Polar Shuffles

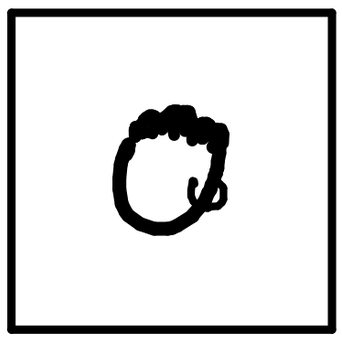
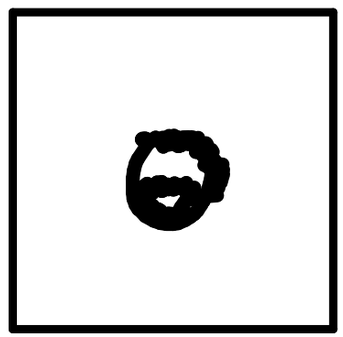
Mario Román inc. j.w.w. Matt Earnshaw, Chad Nester

Tallinn University of Technology

March 18th; chess.

ERC Blast project.
EU Estonian IT Academy.

Polar Interleavings for Deadlock-Free Message Passing

Matt Earnshaw, Chad Nester, Mario Román

Tallinn University of Technology

University of Oxford

Motivation

What are the fundamental structures of concurrency?

Abramsky.

Motivation

A fundamental structure for message passing: message theories.

1. Message theories can be freely constructed over a sym.mon.cat.

2. Message theories are algebras of a universal operad.

   algebras of the freely polarized normal monoidal sym. multicat.

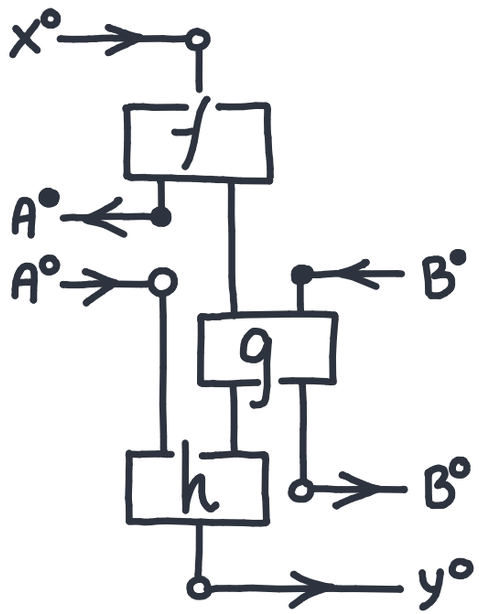
3. Have a concurrency-style internal language.

```
Protocol(request°, login°, get°) {
  Auth(login°, ok°, ack°);
  Serv(ok°, ack°, request°, get°);
}
```

Part 1 : Send / Receive Duality.

CORNERINGS

The horizontal category of the free single object proarrow equipment* represents communicating processes.

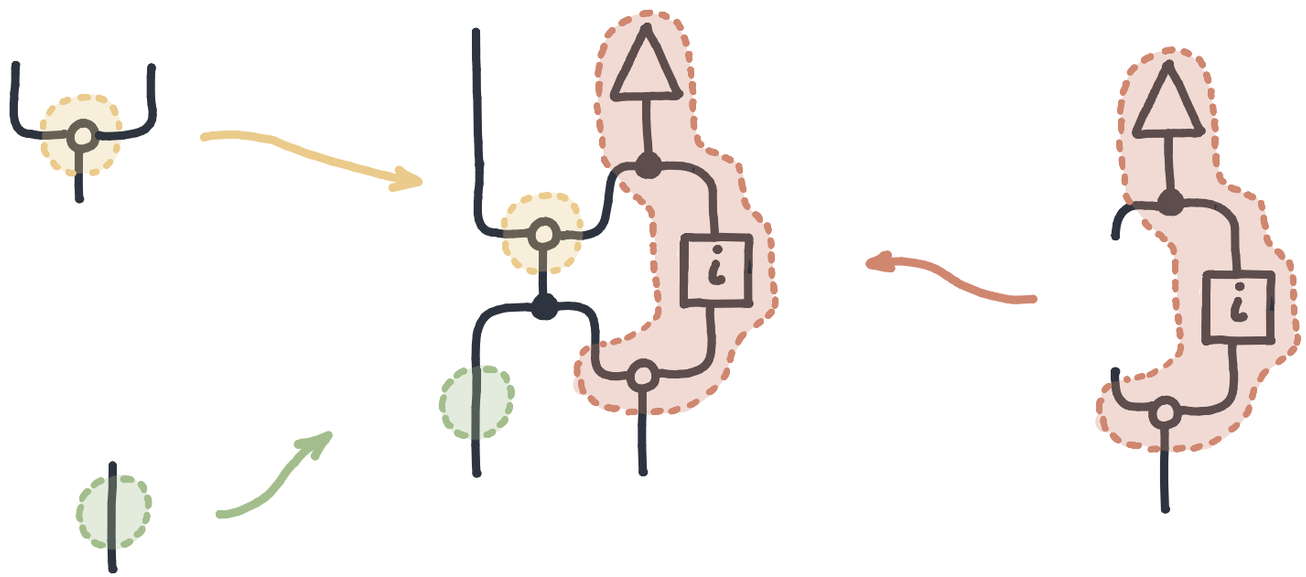


*With pinwheels.

Nester

CORNERINGS

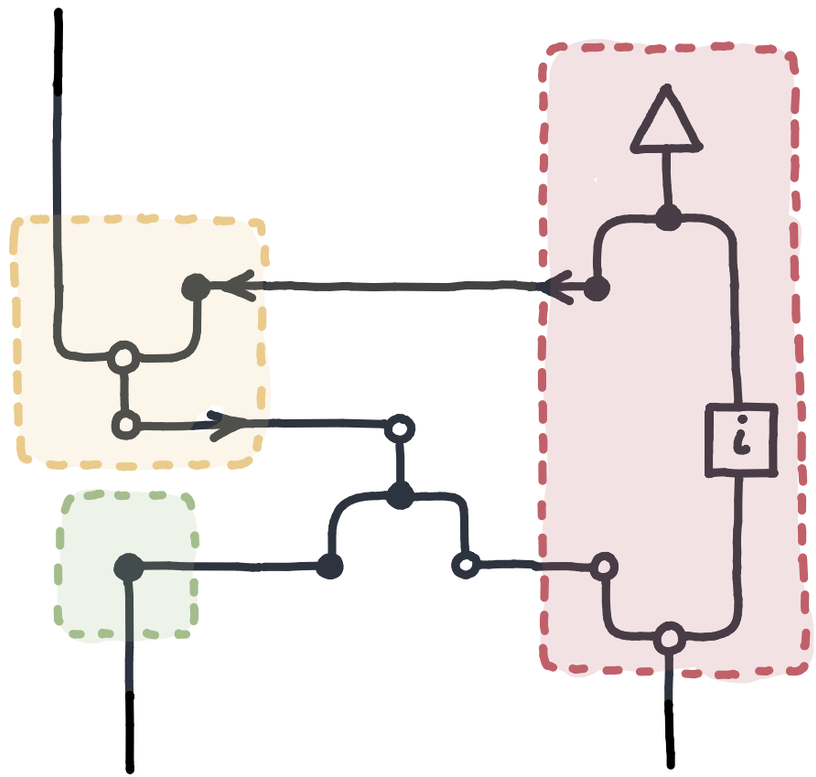
Cornerings allow us to split monoidal morphisms along non-(parallel/sequential) decompositions.



Broadbent, Karvonen.

Cornerings

Cornerings allow us to split monoidal morphisms along non-(parallel/sequential) decompositions.

