Polar Interleavings for Deadlock-Free Message Passing

Matt Earnshaw, Chad Nester, Mario Román

Tallinn University of Technology

Nov 24th, NWPT'23, Västerås.

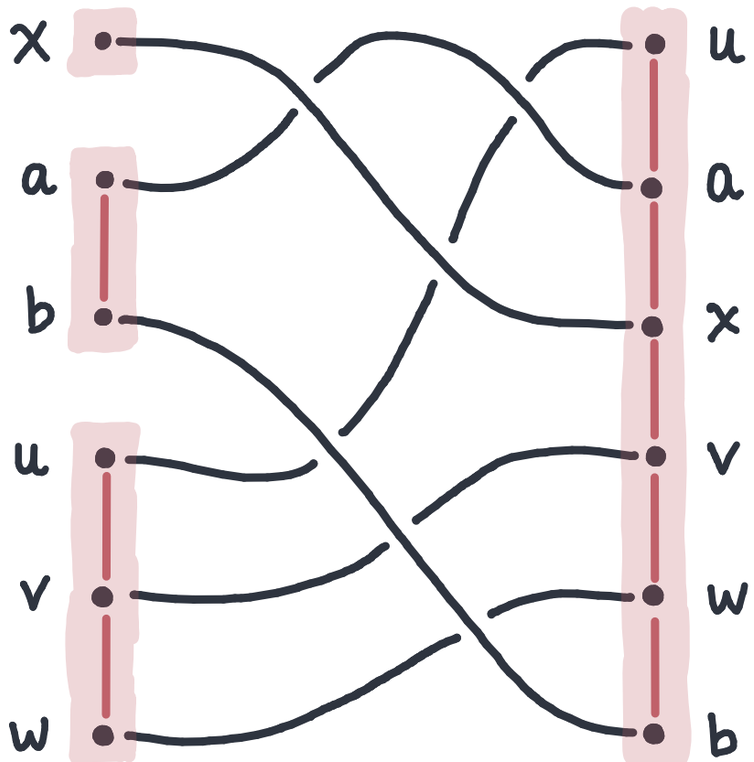
Supported by the EU Estonian IT Academy.

Outline

0. Shuffles
1. Structure: Polar Shuffles.
2. Framework: Monoidal Categories.

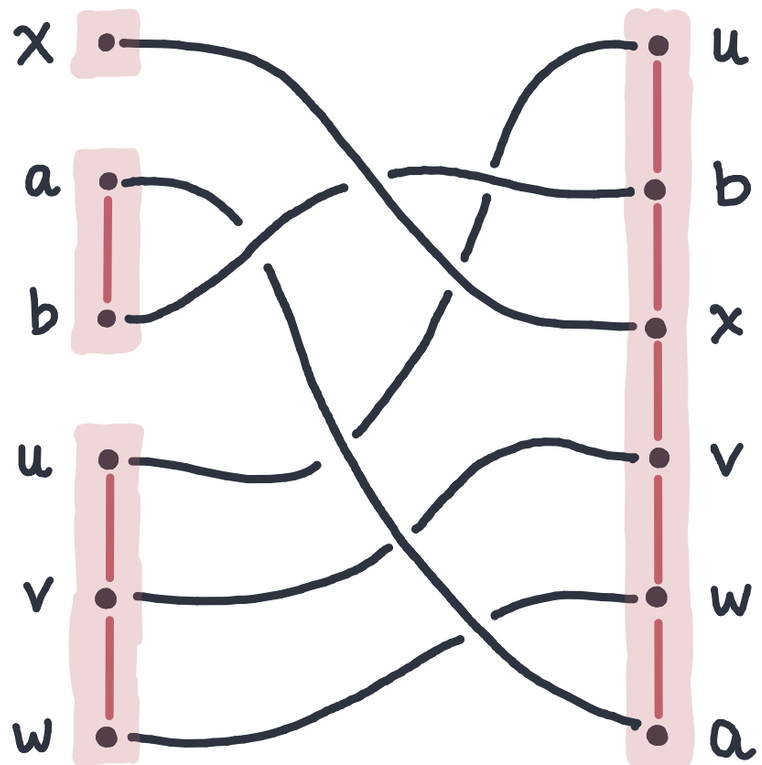
Part 1 : Shuffles

Shuffles



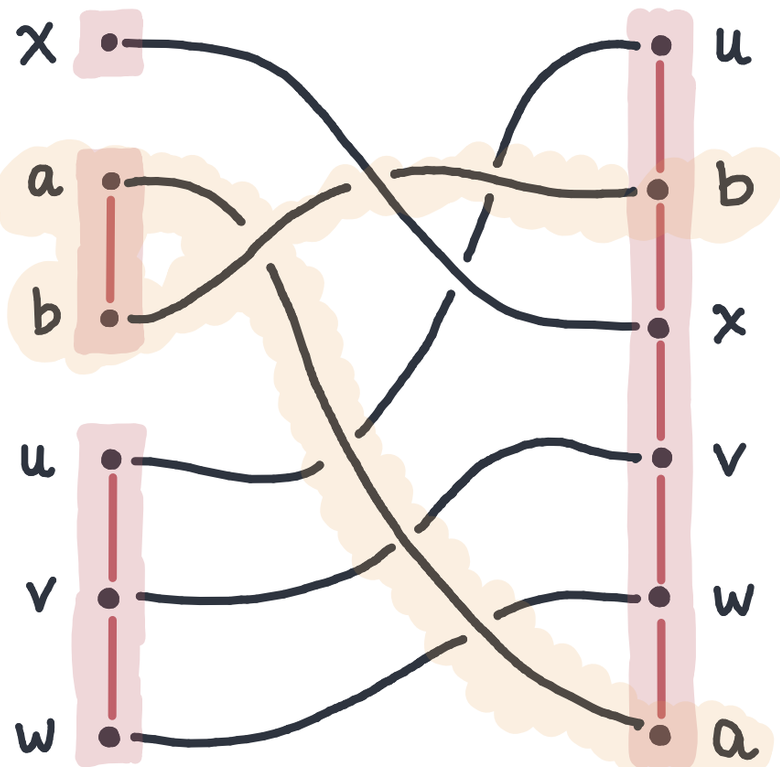
Mix some words, preserving the relative order inside the words.

Shuffles



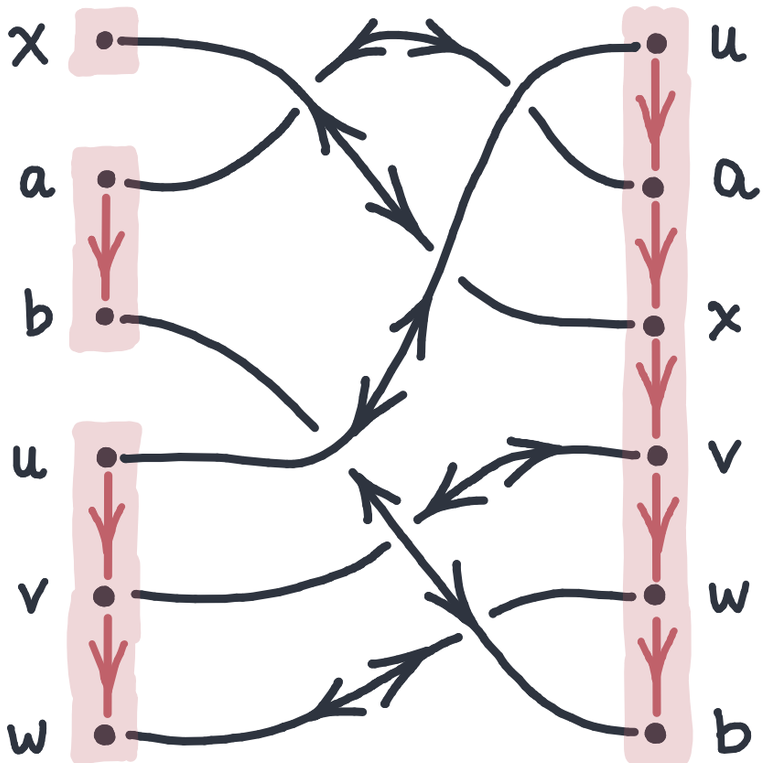
This is not a shuffle: "a" and "b" change order.

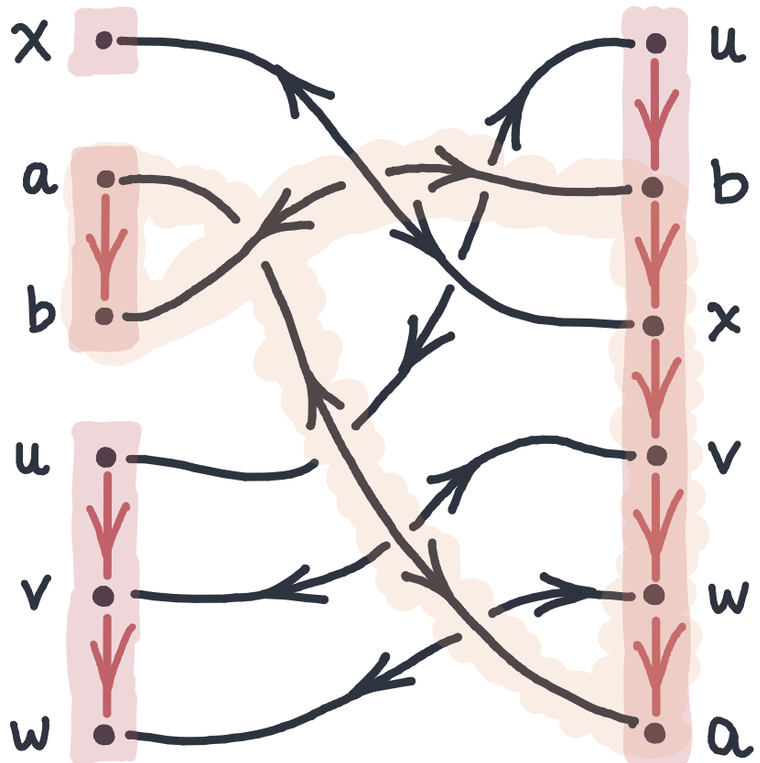
Shuffles



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

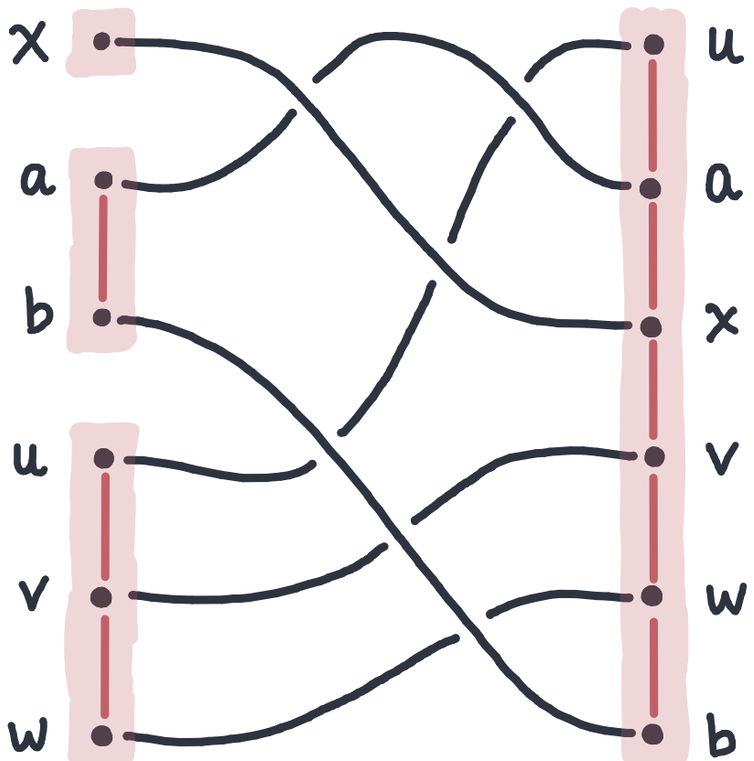
Shuffles





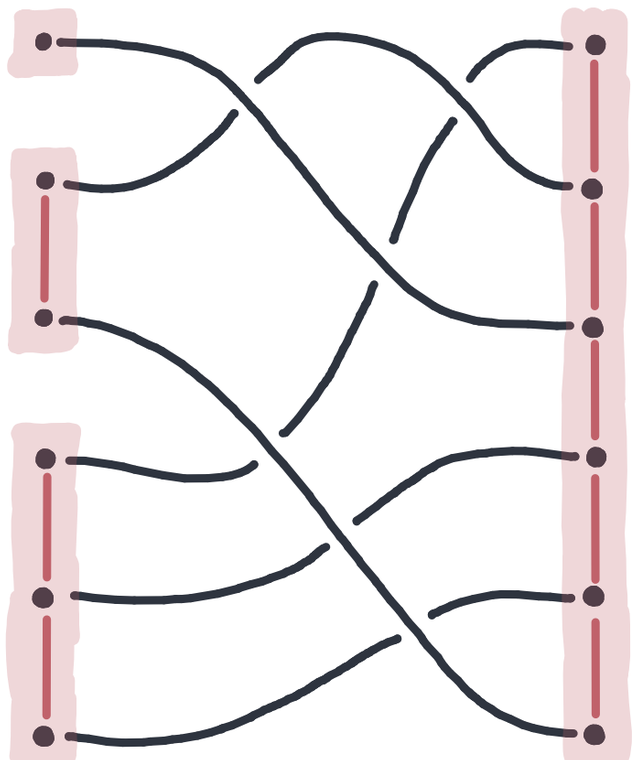
Valid shuffles are acyclic. Invalid shuffles contain cycles.

Shuffles



Mix some words, preserving the relative order.

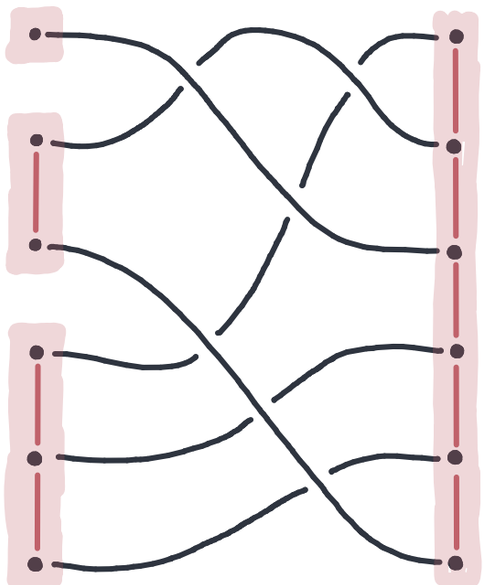
Shuffles

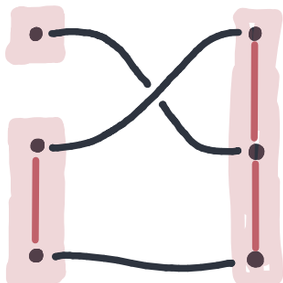


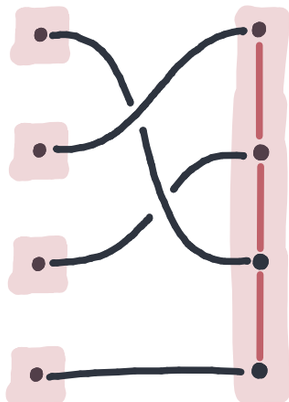
Invariant to renaming, defined up to α-equivalence.

Shuffles

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

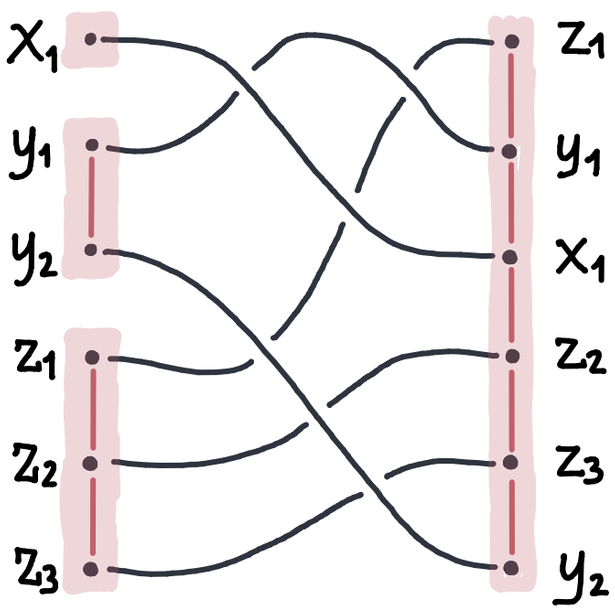


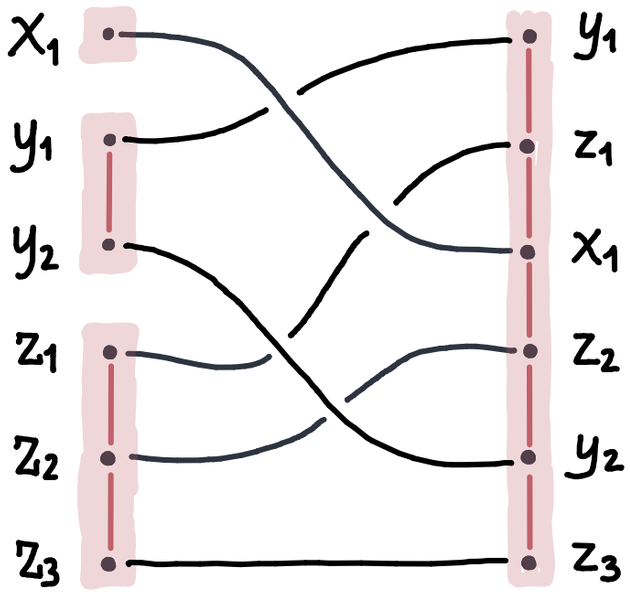


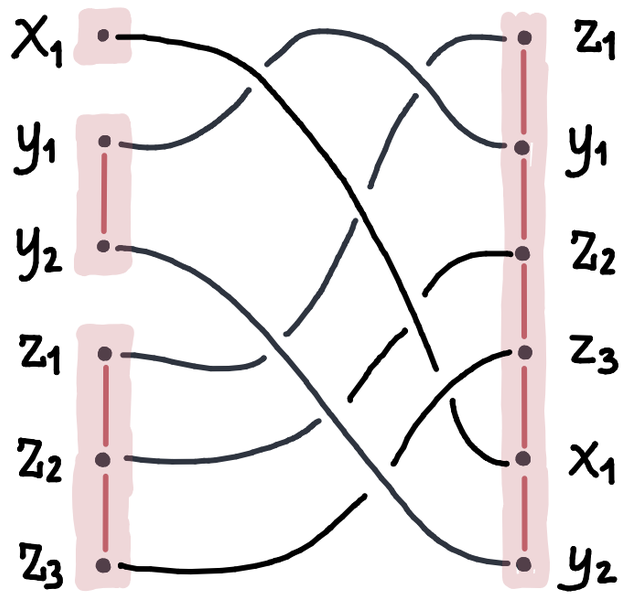


Shuffles are a rich combinatorial structure.

Shuffles







Each shuffle determines a way programs could interleave their traces.

