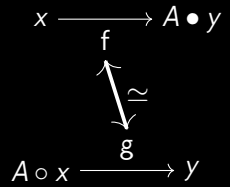


Quantum Message Passing Logic - Day 1

FMCS 2023, Sackville

Priyaa Varshinee Srinivasan

National Institute of Standards and Technology, USA

June 15, 2023

Motivation

Categorical quantum mechanics [1] [2] [3] $\rightarrow$ Diagrammatic framework and type system to describe and reason about quantum processes

Message passing logic [4] [5] $\rightarrow$ A categorical framework and type system to reason about concurrent (message-passing) systems

New!

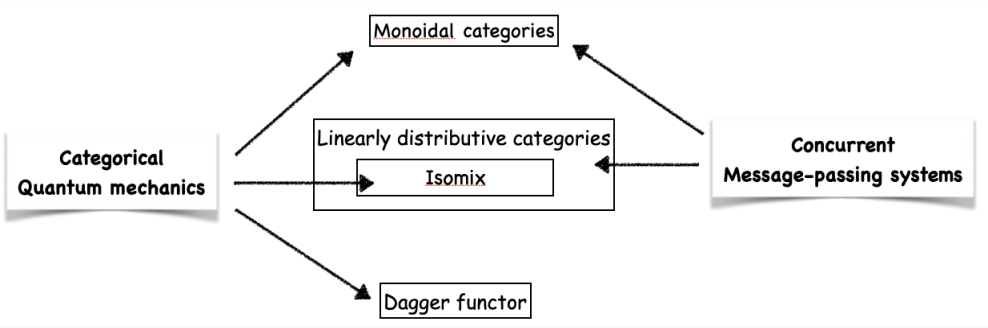
Categorical quantum mechanics + message passing: A type system to describe and reason about quantum concurrent message passing systems ?

[1] Abramsky and Coecke (2009) "A categorical semantics of quantum protocols"
[2] Selinger (2008) "Dagger Compact Closed Categories and Completely Positive Maps"
[3] Srinivasan (2021) "Dagger linear logic for categorical quantum mechanics (Ph.D. thesis)
[4] Coeckett and Pastro (2009) "Logic of Message passing"
[5] Kumar (2018) "Implementation of Message Passing Language" (Master's thesis)

A common framework of both concurrent and quantum systems

Linearly distributive and $*$-**autonomous categories** provide a common framework of both concurrent and quantum systems

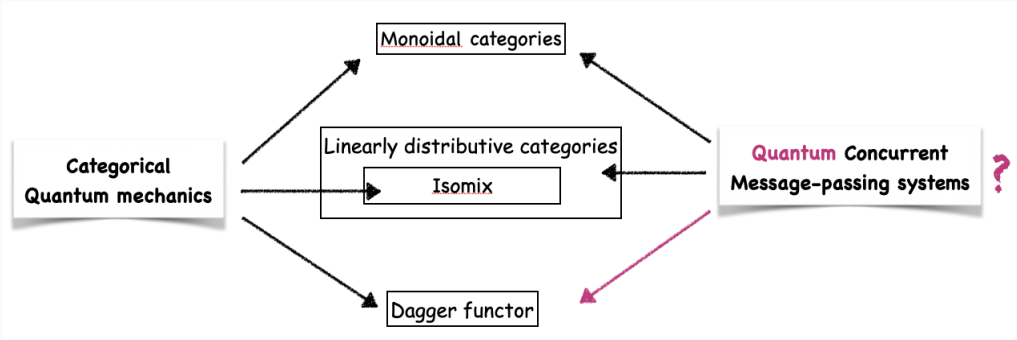
They are the categorical semantics of multiplicative linear logic



A common framework of both concurrent and quantum systems

Linearly distributive and $*$-**autonomous categories** provide a common framework of both concurrent and quantum systems

They are the categorical semantics of multiplicative linear logic



Whats on the menu?

Day 1: Concurrency and the categorical framework for message passing

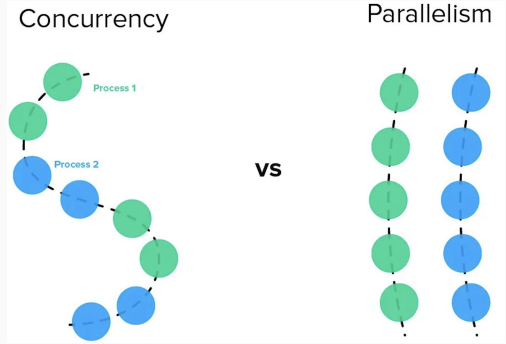
Day 2: Quantum mechanics and its categorical frameworks and
a possible categorical framework for quantum message passing systems
and an example

Message Passing Logic

Robin Cockett, and Craig Pastro. *The logic of message-passing* (2009)

Concurrency = Computation + Synchronization

Modeling a large computation as a collection of several interleaved smaller computations which synchronize with one another as necessary is called as **concurrency**.



Modes of synchronization

A crucial feature of concurrent programs is synchronization of its components.

Shared memory:

The processes synchronize with one another by accessing a common resource like a shared memory.

The synchronization is synchronous like two parties talking over a phone or interlocked gear wheels.

Message passing:

The processes synchronize with one another by passing messages with the communication being governed by a certain protocol (procedure).

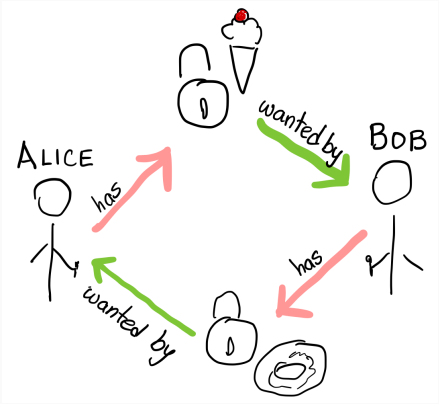
The synchronization is asynchronous like email.

Things that can go wrong with concurrency

Deadlock:

Each process is waiting for another process to release a resource.

