Semantics for Non-Determinism in Categorical Message Passing Language by Sup-Lattice Enrichment

Alexanna (Xanna) Little - FMCS 2023
Undergraduate Research at the University of Calgary with Dr. Robin Cockett

Semantics for Non-Determinism in Categorical Message Passing Language by Sup-Lattice Enrichment

Alexanna (Xanna) Little - FMCS 2023
Undergraduate Research at the University of Calgary with
Dr. Robin Cockett

1. Language

I have been working on undergraduate research with Dr. Robin Cockett for over a year now contributing to the development of the programming language called **Categorical Message Passing Language (CaMPL)**.

CaMPL is a Functional Programming Language.

Functional programming **focuses on what** programs should do **rather than how** they should perform a computation, so programs are **defined by input and output types** and other **details are abstracted away**.

A functional program can be written by composing other programs (also called functions):

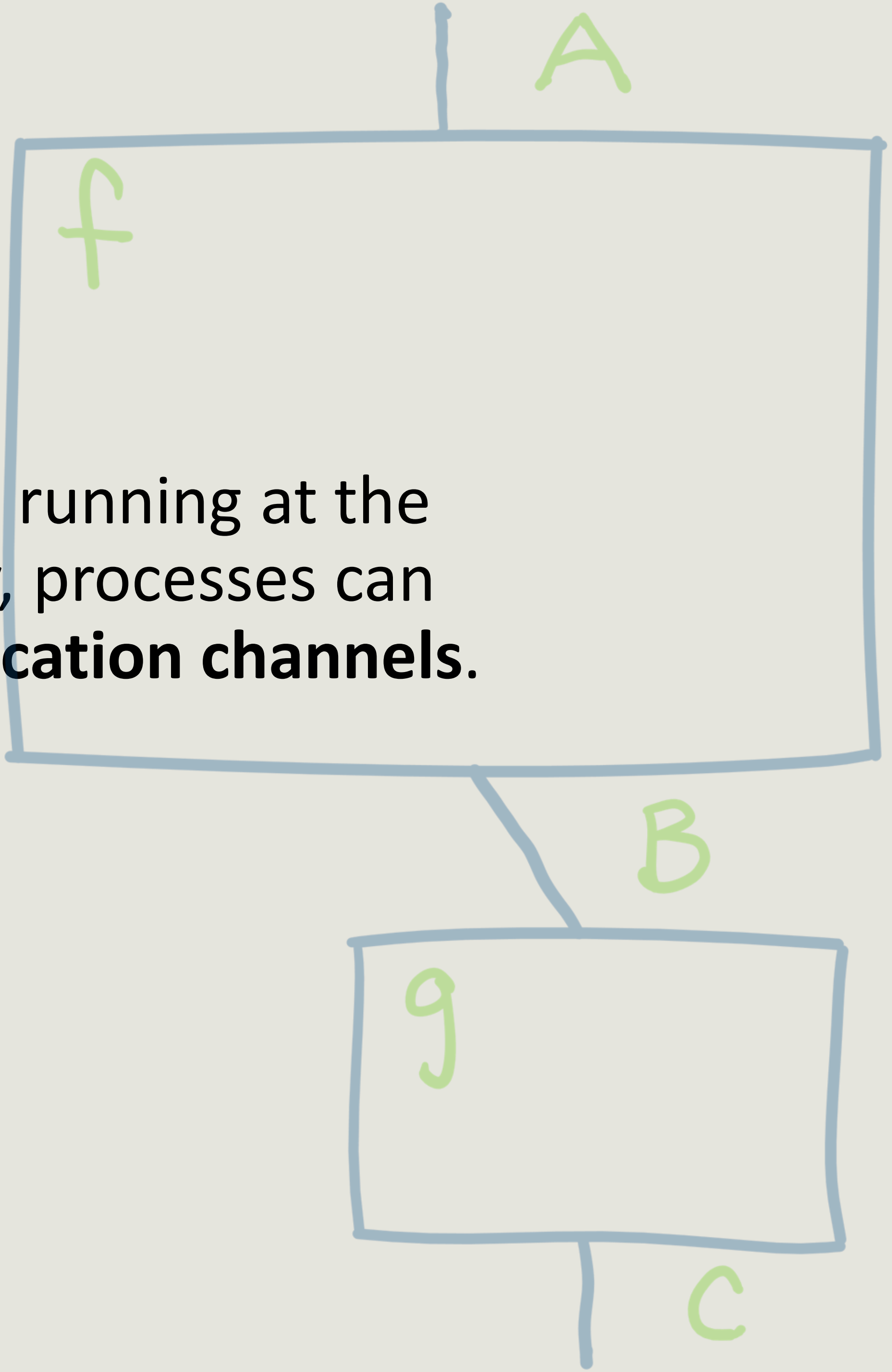
$$X := \overbrace{\underbrace{A \xrightarrow{f} B}_{\text{function } f}\;\underbrace{\xrightarrow{g} C}_{\text{function } g}}^{\text{program } X}$$

CaMPL is a Concurrent Programming Language.