Shuffles

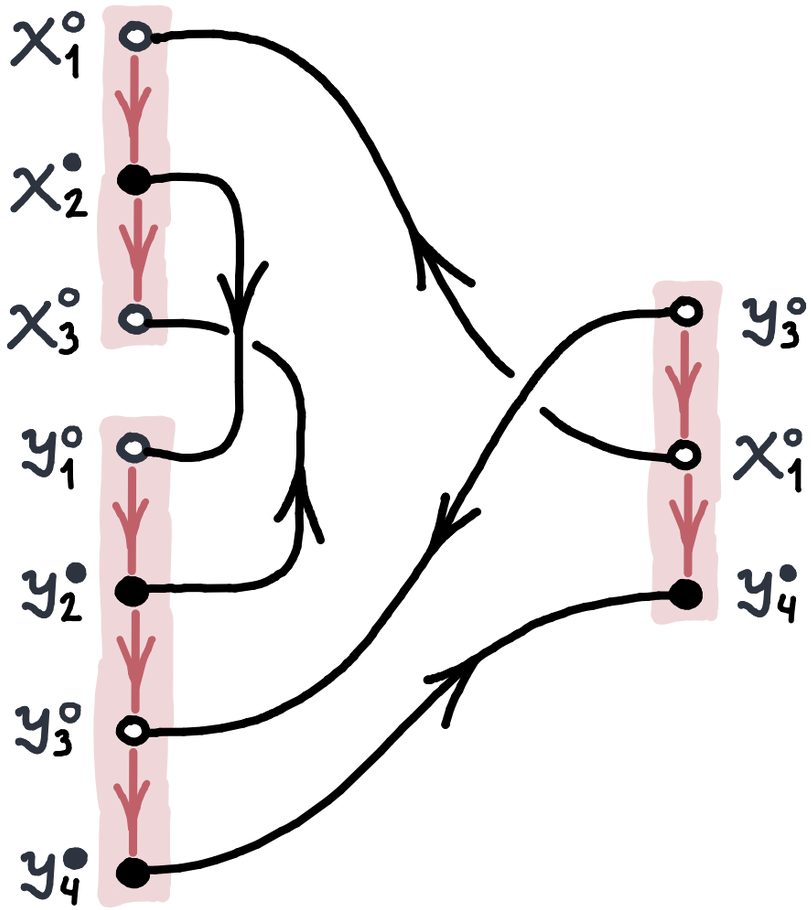
All good if the programs are independent.

What if they communicate?

How can we add message passing?

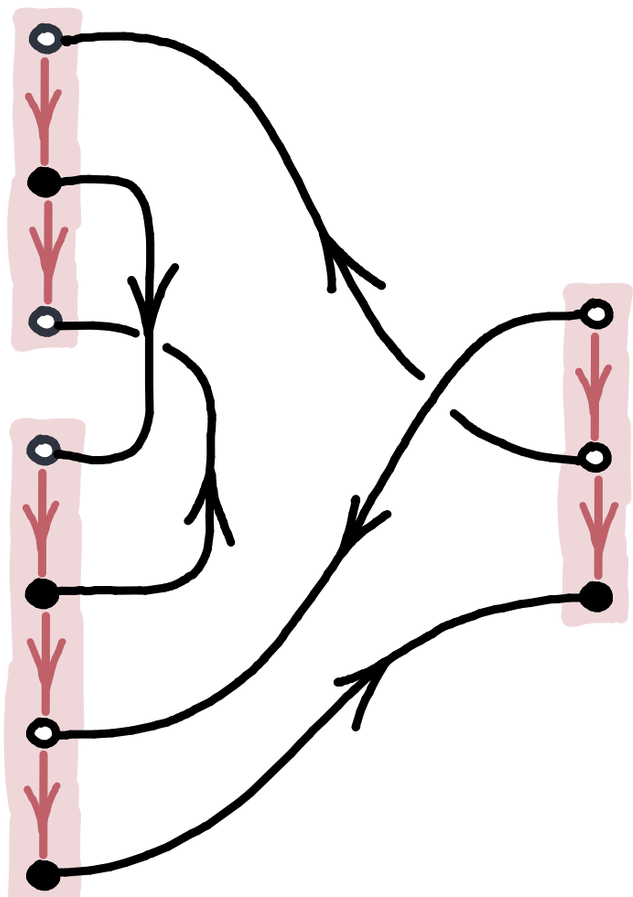
Part 2 : Polar Shuffles

Polar Shuffles



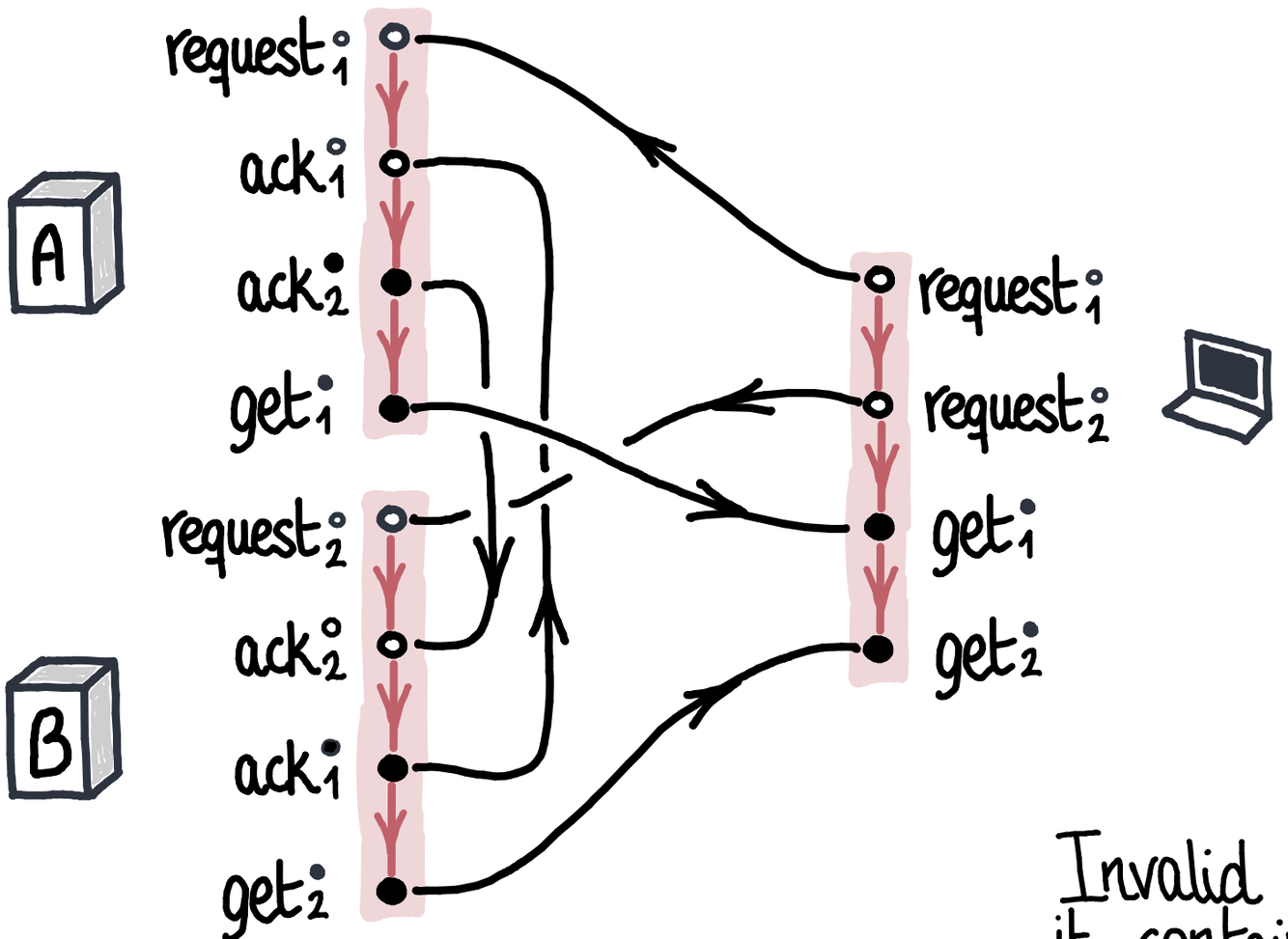
Polar shuffles mix programs that communicate sending$^\bullet$ and receiving$^\circ$.

Polar Shuffles



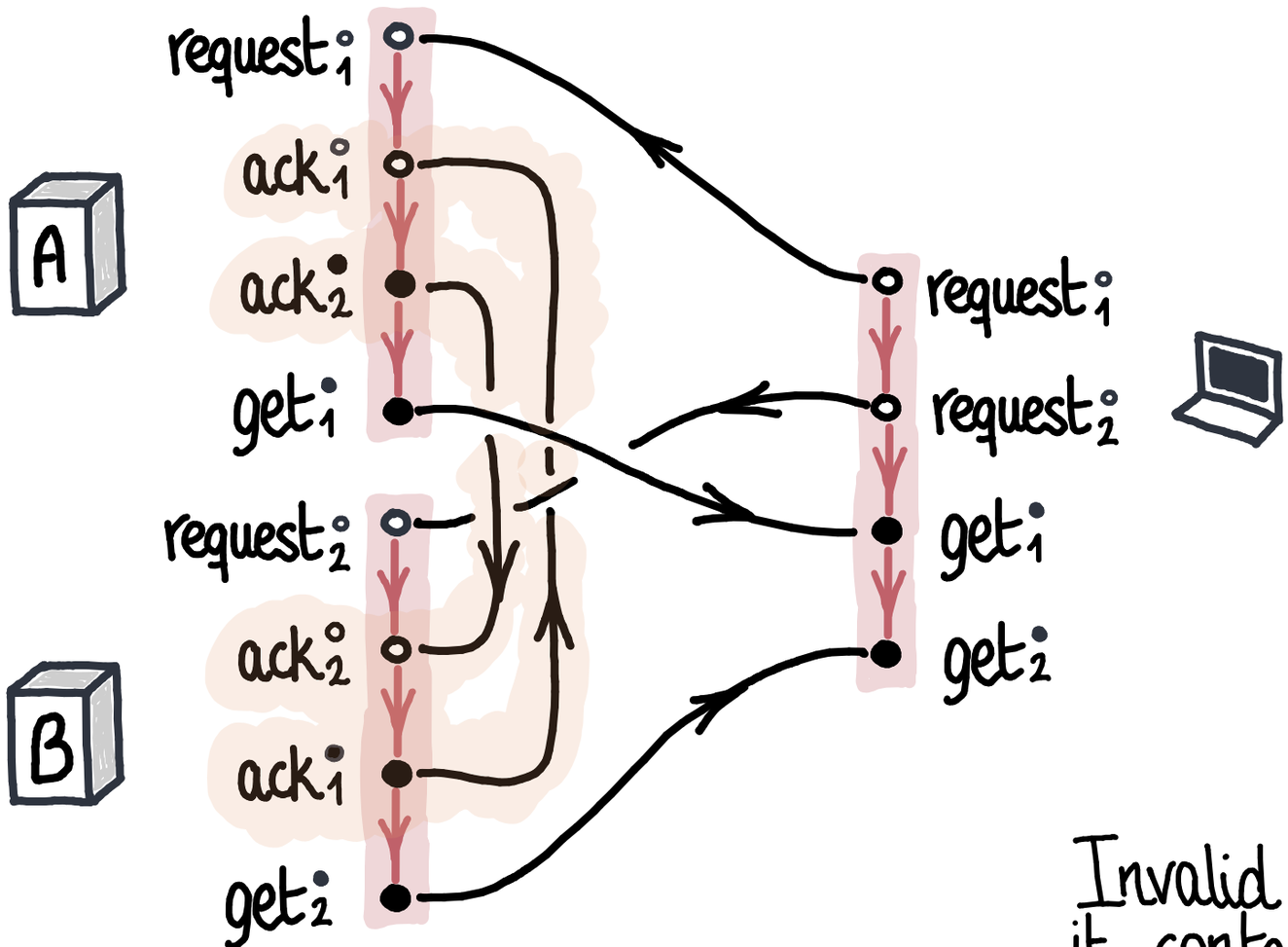
Polar shuffles mix programs that communicate sending• and receiving°.

Polar Shuffles



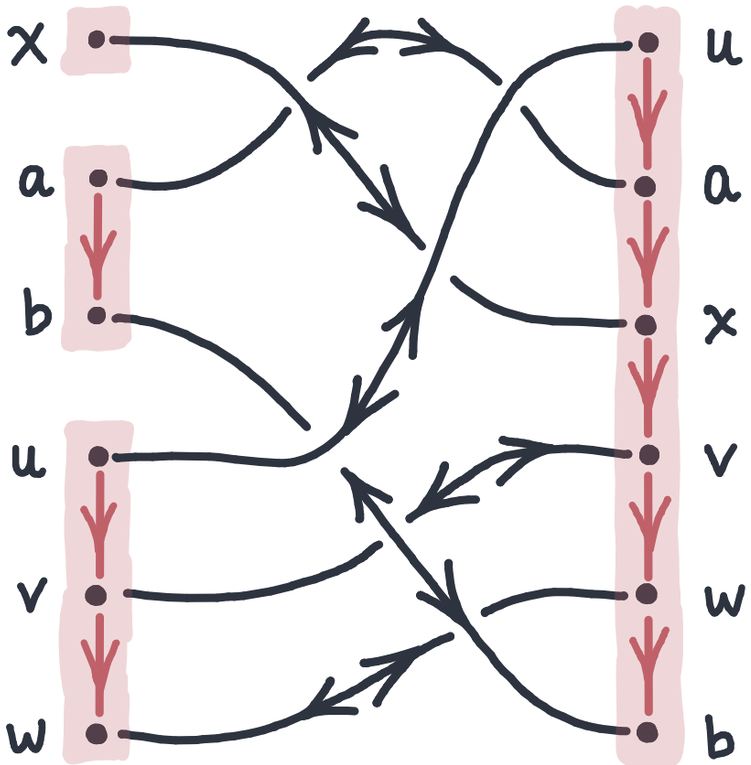
Invalid if and only if it contains a deadlock.

Polar Shuffles



Invalid if and only if it contains a deadlock.

SHUFFLES



DEFINITION. A shuffle is a bijection

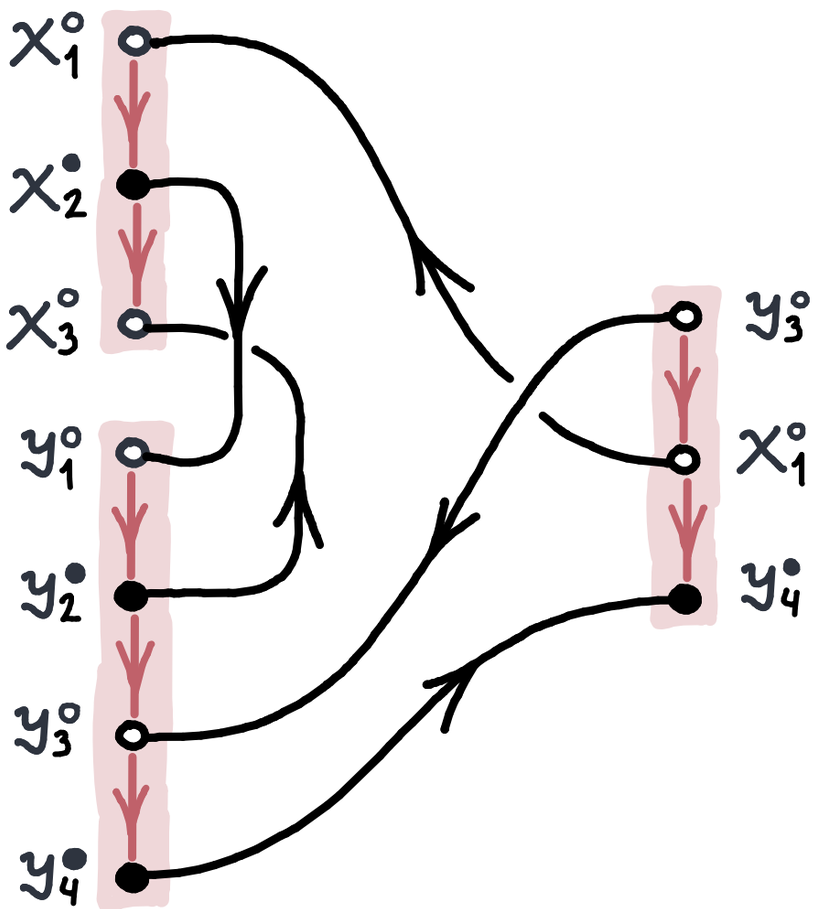
$$f : A_1 + \cdots + A_n \longrightarrow X$$

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

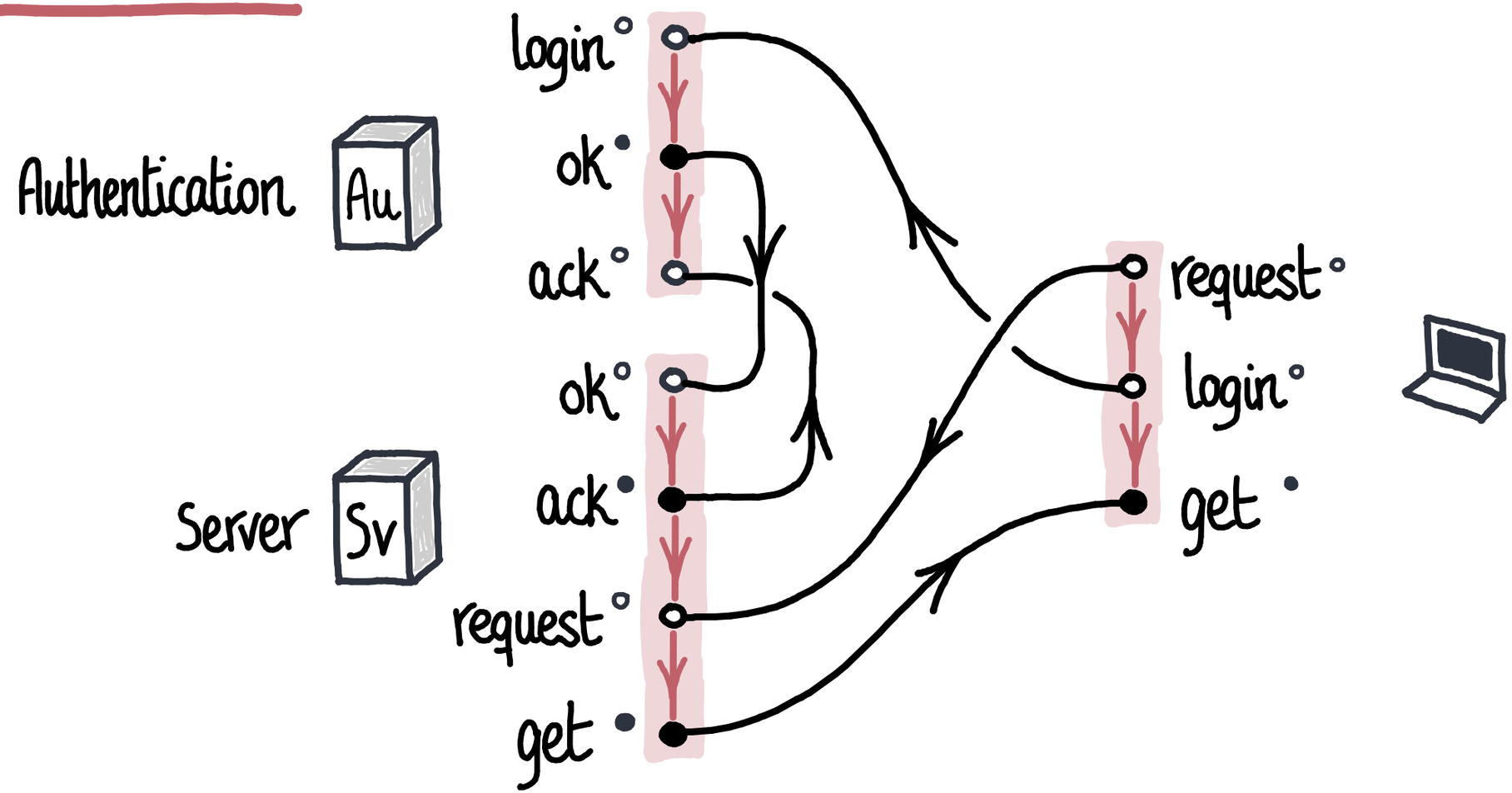
$$A_1^\bullet + \cdots + A_n^\bullet + X^\circ \longrightarrow A_0^\circ + \cdots + A_n^\circ + X^\bullet$$