Hence, progress is stalled.



Things that can go wrong with concurrency

Livelock:

Processes recover from being blocked only to block each other continuously.

Again, progress is stalled.

Analogous to two people stepping aside to allow each other to pass, never taking a step forward.

Race conditions:

Processes race to access a shared resource.

The order of the accesses determines the output of the program.

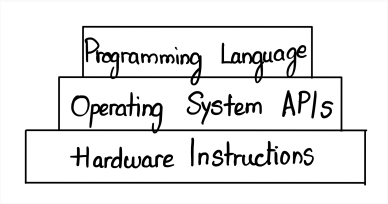
Program output in non-deterministic output

How do we make sense of concurrency?

We do not really make sense of it · · ·

No type system “in practice” to guarantee formal properties of concurrent programs.

How concurrency is widely facilitated in imperative programming languages?



How would one begin to develop a type system for concurrent programming?

Drawing inspiration from sequential programming world

Curry-Howard-Lambek isomorphism: a three-way isomorphism between types (in programming languages), proofs (in logic), and objects of a cartesian closed category.

Observes that proof systems and models of computation (through the *typed* λ-calculus) are identical formalisms.

Types are logical formulas (propositions) and proofs are programs.

Enables type-checking for sequential programs.

Prevents programmers from introducing type errors in programs.

The formality ensures that programs terminate or does not return incorrectly typed results.

Message passing logic

It is desirable to establish a proof system that will guarantee strong formal properties (e.g. programs being deadlock and livelock free).

Can we prove a Curry-Howard-Lambek-like isomorphism for concurrent programming?

Analogous to λ-calculus for sequential programs is the π-calculus for message passing.

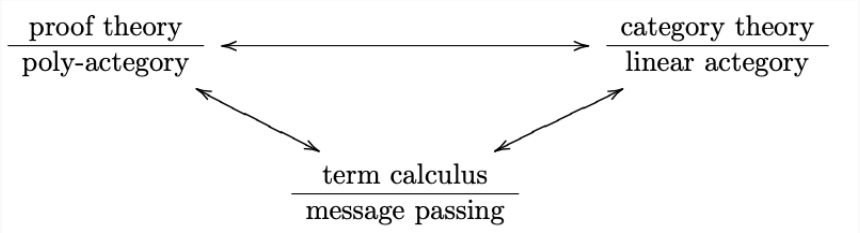
One may describe concurrent computations whose network configuration may change during the computation.

But, π-calculus does not have any type-theoretic underpinning and is purely based on operational semantics.

Message passing logic

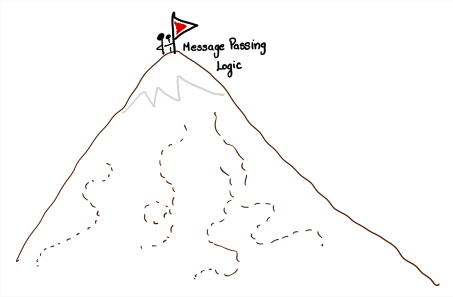
The aim of Cockett and Pastro's paper[6] is:

1. to develop a term calculus, a categorical semantics, and a proof system for message passing
2. to establish Curry-Howard-Lambek-like isomorphism for message passing



[6] Cockett and Pastro (2009) *The logic of message passing*

Caution: This paper is a hard hike!!!



Like most of Robin's other papers, this paper is a steep, almost non-scalable mountain nevertheless with multiple interesting trails to explore!

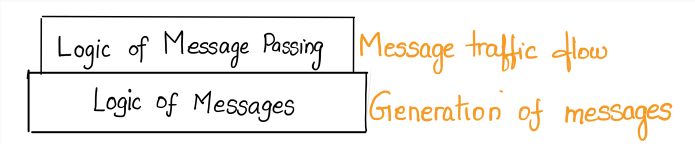
A group of five bright students joined my explorations at the adjoint school of Applied Category Theory Conference 2023 and we roamed around.

This talk shall be mainly be focused on the categorical semantics of message passing.

Message Passing Logic

Design of the message passing logic

Two-tiered logic: Logic of message passing built on top of the logic of messages



- the logic of messages whose proofs should be thought of as ordinary sequential programs

- the logic of message passing is concerned with manipulating the channels of communication, hence concurrency.

- Logic of sequential computation vs concurrent communication

Rationale: The separation of logic allows one to correctly deal with the complexities of both and the interplay between them.

Linear actegories

Linear actegories: an overview

- **Actegory:** A monoidal category acting on a category

- Linear actegory: A symmetric monoidal category acting on a linearly distributive category

  Monoidal category as the semantics for the logic of messages (sequential).

  Linearly distributive category as the semantics for the logic of message passing channels (concurrent).

  Linear acetgory itself as the semantics for the logic of message passing.

- Linearly distributive categories provide a categorical semantics of multiplicative linear logic.

Linear logic

In 1987, Girard introduced linear logic as a logic for resources manipulation.

Classical logic treats statements as truth values; linear logic treats statements as resources which cannot be duplicated or destroyed.

p : to spend a dollar

q : to buy an apple

"$p \Rightarrow q$" has the meaning that if a dollar is spent then an apple can be bought.

A person can either have a dollar or an apple at a given time but not both.

The word "linear" refers to this resource sensitivity of the logic.

Categorical semantics of multiplicative linear logic

Linear logic fragment	Connectives
Negation	$A^{\perp}$
Multiplicative	$(\otimes, 1)$ and $(\parr, \perp)$
Additive	$(\&, \top)$ and $(\oplus, 0)$
Exponentials	! and ?

