On the Lambek Calculus with an Exchange Modality

Jiaming Jiang\textsuperscript{1}, Harley Eades III\textsuperscript{2}, Valeria de Paiva\textsuperscript{3}

\textsuperscript{1}North Carolina State University; \textsuperscript{2}Augusta University; \textsuperscript{3}Nuance Communications
Linearity and Non-Linearity

- Girard bridged linearity with non-linearity via $!A$.

- This modality isolates the structural rules:

\[
\begin{align*}
\Gamma_1, \Gamma_2 \vdash B & \quad \Gamma_1, !A, !A, \Gamma_2 \vdash B \\
\frac{\Gamma_1, \Gamma_2 \vdash B}{\Gamma_1, !A, \Gamma_2} & \quad \frac{\Gamma_1, !A, !A, \Gamma_2 \vdash B}{\Gamma_1, !A, \Gamma_2}
\end{align*}
\]

- Linear Logic = linearity + of-course
Linearity and Non-Linearity

Linear Logic takes for granted the structural rule:

\[
\frac{\Gamma_1, A, B, \Gamma_2 \vdash C}{\Gamma_1, B, A, \Gamma_2 \vdash C} \quad \text{Ex}
\]
Lambek Calculus

- Lambek invented what we call the Lambek Calculus to give a mathematical semantics to sentence structure.

- Lambek Calculus = linearity - exchange
  - Non-commutative tensor: $A \triangleright B$
  - Non-commutative implications: $[[A \prec B]]$ and $[[A \rightarrow B]]$

- No modalities

- Applications
Lambek Calculus

Question posed by computational linguists:
Can we add a modality to the Lambek Calculus that does for exchange what of-course does for weakening and contraction?
Motivation

In process calculi, to model sequential composition of processes:

\[ A \otimes B \]
- Commutative tensor product
- Processes \( A \) and \( B \) run in parallel

\[ A \triangleright B \]
- Non-commutative tensor product
- Process \( A \) runs first, then process \( B \)
Basic Approach

Abstract Benton’s Linear/Non-Linear (LNL) model:
- Remove the exchange structural rule: implicit in $\Phi, \Psi; \Gamma, \Delta$
- Two logics:
  - Intuitionistic linear logic
  - Lambek Calculus
Linear/Non-Linear Model

A symmetric monoidal adjunction $F \dashv G$:

- Counit: $\varepsilon : FG \to id_C$
- Unit: $\eta : id_L \to GF$
Linear/Non-Linear Model

A symmetric monoidal adjunction $F \dashv G$:

- Countit: $\varepsilon : FG \to id_C$
- Unit: $\eta : id_L \to GF$

- Monad ($GF, \eta, \mu = G\varepsilon_F$) on the CCC: strong and commutative
- Comonad ($FG, \varepsilon, \delta = F\eta_G$) on the SMCC: symmetric monoidal
- Of-course modality: $! = FG$
Commutative/Non-Commutative (CNC) Model
Commutative/Non-Commutative (CNC) Model

A monoidal adjunction $F \dashv G$:

- **Counit:** $\varepsilon : FG \to id_C$
- **Unit:** $\eta : id_L \to GF$

$\mathcal{C}$: Symmetric Monoidal Closed Category

$\mathcal{L}$: Lambek Category

$\Rightarrow$: $\mathcal{L}FG$ is symmetric monoidal
Commutative/Non-Commutative (CNC) Model

A monoidal adjunction $F \dashv G$:

- **Counit:** $\varepsilon : FG \to id_C$
- **Unit:** $\eta : id_L \to GF$

Monad $(GF, \eta, \mu = G\varepsilon_F)$ on the SMCC: strong but non-commutative

Comonad $(FG, \varepsilon, \delta = F\eta_G)$ on the Lambek category: monoidal

Exchange: a natural transformation $ex^{FG} : A \triangleright B \to B \triangleright A$ in the co-Eilenberg-Moore category $\mathcal{L}^{FG}$ of the comonad implies $\mathcal{L}^{FG}$ is symmetric monoidal
CNC Logic

- Left: intuitionistic linear logic
- Right: mixed commutative/non-commutative Lambek calculus
CNC Logic: Notation

**Intuitionistic Linear Logic**

\( \mathcal{C} \)-Types: \( W, X, Y, Z \)

\( \mathcal{C} \)-Terms: \( t \)

\( \mathcal{C} \)-Contexts: \( \Phi, \Psi \)

\( \mathcal{C} \)-Typing Judgment: \( \Phi, \Psi \vdash_{\mathcal{C}} t : X \)

**Lambek Calculus**

\( \mathcal{L} \)-Types: \( A, B, C, D \)

\( \mathcal{L} \)-Terms: \( s \)

\( \mathcal{L} \)-Contexts: \( \Gamma, \Delta \)

\( \mathcal{L} \)-Typing Judgment: \( \Gamma; \Delta \vdash_{\mathcal{L}} s : A \)
CNC Logic: Example Typing Rules

Exchange rules:

\[
\frac{\Phi, x : X, y : Y, \Psi \vdash_C t : Z}{\Phi, z : Y, w : X, \Psi \vdash_C \text{ex } w, z \text{ with } x, y \text{ in } t : Z} \quad \text{\(C\)-ex}
\]

\[
\frac{\Gamma; x : X; y : Y; \Delta \vdash_L s : A}{\Gamma; z : Y; w : X; \Delta \vdash_L \text{ex } w, z \text{ with } x, y \text{ in } s : A} \quad \text{\(L\)-ex}
\]
CNC Logic: Example Typing Rules

Functor rules for G:

\[ \Phi \vdash \_{\mathcal{L}} s : A \quad \Phi \vdash \_{\mathcal{C}} G s : GA \quad \text{C-G}\_I \]

\[ \Phi \vdash \_{\mathcal{C}} t : GA \quad \Phi \vdash \_{\mathcal{L}} \text{derelict } t : A \quad \text{C-G}\_E \]
CNC Logic: Example Typing Rules

Functor rules for G:

\[ \begin{align*}
\Phi \vdash \ell s & : A \\
\Phi \vdash \ell Gs & : GA \\
\Phi \vdash \ell \text{derelict} t & : A
\end{align*} \]

Functor rules for F:

\[ \begin{align*}
\Phi \vdash \ell t & : X \\
\Phi \vdash \ell Ft & : FX \\
\Gamma \vdash \ell s_1 & : FX \\
\Delta_1 x & : X; \Delta_2 \vdash \ell s_2 & : A
\end{align*} \]

\[ \Delta_1; \Gamma; \Delta_2 \vdash \ell \text{let} s_1 : FX \text{ be } Fx \text{ in } s_2 : A \]
CNC Logic: Other Results

- $\beta$-reductions: one step $\beta$-reduction rules
- Commuting conversions
- Cut elimination
- Equivalence between sequent calculus and natural deduction
- Strong normalization via a translation to LNL logic
- A concrete model in dialectica categories
Conclusion

- Commutative/Non-commutative Logic:
  - Left: intuitionistic linear logic
  - Right: Lambek calculus
- Categorical model: a monoidal adjunction
  - Left: symmetric monoidal closed category
  - Right: Lambek category

Exchange Natural Transformation

\[ \text{ex}^{FG} : A \triangleright B \rightarrow B \triangleright A \text{ in the co-Eilenberg-Moore category } \mathcal{L}^{FG} \text{ of the comonad on the Lambek category} \]