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

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

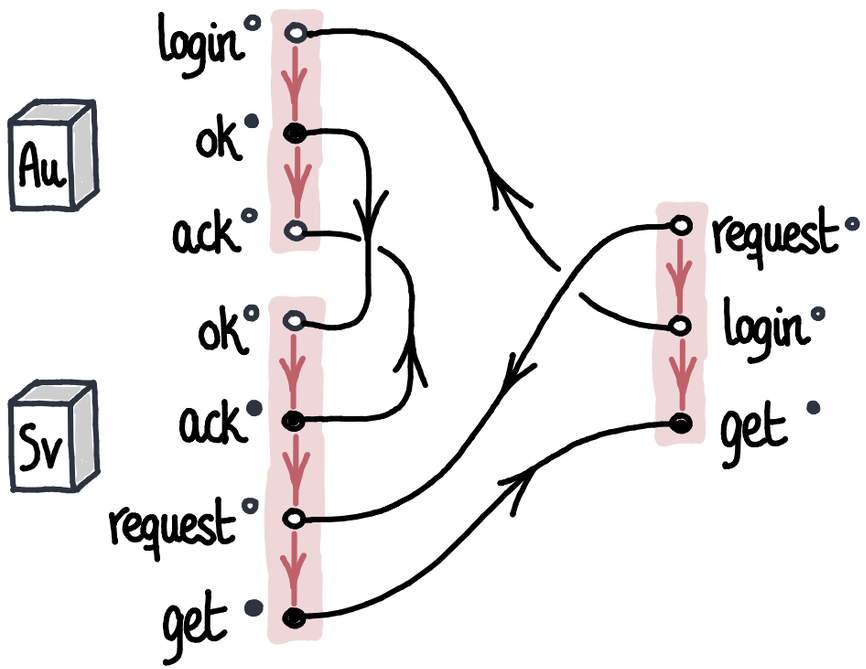
inducing an acyclic graph.

Polar Shuffles



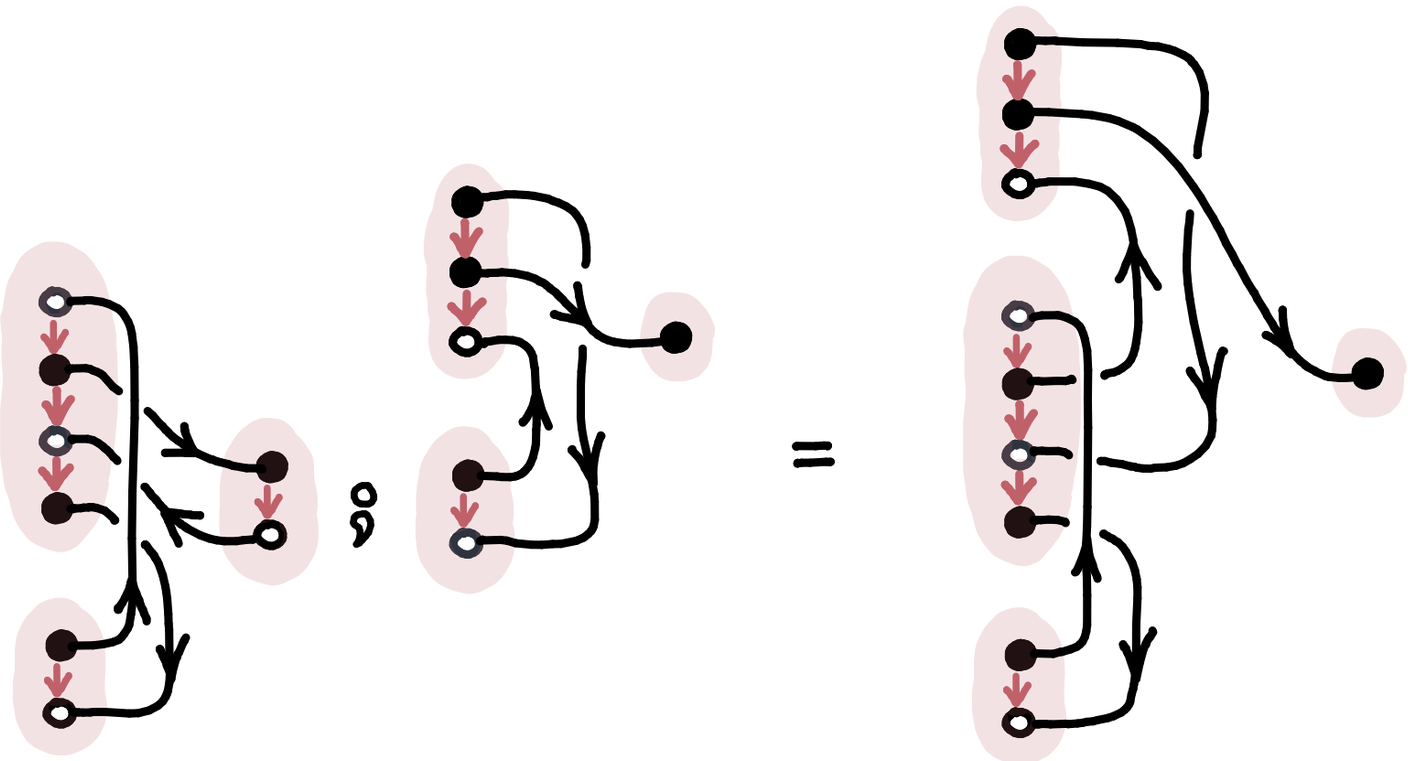
Each polar shuffle determines a way programs could interleave their messages.

Polar Shuffles



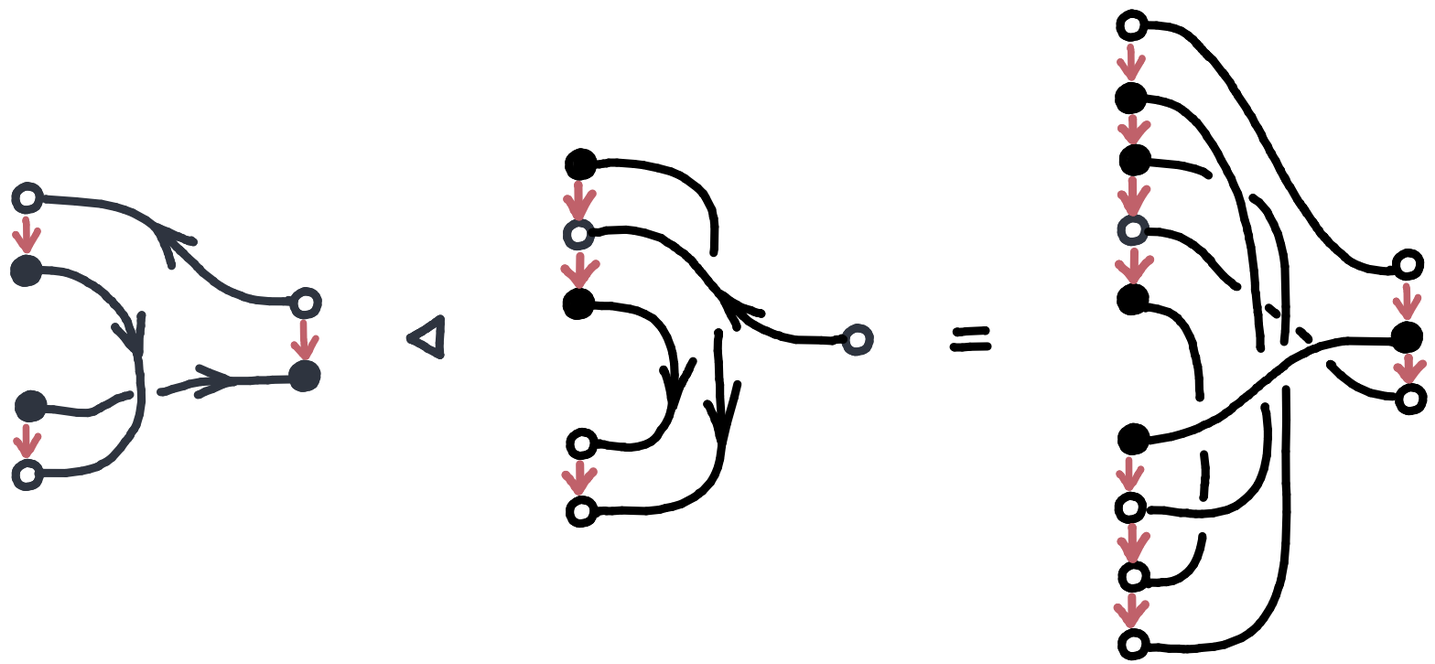

```
Protocol(request°, login°, get•) {
  Auth(login°, ok•, ack°);
  Serv(ok°, ack•, request°, get•);
}
```

Polar Shuffles



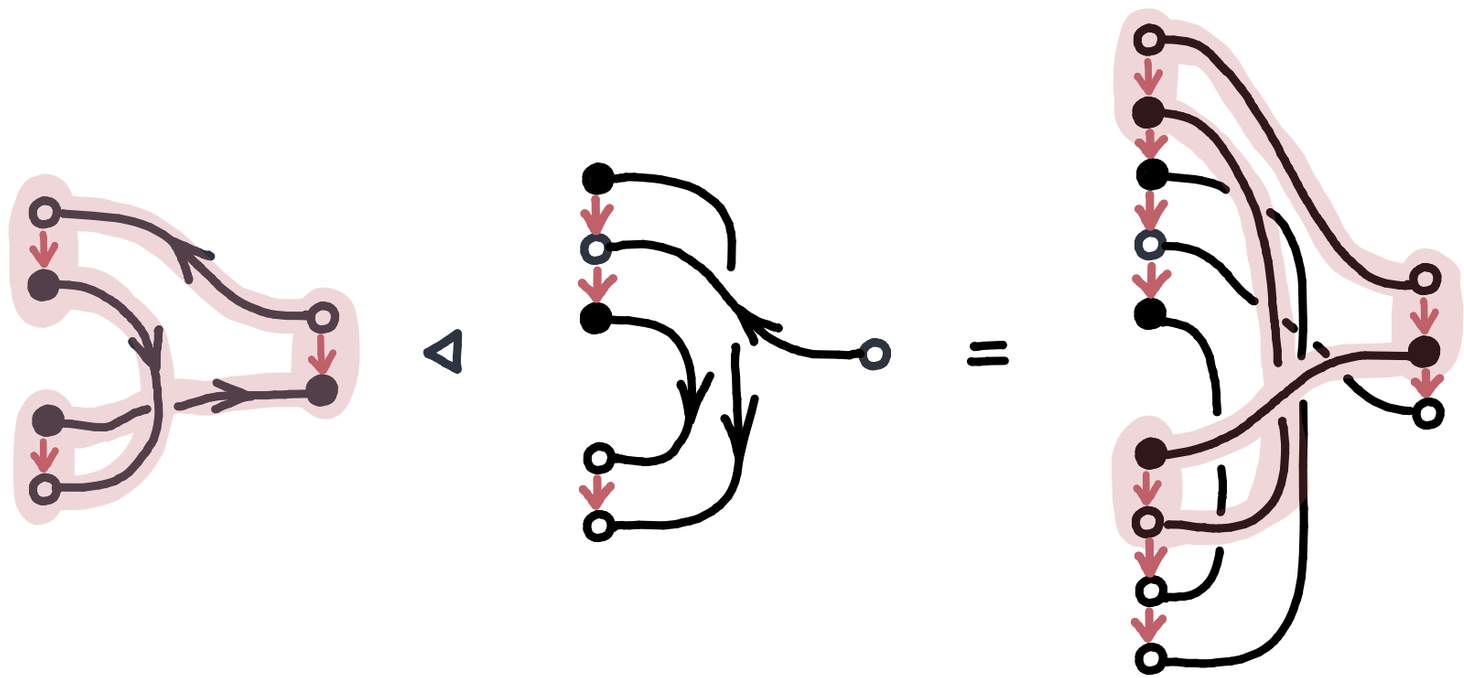
THEOREM. Polar shuffles form the free polar normal monoidal symmetric multicategory.

Polar Shuffles



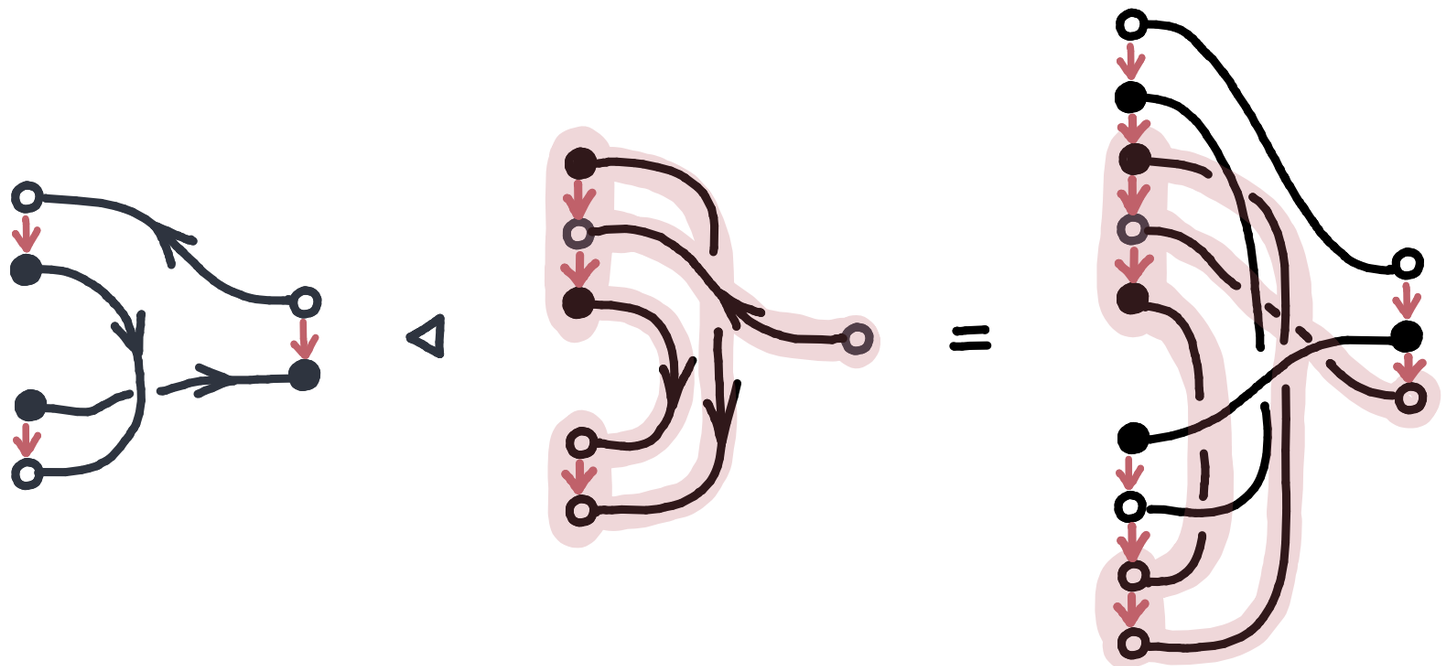
THEOREM. Polar shuffles form the free polar normal monoidal symmetric multicategory.

Polar Shuffles



THEOREM. Polar shuffles form the free polar normal monoidal symmetric multicategory.

Polar Shuffles

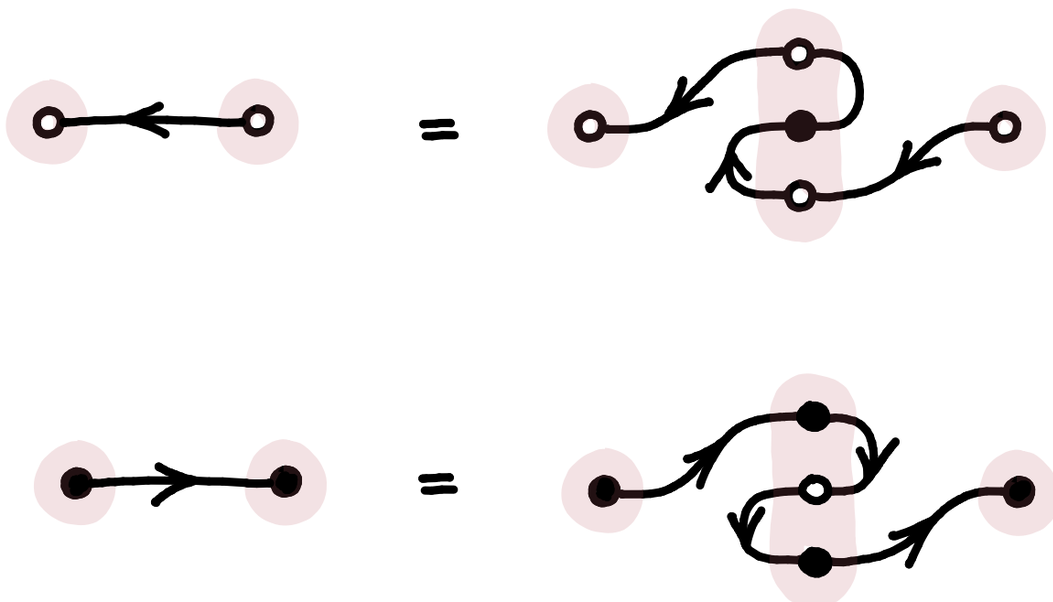


THEOREM. Polar shuffles form the free polar normal monoidal symmetric multicategory.