Negation:

p: to spend a dollar $\quad$ $p^{\perp}$: to receive a dollar

Multiplicative fragment:

$(\otimes, 1)$: an apple $\otimes$ an orange ($p \Rightarrow q \otimes r$ means access to resources at the same time; conjunction from classical logic)

$(\parr, \perp)$ = De Morgan's Rule: $(A^{\perp} \otimes B^{\perp})^{\perp}$ (means don't have access to either resource, so someone else owns it; disjunction from classical logic)

Categorical semantics of multiplicative linear logic

Linear logic fragment	Connectives
Negation	$A^{\perp}$
Multiplicative	$(\otimes, 1)$ and $(\parr, \perp)$
Additive	$(\&, \top)$ and $(\oplus, 0)$
Exponentials	! and ?

Linear logic fragment	Categorical proof theory
MLL $(\otimes, 1)$ and $(\parr, \perp)$	Linearly distributive categories[7] $(\mathbb{X}, \otimes, \top, \oplus, \perp)$
MLL with negation	$*$-autonomous categories[8]
Compact MLL $(\otimes = \parr, 1 = \perp)$	Monoidal categories $(\mathbb{X}, \otimes, I)$
Compact closed categories[9]	Compact MLL with negation

[7] Cockett and Seely (1997) "Weakly Distributive Categories"

[8] Barr (1991) "$*$-autonomous categories and linear logic"

Linearly distributive categories

Linearly distributive categories (LDCs)[10]**:**

$$(\mathbb{X}, \otimes, \top, a_{\otimes}, u^{L}_{\otimes}, u^{R}_{\otimes}) \qquad (\mathbb{X}, \oplus, \bot, a_{\oplus}, u^{L}_{\oplus}, u^{R}_{\oplus})$$

linked by linear distributors:

$$\partial_L : A \otimes (B \oplus C) \to (A \otimes B) \oplus C$$

$$A \times (B + C) \simeq (A \times B) + (A \times C)$$

Intuition: At a restaurant the waiter, A, can choose to address either person at the table, B or C. Once assigned to B, A cannot choose C.

The distributor is not an equality or isomorphism in general!

Monoidal categories: LDCs in which $\otimes = \oplus$; $\top = \bot$

[10] Cockett and Seely (1997) "Weakly distributive categories"

Why linearly distributive categories are a good choice for the logic of channels?

The key is in the graphical calculus of these categories.

Linearly distributive categories come equipped with a graphical calculus which subsumes the graphical calculus of monoidal categories.

In the graphical calculus of *monoidal categories* every possible way of composing its building blocks is valid, resulting in cyclic graphs of sequents – allowing cyclic dependency.

The graphical calculus of LDCs disallows such cyclic dependency.

How?

Linear logic is a two-sided logic

Γ and Δ are collections of resources.

$$\Gamma_1, A, B, \Gamma_2 \vdash \Delta$$

Given a "sequent", the turnstile ($\vdash$) means we can infer.

To combine terms on the left, we tensor ($\otimes$) them.

$$\otimes\text{L}\ \frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, A \otimes B, \Gamma_2 \vdash \Delta}$$

Meaning, we need A and B to produce Δ.

Note: To combine terms on the right, we par ($\oplus$) them.

Linear logic is two-sided

Combining A and B when they are independent ($\otimes$ on the right):

$$\frac{\Gamma_1 \vdash \Gamma_2, A, \Gamma_3 \qquad \Delta_1 \vdash \Delta_2, B, \Delta_3}{\Gamma_1, \Delta_1 \vdash \Gamma_2, \Delta_2, A \otimes B, \Gamma_3, \Delta_3} \otimes\mathrm{R}$$

Interpretation: if we have Γ_1, then we can get Γ_2, A, and Γ_3.

Exchange rule

Exchange rules allow the neighboring premises and antecedents to be swapped.

$$(exch.L)\ \frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, B, A, \Gamma_2 \vdash \Delta}$$

$$(exch.R)\ \frac{\Gamma \vdash \Delta_1, C, D, \Delta_2}{\Gamma \vdash \Delta_1, D, C, \Delta_2}$$

Cut rule

Cut Rule: Formulas can be "cut out" and their respective derivations joined.

$$\frac{B \oplus C \vdash B, C \qquad A, B \vdash A \otimes B}{A, B \oplus C \vdash A \otimes B, C}$$

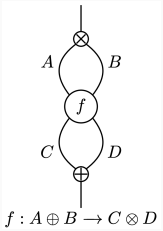
Observation 1: Cut rule lets us combine boxes (graphical calculus) in a sequence (like composition).

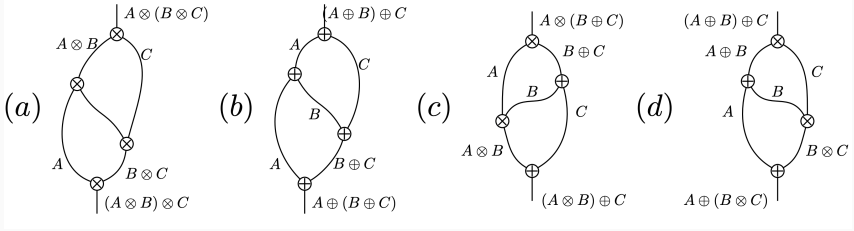
Observation 2: Linear distributivity can be derived from these sequent rules.

Graphical calculus of LDCs: Building blocks

In the graphical calculus for LDCs, wires represent objects and circles represent morphisms. Input wires of a morphism are tensored (with $\otimes$), and the output wires are "par"ed (with $\oplus$).