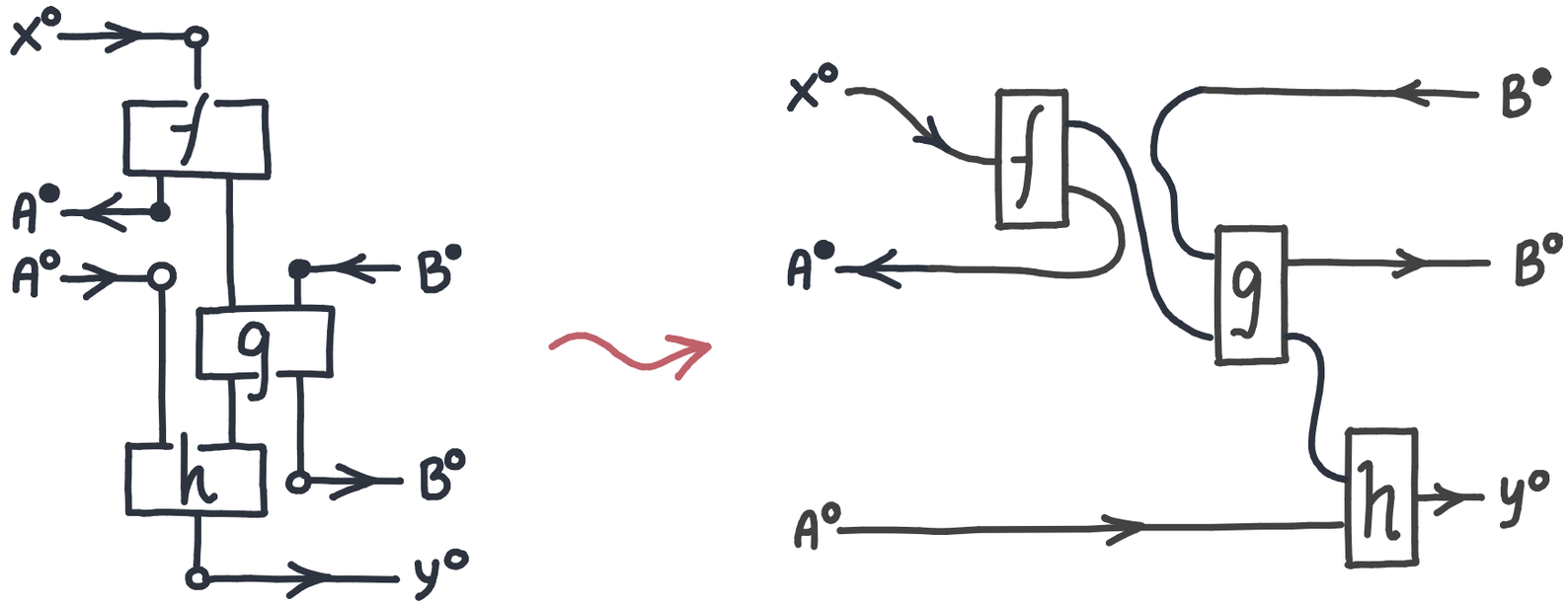


Broadbent, Karvonen.

Nester

Smooth Cornerings

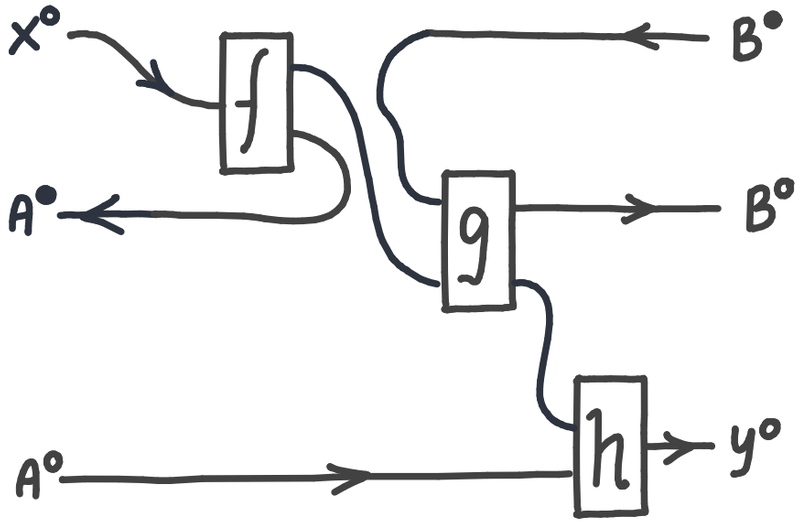
The horizontal category of the free cornering is the original monoidal category plus freely added dualities.



Nester

Smooth Cornerings

The freely polarized monoidal category represents communicating processes.



$$\mathbf{MonCat}_{str} \underset{\text{LAdj}}{\overset{\text{Polar}}{\rightleftarrows}} \mathbf{MonCat}_{str}$$

"Polarization is left adjoint to taking left adjoints."

Part 2: Interleaving

Interleaving

Two programs are running concurrently: in how many ways can they interleave?

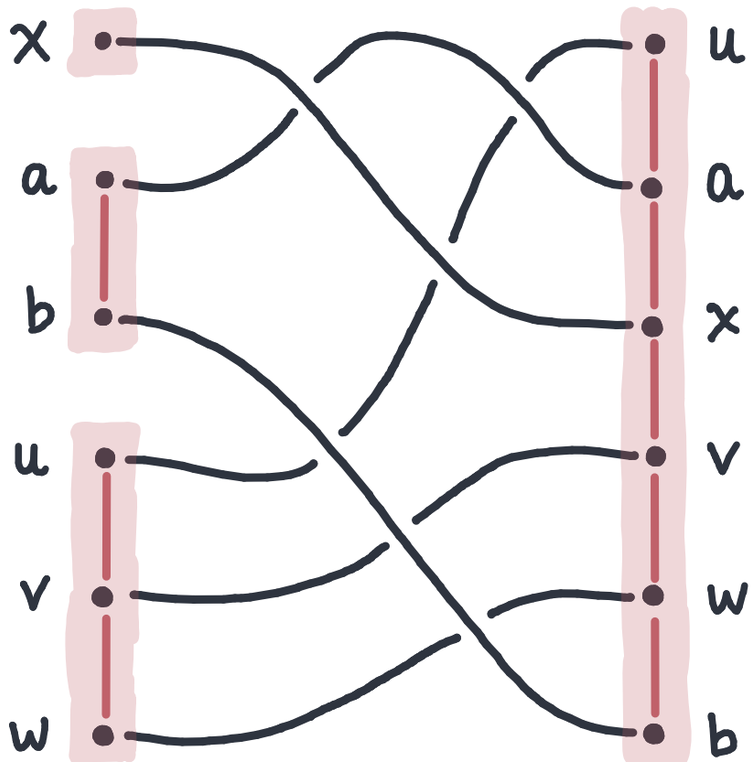
```
alice = do
  x ;
  y ;
  z ;
```

```
bob = do
  p ;
  q ;
```

xyzpq	xpqyz
xypzq	pxyzq
xypqz	pxyqz
xpyzq	pxqyz
xpyqz	pqxyz

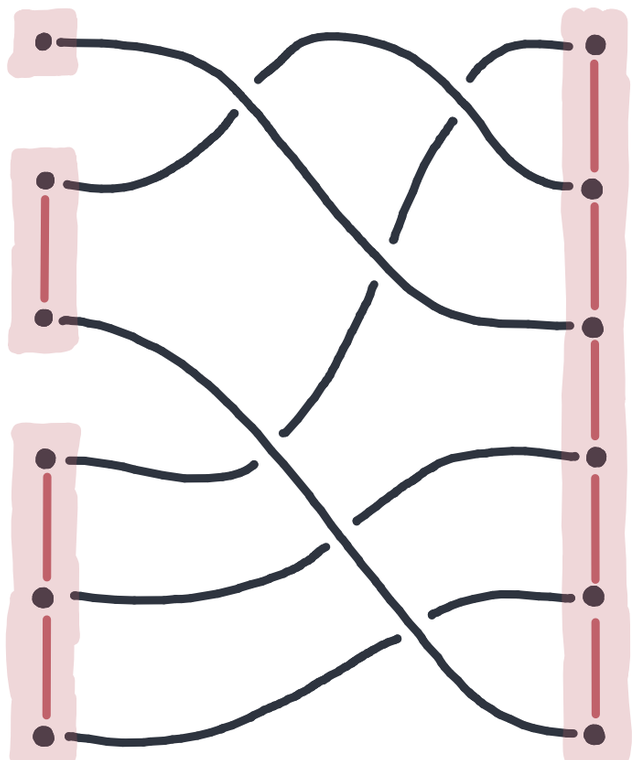
Eilenberg, Maclane.

Shuffles



Mix some words, preserving the relative order inside the words.