Concurrent programs have **multiple processes** running at the same time, and, in some forms of concurrency, processes can communicate with each other along **communication channels**.

CaMPL has **two kinds of functions**:

- **Concurrent processes** defined on concurrent channel types.
- **Sequential functions** defined on sequential input and output types that are run by the processes.



Semantics for Non-Determinism in Categorical Message Passing Language by Sup-Lattice Enrichment

Alexanna (Xanna) Little - FMCS 2023
Undergraduate Research at the University of Calgary with
Dr. Robin Cockett

2. Categorical

The semantics for programming features in CaMPL are defined using categorical structures.

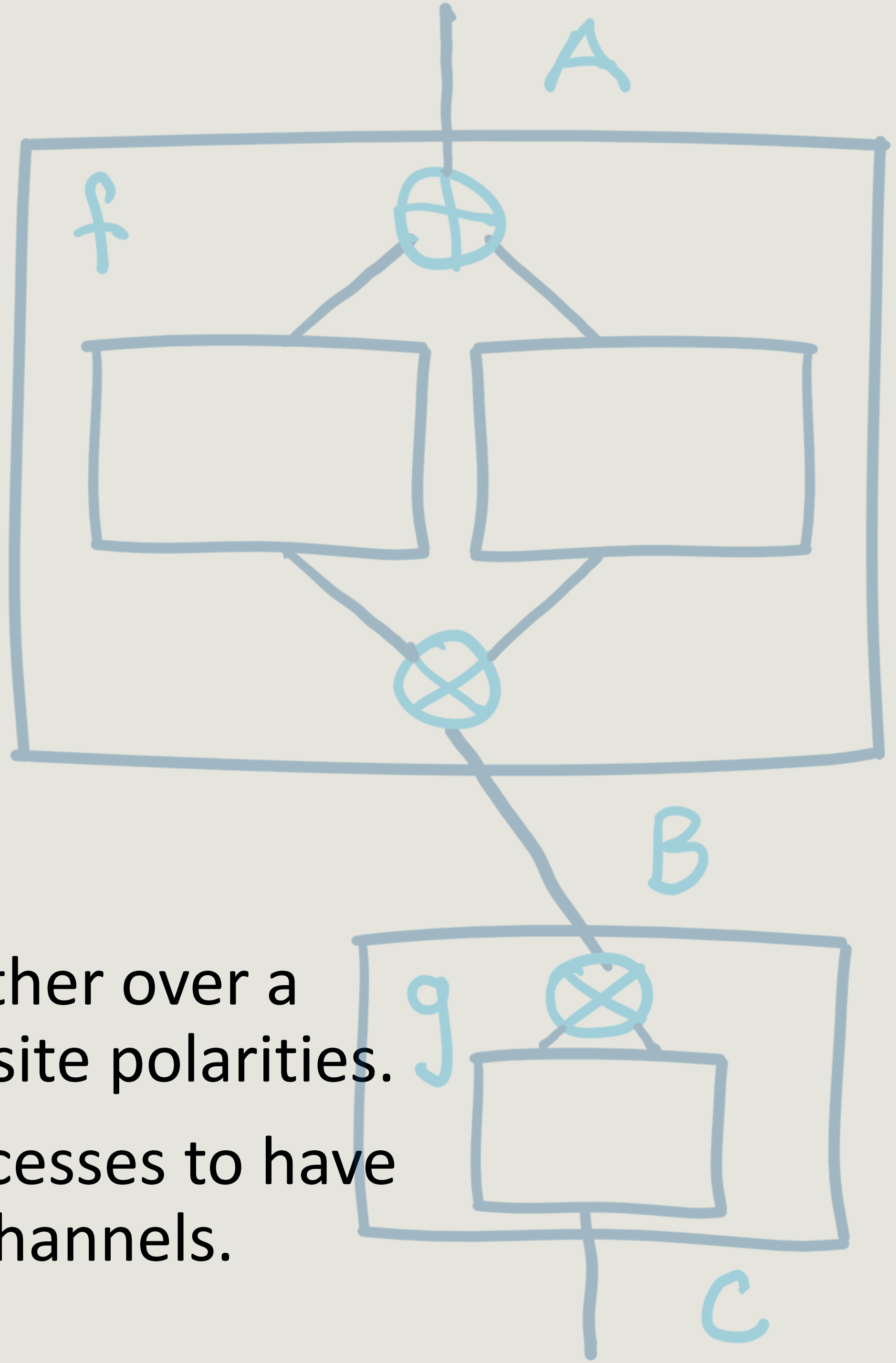
What a CaMPL program does is defined using category theory.

Category theory is a useful mathematical foundation for functional programming because a **function is defined as a map** between objects that are its input and output types.

