Dialectica Categories for the Lambek Calculus

Valeria de Paiva[1] and Harley Eades III[2]

[1] Nuance Communications, Sunnyvale, CA, valeria.depaiva@gmail.com
[2] Computer Science, Augusta University, Augusta, GA, harley.eades@gmail.com

Abstract. We revisit the old work of de Paiva on the models of the Lambek Calculus in dialectica models making sure that the syntactic details that were sketchy on the first version got completed and verified. We extend the Lambek Calculus with a κ modality, inspired by Yetter's work, which makes the calculus commutative. Then we add the of-course modality !, as Girard did, to re-introduce weakening and contraction for all formulas and get back the full power of intuitionistic and classical logic. We also present the categorical semantics, proved sound and complete. Finally we show the traditional properties of type systems, like subject reduction, the Church-Rosser theorem and normalization for the calculi of extended modalities, which we did not have before.

Introduction

Lambek introduced his homonymous calculus (originally called the 'Syntactic Calculus') for proposed applications in Linguistics. However the calculus got much of its cult following and reputation by being a convenient, well-behaved prototype of a Gentzen sequent calculus without any structural rules.

This note recalls a Dialectica model of the Lambek Calculus presented by the first author in the Amsterdam Colloquium in 1991 [10]. Here, like then, we approach the Lambek Calculus from the perspective of Linear Logic, so we are interested in the basic sequent calculus with no structural rules, except associativity of tensors. In that earlier work we took for granted the syntax of the calculus and only discussed the exciting possibilities of categorical models of linear-logic-like systems. Many years later we find that the work on models is still interesting and novel, and that it might inform some of the most recent work on the relationship between categorial grammars and notions of distributional semantics [8].

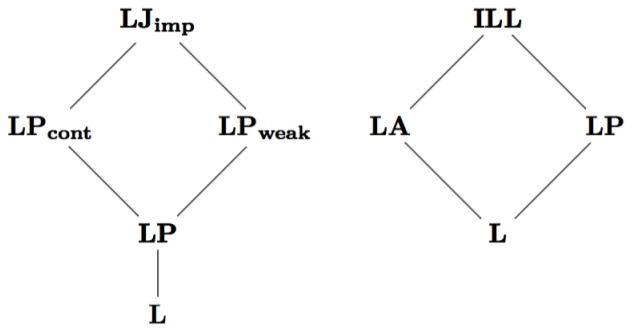
Moreover, using the Agda proof assistant [6] we verify the correctness of our categorical model (Section 4.1), and we add the type theoretical (Section 5) notions that were left undiscussed in the Amsterdam Colloquium presentation. All of the syntax in this paper was checked using `Ott` [23]. The goal is to show that our work can shed new light on some of the issues that remained open. Mostly we wanted to check the correctness of the semantic proposals put forward since Szabo's seminal book [24] and, for future work, on the applicability and fit of the original systems to their intended uses.

Overview. The Syntactic Calculus was first introduced by Joachim Lambek in 1958 [19]. Since then the rechristened Lambek Calculus has had as its main motivation providing an explanation of the mathematics of sentence structure, starting from the author's algebraic intuitions. The Lambek Calculus is the core of logical Categorial Grammar. The first use of the term "categorial grammar" seems to be in the title of Bar-Hillel, Gaifman and Shamir (1960), but categorial grammar began with Ajdukiewicz (1935) quite a few years earlier. After a period of ostracism, around 1980 the Lambek Calculus was taken up by logicians interested in Computational Linguistics, especially the ones interested in Categorial Grammars.

The work on Categorial Grammar was given a serious impulse by the advent of Girard's Linear Logic at the end of the 1980s. Girard [12] showed that there is a full embedding, preserving proofs, of Intuitionistic Logic into Linear Logic with a modality "!". This meant that Linear Logic, while paying attention to resources, could always code up faithfully Classical Logic and hence one could, as Girard put it, 'have one's cake and eat it', paying attention to resources, if desired, but always forgetting this accounting, if preferred. This meant also that several new systems of resource logics were conceived and developed and these refined resource logics were applied to several areas of Computer Science.