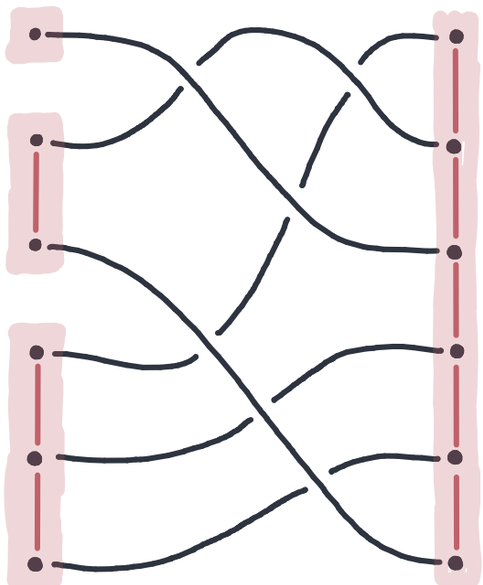
Shuffles

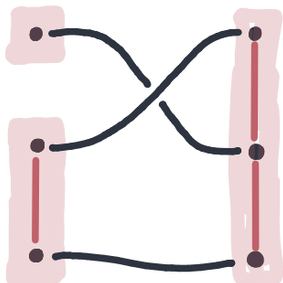


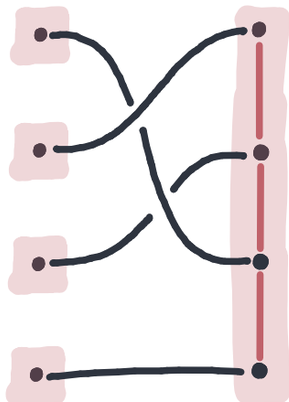
Invariant to renaming, defined up to α-equivalence.

Shuffles

$$\mathrm{Shuf}(n_1, \ldots, n_k) = \frac{(n_1 + \ldots + n_k)!}{n_1! \cdots n_k!} ;$$







Shuffles are a rich combinatorial structure.

Part 3 : Interlude - Duoidals

Duoidal Categories

Duoidal categories have two tensors and one distributes over the other. We interpret

- $X \triangleleft Y$, sequential tensor, "X, and then Y";
- $X \otimes Y$, parallel tensor, "X and Y at the same time";

$$(X \triangleleft Y) \otimes (U \triangleleft V) \longrightarrow (X \otimes U) \triangleleft (Y \otimes V).$$

When the unit distributor is an isomorphism, $I \xrightarrow{\simeq} N$, it is normal. A normal, $\otimes$-symmetric duoidal, is a physical duoidal.

Aguiar & Mahajan '10, Shapiro & Spivak '23.

Duoidal Categories

Duoidal categories are not coherent, there are two formal maps

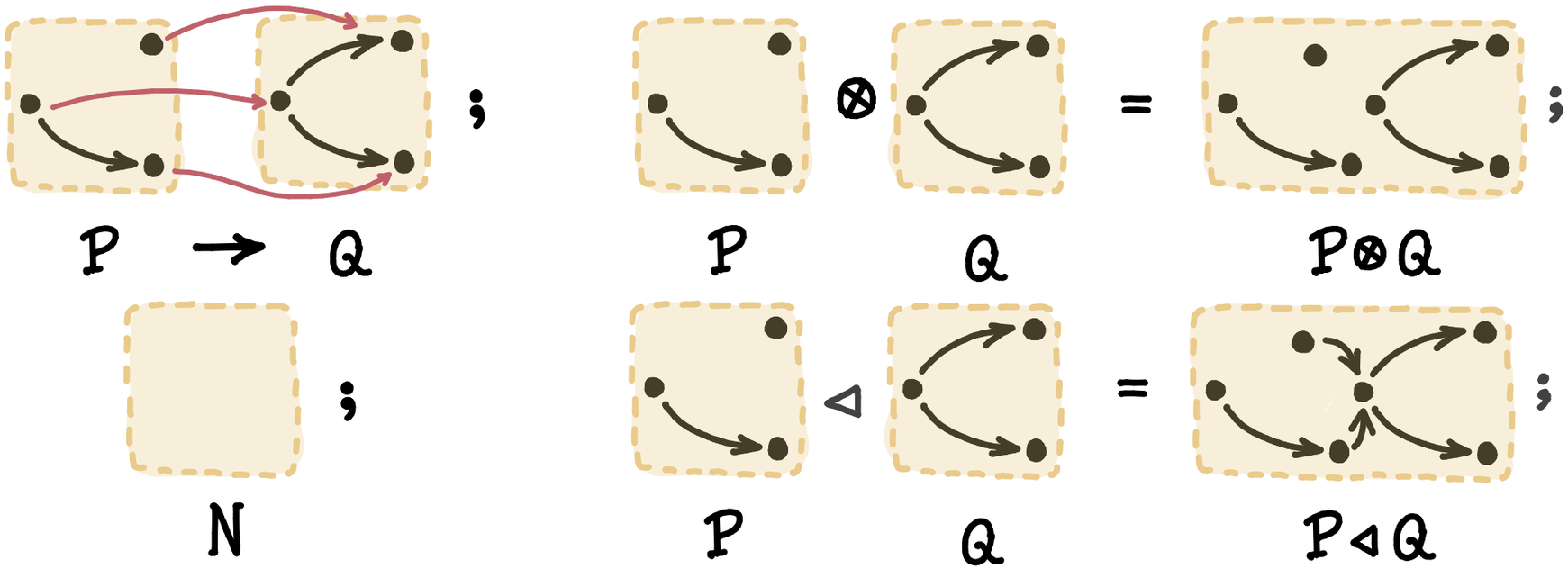
$$I \triangleleft I \to I.$$

However, physical duoidal categories are coherent. There is at most a single morphism between any two objects where every type appears exactly once with each variance.

$$A \otimes (B \triangleleft C) \rightleftarrows (A \otimes B) \triangleleft C.$$

Physical Duoidal Categories

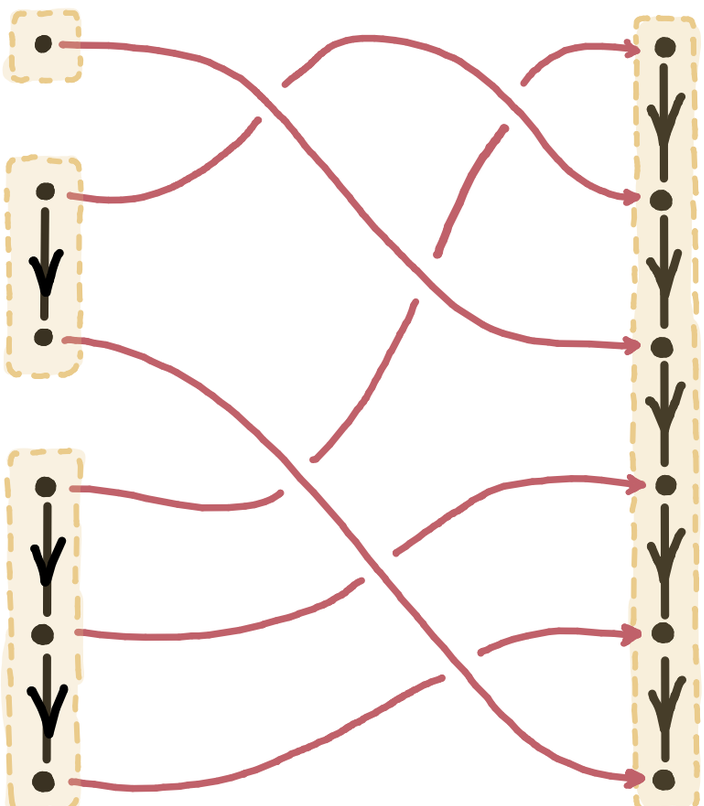
The free physical duoidal category over a single object is fully-faith. into posets and bijective-on-objects inclusions.



Grabowski '81, Gischer '88, Shapiro & Spivak '23

Physical Monoidal Multicategories

When only sequential composition is representable, we recover shuffles as poset inclusions.



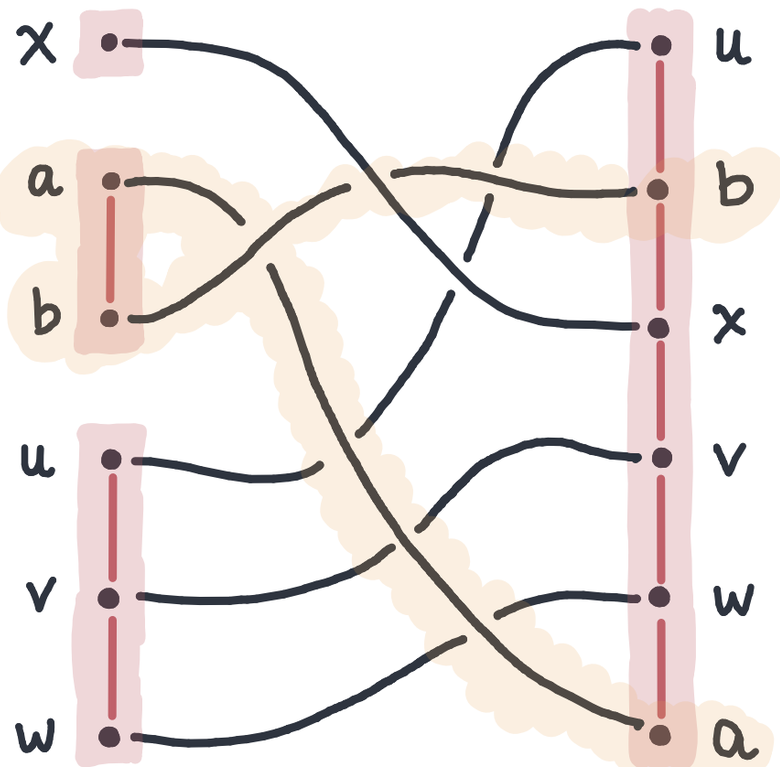
THEOREM. Shuffles form the free physical monoidal multicategory over a single object.