Concurrent part

The categorical semantics is defined by a **linearly distributive category** where:

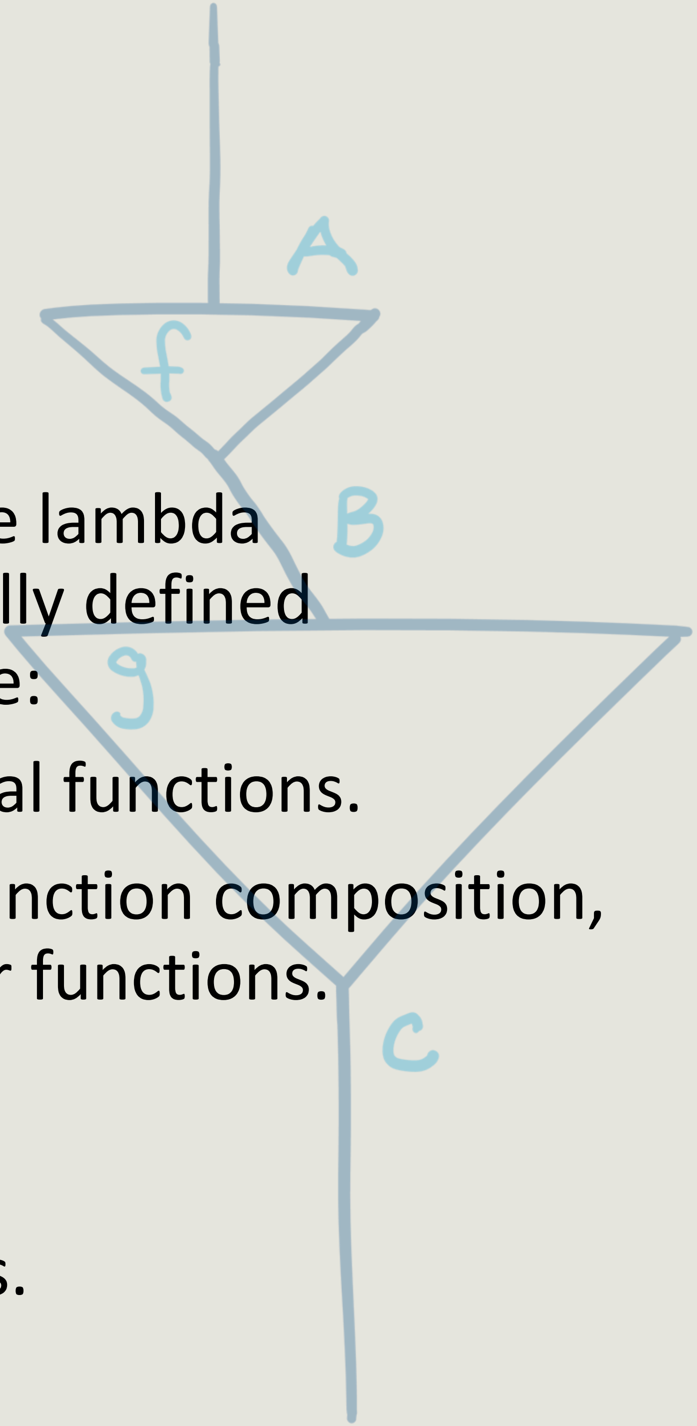
- Objects are concurrent channel types.
- Maps are concurrent processes.
- Identity is just a channel.
- Composition is plugging two processes together over a channel type that they both have with opposite polarities.
- The $\otimes$ tensor and $\oplus$ par functors allow processes to have multiple input polarity and output polarity channels.



Sequential part

The sequential part is a programming language like the lambda calculus or Haskell. Its categorical semantics is minimally defined by a **distributive symmetric monoidal category**, where:

- Objects are sequential types, and maps are sequential functions.
- Identity is an identity function, and composition is function composition, so inputs can be substituted with outputs from other functions.
- The $+$ coproduct allows functions to be defined in multiple ways on different types of input.
- The $*$ tensor allows functions to have multiple inputs.



Semantics for Non-Determinism in Categorical Message Passing Language by Sup-Lattice Enrichment

Alexanna (Xanna) Little - FMCS 2023
Undergraduate Research at the University of Calgary with
Dr. Robin Cockett

3. Message Passing

CaMPL uses a form of concurrency where **processes can communicate by passing messages** along channels.

Concurrent processes can pass messages in both directions along a channel:

- Output from a sequential function can be sent.
- Input for a sequential function can be received.