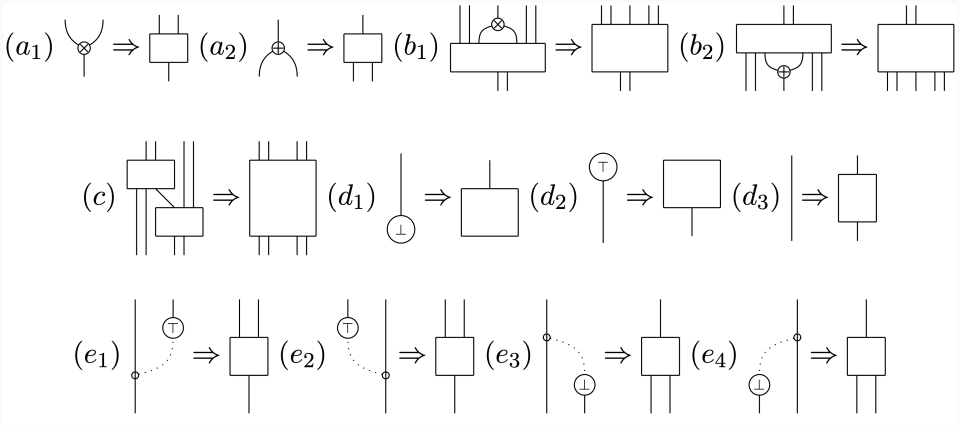
For instance, $f : A \otimes B \to C \oplus D$, the $\otimes$-associator, the $\oplus$-associator, δ_L^R, and δ_R^R are drawn as follows:





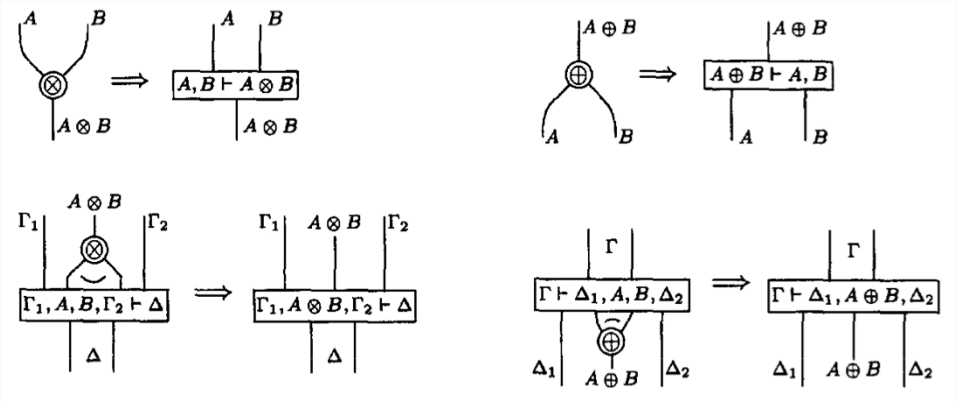
Boxing Algorithm

An LDC circuit is *valid* if and only if the entire circuit can be "boxed" using the rules below [11].



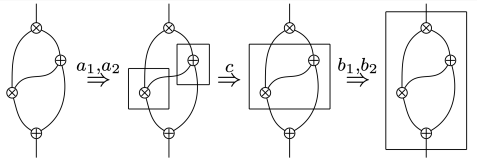
[11] Blute, Cockett, Seely, Trimble (1996) "Natural deduction and coherence for weakly distributive categories"

Boxing Algorithm: Sequentialization



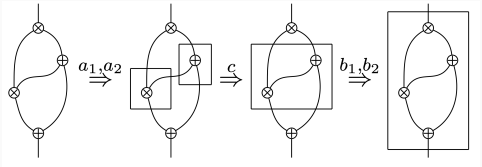
Verifying that δ_L is Valid

Applying the boxing algorithm to δ_L, we can that the circuit can be *encapsulated by one box*, making it a valid circuit.

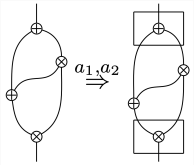


Verifying that δ_L is Valid

Applying the boxing algorithm to δ_L, we can that the circuit can be *encapsulated by one box*, making it a valid circuit.



But the reverse cannot because a $\otimes$-elimination cannot be absorbed by a box above and $\oplus$-introduction cannot be absorbed by a box below.



What does the non-cyclicity imply?

What would the non-cyclicity property imply for the message passing semantics?

What does the non-cyclicity imply?

What would the non-cyclicity property imply for the message passing semantics?

The type system for process communication based on LDCs will avoid cyclic dependency hence will guarantee deadlock and livelock free programs !!!

Linear actegories

Let $(\mathbb{A}, *, I)$ be a symmetric monoidal category.

A symmetric linear $\mathbb{A}$-actegory consists the following data:

- A symmetric linearly distributive category $(X, \otimes, \top, \oplus, \bot)$

- Functors

$$\circ : \mathbb{A} \times \mathbb{X} \to \mathbb{X} \qquad \bullet : \mathbb{A}^{\mathrm{op}} \times \mathbb{X} \to X$$

such that for all $A \in \mathbb{A}$, $A \circ - \dashv A \bullet -$

- 6 natural isomorphisms

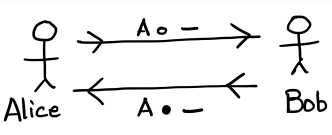
- 3 natural transformations

○ is left-parametrised left adjoint of ●

The functors encode the inherent duality in the direction of information flow between two connected parties!

$$\circ : \mathbb{A} \times \mathbb{X} \to \mathbb{X} \qquad \bullet : \mathbb{A}^{op} \times \mathbb{X} \to X$$

for all $A \in \mathbb{A}$, $A \circ -$ is left adjoint of $A \bullet -$



- ○ means send value to the right and receive from the left
- ● means send value to the left and receive from the right

$\circ$ is left-parametrised left adjoint of $\bullet$

For all $A \in \mathbb{A}$, $A \circ -$ is left adjoint to $A \bullet -$:

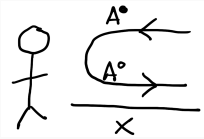


Figure 1: $\eta_X : X \to A \bullet (A \circ X)$

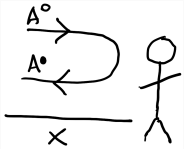


Figure 2: $\epsilon_X : A \circ (A \bullet X) \to X$

Natural isormorphisms of linear actegories

For all $A, B \in \mathbb{A}$ and $X, Y \in \mathbb{X}$ natural isomorphisms in $\mathbb{X}$:

$$u_\circ : I \circ X \to X \qquad u_\bullet : X \to I \bullet X$$

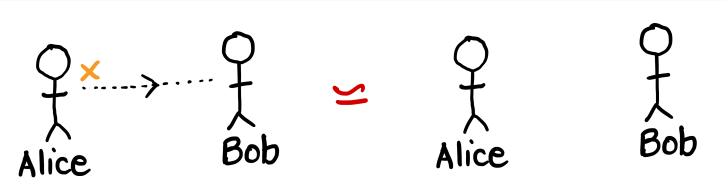
$$a^*_\circ : (A * B) \circ X \to A \circ (B \circ X) \qquad a^*_\bullet : A \bullet (B \bullet X) \to (A * B) \bullet X$$

$$a^\circ_\otimes : A \circ (X \oplus Y) \to (A \circ X) \circ Y \qquad a^\bullet_\oplus : (A \bullet X) \oplus X \to A \bullet (X \oplus Y)$$

Natural isomorphisms of linear actegories

For all $A, B \in \mathbb{A}$ and $X, Y \in \mathbb{X}$ natural isomorphisms in $\mathbb{X}$:

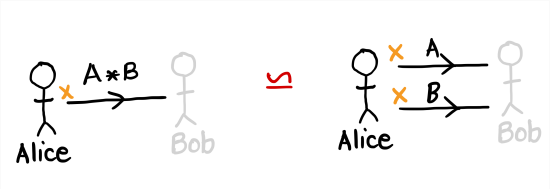
$$u_\circ : I \circ X \to X \qquad u_\bullet : X \to I \bullet X$$



Natural isomorphisms of linear actegories

For all $A, B \in \mathbb{A}$ and $X, Y \in \mathbb{X}$ natural isomorphisms in $\mathbb{X}$:

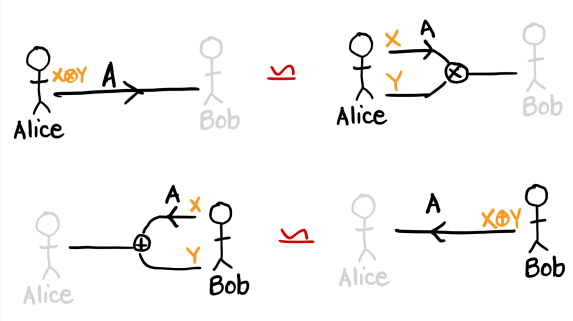
$$a^*_\circ : (A * B) \circ X \to A \circ (B \circ X) \qquad a^*_\bullet : A \bullet (B \bullet X) \to (A * B) \bullet X$$



Natural isormorphisms of linear actegories

For all $A, B \in \mathbb{A}$ and $X, Y \in \mathbb{X}$ natural isomorphisms in $\mathbb{X}$:

$$a^{\circ}_{\otimes} : A \circ (X \otimes Y) \to (A \circ X) \circ Y \qquad a^{\bullet}_{\oplus} : (A \bullet X) \oplus X \to A \bullet (X \oplus Y)$$



Natural transformations of linear actegories

For all $A, B \in \mathbb{A}$ and $X, Y \in \mathbb{X}$ there exists following natural transformations:

$$d^{\circ}_{\oplus} : A \circ (X \oplus Y) \to (A \circ X) \oplus Y$$

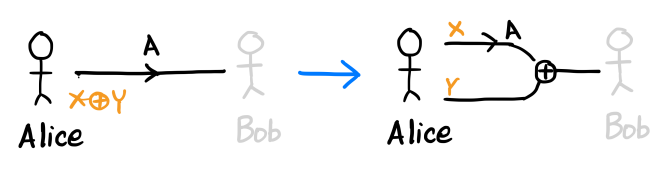
$$d^{\bullet}_{\otimes} : (A \bullet X) \otimes Y \to A \bullet (X \otimes Y)$$

$$d^{\circ}_{\bullet} : A \circ (B \bullet X) \to B \bullet (A \circ Y)$$

Natural transformations of linear actegories

For all $A, B \in \mathbb{A}$ and $X, Y \in \mathbb{X}$ there exists following natural transformations:

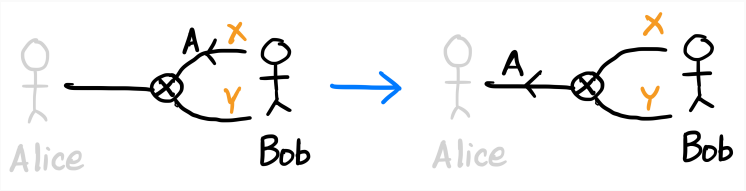
$$d^{\circ}_{\oplus} : A \circ (X \oplus Y) \to (A \circ X) \oplus Y$$



Natural transformations of linear actegories

For all $A, B \in \mathbb{A}$ and $X, Y \in \mathbb{X}$ there exists following natural transformations:

$$d^{\bullet}_{\otimes} : (A \bullet X) \otimes Y \to A \bullet (X \otimes Y)$$

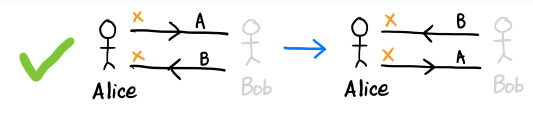


Natural transformations of linear actegories

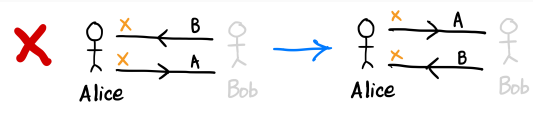
For all $A, B \in \mathbb{A}$ and $X, Y \in \mathbb{X}$ there exists following natural transformations:

$$d^{\circ}_{\bullet} : A \circ (B \bullet X) \to B \bullet (A \circ Y)$$

Value of A **independent** of value of B



Value of A **may be dependent** of value of B



Programming syntax - cut

$$\text{cut}\ \frac{s\quad t}{\text{on } \alpha = \beta \textbf{ plug } s \text{ to } t}$$