Part 4: Polar Shuffles

Send
Receive

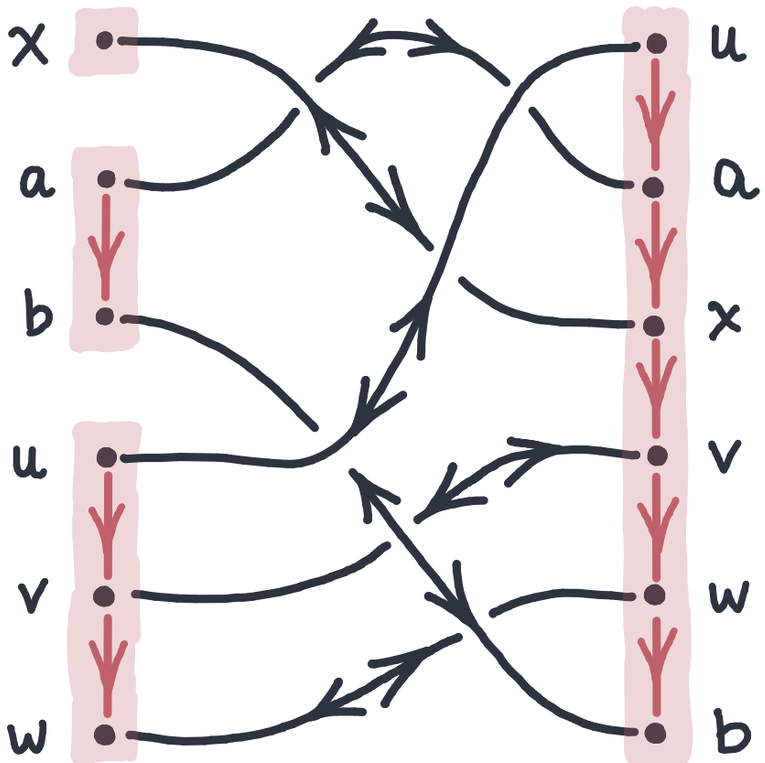
Interleave

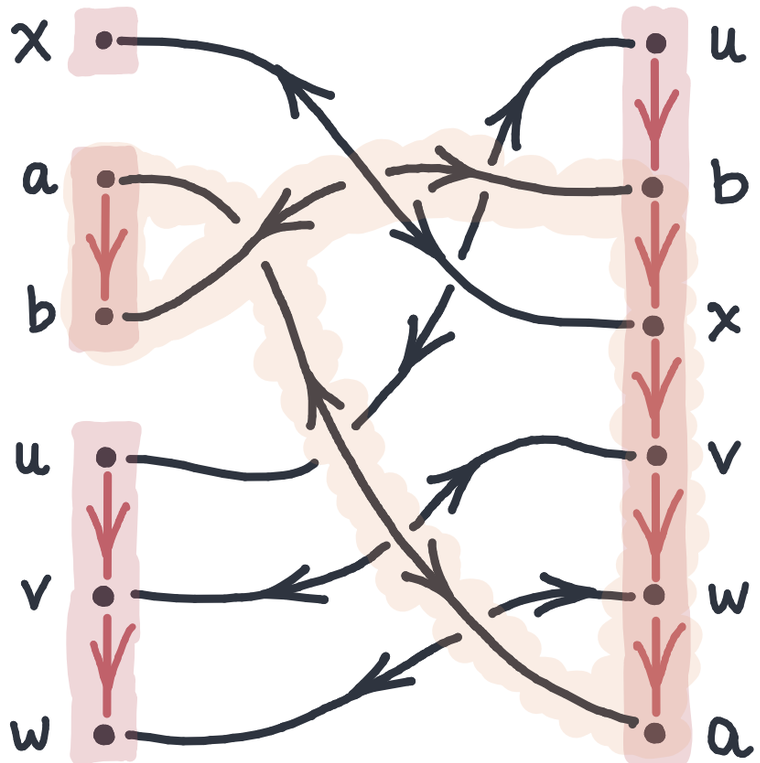
Shuffles



This is not a shuffle: "a" and "b" change order.

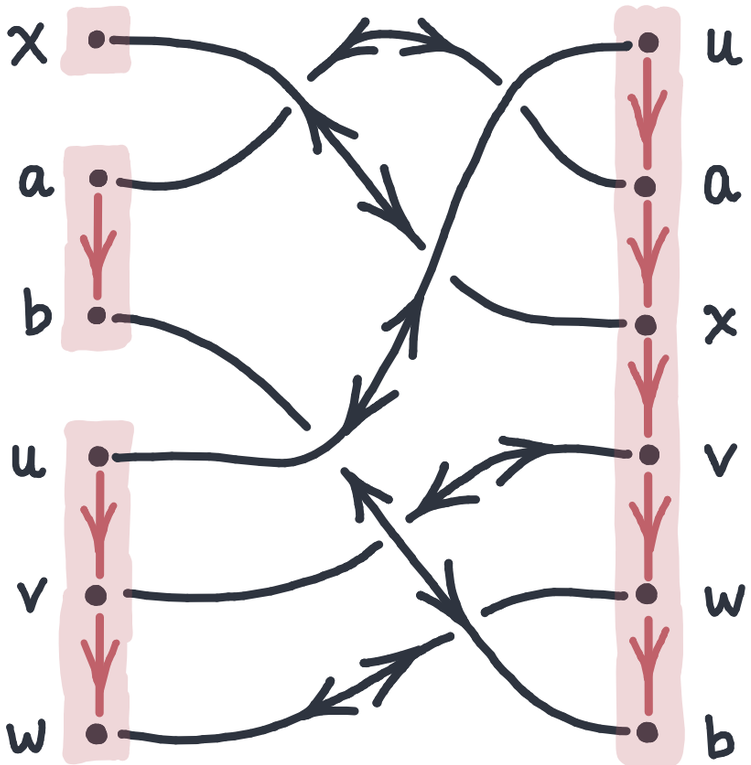
Shuffles





Valid shuffles are acyclic. Invalid shuffles contain cycles.

SHUFFLES



DEFINITION. A shuffle is a bijection

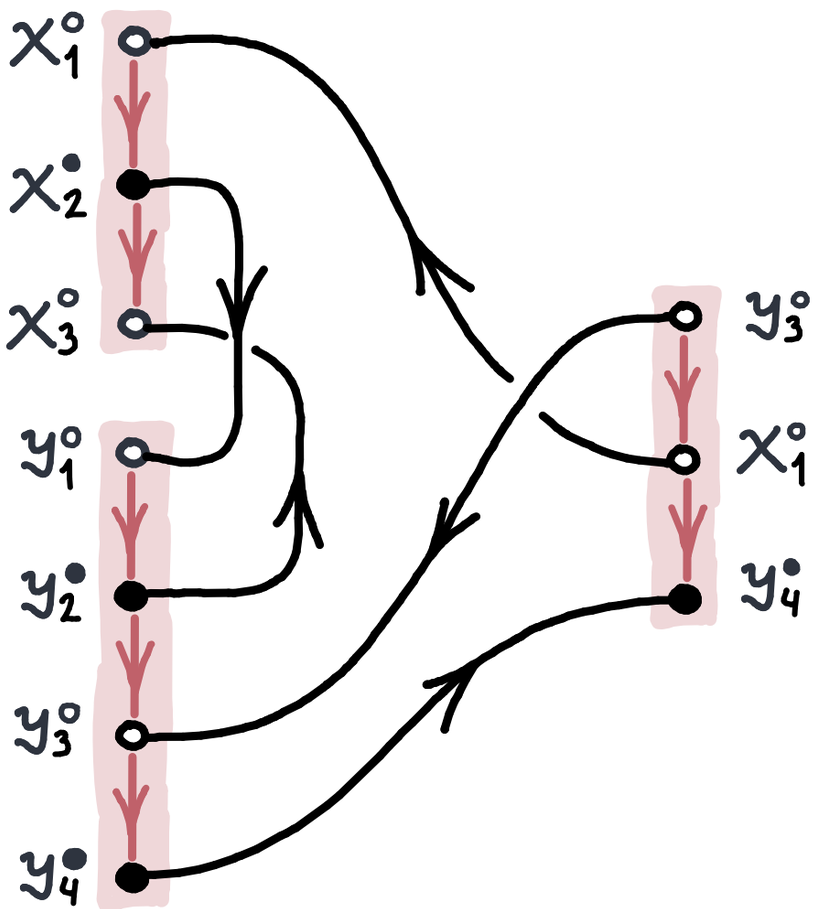
$$f : A_1 + \cdots + A_n \longrightarrow X$$

such that the edges from relative orders (↓) and the edges from the bijection

$$(x \bullet\!\!\rightarrow\!\!\bullet f(x)) \qquad (f(x) \bullet\!\!\rightarrow\!\!\bullet x)$$

form an acyclic graph.