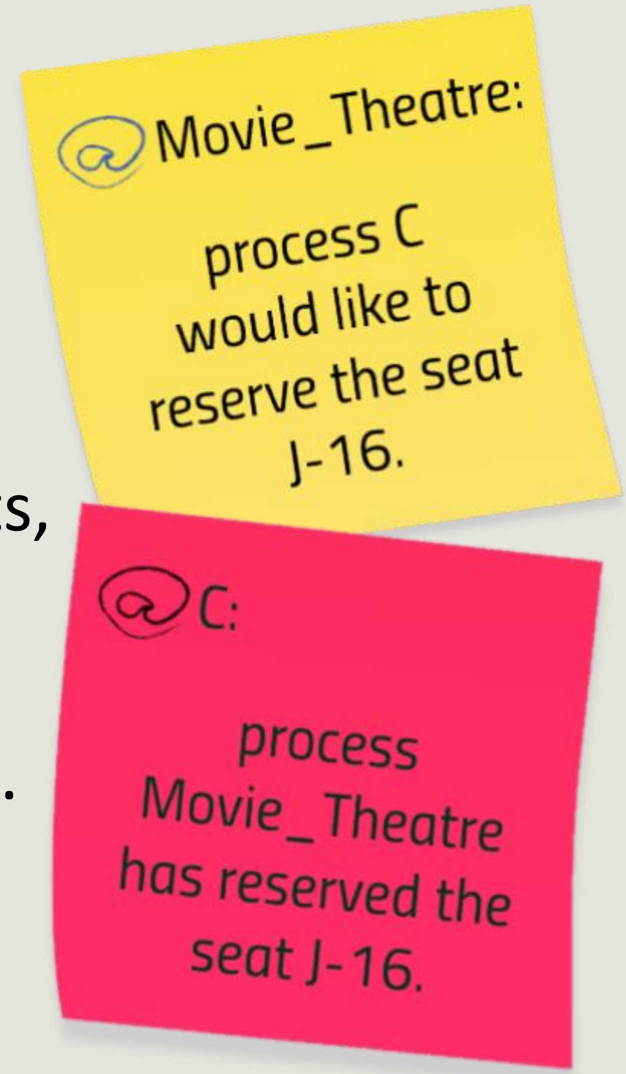
Message Passing

The categorical semantics is defined by an **additive linear actegory** where:

- Distributive monoidal category is the sequential part.
- Linearly distributive category is the channel types and concurrent processes.
- The $\circ$ and $\bullet$ action functors are used to express messages passed on channels.
- Passing a message is given by an adjunction between left parameterized functors $A \circ -$ and $A \bullet -$.

Movie Theatre

- Imagine a movie theatre wants a program for customers to book movie seats online.
- The Movie_Theatre process handles customer requests, and it can **book seats for multiple customers at once**.
- A customer uses their process C to input a seat, and a message with this request is passed to Movie_Theatre.
- Then Movie_Theatre books the seat and passes a confirmation message back.



Semantics for Non-Determinism in Categorical Message Passing Language by Sup-Lattice Enrichment

Alexanna (Xanna) Little - FMCS 2023
Undergraduate Research at the University of Calgary with
Dr. Robin Cockett

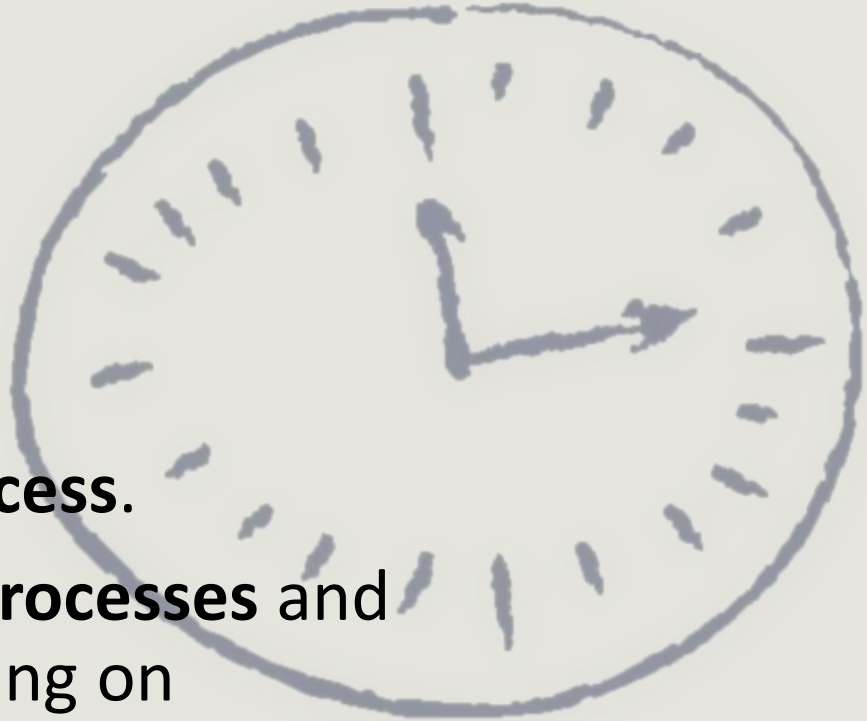
4. Non-Determinism

I have been working on the semantics for the programming feature we use to write **non-deterministic CaMPL programs**.

A program is non-deterministic if its **output cannot be completely determined** by its input values.

In CaMPL, non-determinism was introduced by adding **races**.