In Computational Linguistics, the Lambek Calculus has seen a significant number of works written about it, apart from a number of monographs that deal with logical and linguistic aspects of the generalized type-logical approach. For a shorter introduction, see Moortgat's entry on the Stanford Encyclopedia of Philosophy on Type Logical Grammar [20]. Type Logical Grammar situates the type-logical approach within the framework of Montague's Universal Grammar and presents detailed linguistic analyses for a substantive fragment of syntactic and semantic phenomena in the grammar of English. Type Logical Semantics offers a general introduction to natural language semantics studied from a type-logical perspective.

This meant that a series of systems, implemented or not, were devised that used the Lambek Calculus or its variants as their basis. These systems can be as expressive as Intuitionistic Logic and the claim is that they are more precise i.e. they make finer distinctions. Some of the landscape of calculi can be depicted as follows:



From the beginning it was clear that the Lambek Calculus is the multiplicative fragment of non-commutative Intuitionistic Linear Logic. In the diagrams **L** stands for the Lambek Calculus, as expounded in [19] but with the unit I added for the tensor connective (there was a certain amount of dispute on that, as the original system did not introduce the constant corresponding to the nullary case of the tensor product, here written as I). The system **LP** is the Lambek Calculus with permutation, sometimes called the van Benthem calculus. We are calling **LA** the Lambek Calculus with additives, the more usual algebraic connectives corresponding to meet and join. Hence adding both permutation and additives to the Lambek Calculus we get to intuitonistic linear logic. On the other diagram to the Lambek Calculus with permutation we add either weakening ($\mathbf{LP_{weak}}$) or contraction $\mathbf{LP_{cont}}$) or both to get the implicational fragment of Intuitionsitic Propositional Logic.

The Lambek Calculus also has the potential for many applications in other areas of computer science, such as, modeling processes. Linear Logic has been at the forefront of the study of process calculi for many years [14,22,1]. We can think of the commutative tensor product of linear logic as a parallel operator. For example, given a process A and a process B, then we can form the process $A \otimes B$ which runs both processes in parallel. If we remove commutativity from the tensor product we obtain a sequential composition instead of parallel composition. That is, the process $A \triangleright B$ first runs process A and then process B in that order. Paraphrasing Vaughan Pratt, "The sequential composition operation has no evident counterpart in type theory" see page 11 of [22]. We believe that the Lambek Calculus will lead to filling this hole, and the results of this paper as a means of obtaining a theory with both a parallel operator and a sequential composition operator. This work thus has a potential to impact research in programming languages and computer security where both linear logic and sequential composition play important roles.

There are several interesting questions, considered for Linear Logic, that could also be asked of the Lambek Calculus or its extensions. One of them, posed by Morrill et al is whether we can extend the Lambek Calculus with a modality that does for the structural rule of *(exchange)* what the modality *of course* '!' does for the rules of *(weakening)* and *(contraction)*. A preliminary proposal, which answers this question affirmatively, is set forward in this paper. The answer was provided in semantical terms in the first version of this work. Here we provide also the more syntactic description of these modalities. Building up from work of Ciabattoni, Galatos and Terui in [7] and others that describe how to transform systems of axioms into cut-free sequent rules, we aim to refine the algebraization of proof theory.

1 Related Work

Lamarche and Retoré [18] give an overview of proof nets for the Lambek Calculus where they discuss the addition of various exchange rules to the calculus. For

example, the following permutation and cycle rules:

$$\frac{A_1, A_2, \Gamma \vdash B}{A_2, A_1, \Gamma \vdash B}\ \text{PERM} \qquad \frac{A_1, A_2, \Gamma \vdash B}{A_1, \Gamma, A_2, \Gamma \vdash B}\ \text{CYCL}$$

Taken in isolation each rule does not imply general exchange, but taken together they do. Thus, it is possible to take a restricted notion of exchange to the Lambek Calculus without taking general exchange. However, applications where one needs a completely non-commutative tensor product and a commutative tensor product cannot be modeled in the Lambek Calculus with these rules.

Polakow and Pfenning [21] combine intuitionistic, commutative linear, and non-commutative linear logic into a system called Ordered Linear Logic (OLL). Polakow and Pfenning then extend OLL into two new systems: a term assignment of OLL called OLLi, and a logical framework in the tradition of LF called OLF.

OLL's sequent is of the form $\Gamma, \Delta, \Omega \vdash A$ where Γ is a multiset of intuitionistic assumptions, Δ is a multiset of commutative linear assumptions, and Ω is a list of non-commutative linear assumptions. Furthermore, OLL contains logical connectives from each logic. For example, there are two different tensor products and three different implications. Thus, the systems developed here are a simplification of OLL showing how to get all of these systems using modalities.

