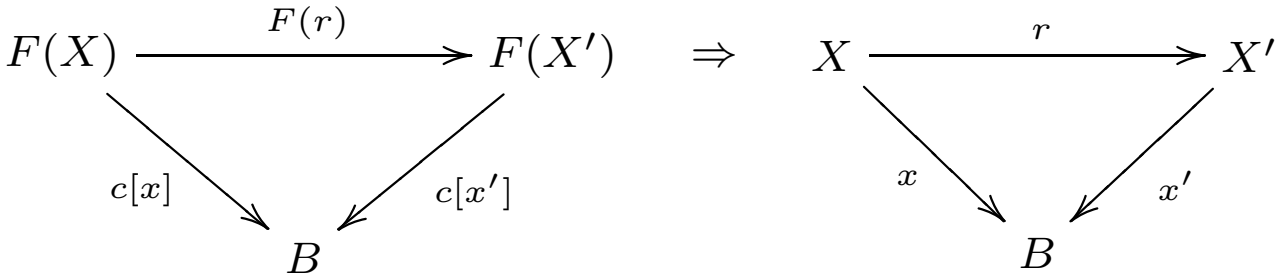
$\{\!|-|\!\}$ and $(\!|-|\!)$ are examples of combinators.

Inductice circular datatypes

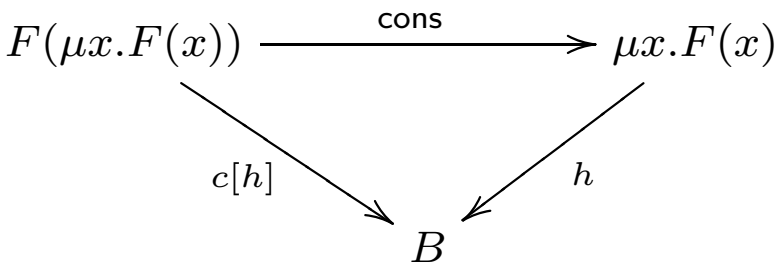
A combinator:

$$\frac{f : X \longrightarrow B}{c[f] : F(X) \longrightarrow B}$$

where



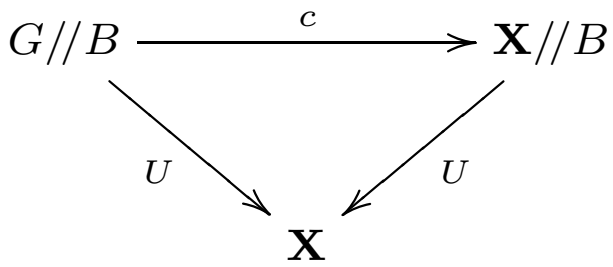
delivers a **circular map** $\mu a.c[a] : \mu x.F(x) \longrightarrow B$ such that the following diagram commutes



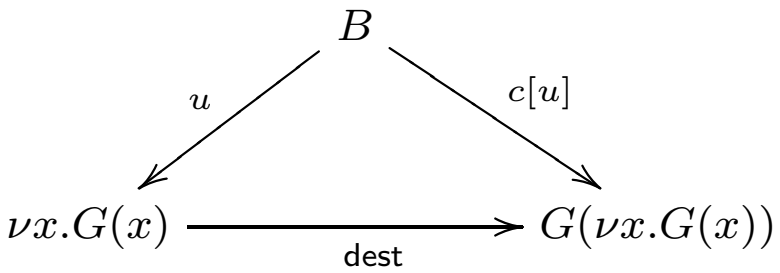
if and only if $h = \mu a.c[a]$.
Note that, in particular this means, substituting $\mu a.c[a]$ for h, that cons; $\mu a.c[a] = c[\mu a.c[a]]$.

Coinductive circular datatypes

Dually we have for coinductive datatypes the following circular style definition. Given a combinator



where B is a fixed object in $\mathbf{X}$, there is a **cocircular map** $\nu b.c[b] : B \longrightarrow \nu x.G(x)$ such that



commutes if and only if $u = \nu b.c[b]$.
In particular, this means $(\nu b.c[b]); \mathsf{dest} = c[\nu b.c[b]]$.

See also Tarmo Uustalu and Varmo Vene (thesis).

Circular rules for polycategories

Given morphism of polycategories P and Q here are the term formation rules for circular terms:

$$\frac{\Gamma \vdash_A \Delta, \beta : P(\mu x.P(x)), \Delta'}{\Gamma \vdash_{\overline{\beta}[\mu]\cdot A} \Delta, \beta : \mu x.P(x), \Delta'}\ \mu\text{-Cons}$$

$$\frac{\Gamma, \alpha : Q(\nu x.Q(x)), \Gamma' \vdash_A \Delta}{\Gamma, \alpha : \nu x.Q(x), \Gamma' \vdash_{\overline{\alpha}[\nu]\cdot A} \Delta}\ \nu\text{-Cons}$$

$$\frac{\begin{array}{c} X = \mu x.P(x) \mid \Gamma, \alpha : X, \Gamma' \vdash_X \Delta \\ \hline \vdots \\ \hline \Gamma, \alpha : P(X), \Gamma' \vdash_{A(X)} \Delta \end{array}}{\Gamma, \alpha : \mu x.P(x), \Gamma' \vdash_{\alpha[\mu]X\cdot A(X)} \Delta}$$

$$\frac{\begin{array}{c} X = \nu x.Q(x) \mid \Gamma \vdash_X \Delta, X, \Delta' \\ \hline \vdots \\ \hline \Gamma \vdash_{A(X)} \Delta, Q(X), \Delta' \end{array}}{\Gamma \vdash_{\beta[\mu]X\cdot A(X)} \Delta, \beta : \nu x.Q(x), \Delta'}$$

Expressiveness of datatypes

Adding datatypes increases expressiveness dramatically!
One can defne the Burroni natural numbers by:

$$\mathbb{N}(A) = \mu X.A + X$$

Having the Burroni natural numbers means:

- All primitive recursive functions on the natural numbers are present (Pare and Roman)!
- The decision problem for equality of maps is undecideable.

Conclusions ...

Was this carving in snow!!???

- Logic of products and coproducts precisely describes communication on a channel.
- Polycategories (the logic of cut) and polyadditives model communication on many channels.
- Multiplicatives given by representability.
- Protocols given by datatypes.
- *There is no choice!*

SO linearly distributive categories with products and coproducts provide a basic setting for the semantics of communication ... (mutually recursive) datatypes give sophisticated communication protocols.

Conclusions ...

Mathematics had something to say ...

... and the foundations of the mathematics involved was not exactly new!

More conclusions ...

Is that all? NO!

- Message passing ...
- What can you do with protocols ...
- Details of implementing these interactions ...
- A formal programming language for communicating processes which has a guarantee of no deadlock, no livelock
  ... *this will be very neat!!*

  Looking for volunteers!