Polar Shuffles

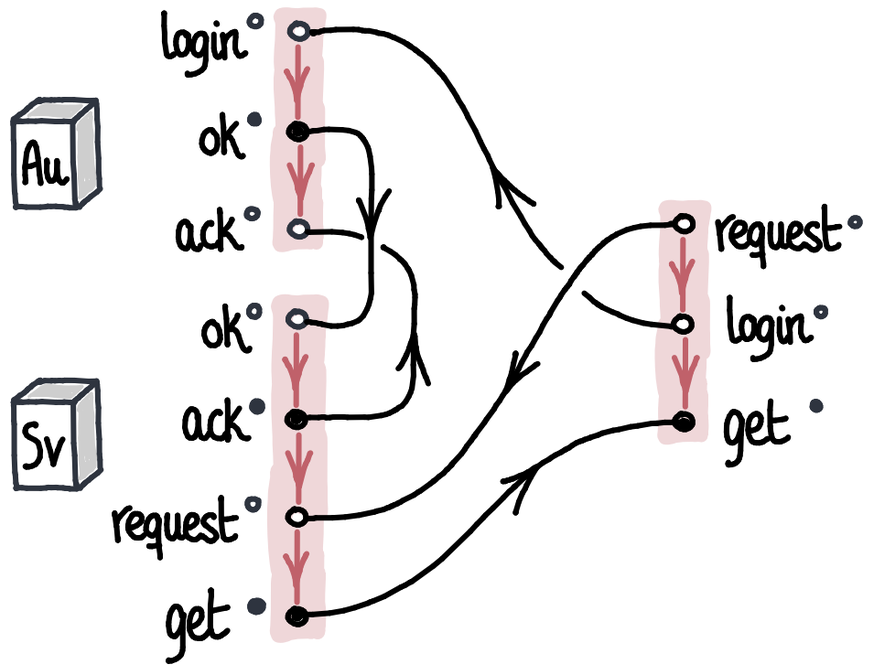
THEOREM. Polar shuffles form the free polar normal monoidal symmetric multicategory.

Polar Shuffles



THEOREM. Polar shuffles form the free polar normal monoidal symmetric multicategory.

Polar Shuffles

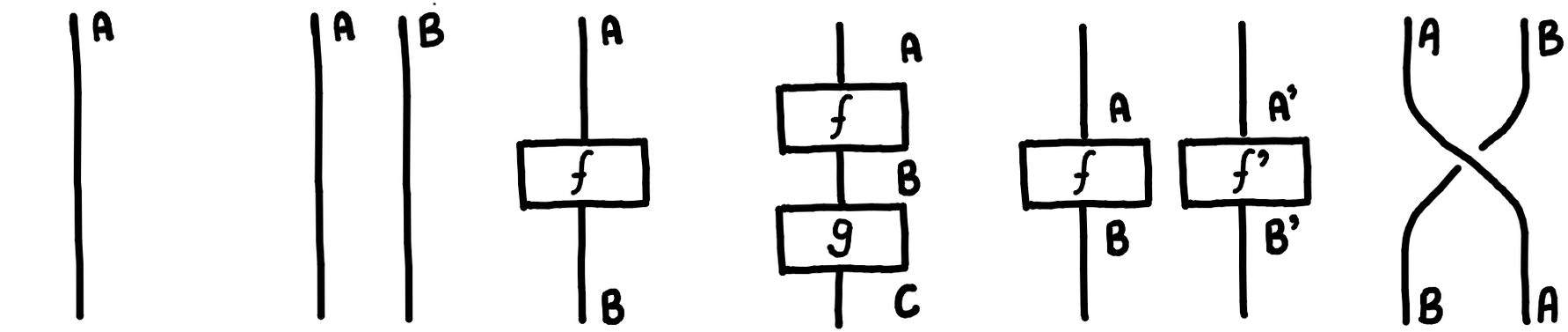



```
Protocol(request°, login°, get•) {
  Auth(login°, ok•, ack°);
  Serv(ok°, ack•, request°, get•);
}
```

THEOREM. Polar shuffles form the free polar normal monoidal symmetric multicategory.

Part 3: Monoidal Categories

String Diagrams

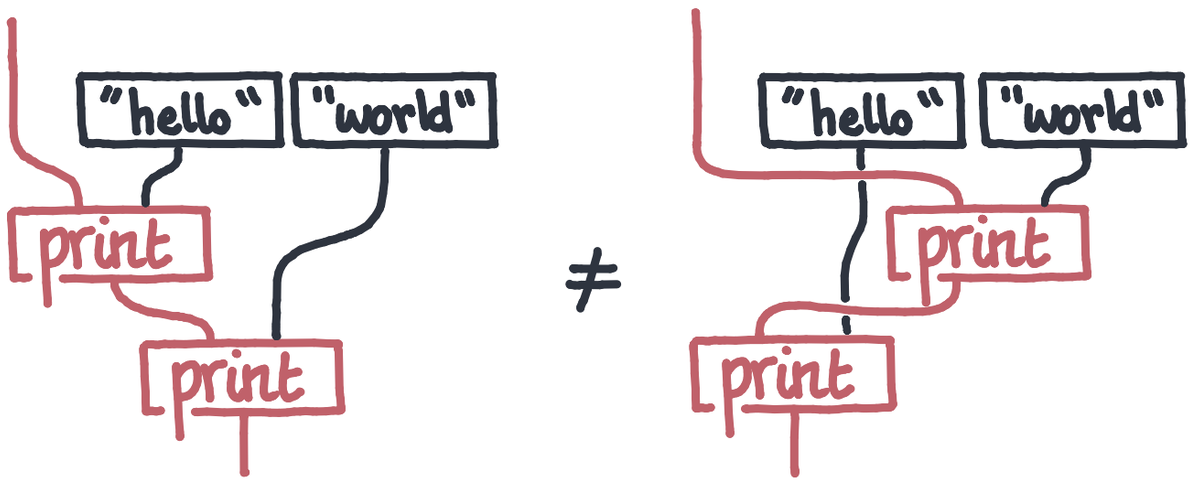


Resource Joint Process Sequence Parallel Swap

THEOREM. Acyclic string diagrams are sound & complete for sym. monoidal categories.

- Bonchi, Sobociński, Kissinger, Zanasi.
- Joyal, Street.

String Diagrams



- Power, Robinson. Premonoidal Categories and Notions of Computation.
- Jeffrey. A Graphical View of Programs.
- Román. Promonads and String Diagrams for Effectful Categories.
- Staton, Møgelberg. Linear Usage of State.

String Diagrams

```
f() = do
  let a = "hello"
  let b = "world"
  print(a)
  print(b)
```

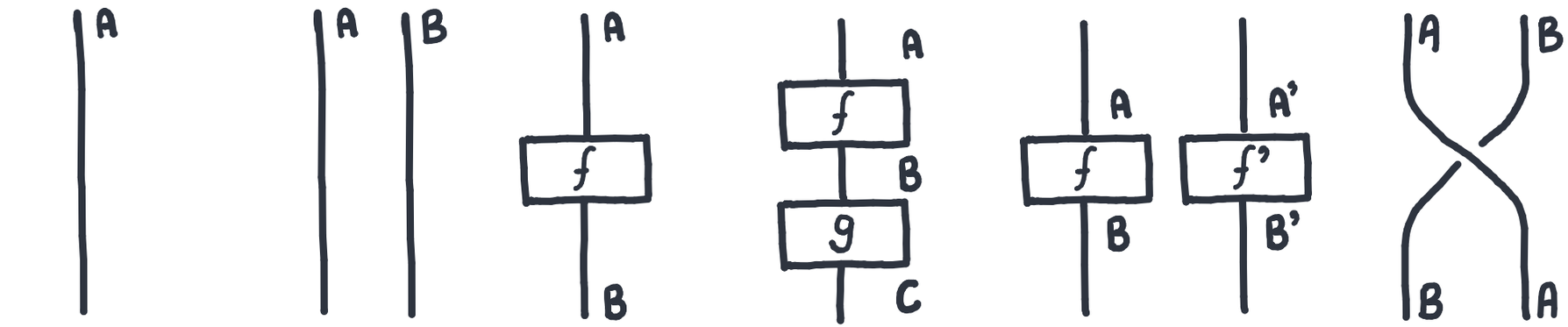
≠

```
f() = do
  let b = "world"
  let a = "hello"
  print(b)
  print(a)
```

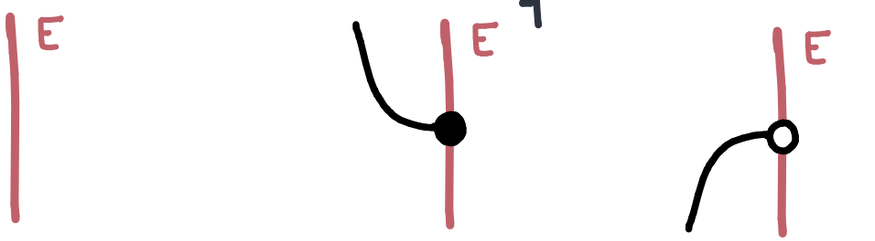
;

- Staton, Levy. Universal Properties of Impure Programming Languages.
- Jacobs, Hasuo. Freyd is Kleisli, for Arrows.
- Power, Robinson. Premonoidal Categories and Notions of Computation.
- Hughes. Generalising monads to arrows.

String Diagrams + Send/Receive



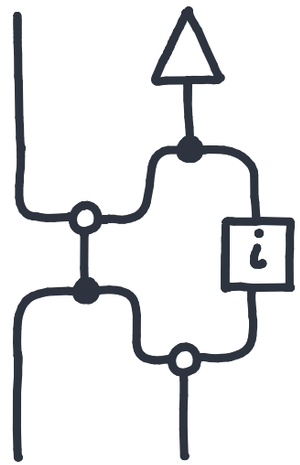
Resource Joint Process Sequence Parallel Swap

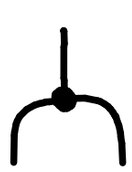


Environment Send Receive

THEOREM. Send/receive diagrams can be combined with polar shuffles.

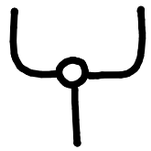
One-Time Pad





copy

random

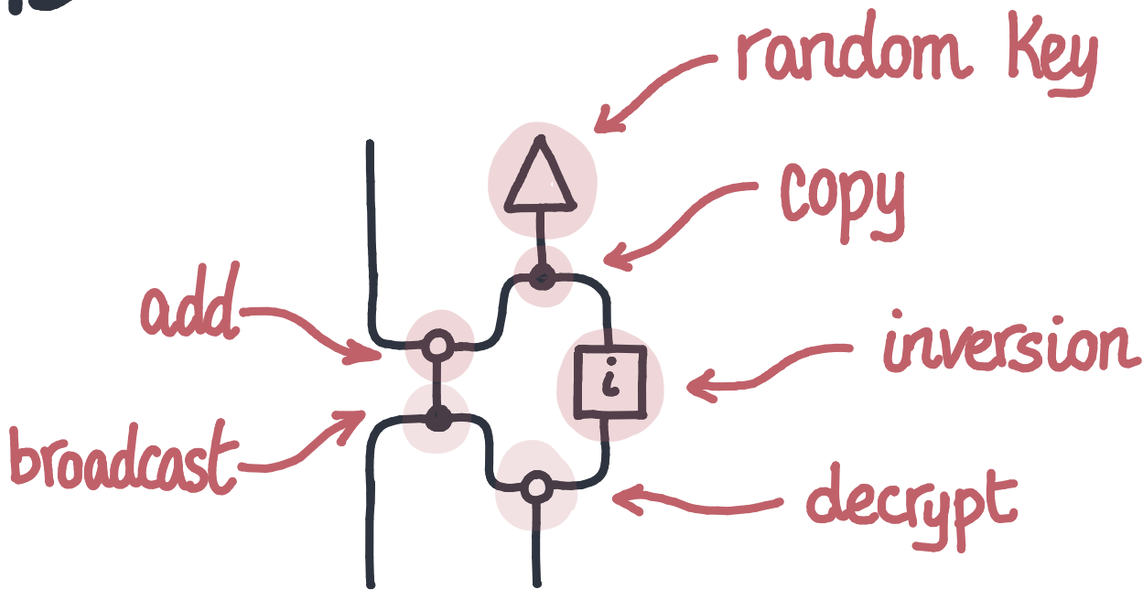


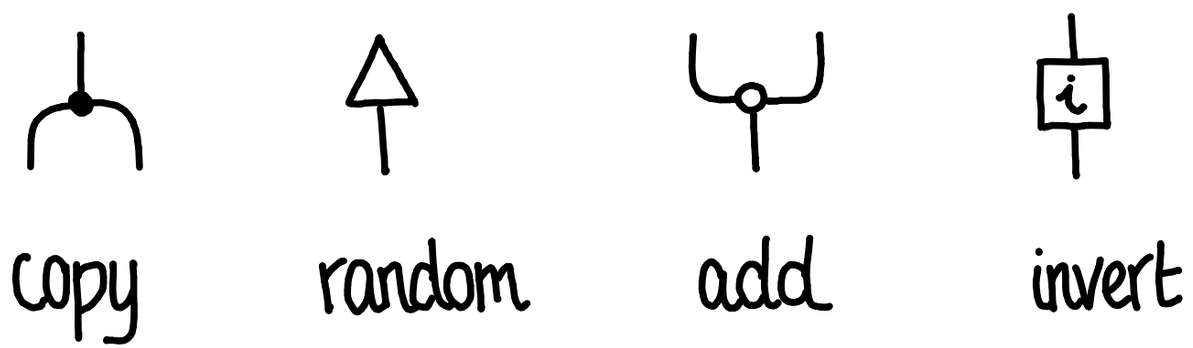
add

invert

Broadbent & Karvonen. Categorical Composable Cryptography.

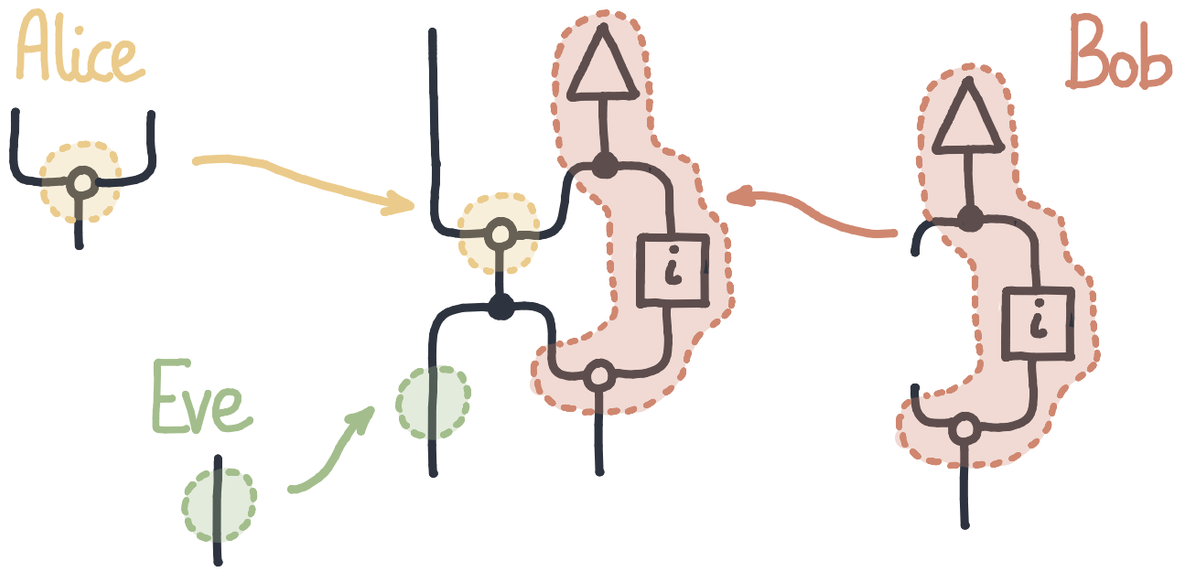
One-Time Pad





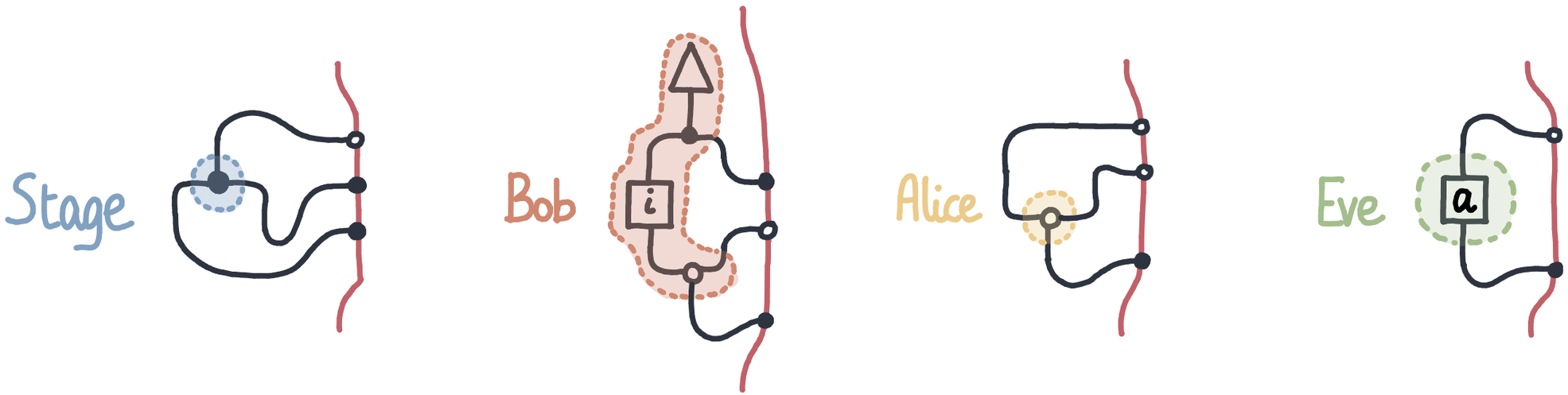
Broadbent & Karvonen. Categorical Composable Cryptography.

Multi-Party Processes



We want to split the morphism into different agents: Alice does not control the broadcast; Eve can only attack at the end; Bob keeps a bit in memory.