Greco and Palmigiano [13] give a type of logic called a proper display logic for the Lambek Calculus with exponentials. However, they decompose the linear exponential into an adjunction in the spirit Benton's LNL models. In this paper we concentrate on sequent calculi rather than display logic.

2 The Lambek Calculus

The Lambek Calculus, formerly the Syntactic Calculus L, due to J. Lambek [19], was created to capture the logical structure of sentences. Lambek introduced what we think of as a substructural logic with an operator denoting concatenation, $A \otimes B$, and two implications relating the order of phrases, $A \leftharpoonup B$ and $A \rightharpoonup B$. The first implication corresponds to a phrase of type A when followed by a phrase of type B, and the second is a phrase of type B when preceded by a phrase of type A.

The Lambek Calculus, defined in Figure 1, can be presented as a non-commutative intuitionistic multiplicative linear logic. Contexts are sequences of formulas, and we denote mapping the modalities over an arbitrary context by $!\Gamma$ and $\kappa\Gamma$.

Because the operator $A \otimes B$ denotes the type of concatenations the types $A \otimes B$ and $B \otimes A$ are not equivalent, and hence, L is non-commutative which explains why implication must be broken up into two operators $A \leftharpoonup B$ and $A \rightharpoonup B$. In the following subsections we give two extensions of L: one with the well-known modality of-course of linear logic which adds weakening and contraction, and a second with a new modality adding exchange.

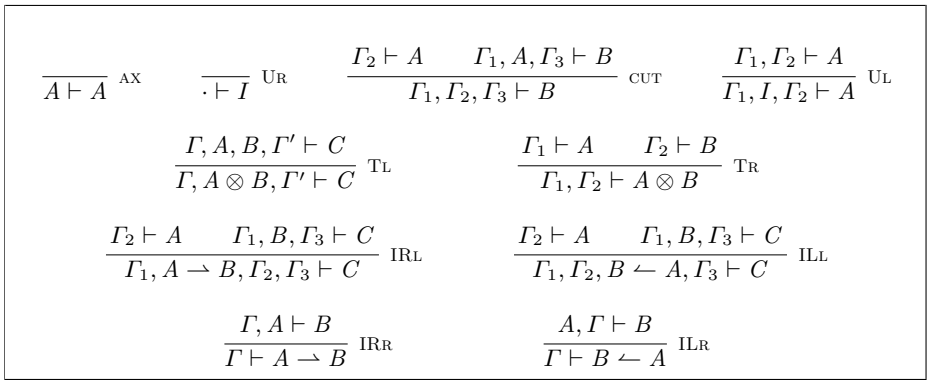


Fig. 1. The Lambek Calculus: L

3 Extensions to the Lambek Calculus

The linear modality, $!A$, read "of-course A" was first proposed by Girard [12] as a means of encoding non-linear logic in both classical and intuitionistic forms in linear logic. For example, non-linear implication $A \rightharpoonup B$ is usually encoded into linear logic by $!A \multimap B$. Since we have based L on non-commutative intuitionistic linear logic it is straightforward to add the of-course modality to L. The rules for the of-course modality are defined by the following rules:

$$\frac{\Gamma_1, !A, \Gamma_2, !A, \Gamma_3 \vdash B}{\Gamma_1, !A, \Gamma_2, \Gamma_3 \vdash B}\ \mathrm{C} \qquad \frac{\Gamma_1, \Gamma_2 \vdash B}{\Gamma_1, !A, \Gamma_2 \vdash B}\ \mathrm{W} \qquad \frac{!\Gamma \vdash B}{!\Gamma \vdash !B}\ \mathrm{B_R} \qquad \frac{\Gamma_1, A, \Gamma_2 \vdash B}{\Gamma, !A, \Gamma_2 \vdash B}\ \mathrm{B_L}$$

The rules C and W add contraction and weakening to L in a controlled way. Then the other two rules allow for linear formulas to be injected into the modality; and essentially correspond to the rules for necessitation found in S4 [5]. Thus, under the of-course modality the logic becomes non-linear. We will see in Section 4.1 that these rules define a comonad. We call the extension of L with the of-course modality $\mathrm{L}_!$.