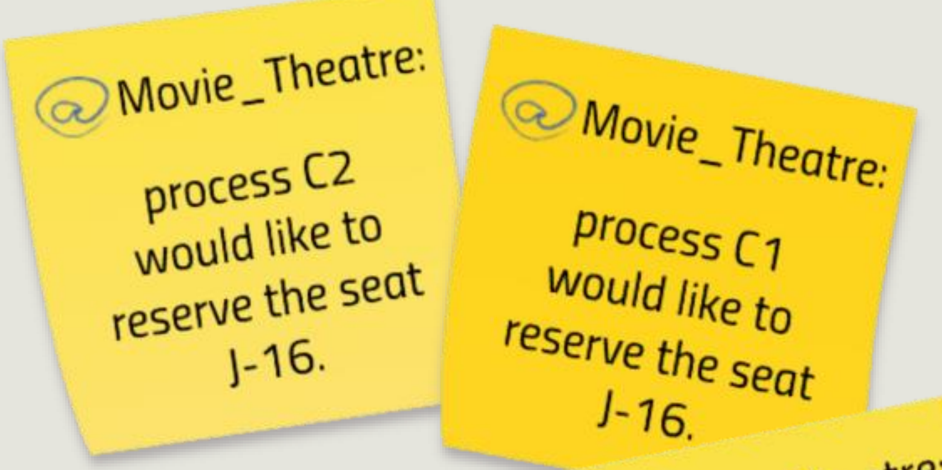
Races

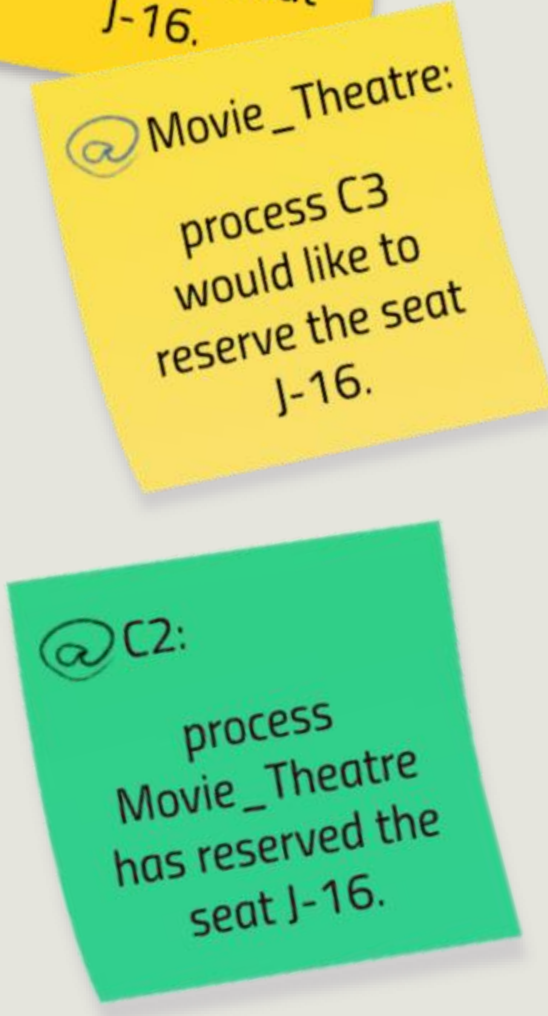


- A CaMPL program is non-deterministic if it has a **non-deterministic concurrent process**.
- A non-deterministic process **races other processes** and **changes what it does** in real time depending on **which process wins the race** by passing the first message.
- This feature is important because **life is non-deterministic**, so some solutions need dynamic processes that can change what they do while the program is running.

Movie Theatre

- Imagine we are the movie theatre again, and business is booming because our booking system was so efficient!
- We ran into a problem though when **3 customers have asked for the same seat**.
- We cannot confirm the booking for all of them or a crisis would certainly arise, so what should we do?
- One solution could be to handle requests from customers in a pre-determined order, but this is rather impractical.
- An efficient customer-friendly solution is to give the seat to the first customer who asks for it!





Semantics for Non-Determinism in Categorical Message Passing Language by Sup-Lattice Enrichment

Alexanna (Xanna) Little - FMCS 2023
Undergraduate Research at the University of Calgary with
Dr. Robin Cockett

5. Sup-Lattice

Concurrent processes are the maps between concurrent channel types, and in the **deterministic semantics**, the **maps were sets**.

Now, we want the maps to be a structure that can represent non-deterministic processes too.

For the **non-deterministic semantics**, the maps are **sup-lattices** constructed by taking the **power set of a map**.

What is a Sup-Lattice?

- A **sup-lattice** $\mathcal{L} = (L, \bot, \vee)$ has carrier L, bottom element $\bot \in L$, and join operation $\vee$.
- A partial ordering is given by $\vee$ which can be expressed as $x \leq y \iff x \vee y = y$ and $\bot \leq x$.
- Additionally, sup is a unique map from elements to their **least upper bound** (supremum) element if it exists, so $x \leq sup(x, y),\ y \leq sup(x, y)$ and $x \leq z,\ y \leq z \implies sup(x, y) \leq z$.
- We can also express the notion of arbitrary sups as joins:

$$x_i \leq \bigvee_{i \in I} x_i x_i \leq z,\ \forall i \in I \implies \bigvee_{i \in I} x_i \leq z$$