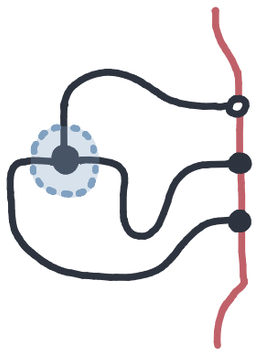
String Diagrams + Send/Receive



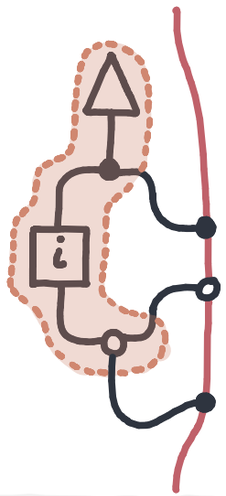
copy random add invert send receive

String Diagrams + Send/Receive

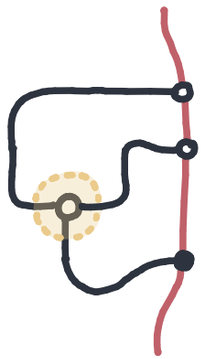
Stage



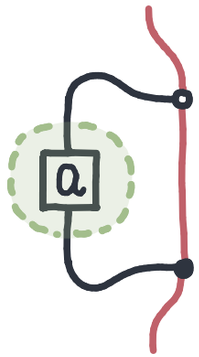
Bob



Alice



Eve




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

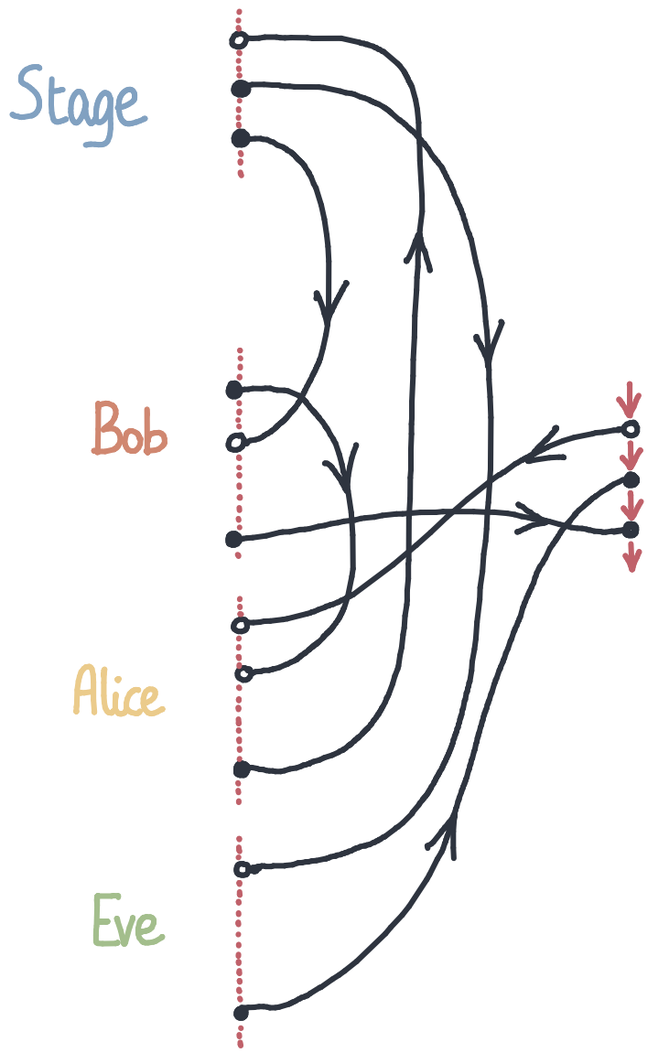
```
bob = do
    k ← random
    send(k)
    c ← get
    send(i(k)⊕c)
```

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

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

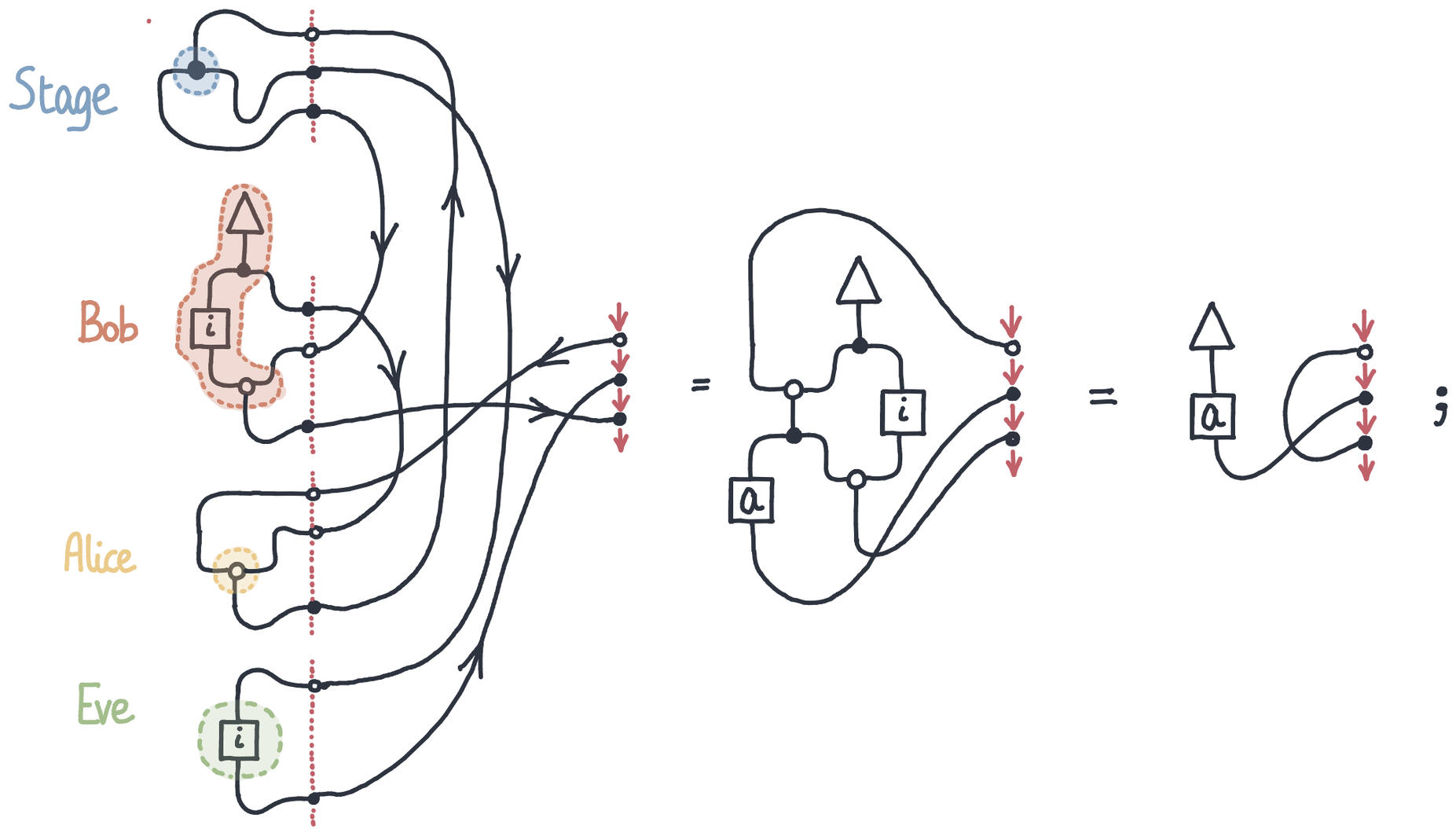
We split a protocol into multiple agents.

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•); }
```

In the Paper

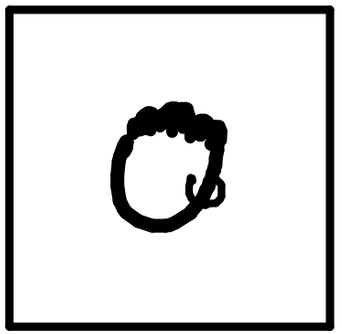
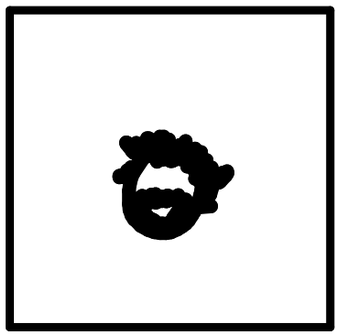
- Polar shuffles form a polarized physical monoidal multicategory.
  The universal one.

- Logic for deadlock-free-by-construction message-passing protocols.

- Adjunction between monoidal categories and algebras of polar shuffles.
  Universal message-passing for stochastic functions or Hilbert spaces.

END

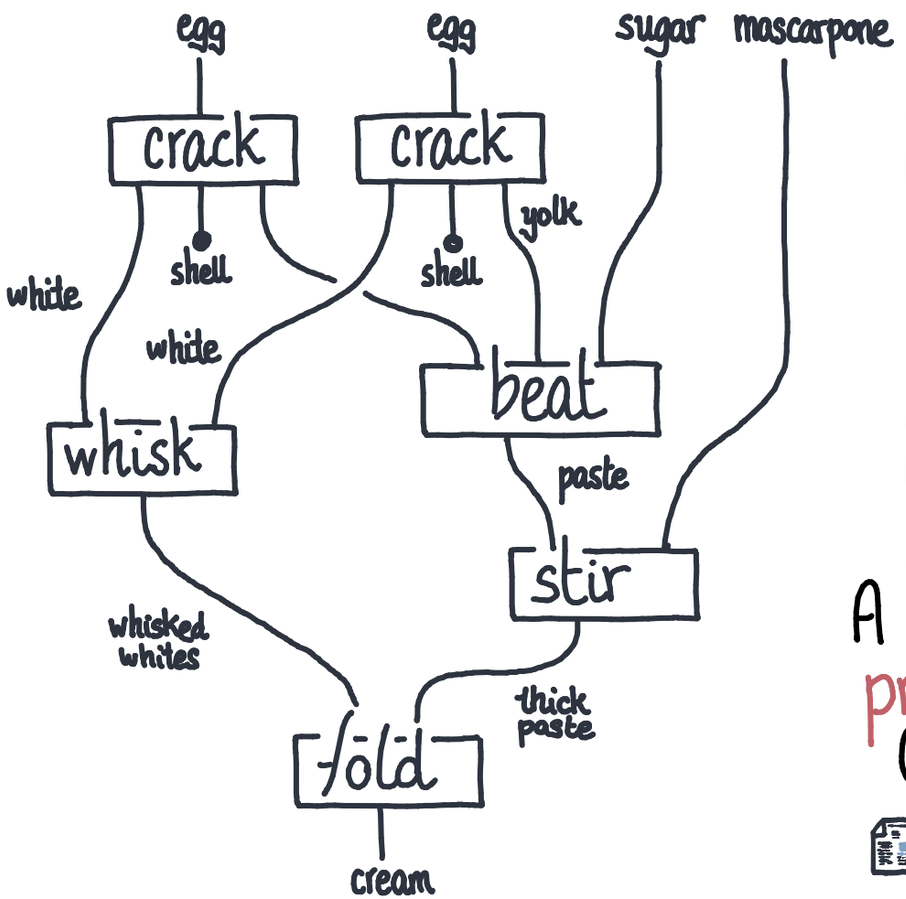
Polar Interleavings for Deadlock-Free Message Passing

Matt Earnshaw, Chad Nester, Mario Román

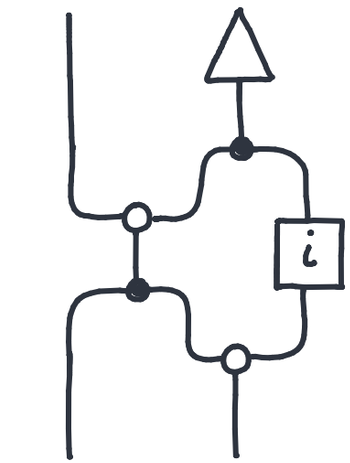
Tallinn University of Technology

Examples



Cooking recipe: crema di Mascarpone

Soboniński

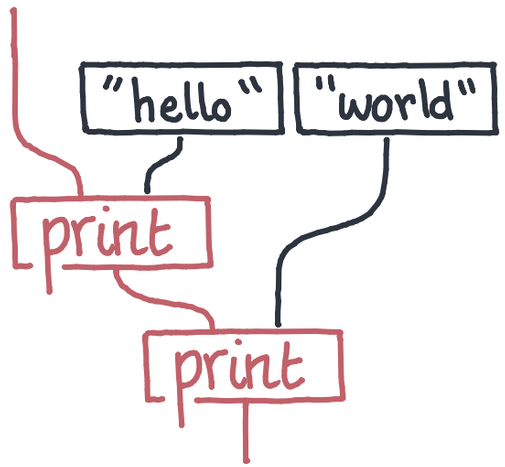


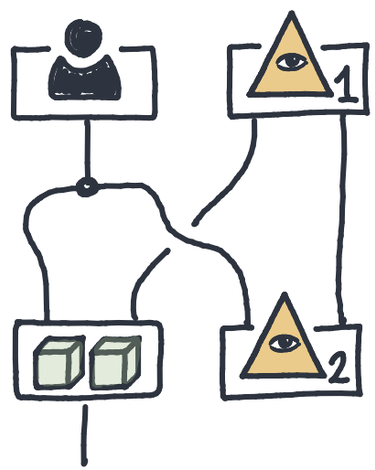
A cryptographic protocol:
One-time Pad

Broadbent, Karvonen

An imperative program:
"hello world"

Jeffrey

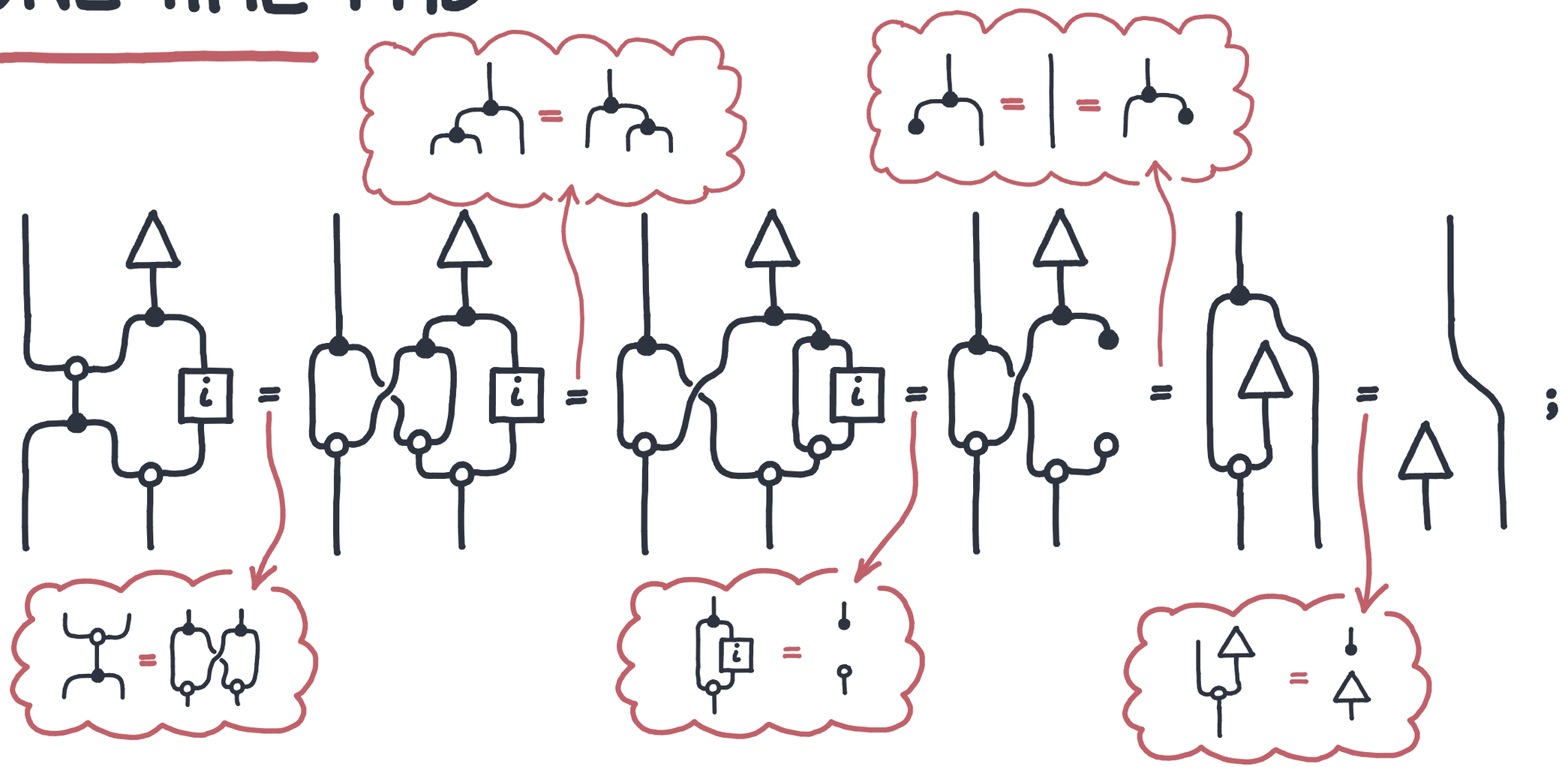




A decision problem:
Newcomb's paradox

Di Lavore, Román

One-Time Pad

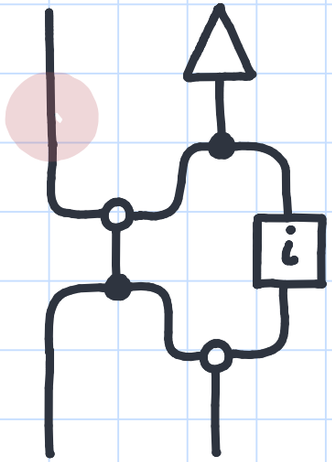


Broadbent & Karvonen. Categorical Composable Cryptography.

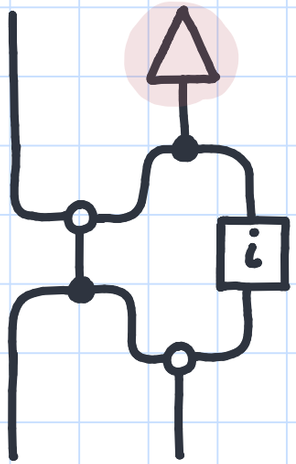
One-Time Pad

nwpt

message



One-Time Pad



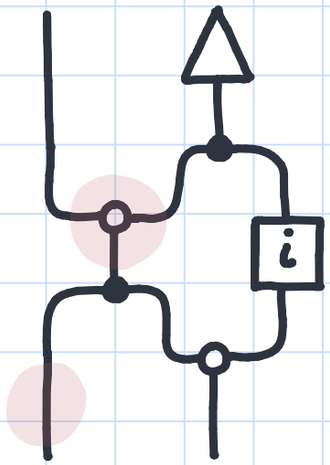
n w p t

1 0 -2 -4

message

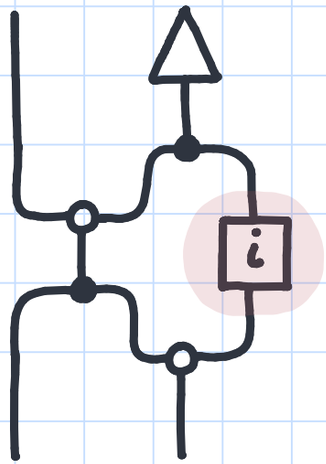
random key

One-Time Pad



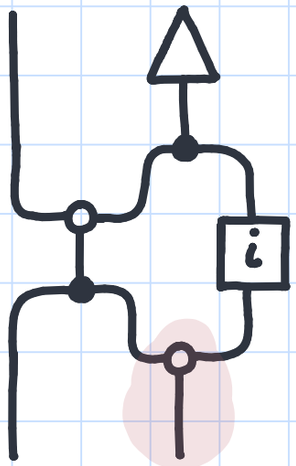
n w p t	message
1 0 -2 -4	random key
m w n p	crypted message

One-Time Pad



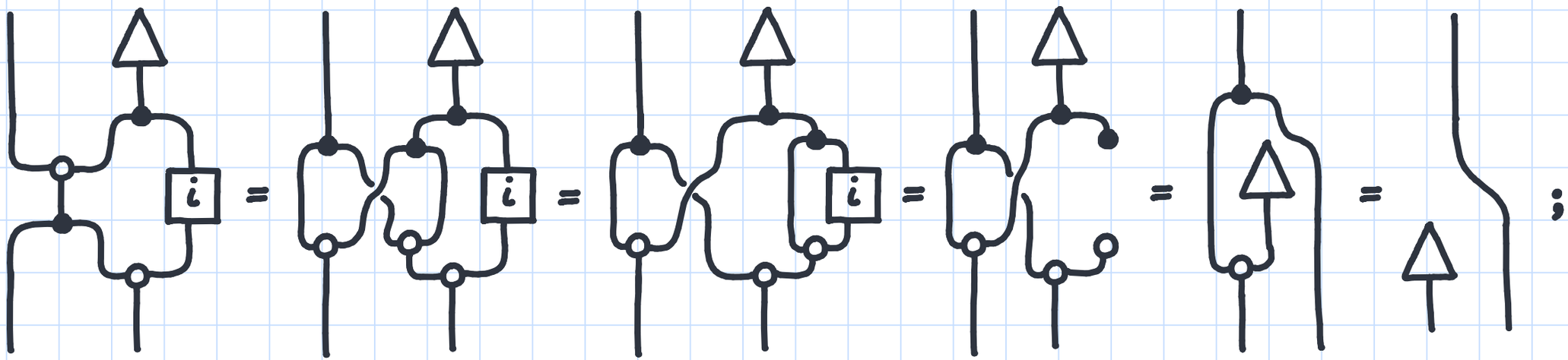
n w p t	message
1 0 -2 -4	random key
m w n p	crypted message
-1 0 2 4	inverted key

One-Time Pad



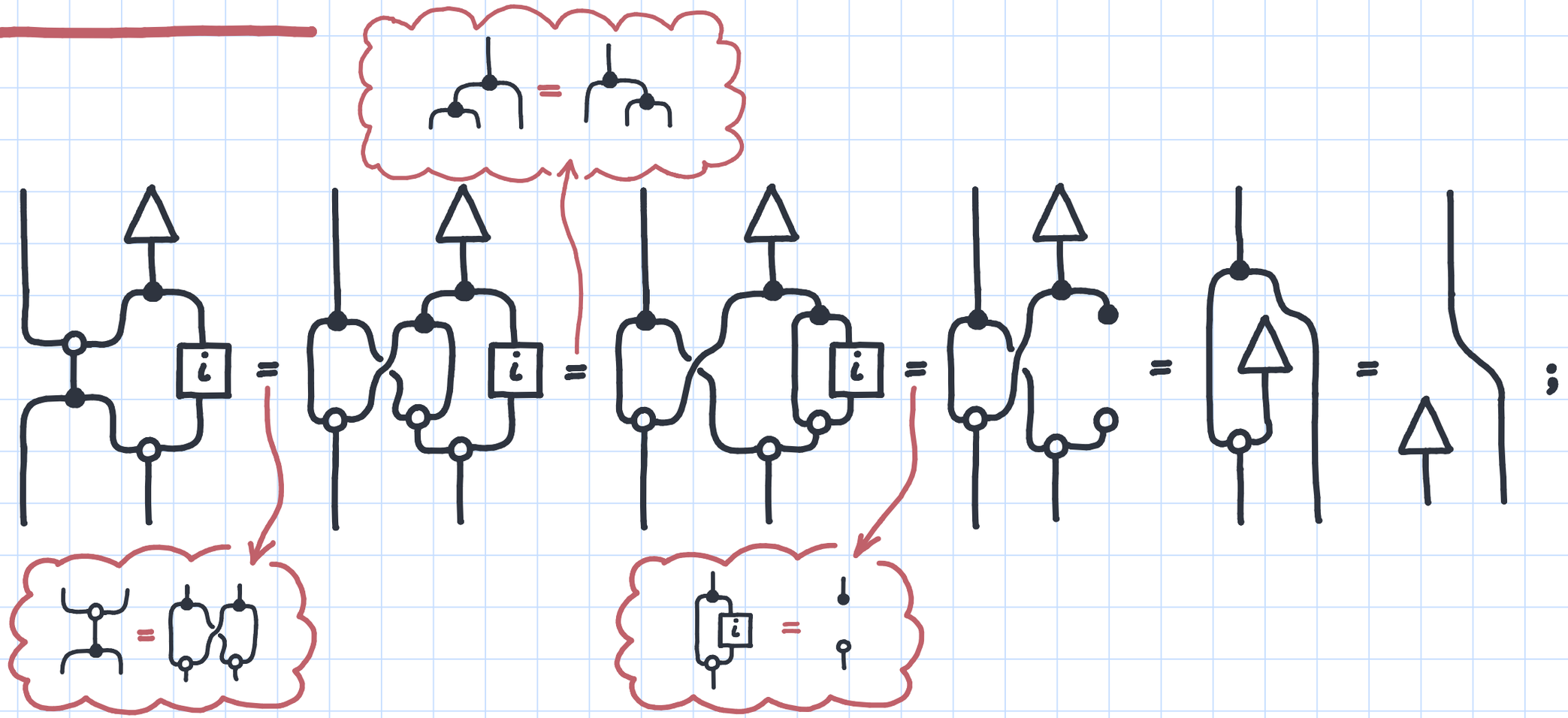
n w p t	message
1 0 -2 -4	random Key
m w n p	crypted message
-1 0 2 4	inverted Key
n w p t	message

One-Time Pad



Broadbent & Karvonen. Categorical Composable Cryptography.

One-Time Pad



Broadbent & Karvonen. Categorical Composable Cryptography.

Part 3 : 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

$$\frac{\Gamma_1 \quad \dots \quad \Gamma_n}{[\Gamma_1, \dots, \Gamma_n]_\sigma}\ \mathrm{SHF}_\sigma \qquad \frac{}{X^\circ, X^\bullet}\ \mathrm{SPW} \qquad \frac{\Gamma, X^\bullet, X^\circ, \Delta}{\Gamma, \Delta}\ \mathrm{LNK}$$

1. We can create a receive-send "echo" session.
2. We can receive what we just sent.
3. Events can be interleaved in any order.

This is the free polarized physical monoidal multicategory.

Message Theories

Shuffles

$$\frac{\Gamma_1 \quad \dots \quad \Gamma_n}{[\Gamma_1, \dots, \Gamma_n]_\sigma}\ \mathrm{SHF}_\sigma \qquad \frac{}{X^\circ, X^\bullet}\ \mathrm{SPW} \qquad \frac{\Gamma, X^\bullet, X^\circ, \Delta}{\Gamma, \Delta}\ \mathrm{LNK}$$

1. We can create a receive-send "echo" session.
2. We can receive what we just sent.
3. Events can be interleaved in any order.

This is the free polarized physical monoidal multicategory.

Message Theories

$$\frac{\Gamma_1 \quad \dots \quad \Gamma_n}{[\Gamma_1, \dots, \Gamma_n]_\sigma}\ \mathrm{SHF}_\sigma$$

Dualities

$$\frac{}{X^\circ, X^\bullet}\ \mathrm{SPW} \qquad \frac{\Gamma, X^\bullet, X^\circ, \Delta}{\Gamma, \Delta}\ \mathrm{LNK}$$

1. We can create a receive-send "echo" session.
2. We can receive what we just sent.
3. Events can be interleaved in any order.

This is the free polarized physical monoidal multicategory.

Message Theories: Axioms

Linking is natural with respect to shufflings.

$$\dfrac{\dfrac{\vdots m_1}{\Gamma_1, X^\bullet, X^\circ, \Gamma_2} \quad \vdots m_2}{\dfrac{\Gamma_1, \Gamma_2 \qquad \Delta_1, \Delta_2}{[\Gamma_1;\Delta_1]_\sigma, [\Gamma_2;\Delta_2]_\tau}} = \dfrac{\dfrac{\vdots m_1 \qquad \vdots m_2}{\Gamma_1, X^\bullet, X^\circ, \Gamma_2 \qquad \Delta_1, \Delta_2}}{[\Gamma_1;\Delta_1]_\sigma, X^\bullet, X^\circ, [\Gamma_2;\Delta_2]_\tau} ;$$

Spawning is natural with respect to shufflings.

$$\dfrac{\dfrac{\vdots m_1}{\Gamma_1, \Gamma_2} \quad \vdots m_2}{\dfrac{\Gamma_1, X^\circ, X^\bullet, \Gamma_2 \qquad \Delta_1, \Delta_2}{[\Gamma_1;\Delta_1]_\sigma, X^\circ, X^\bullet, [\Gamma_2;\Delta_2]_\tau}} = \dfrac{\dfrac{\vdots m_1 \qquad \vdots m_2}{\Gamma_1, \Gamma_2 \qquad \Delta_1, \Delta_2}}{[\Gamma_1;\Delta_1]_\sigma, [\Gamma_2;\Delta_2]_\tau, X^\circ, X^\bullet, [\Gamma_2;\Delta_2]_\tau} ;$$

Message Theories: Axioms

Spawning and linking are duals.

$$\dfrac{\dfrac{\vdots\, m}{\Gamma, X^\circ, \Delta}}{\Gamma, X^\circ, X^\circ, X^\circ, \Delta}\,(4_L) = \dfrac{\vdots\, m}{\Gamma, X^\circ, \Delta} \quad ; \quad \dfrac{\dfrac{\vdots\, m}{\Gamma, X^\bullet, \Delta}}{\Gamma, X^\bullet, X^\bullet, X^\bullet, \Delta}\,(4_R) = \dfrac{\vdots\, m}{\Gamma, X^\bullet, \Delta} \quad ;$$

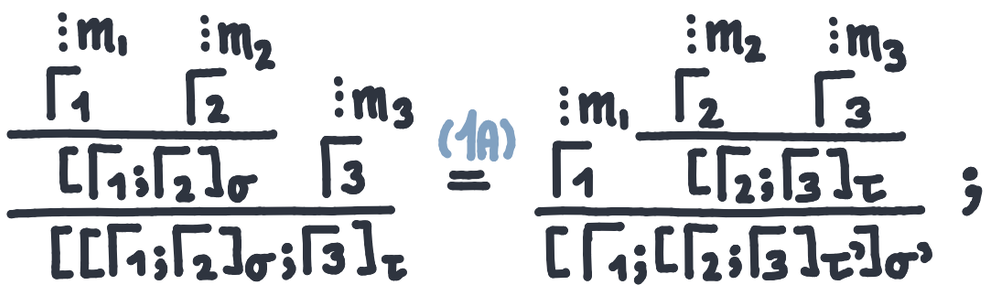
Spawning and linking interchange.

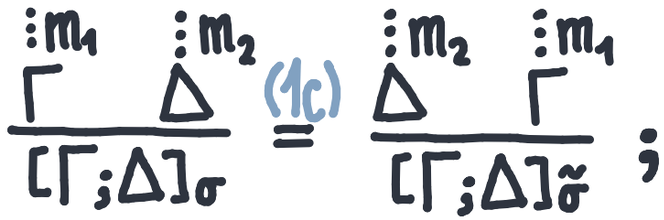
$$\dfrac{\dfrac{\vdots\, m}{\Gamma_1, X^\bullet, X^\circ, \Gamma_2, \Gamma_3}}{\Gamma_1, X^\bullet, X^\circ, \Gamma_2, Y^\circ, Y^\bullet, \Gamma_3}\,(5_A) = \dfrac{\dfrac{\vdots\, m}{\Gamma_1, X^\bullet, X^\circ, \Gamma_2, \Gamma_3}}{\Gamma_1, \Gamma_2, \Gamma_3} \quad ;$$

This is the free polarized normal monoidal multicategory on a set of types.

Message Theories: Axioms

Shuffles compose as in the multicategory of shuffles.

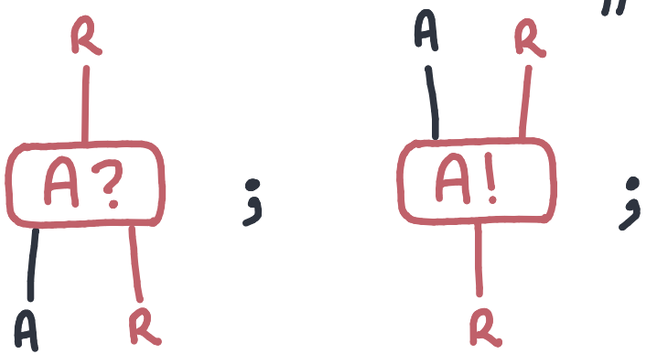




This is the free normal monoidal multicategory on a set of types.

Message Theories vs Process Theories

Sessions of the free message theory over a process theory are string diagrams extended with "send" and "receive" effects.

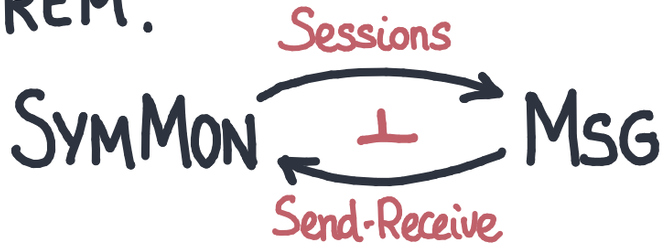


for each object $A \in M_{obj}$

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

$$X^{\circ}, Y^{\bullet}.$$

THEOREM.



Examples

THEOREM. Acyclic diagrams are symmetric monoidal terms.

Bonchi, Sobociński, Kissinger, Zanasi.

Multi-Party Processes