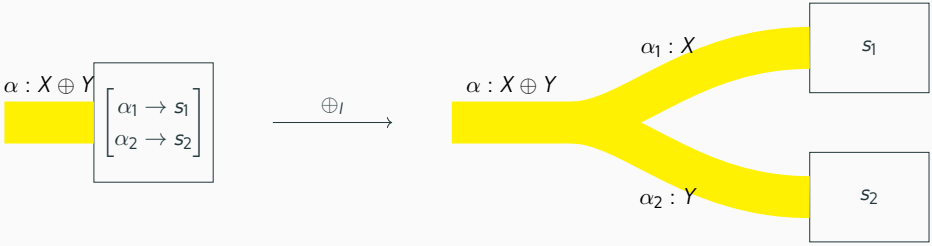
Programming syntax - $\otimes_l / \oplus_r$

$$\otimes_l \ \frac{s}{\textbf{split } \alpha \text{ as } \alpha_1, \alpha_2; s}$$

Have the channel on the right side and replace $\otimes$ with $\oplus$ and you get $\oplus_r$

Programming syntax - $\otimes_r/\oplus_l$

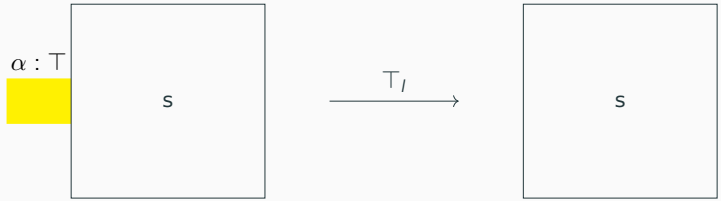
$$\otimes_r \quad \frac{s_1 \quad s_2}{\textbf{fork}\ \alpha\ \text{as} \begin{array}{l} \mid \alpha_1 \to s_1 \\ \mid \alpha_2 \to s_2 \end{array}}$$



Have the channel on the right side and replace $\oplus$ with $\otimes$ and you get $\otimes_r$

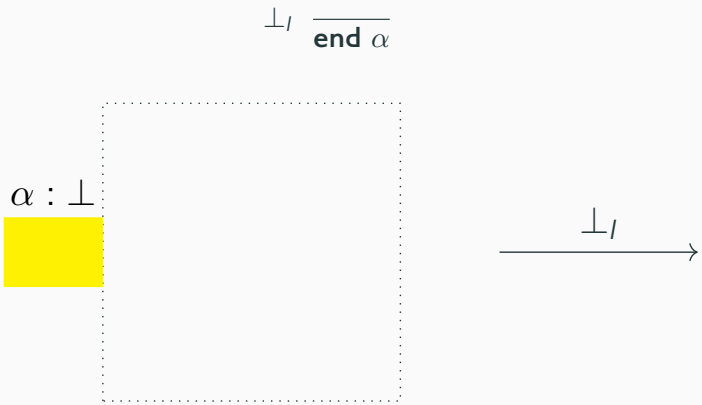
Programming syntax - $\top_l/\bot_r$

$$\top_l \quad \frac{s}{\textbf{close } \alpha; s}$$



Have the channel on the right side and replace $\top$ with $\bot$ and you get $\bot_r$

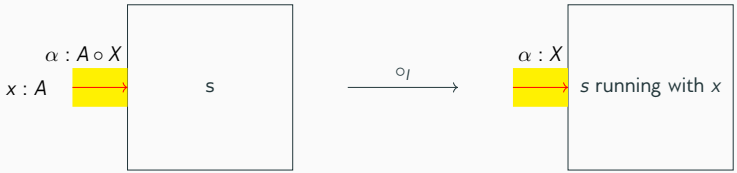
Programming syntax - $\bot_l/\top_r$



Have the channel on the right side and replace $\bot$ with $\top$ and you get $\top_r$

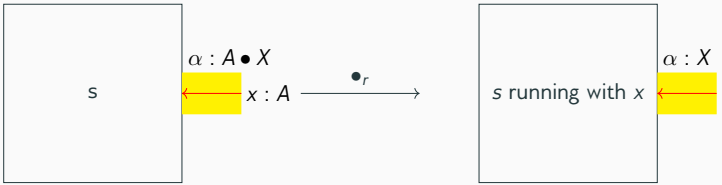
Programming syntax - $\circ_I$

$$\circ_I \quad \frac{s}{\textbf{get}\ \alpha \quad x \Rightarrow s}$$



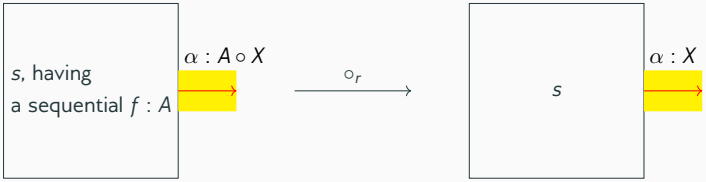
Programming syntax - $\bullet_r$

$$\circ_I \ \frac{s}{\text{get } \alpha \quad x \Rightarrow s}$$



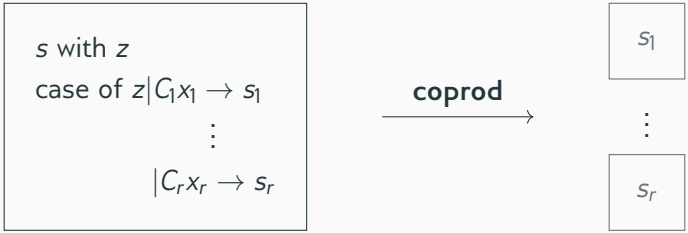
Programming syntax - $\bullet_l/\circ_r$

$$\bullet_l \quad \frac{f \quad s}{\text{put } \alpha \quad f; s}$$



Send the output to the left, replace $\circ$ with $\bullet$ and you get $\bullet_l$

Programming syntax - coprod



Summary: Linear actegories

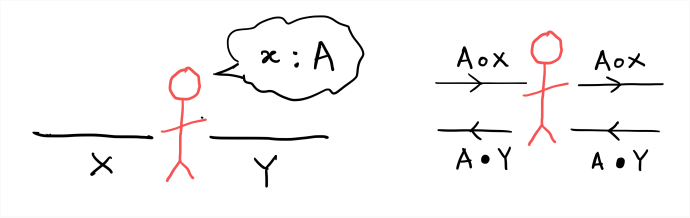
Loosely, a linear actegory is a symmetric monoidal category acting on a linearly distributive category on the left and the right.