Polar Shuffles



DEFINITION. Polar shuffles are bijections

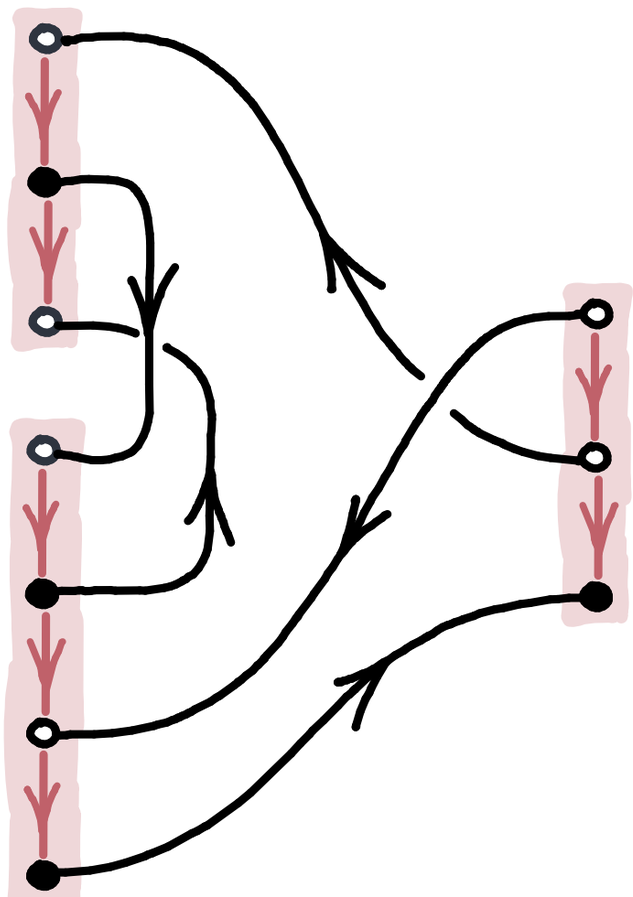
$$A_1^\bullet + \cdots + A_n^\bullet + X^\circ \longrightarrow A_0^\circ + \cdots + A_n^\circ + X^\bullet$$

such that the edges from relative orders (↓) and the edges from the bijection

$$(x \bullet\!\!\rightarrow\!\!\bullet f(x))$$

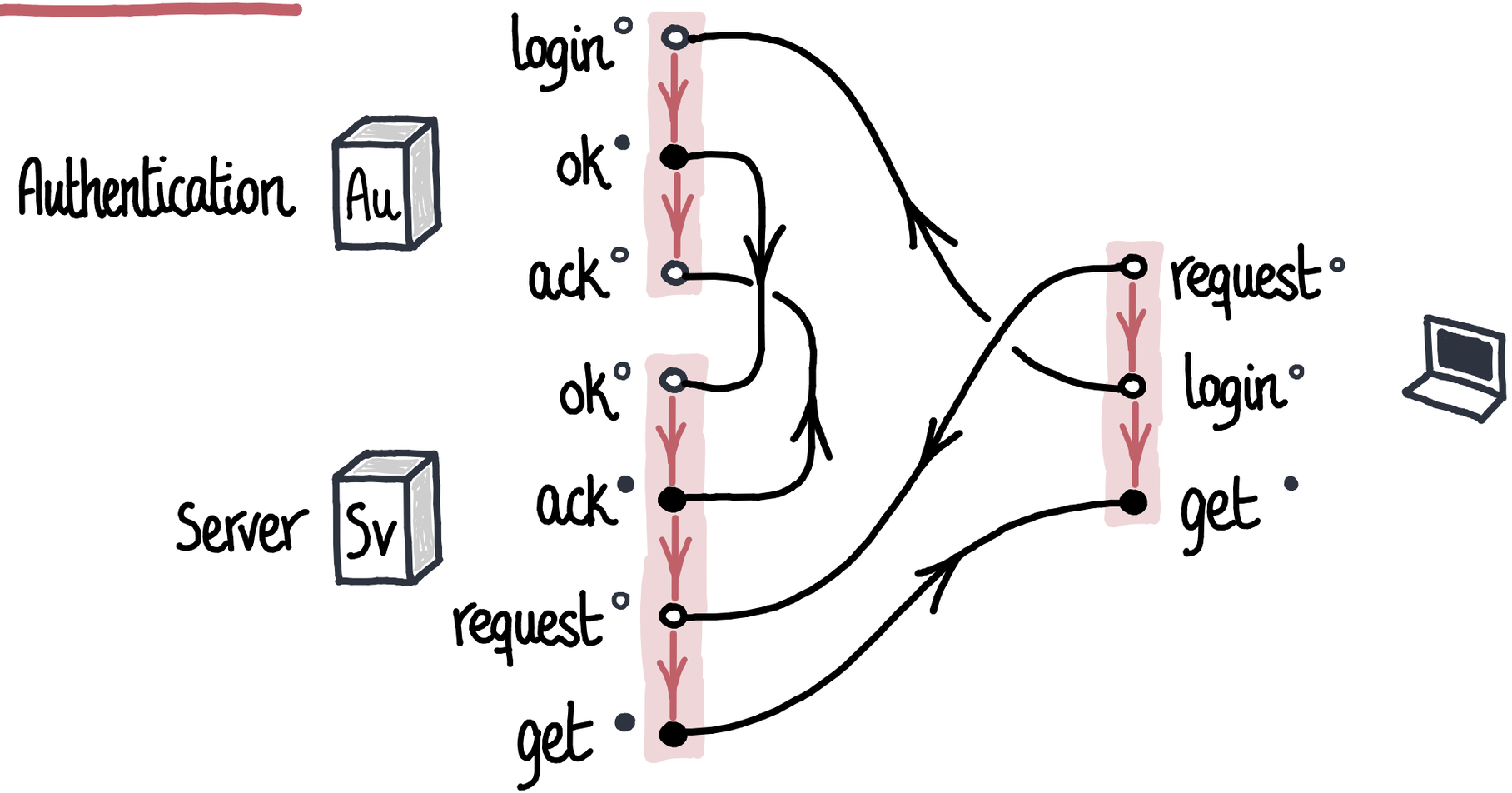
induce an acyclic graph.