```
oneTimePad(alice,bob,eve,msg) = do
  key <- bob₀()
  crypt <- alice(msg, key)
  () <- eve(crypt)
  msg <- bob₁(crypt)
  return msg
```

```
alice(msg, key) = do
  crypt <- xor(msg,key)
  return crypt
```

```
bob() = do
  key <- randomBit
  !key
  ?crypt
  msg <- xor(crypt,key)
  return msg
```

```
eve(crypt) = do
  return crypt
```

These allow for code modularity. Send and receive types arise naturally.

github.com/mroman42/prodvoidal-algebra-code

Next Steps

- **CONJECTURE.** Runtime wire diagrams for Premonoidal & Freyd bicategories work "as expected".
  - Paquet, Saville. Strong Pseudomonads and Premonoidal Bicategories.
  - Bartlett. Quasistrict Symmetric Monoidal 2-Categories via Wire Diagrams.
- **CONJECTURE.** Type theory for message-passing over a monoidal mimicks send/receive types.
  We can extend to branching ⊕/& and iteration, in multi-party session using linear actions and feedback.
  - Honda, Yoshida, Carbone. Multiparty Asynchronous Session Types.
  - Cockett, Pastro. The Logic of Message-Passing.
  - Earnshaw, Hefford, Román. Contouring Prostar-Autonomous Categories. UNPUBLISHED.

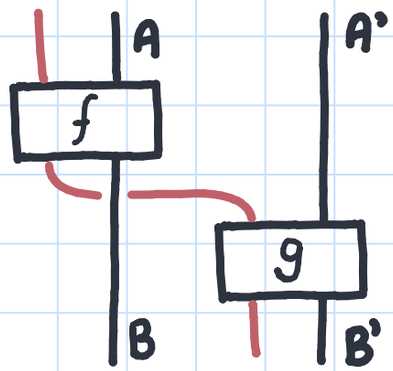
  The "Markov with conditionals" case is of particular interest: good characterization
- **GAME SEMANTICS.** Vast literature to compare to, outside the original scope.

Do-Notation

$$\frac{}{a_0 : A_0, \ldots, a_n : A_n \gg_\tau () \vdash \mathsf{return}\,(a_0, \ldots, a_n) : A_0 \otimes \cdots \otimes A_n}\ \mathrm{RETURN}_\tau$$

$$\frac{b_0 : B_0, \ldots, b_m : B_m,\ \Gamma \vdash t : \Delta}{a_0 : A_0, \ldots, a_n : A_n \gg_\tau \Gamma \vdash \mathsf{let}\ b_0, \ldots, b_m = f(a_0, \ldots, a_n);\ t : \Delta}\ \mathrm{GEN}_{f,\tau}$$

Linear do-notation is an internal language for sym. monoidals.