Let $(\mathbb{A}, *, I)$ be a symmetric monoidal category. A symmetric linear $\mathbb{A}$-actegory consists the following data:

- A symmetric linearly distributive category $(X, \otimes, \top, \oplus, \bot)$

- Functors

$$\circ : \mathbb{A} \times \mathbb{X} \to \mathbb{X} \qquad \bullet : \mathbb{A}^{\mathrm{op}} \times \mathbb{X} \to \mathbb{X}$$



Summary: Linear actegories (cont...)

For all $A \in \mathbb{A}$, $A \circ -$ is left adjoint to $A \bullet -$:

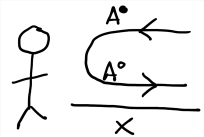
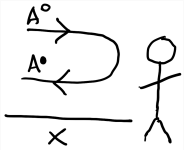


Figure 3: $\eta_X : X \to A \bullet (A \circ X)$



$\epsilon_X : A \circ (A \bullet X) \to X$

Linear actegories: an analogy

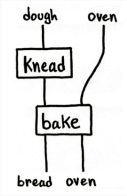
The network of roads	::	Linearly distributive cats
Market, school, house..	::	Monoidal categories
Vehicles	::	Messages
Getting vehicles on to the road?	::	Adjoint functors: $A \circ - \dashv A \bullet -$

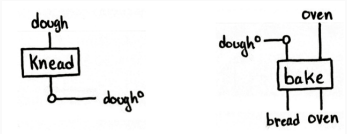
A Toy Model of Linear actegories: Chad Nester's *Concurrent process histories*

Concurrent process histories and resource transducers

Chad Nester (2022)

Overview





Motivation:

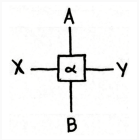
Capture the *movement* of resources or information across different components of a concurrent process

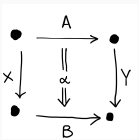
Methodology:

- Considers the concurrent process as a resource theory (strict symmetric monoidal category)
- Augments the string diagrams for symmetric monoidal categories with corners
- Resources flow between different components of the systems through the corners

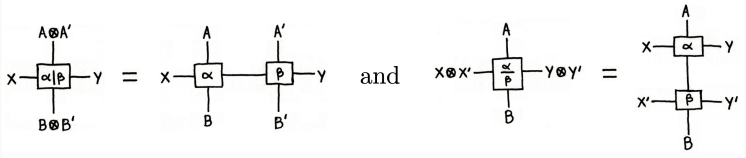
Single object double category

Start with single object double category





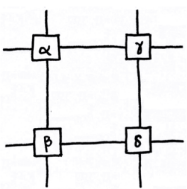
When cells α & β have appropriate matching boundary types, we can have **horizontal composition** $\alpha|\beta$ or **vertical composition** $\frac{\alpha}{\beta}$ as defined below.



Single object double category: Interchange rule

This horizontal & vertical composition needs to satisfy the interchange rule

$$\frac{\alpha}{\beta}|\frac{\gamma}{\delta} = \frac{\alpha|\gamma}{\beta|\delta}$$

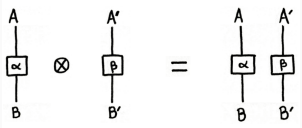


Monoidal categories

* We shall refer to the horizontal 1-cells as **sequential** 1-cells, and vertical 1-cells as **parallel** 1-cells.

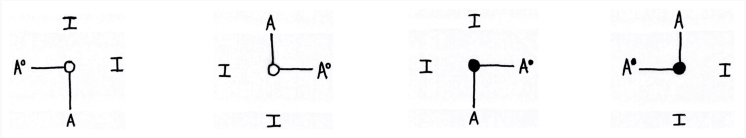
Given a single object double category $\mathbb{A}$, we get two strict monoidal categories, $\mathbf{S}(\mathbb{A})$, $\mathbf{P}(\mathbb{A})$ as follows:

The tensor product of sequential morphisms given as follows:

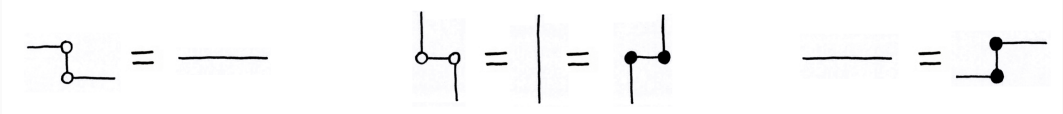


Adding Cornerings

A single object double category is a pro-arrow equipment if for every horizontal 1-cell A, there exists parallel 1-cells A° and $A^\bullet$ with $A^\circ \neq A^\bullet$ and the following 2-cells:



called $\circ$-corners and $\bullet$-corners, respectively, which satisfy the yanking equations:



Resource theories as strict monoidal categories

This work considers a concurrent process as a resource theory (strict symmetric monoidal category).

A strict symmetric monoidal category can be interpreted as resource theory [12] -

Objects: Resources

Arrows: Resource transformations that can be implemented without any cost

Composition: Sequential composition of resource transformations

Tensor: Parallel composition of resources and transformations

Unit object: Trivial resource

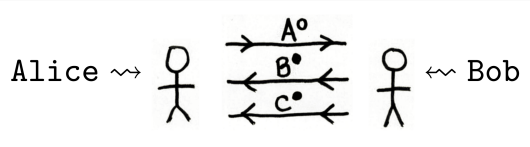
[12] Coecke, Spekkens, Fritz (2013) "A mathematical theory of resources"

$\mathbb{A}$-valued exchanges

Given a resource theory (i.e., a strict SMC) $\mathbb{A}$, the *A*-valued exchanges $\mathbb{A}^{\circ\bullet}$ is the free monoid on the set $(\mathrm{ob}(\mathbb{A}) \times \{\circ, \bullet\})$, whose elements are denoted by A° and $A^{\bullet}$.