Polar Shuffles



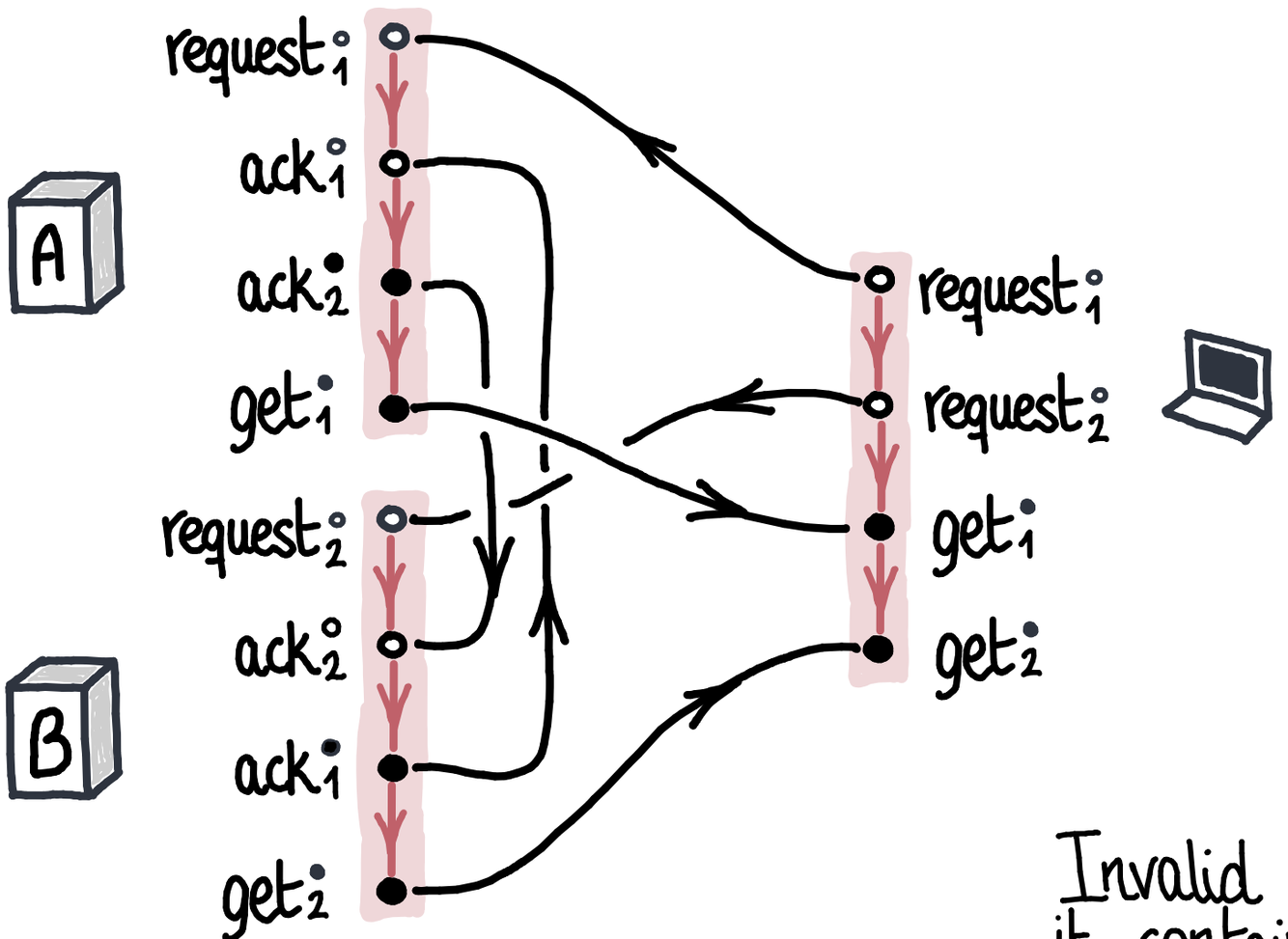
Polar shuffles mix programs that communicate sending• and receiving°.

Polar Shuffles



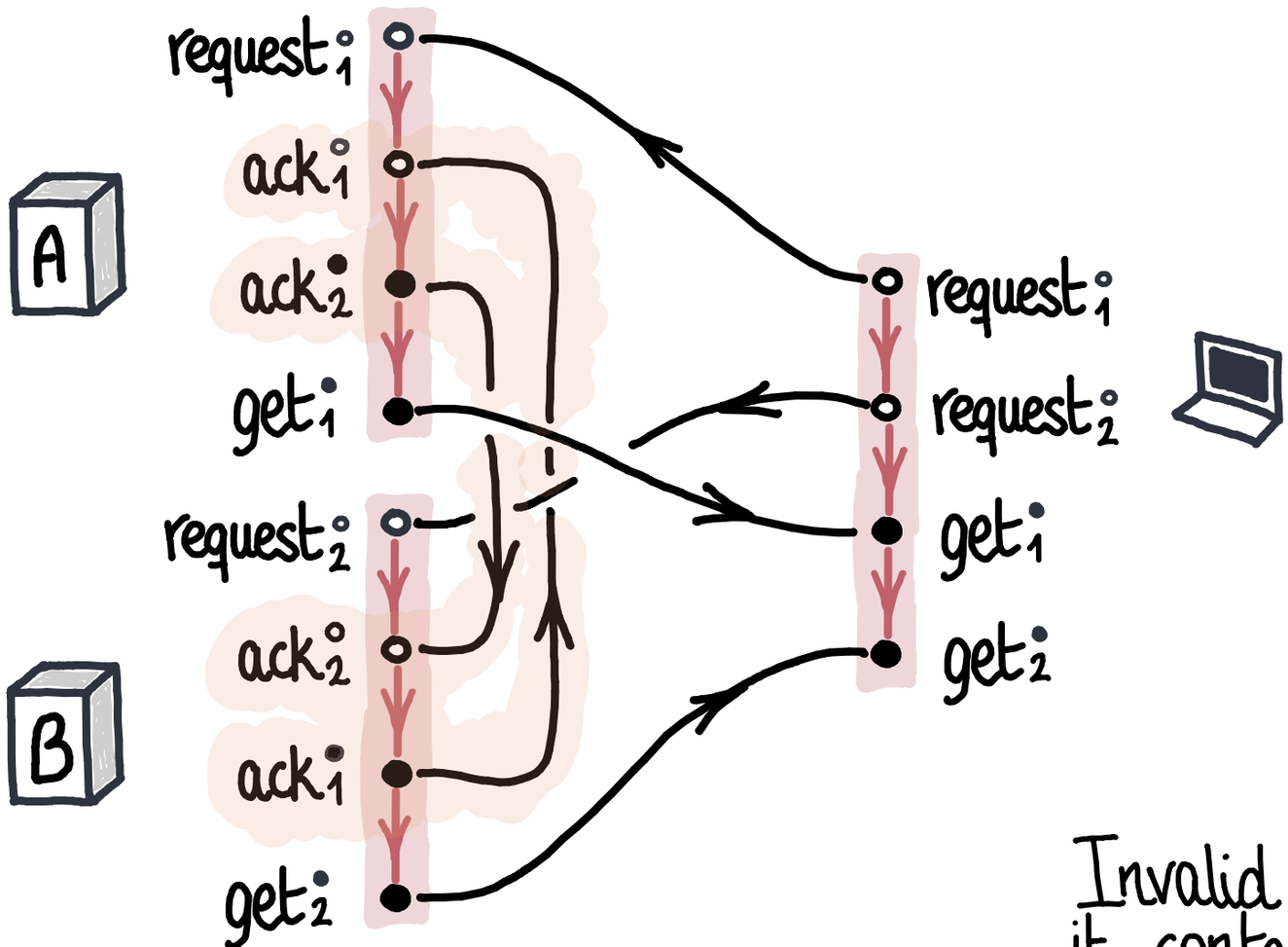
Each polar shuffle determines a way programs could interleave their messages.

Polar Shuffles



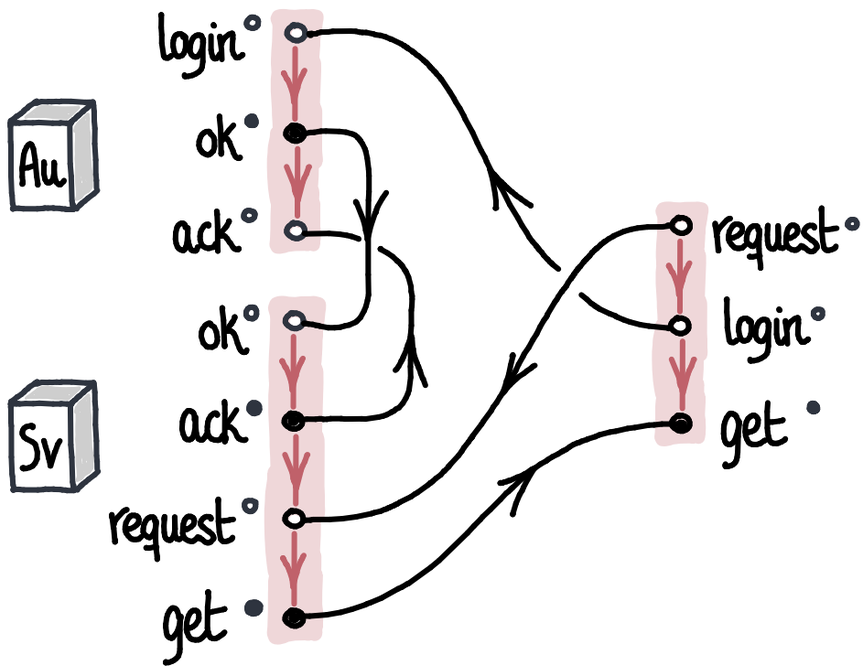
Invalid if and only if it contains a deadlock.

Polar Shuffles



Invalid if and only if it contains a deadlock.

Polar Shuffles




```
Protocol(request°, login°, get•) {
  Auth(login°, ok•, ack°);
  Serv(ok°, ack•, request°, get•);
}
```

Part 5: Message Theories

Message Theories

Set of types, representing resource types: $X, Y, Z, \ldots$

Two actions for each resource: send and receive,

$X^\bullet$ means "send X".
X° means "receive X".