Tensor of Sup-Lattices

- **SupLat** is the category of sup-lattices where the objects are sup-lattices and the morphisms are sup-preserving maps.
- A sup-preserving map ensures the sup for a set of elements is mapped to the sup for the mappings of those elements: $f(\bigvee_{i \in I} x_i) = \bigvee_{i \in I} f(x_i)$
- **SupLat** is a monoidal category, and the unit sup-lattice has carrier $\{\top, \bot\}$. The tensor $\mathcal{L}_1 \otimes \mathcal{L}_2$ has carrier $\{ \bigvee(x_1 \otimes x_2) \mid x_1 \in L_1, x_2 \in L_2 \}$, where $\bot \otimes x = \bot = x \otimes \bot$. Maps out of the tensor are sup-preserving and bi-sup-preserving so we have:

$$(\bigvee_{i \in I} x_i) \otimes y = \bigvee_{i \in I} (x_i \otimes y) \quad x \otimes (\bigvee_{j \in J} y_j) = \bigvee_{j \in J} (x \otimes y_j)$$

Power set construction

- Let $\mathscr{P}$: **Set** $\to$ **Set** denote the power set functor. We can write $\mathscr{P} = \mathcal{PU}$ where $\mathcal{P}$: **Set** $\to$ **SupLat** lifts a set to a sup-lattice where the carrier is its power set and $\perp := \emptyset, \vee := \cup$ and $\mathcal{U}$: **SupLat** $\to$ **Set** is the underlying functor that maps a sup-lattice back to the set that is its carrier.
- These functors form an adjunction (η, ϵ) : $\mathcal{P} \dashv \mathcal{U}$: **Set** $\to$ **SupLat** with $\eta : 1_{\textbf{Set}} \Rightarrow \mathcal{PU}$ which sends elements to singletons.
  An abstract component map is $\eta_X : X \to \mathcal{PU}(X)\ ;\ x \mapsto \{x\}$.
  The power set is formed by the sup-lattice with bottom element $\emptyset$ and the singletons are joined with $\cup$.

Semantics for Non-Determinism in Categorical Message Passing Language by Sup-Lattice Enrichment

Alexanna (Xanna) Little - FMCS 2023
Undergraduate Research at the University of Calgary with Dr. Robin Cockett

6. Enrichment

We want to give our maps additional structure so we can **represent non-deterministic processes**.

Enriched category theory **gives maps additional structure**, and changing the structure of the maps in a category is called a **change of base**.

We can define the non-deterministic semantics for CaMPL by changing the base of the semantics **from set-enriched into sup-lattice enriched**.

What is an Enriched Category?

- An enriched category has maps that are **hom-objects** taken from a monoidal category.
- Hom-objects must come from a monoidal category because the unit object is required to define identity maps and the tensor is required to define composition of maps.
- Let $\mathbb{X}$ be a monoidal category, with a tensor $\otimes$ and unit I, an $\mathbb{X}$-enriched category $\mathbb{C}$ can be defined as follows:

$$\text{Objects} : \text{Obj}(\mathbb{C})$$

$$\text{Hom-objects} : \mathbb{C}(A, B) \in \text{Obj}(\mathbb{X}),\ \ A, B \in \text{Obj}(\mathbb{C})$$

$$\text{Identity} : 1_A : I \to \mathbb{C}(A, A),\ \ \forall A \in \text{Obj}(\mathbb{C})$$

$$\text{Composition} : m_{ABC} : \mathbb{C}(A, B) \otimes \mathbb{C}(B, C) \to \mathbb{C}(A, C),\ \ A, B, C \in \text{Obj}(\mathbb{C})$$

Change of base

- We can change the monoidal category we are enriching in with a **monoidal functor** from the original monoidal category to the new monoidal category.
- We want to show that $\mathcal{P}$: **Set** $\rightarrow$ **SupLat** is a monoidal functor; that is, the monoidal structure in **Set** $(\times, \emptyset, a_\times, u^L_\times, u^R_\times)$ is lifted to the monoidal structure in **SupLat** $(\otimes, I, a_\otimes, u^L_\otimes, u^R_\otimes)$.
- To show that $\mathcal{P}$ is a monoidal functor, we must define comparison maps so that the necessary coherence diagrams commute: $m_1 : I \rightarrow \mathcal{P}(\emptyset)$ and $m_\otimes : \mathcal{P}(X) \otimes \mathcal{P}(Y) \rightarrow \mathcal{P}(X \times Y)$.