```
(a, a') ⇒ do
    f(a) → b
    g(a') → b'
    return (b, b')
```

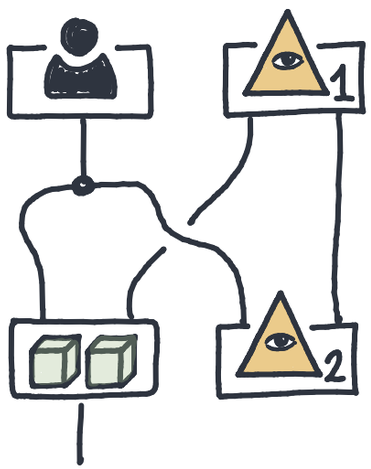
Message Theories

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

- Doing nothing is a session.
- We can create a receive-send "echo" session.
- We can receive what we just sent.
- Events can be interleaved in any order.
- This presents a monoidal multicategory.

PROOF: Polar shuffles are Message derivations

1. Shuffles form the free physical monoidal multicategory. Prop 4.2.9
2. Message theories are shuffles with duals, by definition.
3. Message theories are the free polarized physical monoidal multicategory.
4. Message theories are coherent, by finding their normal form. Thm 4.1.8
5. Polar shuffles are coherent, by definition. Prop 4.4.4
6. A polar shuffle between some types exists iif a message theory derivation exists.
7. Polar shuffles are message theory derivations. Prop 4.4.9
8. Polar shuffles form the free polarized physical mon. multicategory. Thm 4.4.11


1
2

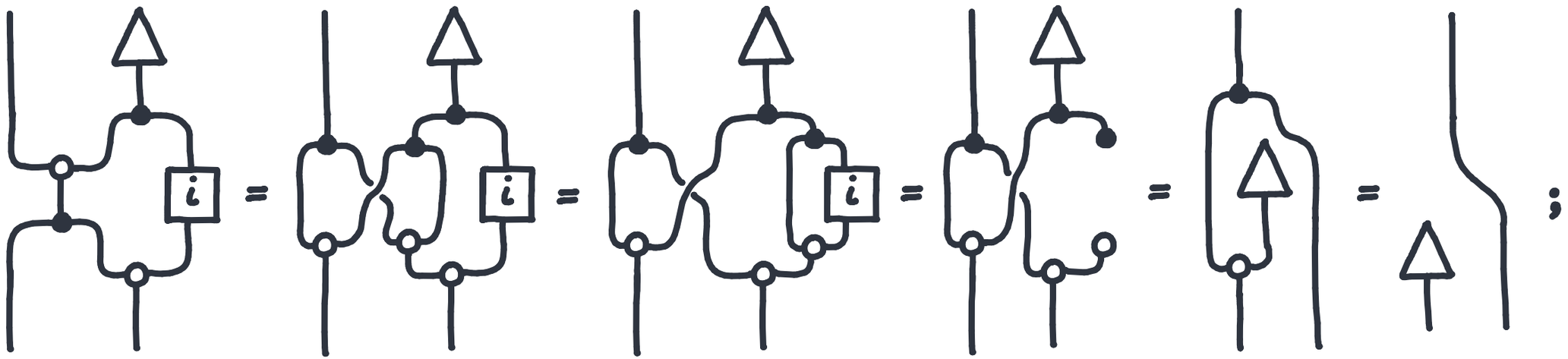
Proof: Sessions and Processes form an adjunction

1. We construct a effectful category of sessions over a strict sym.mon.cat. Def 4.5.3
2. Sessions form a message theory, by topology. Prop 4.5.6
3. Sessions are combs. Prop 4.5.5
4. There exists a unit $\mathrm{inProc} : \mathbb{C} \to \mathrm{Proc}(\mathrm{Session}(\mathbb{C}))$. Lem. 4.5.8
5. Sessions and Processes form an adjunction. Thm 4.5.9

Part 6: Examples

One-Time Pad

Broadbent & Karvonen propose a formalization of the one-time pad in a monoidal category with a Hopf algebra with an integral.



We can reason about security using string diagrams.

Broadbent & Karvonen. Categorical Composable Cryptography.

PROOF: String diagrams for effectful categories

1. Braiding runtime forms cliques.
2. Braid cliques on the runtime monoidal form a effectful, $\mathbf{Eff}(\mathcal{V}, \mathcal{G})$. Lem 1.7.5
3. There exists an id.on objs. $\mathbf{Mon}(\mathcal{V}) \to \mathbf{Eff}(\mathcal{V}, \mathcal{G})$ preserving mon.structure. Lem 1.7.6
4. There exists a unique effectful functor out of $\mathbf{Mon}(\mathcal{V}) \to \mathbf{Eff}(\mathcal{V}, \mathcal{G})$. Lem 1.7.7
5. The free efectful has morphisms $A \to B$ the $R \otimes A \to R \otimes B$ of the runtime monoidal. Thm 1.7.8
6. String diagrams with runtime are a language for effectful categories. Cor 1.7.9

Duoidal Categories

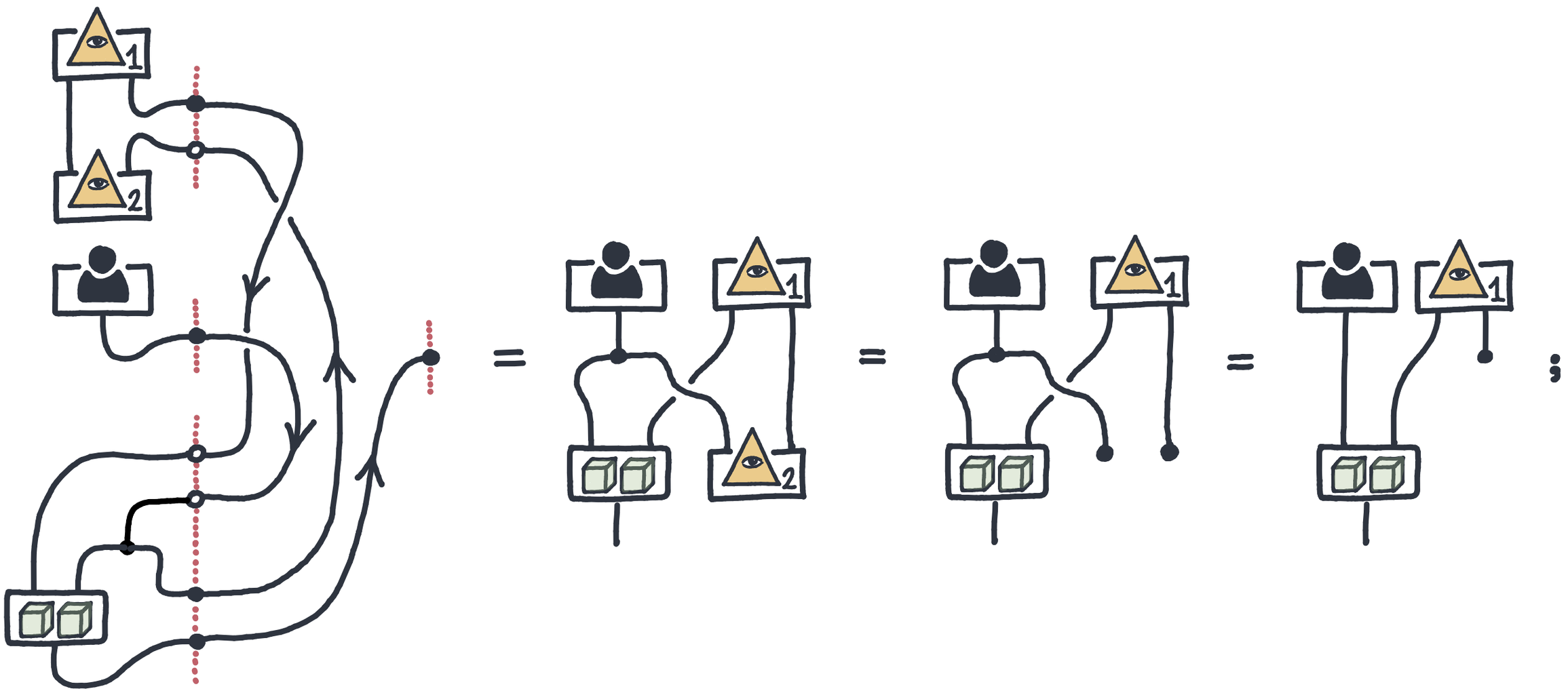
Duoidal categories are not coherent, there are two formal maps

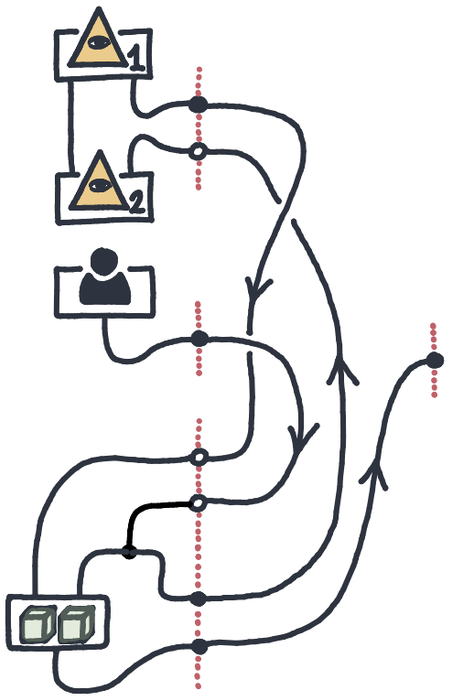
$$I \triangleleft I \to I.$$

However, physical duoidal categories are coherent: the free physical duoidal over a set of objects is a full subcategory of poset inclusions. There is at most a single morphism between any two objects where every type appears exactly once with each variance.

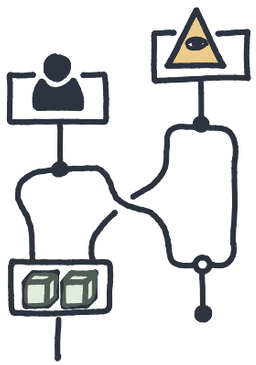
$$A \otimes (B \triangleleft C) \rightleftarrows (A \otimes B) \triangleleft C.$$

Grabowski '81
Gischer '88


1
2
=
1
2
=
1
=
1
;

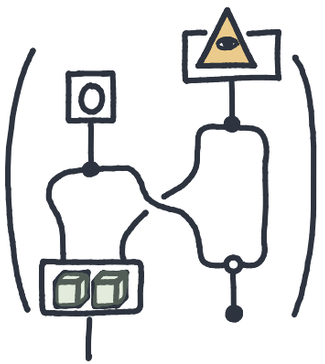


(i) =

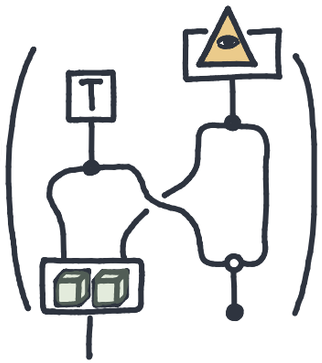


(ii) =

a

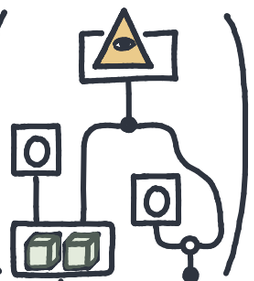


$\oplus (1-a)$

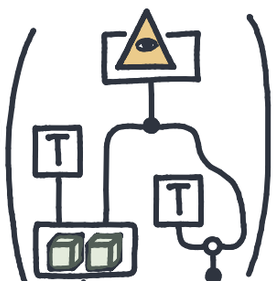


(iii) =

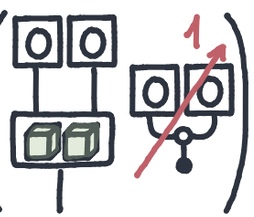
a



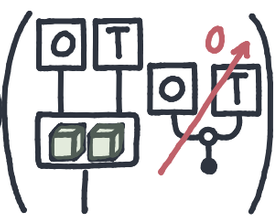
$\oplus (1-a)$