Lists of actions represent sequencing of the actions.

$\Gamma = X^\circ, Y^\bullet, Z^\bullet, W^\circ$ means "ask for X; send Y and then Z; finally, receive W".

Message Theories, a monoidal multicategory.

$$\frac{}{\varepsilon}\,\text{NOP} \qquad \frac{}{X^\circ, X^\bullet}\,\text{ECHO} \qquad \frac{\Gamma, X^\bullet, X^\circ, \Delta}{\Gamma, \Delta}\,\text{LNK} \qquad \frac{\Gamma_1 \quad \dots \quad \Gamma_n}{[\Gamma_1, \dots, \Gamma_n]_\sigma}\,\text{SHF}_\sigma$$

1. Doing nothing is a session.
2. We can create a receive-send "echo" session.
3. We can link a SEND to a RECEIVE port.
4. Events can be interleaved in any order.

Message Theories

Definition. A message theory $\mathbb{M}$ consists of a set of types, $\mathbb{M}_{\text{obj}}$, and, for each list of polarized types, a set "of sessions" typed by that list.

$\mathbb{M}(X_1^{\ominus_1}, \ldots, X_n^{\ominus_n})$, for any list of objects $X_i \in \mathbb{M}_{\text{obj}}$ and $\ominus_i \in \{\circ, \bullet\}$.

A message theory must contain (reasonably axiomatized) operations for

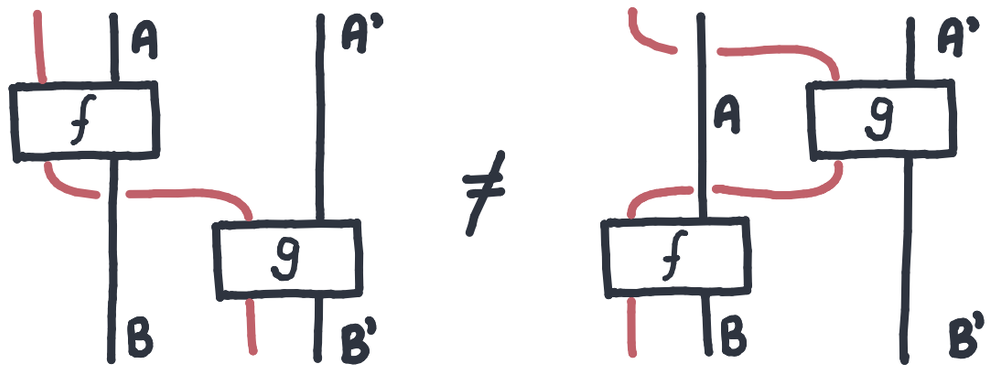
i. binary shuffling, $\textsc{shf}_\sigma : \mathbb{M}(\Gamma) \times \mathbb{M}(\Delta) \to \mathbb{M}(\sigma(\Gamma, \Delta))$;
ii. a no-operation, $\textsc{nop} : \mathbb{M}()$;
iii. a receive to send channel, $\textsc{spw}_X^\Gamma : \mathbb{M}(\Gamma, \Delta) \to \mathbb{M}(\Gamma, X^\circ, X^\bullet, \Delta)$;
iv. linking a send to receive channel, $\textsc{lnk}^\Gamma : \mathbb{M}(\Gamma, X^\bullet, X^\circ, \Delta) \to \mathbb{M}(\Gamma, \Delta)$.

Thm. Message theories are algebras for the monoidal operad of shuffles.

Part 6: Process/Message Adj.

PreMonoidal Categories

Premonoidal categories extend monoidal categories with effects.



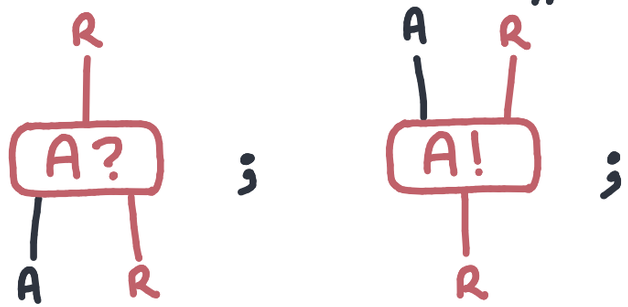
Failure of Interchange

THEOREM. String diagrams with runtime are the internal language of premonoidal categories.

Jeffrey Román

Message Theories vs Process Theories

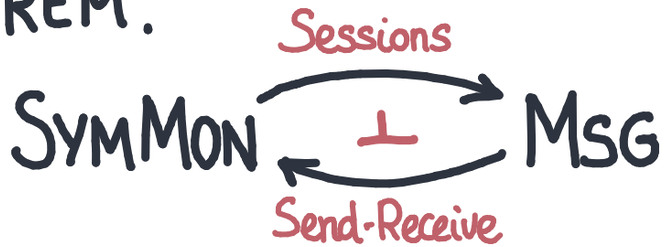
The free message theory **contains** string diagrams extended with "send" and "receive" effects.



The cofree process theory on a message theory has as morphisms the "receive-then-send" sessions:

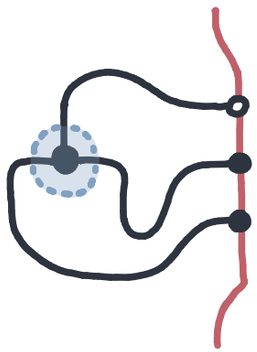
$$M(X^{\circ}, Y^{\bullet}).$$

THEOREM.

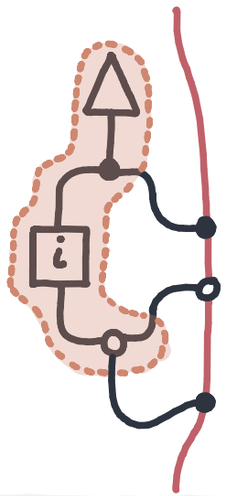


String Diagrams + Send/Receive

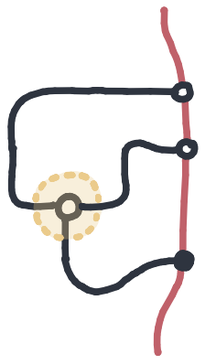
Stage



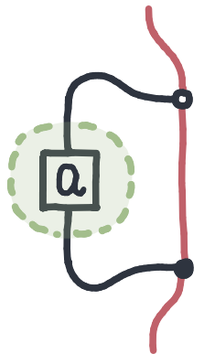
Bob



Alice



Eve




```
stage = do
    x ← get
    send(x)
    send(x)
```

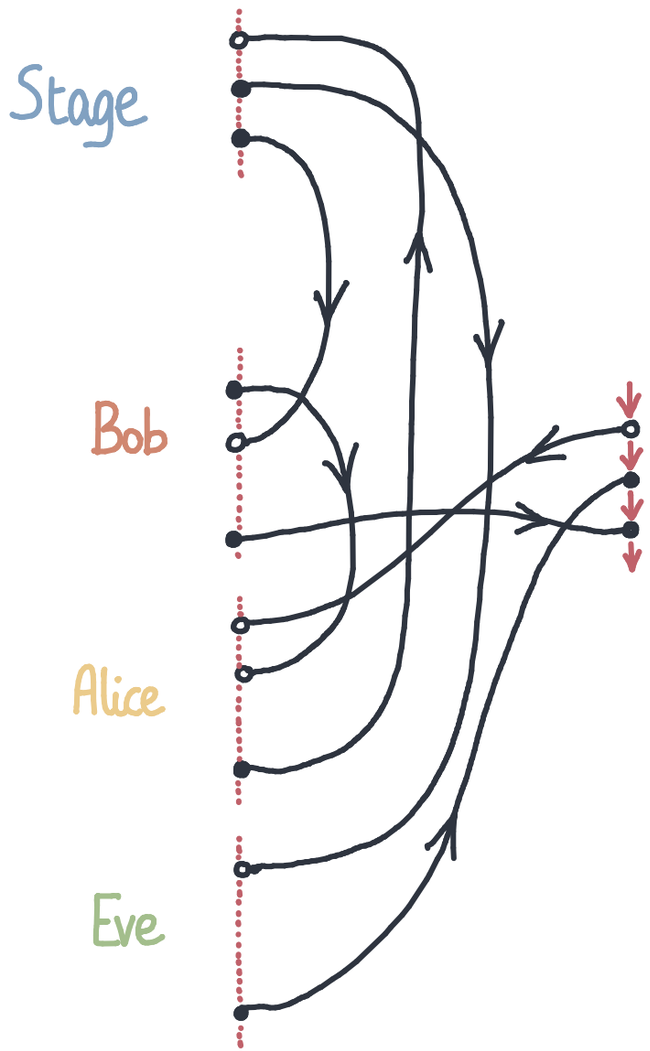
```
bob = do
    k ← random
    send(k)
    c ← get
    send(i(k)⊕c)
```

```
alice = do
    m ← get
    K ← get
    send(m⊕K)
```

```
eve = do
    c ← get
    attack(c)
```