The necessary coherence diagrams commute:

- Left and right unitors:

$$\begin{tikzcd} I \otimes \mathcal{P}(X) \arrow[r, "m_1 \otimes 1_{\mathcal{P}(X)}"] \arrow[d, "u^L_\otimes"'] & \mathcal{P}(\emptyset) \otimes \mathcal{P}(X) \arrow[d, "m_\otimes"] \\ \mathcal{P}(X) & \mathcal{P}(\emptyset \times X) \arrow[l, "\mathcal{P}(u^L_\times)"] \end{tikzcd} \qquad \begin{tikzcd} \mathcal{P}(X) \otimes I \arrow[r, "1_{\mathcal{P}(X)} \otimes m_1"] \arrow[d, "u^R_\otimes"'] & \mathcal{P}(X) \otimes \mathcal{P}(\emptyset) \arrow[d, "m_\otimes"] \\ \mathcal{P}(X) & \mathcal{P}(X \times \emptyset) \arrow[l, "\mathcal{P}(u^R_\times)"] \end{tikzcd}$$

- Associativity

$$\begin{tikzcd} (\mathcal{P}(X) \otimes \mathcal{P}(Y)) \otimes \mathcal{P}(Z) \arrow[r, "a_\otimes"] \arrow[d, "m_\otimes \otimes 1_{\mathcal{P}(Z)}"'] & \mathcal{P}(X) \otimes (\mathcal{P}(Y) \otimes \mathcal{P}(Z)) \arrow[d, "1_{\mathcal{P}(X)} \otimes m_\otimes"] \\ \mathcal{P}(X \times Y) \otimes \mathcal{P}(Z) \arrow[d, "m_\otimes"'] & \mathcal{P}(X) \otimes \mathcal{P}(Y \times Z) \arrow[d, "m_\otimes"] \\ \mathcal{P}((X \times Y) \times Z) \arrow[r, "\mathcal{P}(a_\times)"'] & \mathcal{P}(X \times (Y \times Z)) \end{tikzcd}$$

Semantics for Non-Determinism in Categorical Message Passing Language by Sup-Lattice Enrichment

Alexanna (Xanna) Little - FMCS 2023
Undergraduate Research at the University of Calgary with
Dr. Robin Cockett

7. Semantics

I have been working on showing that, if we add non-determinism to CaMPL as I have described, "everything" "still works".

Today we will talk about how sup-lattice enriching a linearly distributive category gives a sup-lattice enriched linearly distributive category.

And how initial and final algebras give semantics for non-deterministic protocols and coprotocols.

Lifting a LDC

- If $\mathcal{P}$ is a monoidal functor, then it **preserves monoidal structure**, and we have shown that it is a monoidal functor.
- We also need to show $\mathcal{P}$ **preserves natural transformations** and will therefore lift the linear distributors δ_L and δ_R.
- That is, if $\alpha : F \Rightarrow G : \mathbb{X} \to \mathbb{Y}$ is an arbitrary natural transformation, then $\eta(\alpha) : \mathcal{P}(F) \Rightarrow \mathcal{P}(G) : \mathcal{P}(\mathbb{X}) \to \mathcal{P}(\mathbb{Y})$ is also a natural transformation.
- We define $\eta(\alpha)$ with a naturality square:

$$\begin{tikzcd}
\mathcal{P}(F)(X) \arrow[r, "\eta(\alpha)_X"] \arrow[d, "\mathcal{P}(F)(f)"'] & \mathcal{P}(G)(X) \arrow[d, "\mathcal{P}(G)(f)"] \\
\mathcal{P}(F)(Y) \arrow[r, "\eta(\alpha)_Y"'] & \mathcal{P}(G)(Y)
\end{tikzcd}$$

  And check that it commutes:

$$\eta(\alpha)_X\, \mathcal{P}(G)(f) = \{\alpha_X\}\,\{G(x) \mid x \in f\} = \{\alpha_X\, G(x) \mid x \in f\} = \{F(x)\, \alpha_Y \mid x \in f\} = \{F(x) \mid x \in f\}\,\{\alpha_Y\} = \mathcal{P}(F)(f)\, \eta(\alpha)_Y$$

Lifting products and coproducts

- Deterministic products have the universal property that there exists a unique map $\langle f, g\rangle$ such that $\langle f, g\rangle\pi_0 = f$ and $\langle f, g\rangle\pi_1 = g$.
- In a sup-lattice enriched category, f and g are non-deterministic, so the **unique maps for non-deterministic products** are $\langle f, g\rangle = \{\langle f_i, g_j\rangle \mid f_i \in f, g_j \in g\}$.
- Deterministic coproducts have the universal property that there exists a unique map $\langle f \mid g\rangle$ such that $\sigma_0\langle f \mid g\rangle = f$ and $\sigma_1\langle f \mid g\rangle = g$.
- The **unique maps for non-deterministic coproducts** are $\langle f \mid g\rangle = \{\langle f_i \mid g_j\rangle \mid f_i \in f, g_j \in g\}$.

Lax universal property

- Since f and g are non-deterministic, we can make the necessary diagrams commute **with subset inclusion equalities** where $\langle f', g' \rangle \subseteq \langle f, g \rangle$ and $\langle f' \mid g' \rangle \subseteq \langle f \mid g \rangle$ **rather than strict equalities**.

$$
\begin{tikzcd}
& & A \\
C \arrow[urr, bend left, "f"] \arrow[r, "\langle f', g' \rangle"] \arrow[drr, bend right, "g"'] & A \times B \arrow[ur, "\pi_0"'] \arrow[dr, "\pi_1"] \arrow[u, phantom, "\supseteq"] \arrow[d, phantom, "\supseteq"] & \\
& & B
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[dr, "\sigma_0"'] \arrow[drr, bend left, "f"] & & \\
& A + B \arrow[r, "\langle f' | g' \rangle"] \arrow[u, phantom, "\subseteq"] \arrow[d, phantom, "\subseteq"] & C \\
B \arrow[ur, "\sigma_1"] \arrow[urr, bend right, "g"'] & &
\end{tikzcd}
$$

- However, $\langle f', g' \rangle$ and $\langle f' \mid g' \rangle$ are no longer unique as many subsets of $\langle f, g \rangle$ or $\langle f \mid g \rangle$ could make the diagram commute.
- So rather than a universal property, we have a **lax universal property** for non-deterministic products and coproducts.