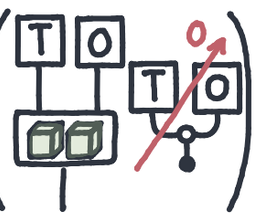
(iv) $= ap$



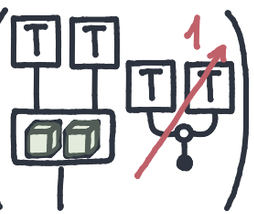
$\oplus a(1-p)$



$\oplus (1-a)\, p$



$\oplus (1-a)(1-p)$



(v) $= ap(1000€) \oplus (1-a)(1-p)\,\varepsilon(1€) \overset{(vi)}{\leq} p(1000€).$

SHUFFLES

The physical monoidal multicategory of shuffles over an alphabet Σ has

- objects the words of Σ^*;
- mutimorphisms $\mathrm{Shuf}(w_1, \ldots, w_n ; w)$ are shufflings into w.

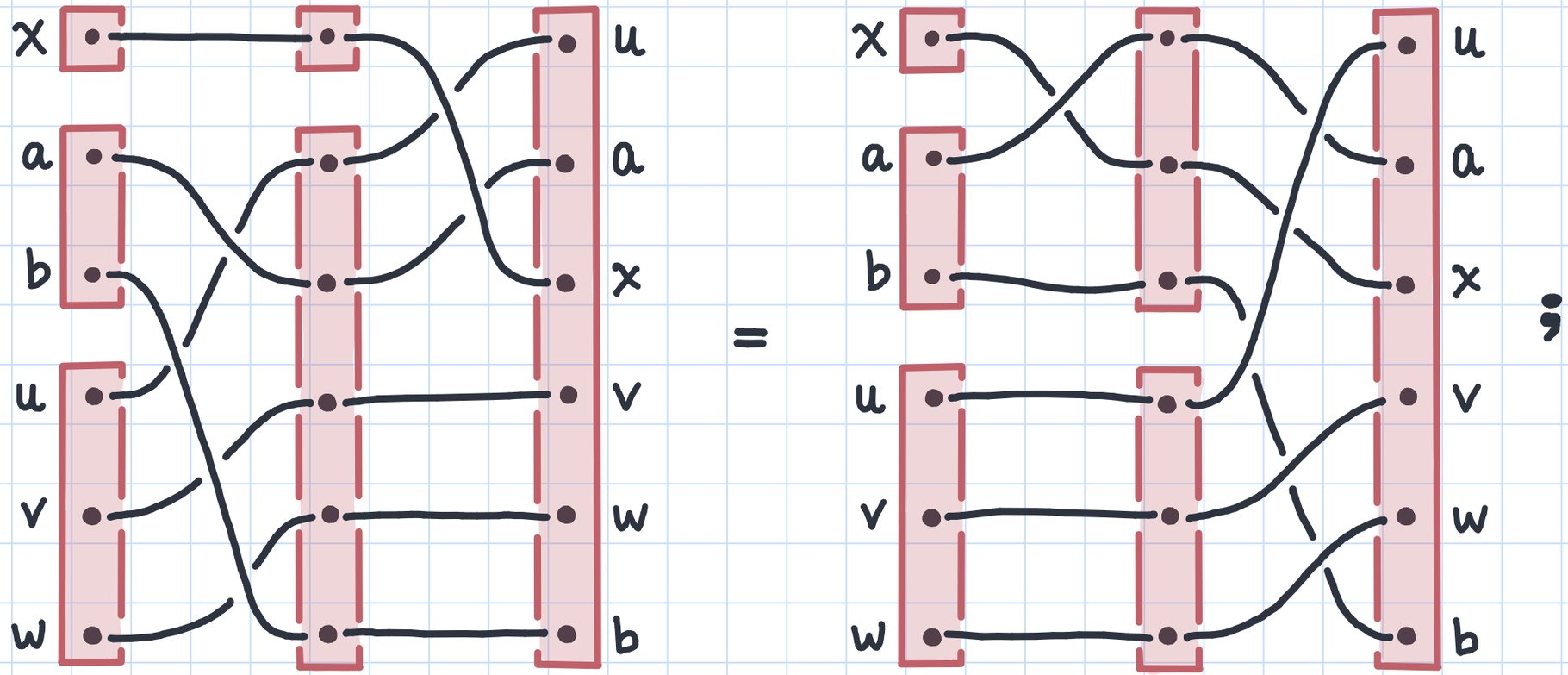
This is not posetal, in fact,

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

However, if each letter appears exactly once, then the suffle, if it exists, it is unique.

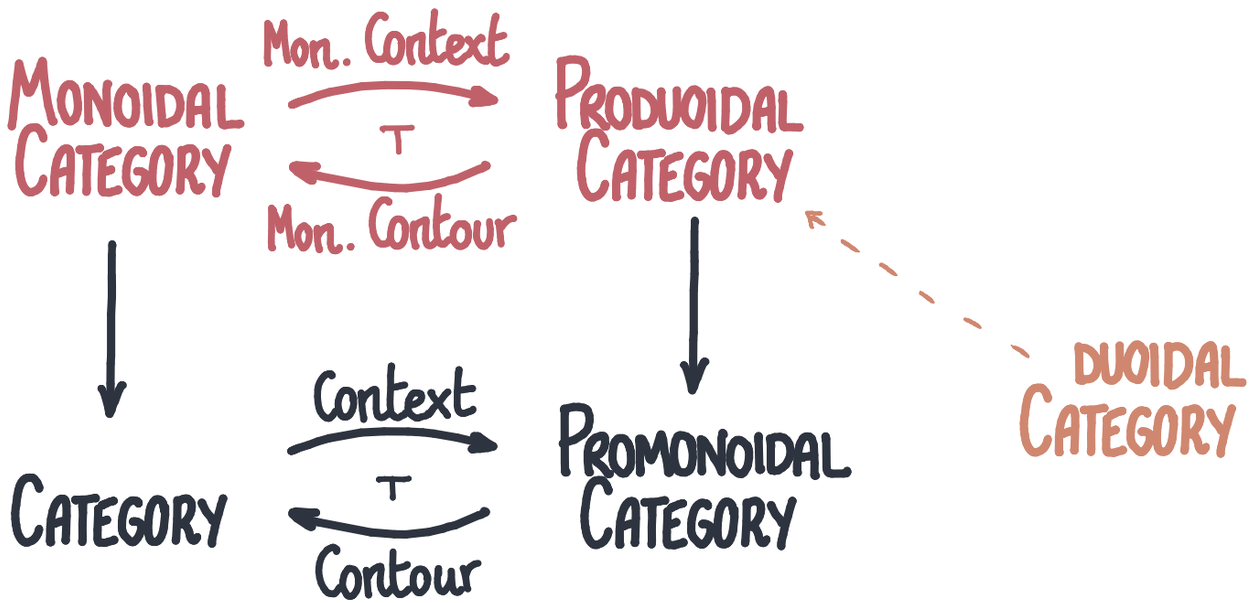
EXAMPLE. Unique possible shuffle $xy, wz \to xwyz$.

SHUFFLES



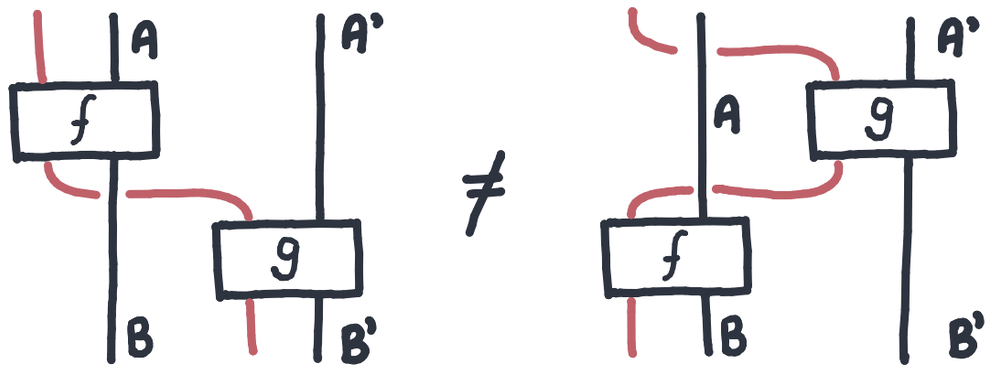
THEOREM. Shuffles form the free physical monoidal multicategory.

Next


Monoidal Category
Mon. Context
T
Mon. Contour
Produoidal Category
Duoidal Category
Category
Context
T
Contour
Promonoidal Category

PreMonoidal Categories

Premonoidal categories extend monoidal categories with effects.

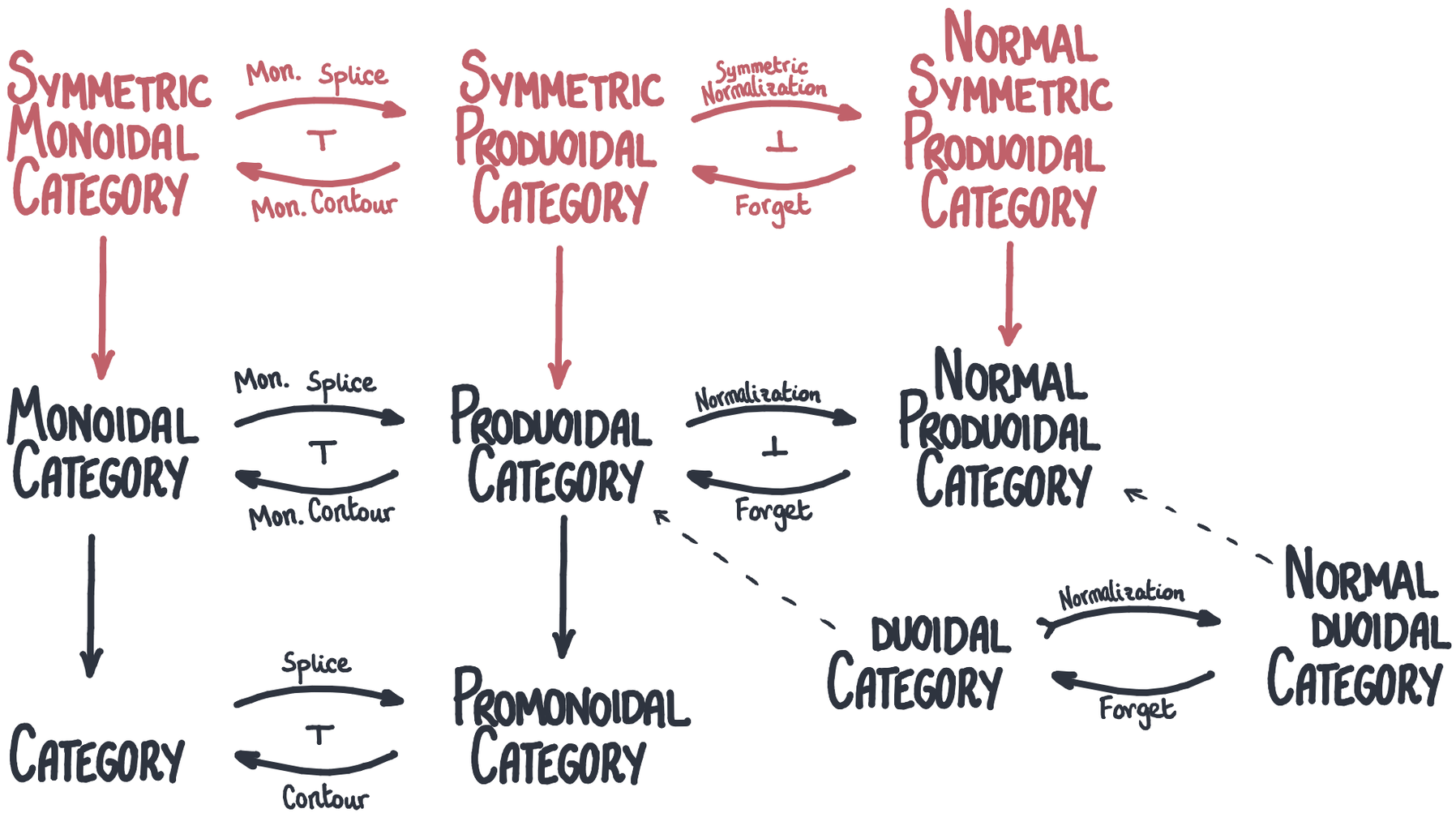


Failure of Interchange

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

Jeffrey Román

Next


Symmetric Monoidal Category
Mon. Splice
⊤
Mon. Contour
Symmetric Produoidal Category
Symmetric Normalization
⊥
Forget
Normal Symmetric Produoidal Category
Monoidal Category
Mon. Splice
⊤
Mon. Contour
Produoidal Category
Normalization
⊥
Forget
Normal Produoidal Category
Category
Splice
⊤
Contour
Promonoidal Category
Duoidal Category
Normalization
Forget
Normal Duoidal Category

Monoidal Categories: Process Theories

Monoidal categories are coherent: there is at most a single morphism between any two objects in the free monoidal category over some objects.

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

Symmetric monoidal categories do not satisfy the same properly.

$$A \otimes A \underset{\text{swap}}{\overset{\text{id}}{\rightleftarrows}} A \otimes A$$

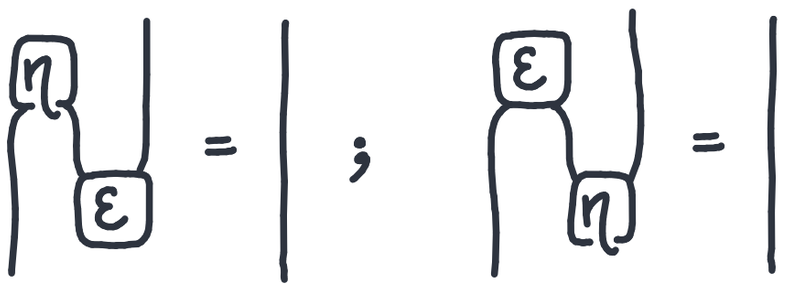
Still, they satisfy another coherence property: there is at most a single morphism between any two objects where every type appears exactly once with each variance.

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

Polarization

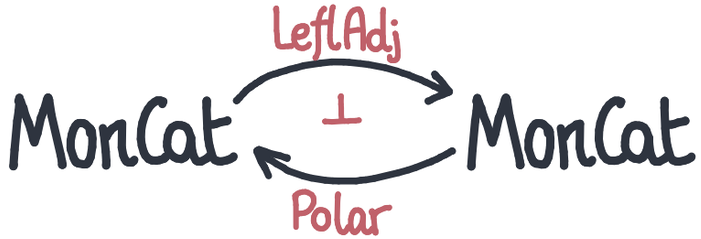
The second ingredient for message passing is the duality Send/Receive.

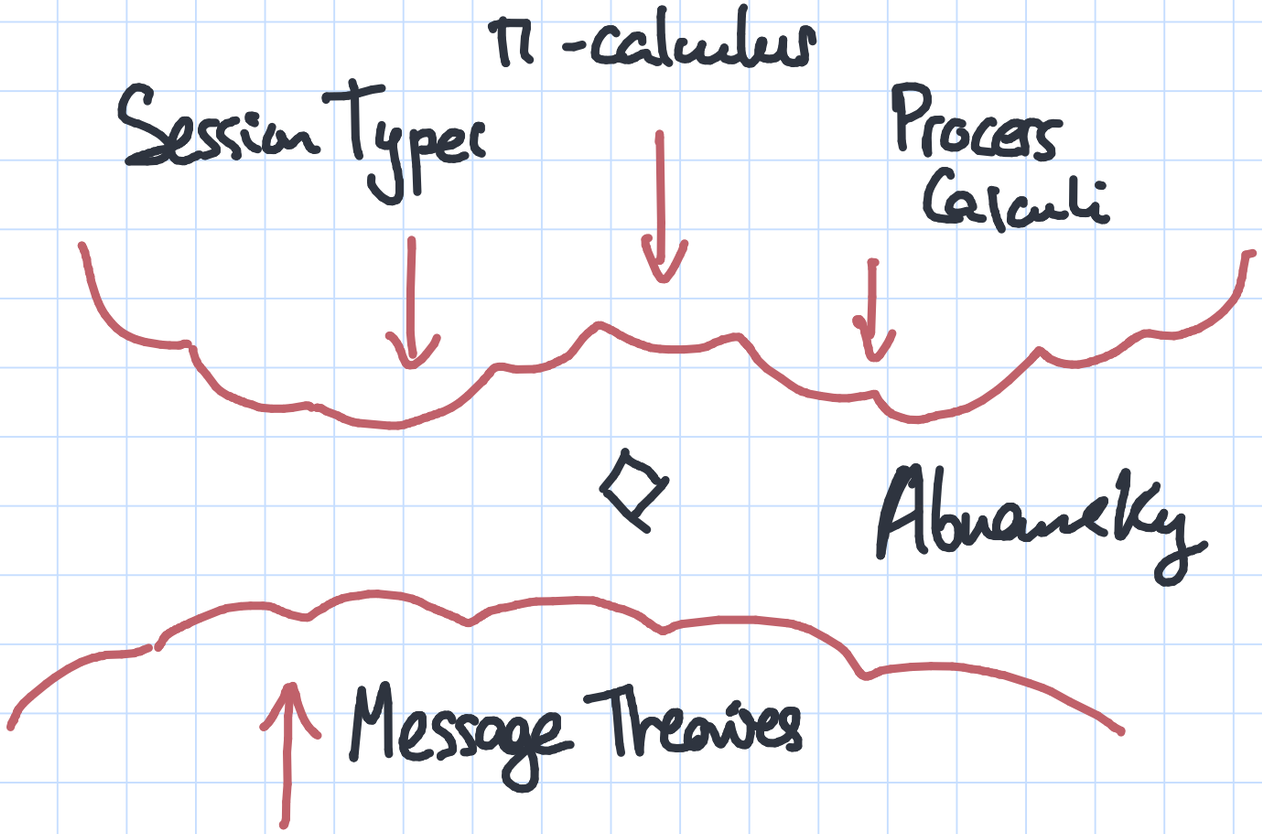
A duality is a pair of objects with two morphisms $(L \dashv R, \varepsilon, \eta)$ such that



"Polarization is left adjoint to taking left adjoints."

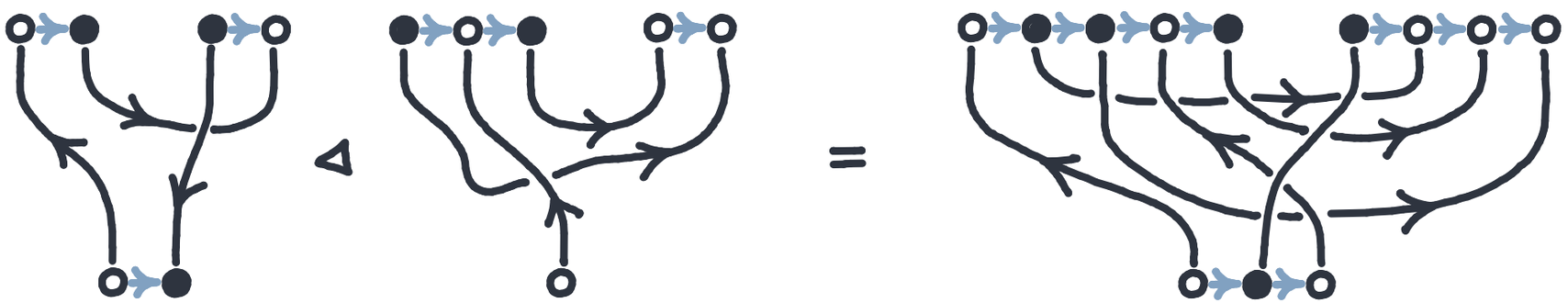
The free polarized monoidal category over a monoidal h

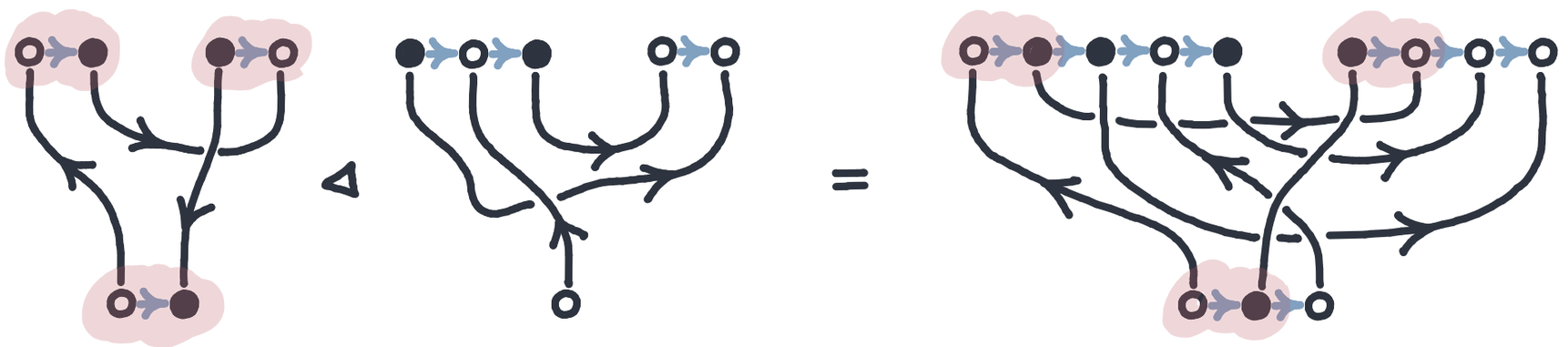


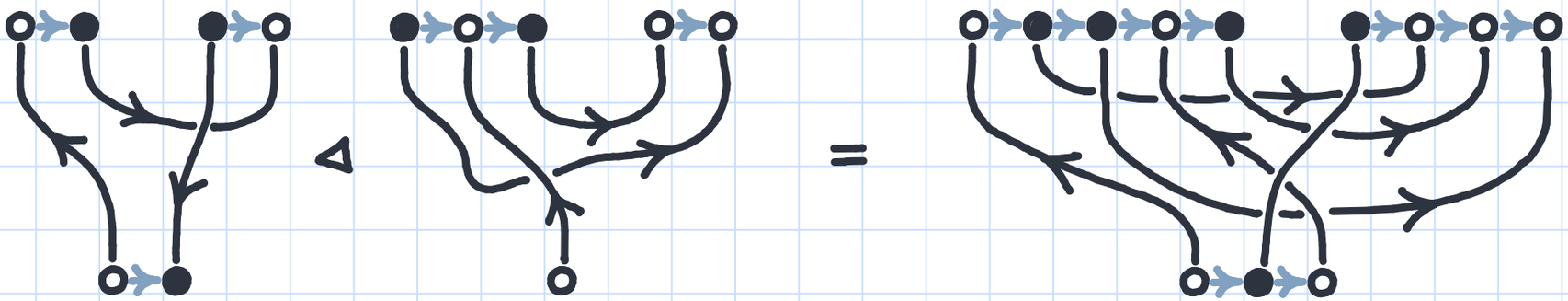


Is it correct? Expressive
Why this one?
Too successful!
Too easy to cook a new one!

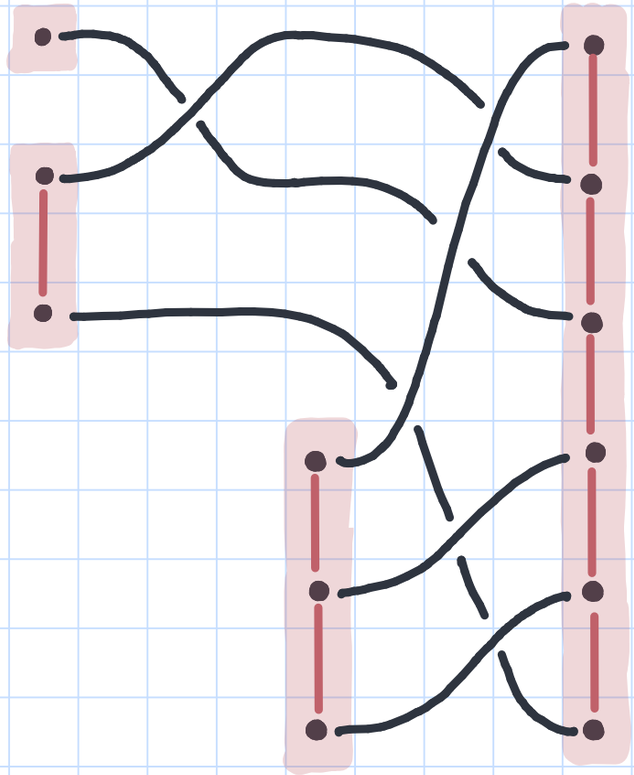
Why not more expressivity? Universally Characterized





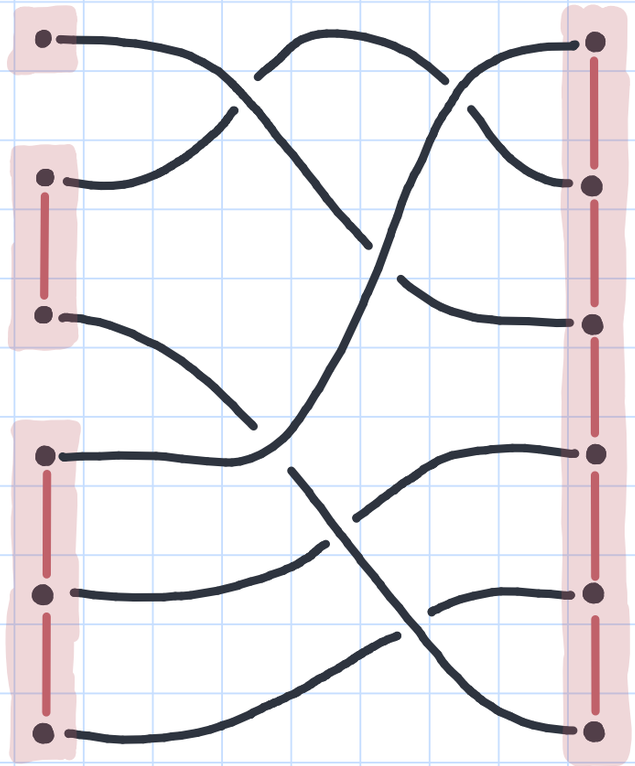


SHUFFLES

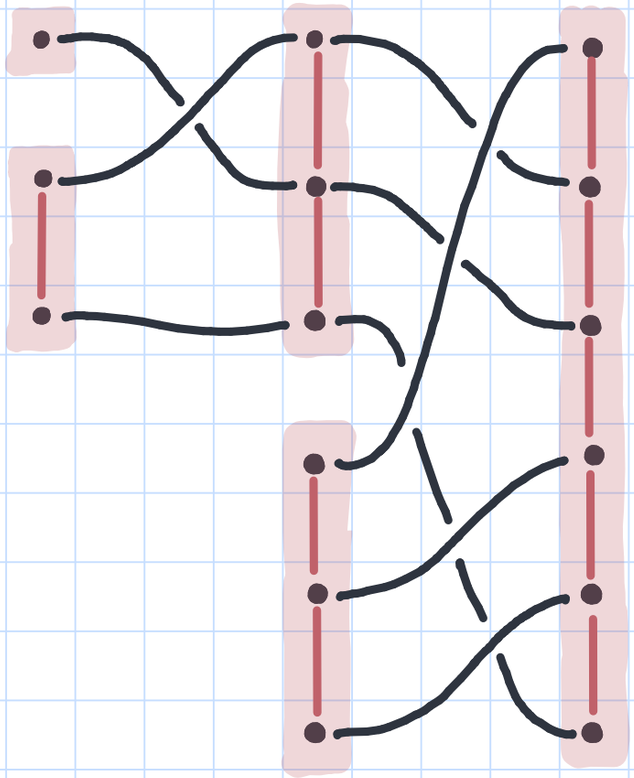


THEOREM. Shuffles form the free physical monoidal multicategory.

Shuffles

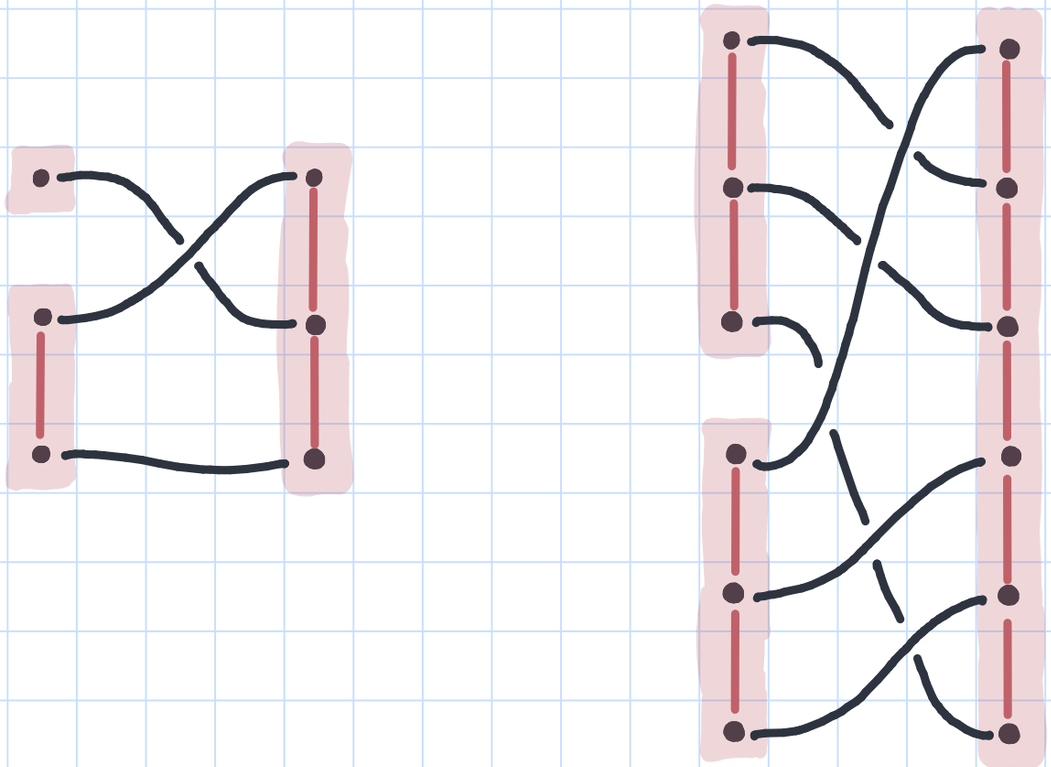


SHUFFLES



THEOREM. Shuffles form the free physical monoidal multicategory.

SHUFFLES



THEOREM. Shuffles form the free physical monoidal multicategory.