The monoid is NOT commutative.

Intuitively, elements of $\mathbb{A}^{\circ\bullet}$ of describe a sequence of resources moving between participants in the exchange.



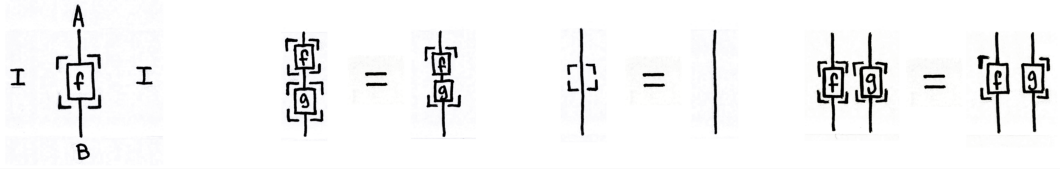
Free cornering of $\mathbb{A}$

Given a resource theory (i.e., a strict SMC) $\mathbb{A}$, the free cornering of $\mathbb{A}$ is a proarrow equipment $\ulcorner \mathbb{A} \urcorner$ where,

$\ulcorner \mathbb{A} \urcorner_H$: are objects of $\mathbb{A}$ (Resources)

$\ulcorner \mathbb{A} \urcorner_V$ are elements of $\mathbb{A}$-valued exchange monoid (Sequence of resources on the move)

Generating 2 -cells: the $\circ$-corners and $\bullet$-corners along with a vertical cell $\ulcorner f \urcorner$ for each $f : A \to B$ subject to the equations:

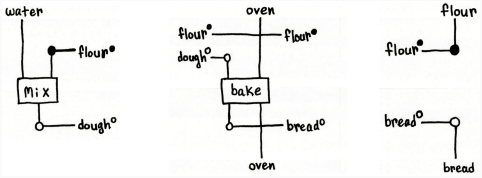


Free cornering captures concurrency

For $\ulcorner \mathbb{A} \urcorner$, we interpret

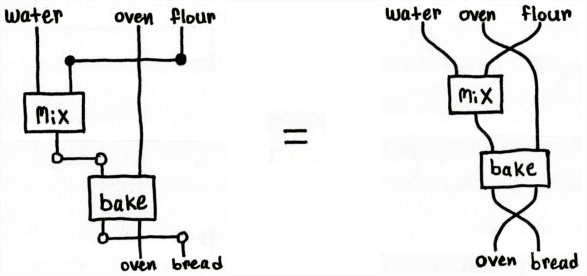
1. $\ulcorner \mathbb{A} \urcorner_{\mathsf{H}} =$ as resources
2. $\ulcorner \mathbb{A} \urcorner_{\mathsf{V}} = \mathbb{A}^{\circ\bullet}$ exchange of resources from one party to the other.
3. cells of $\ulcorner \mathbb{A} \urcorner$ as **concurrent transformations**. Each cell describes a way to transform the collection of resources given by the top boundary into that given by the bottom boundary via participating in A-exchanges along the left and right boundaries.

For example, in the free cornering of our bread category,



$\mathbf{S}[\mathbb{A}]$ is isomorphic to the resource theory

Combining the prior three bread concurrent transformations, we get



Theorem: *There is an isomorphism of categories* $\mathbf{S}[\mathbb{A}] \cong \mathbb{A}$.

$\mathbf{P}\lceil\mathbb{A}\rceil$ the category of resource transducers

Consider the category $\mathbf{P}\lceil\mathbb{A}\rceil$ as the category of resource transducers.

Lemma: There are strong monoidal functors $(-)^\circ : \mathbb{A} \to \mathbf{P}\lceil\mathbb{A}\rceil$ and $(-)^\bullet : \mathbb{A}^{op} \to \mathbf{P}\lceil\mathbb{A}\rceil$ defined respectively on $f : A \to B$ of $\mathbb{A}$ by:

$\mathbf{P}\lceil\mathbb{A}\rceil$ is a linear actegory

For the category of resource transducers $\mathbf{P}\lceil\mathbb{A}\rceil$ define -

$$\circ : \mathbb{A} \times \mathbf{P}\lceil\mathbb{A}\rceil \to \mathbf{P}\lceil\mathbb{A}\rceil; f \circ h = f^{\circ} \otimes h$$
$$\bullet : \mathbb{A}^{\mathrm{op}} \times \mathbf{P}\lceil\mathbb{A}\rceil \to \mathbf{P}\lceil\mathbb{A}\rceil; f \bullet h = f^{\bullet} \otimes h$$

for all resource transducer $h \in \mathbf{P}\lceil\mathbb{A}\rceil$.

Lemma In $\mathbf{H}\lceil\mathbb{A}\rceil$, for each A, the functors $A \circ -$ is left adjoint to $A \bullet -$:

$\mathbf{P}[\mathbb{A}]$ is a linear actegory (cont...)

We seek families of morphisms $\eta_{A,X} : X \to A \bullet (A \circ X)$ and $\varepsilon_{A,X} : A \circ (A \bullet X) \to X$ in $\mathbf{P}[\mathbb{A}]$ that satisfy the triangle identities. Define $\eta_{A,X}$ and $\epsilon_{A,X}$, repressively, by

By the yanking equations, the triangles identities hold, as shown below.

Theorem: Let $\mathbb{A}$ be a resource theory. Then, $\mathbf{P}[\mathbb{A}]$ is a linear actegory.

Acknowledgement

Thanks to Brandon Baylor, Durgesh Kumar, Fabian Wesner, Isaiah B. Hilsenrath, Paige Frederick, Rose Kudzman-Blais from the ACT adjoint school 2023 for all the fun discussions! Thanks to Fabian for the illustration of the programming syntax!