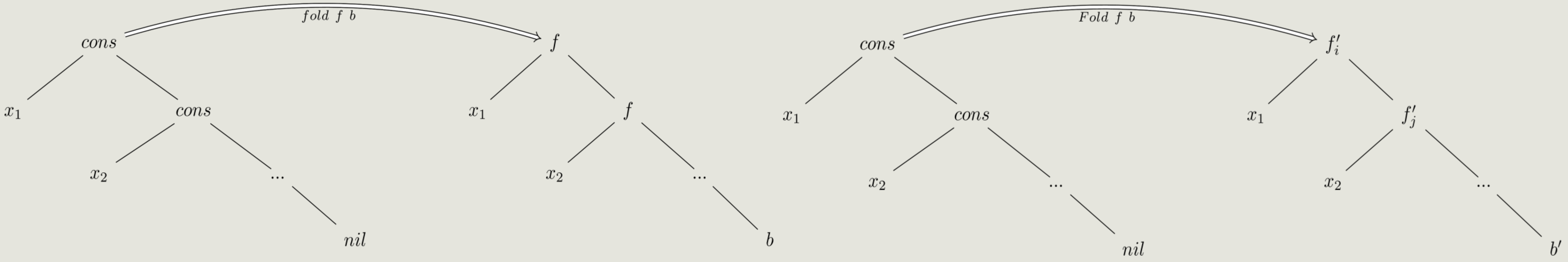
Lifting initial and final algebras

- In the deterministic semantics, initial and final algebras define the inductive and coinductive data types which give the **semantics for protocols and coprotocols**.
- Initial and final algebras also exist with subset inclusion equalities rather than strict equalities.
- In the case of initial algebras, we have $f' \subseteq Fold(f)$ where $f = \{f_i \mid f_i \in f\}$ and $Fold(f) = \bigcup_{cons\ f' \subseteq F(f')\ f} \{fold(f_i)\}$ which makes the following diagram commute:

$$\begin{tikzcd} F(F^{@}) \arrow[r, "cons"] \arrow[d, dashed, "F(f')"'] \arrow[dr, phantom, "="] & F^{@} \arrow[d, dashed, "f' = fold(f)"] \\ F(A) \arrow[r, "f"'] & A \end{tikzcd}
\qquad
\begin{tikzcd} F(F^{@}) \arrow[r, "cons"] \arrow[d, dashed, "F(f')"'] \arrow[dr, phantom, "\supseteq"] & F^{@} \arrow[d, dashed, "f' \subseteq Fold(f)"] \\ F(A) \arrow[r, "f = \{f_i \mid f_i \in f\}"'] & A \end{tikzcd}$$

Non-deterministic folds and unfolds

- Inductive and coinductive data types have folds and unfolds.
- Non-deterministic folds and unfolds have constructors and destructors where **each iteration's function can be selected non-deterministically**.
- For example, $fold\ f\ b$ becomes $Fold\ f\ b$ with $f_i', f_j' \in f, b' \in b$:



Semantics for Non-Determinism in Categorical Message Passing Language by Sup-Lattice Enrichment

Alexanna (Xanna) Little - FMCS 2023
Undergraduate Research at the University of Calgary with
Dr. Robin Cockett

References

1. J. R. B. Cockett and Craig Pastro. The Logic of Message Passing. Science of Computer Programming, 74(8):498–533, 2009. (Determinsitic semantics for CaMPL.)
2. G. S. H. Cruttwell. Normed Spaces and the Change of Base for Enriched Categories. PhD thesis, Dalhousie University, Halifax, Nova Scotia, December 2008.
(Enriched categories, change of base, monoidal functors.)
3. Andre Joyal and Myles Tierney. An extension of the Galois theory of Grothendieck, volume 309 of Memoirs of the American Mathematical Society. American Mathematical Society, Providence, R.I, 1984. (Sup-lattices.)
4. Reginald Lybbert. Progress for the Message Passing Logic. Undergraduate thesis, University of Calgary, April 2018. Provided by the author. (Progress for deterministic CaMPL programs.)
5. Jared Pon. Implementation Status of CMPL. Undergraduate thesis Interim Report, University of Calgary, December 2021. Provided by the author. (Races in CaMPL.)
6. Masuka Yeasin. Linear Functors and their Fixed Points. Master’s thesis, University of Calgary, Calgary, Alberta, December 2012. Provided by the author.
(Deterministic protocols and coprotocols.)