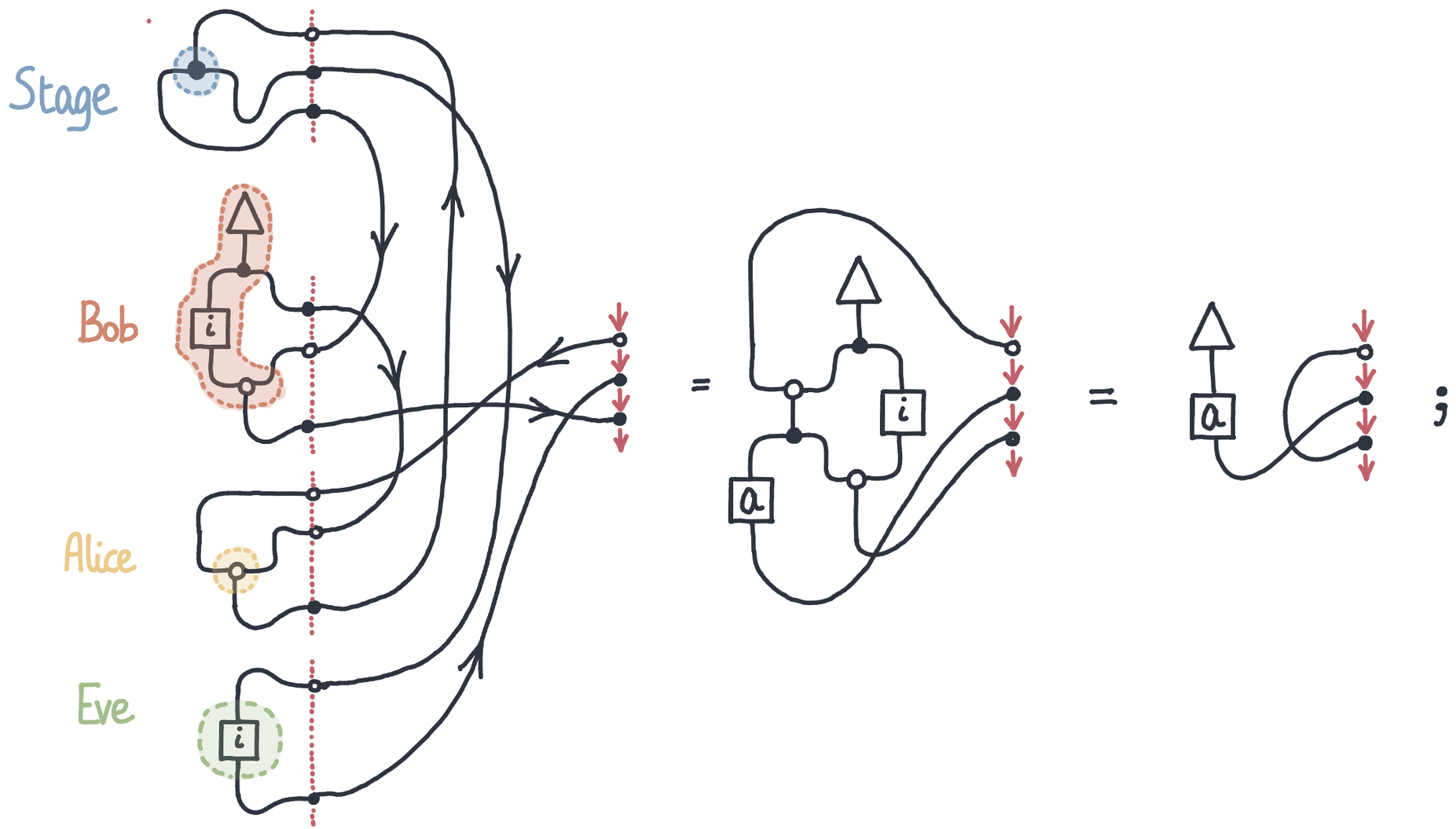
We split a protocol into multiple agents.

Polar Shuffles for Protocols




```
otp (msg°, attack•, decrypt•) {
    stage (crypt°, crypt•1, crypt•2);
    bob (key•, crypt°1, decrypt•);
    alice (msg°, key°, crypt•);
    eve (crypt°2, attack•); }
```

Polar Shuffles for Protocols



Polar Shuffles for Protocols

```
stage = do
    x ← get
    send(x)
    send(x)
```

```
eve = do
    c ← get
    attack(c)
```

```
bob = do
    k ← random
    send(k)
    c ← get
    send(i(k)⊕c)
```

```
alice = do
    m ← get
    k ← get
    send(m⊕k)
```

```
otp (msg°, attack•, decrypt•) {
    stage (crypt°, crypt•1, crypt•2);
    bob (key•, crypt°1, decrypt•);
    alice (msg°, key°, crypt•);
    eve (crypt°2, attack•); }
```

Motivation

A fundamental structure for message passing: message theories.

1. Message theories can be freely constructed over a sym.mon.cat.

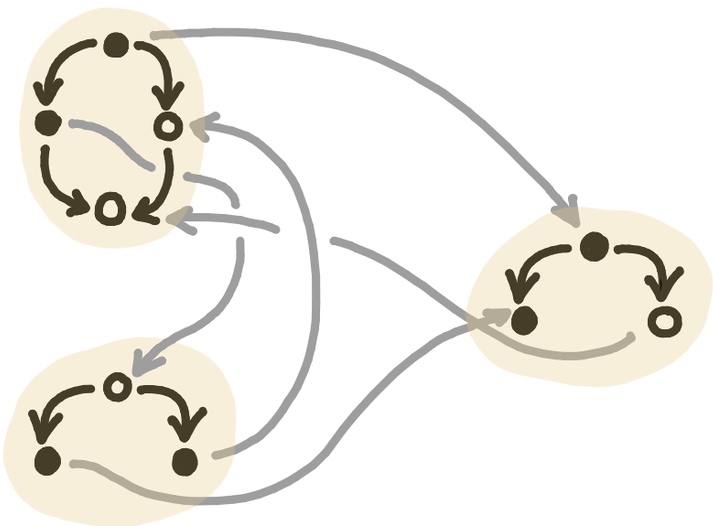
2. Message theories are algebras of a universal operad.

   algebras of the freely polarized normal monoidal sym. multicat.

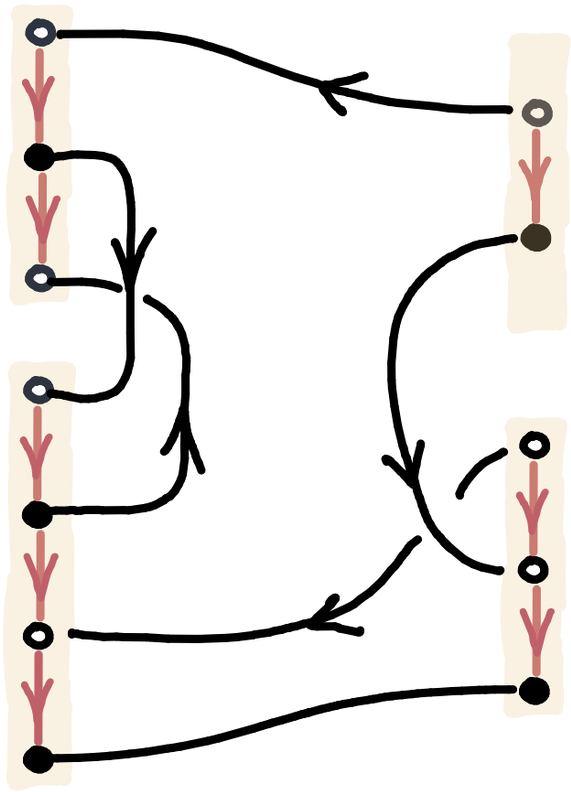
3. Have a concurrency-style internal language.

```
Protocol(request°, login°, get°) {
  Auth(login°, ok°, ack°);
  Serv(ok°, ack°, request°, get°);
}
```

Next Steps



Polar Posets
(c.f. Event Strs.)



*-Polycategory
via Chu Construction

END