As we remarked above, one leading question of the Lambek Calculus is: can exchange be added in a similar way to weakening and contraction? That is, can we add a new modality that adds the exchange rule to L in a controlled way? The answer to this question is positive, and the rules for this new modality are as follows:

$$\frac{\kappa\Gamma \vdash B}{\kappa\Gamma \vdash \kappa B}\ \mathrm{E_R} \qquad \frac{\Gamma_1, A, \Gamma_2 \vdash B}{\Gamma_1, \kappa A, \Gamma_2 \vdash B}\ \mathrm{E_L} \qquad \frac{\Gamma_1, \kappa A, B, \Gamma_2 \vdash C}{\Gamma_1, B, \kappa A, \Gamma_2 \vdash C}\ \mathrm{E1} \qquad \frac{\Gamma_1, A, \kappa B, \Gamma_2 \vdash C}{\Gamma_1, \kappa B, A, \Gamma_2 \vdash C}\ \mathrm{E2}$$

The first two rules are similar to of-course, but the last two add exchange to formulas under the κ-modality. We call L with the exchange modality L_κ. Thus, unlike intuitionistic linear logic where any two formulas can be exchanged, L_κ restricts exchange to only formulas under the exchange modality. Just like of-course the exchange modality is modeled categorically as a comonad; see Section 4.1.

4 Categorical Models

We now turn to the categorical models, ones where one considers different proofs of the same theorem. Since the Lambek Calculus itself came from its categorical models, biclosed monoidal categories, there is no shortage of these models. However, Girard's insight of relating logical systems via modalities should also be considered in this context.

Lambek's work on monoidal biclosed categories happened almost three decades before Girard introduced Linear Logic, hence there were no modalities or exponentials in Lambek's setting. The categorical modelling of the modalities (of-course! and why-not?) was the difficult issue with Linear Logic. This is where there are design decisions to be made.

4.1 Dialectica Lambek Spaces

A sound and complete categorical model of the Lambek Calculi can be given using a modification of de Paiva's dialectica categories [9]. Dialectica categories arose from de Paiva's thesis on a categorical model of Gödel's Dialectica interpretation, hence the name. Dialectica categories were one of the first sound categorical models of intuitionistic linear logic, with linear modalities. We show in this section that they can be adapted to become a sound and complete model for the Lambek Calculus, with both the exchange and of-course modalities. We call this model *dialectica Lambek spaces*.

Due to the complexities of working with dialectica categories we have formally verified[3] this section in the proof assistant Agda [6]. Dialectica categories arise as constructions over a given monoidal category. Suppose $\mathcal{C}$ is such a category. Then in complete generality the objects of the dialectica category over $\mathcal{C}$ are triples (U, X, α) where U and X are objects of $\mathcal{C}$, and $\alpha : A \longrightarrow U \otimes X$ is a subobject of the tensor product in $\mathcal{C}$ of U and X. Thus, we can think of α as a relation over $U \otimes X$. If we specialize the category $\mathcal{C}$ to the category of sets and functions, Set, then we obtain what is called a dialectica space. Dialectica spaces are a useful model of full intuitionistic linear logic [15].

Morphisms between objects (U, X, α) and (V, Y, β) are pairs (f, F) where $f : U \longrightarrow V$ and $F : Y \longrightarrow X$ are morphisms of $\mathcal{C}$ such that the pullback condition $(U \otimes F)^{-1}(\alpha) \leq (f \otimes Y)^{-1}(\beta)$ holds. In dialectica spaces this condition becomes $\forall u \in U. \forall y \in Y. \alpha(u, F(y)) \leq \beta(f(u), y)$. The latter reveals that we can think of the condition on morphisms as a weak form of an adjoint condition. Finally, through some nontrivial reasoning on this structure we can show that this is indeed a category; for the details see the formal development. Dialectica categories are related to the Chu construction [11] and to Lafont and Streicher's category of games GAME_κ [17].

To some extent the underlying category $\mathcal{C}$ controls the kind of structure we can expect in the dialectica category over $\mathcal{C}$. However, de Paiva showed [11]

[3] The complete formalization can be found online at `https://github.com/heades/dialectica-spaces/blob/Lambek/NCDialSets.agda`.

that by changing the relations used in the objects and the order used in the 'adjoint condition' (which also controls the type of structure) we can obtain a non-symmetric tensor in the dialectica category, if the structure of the underlying category and the structure of the underlying relations are compatible. She also showed that one can abstract the notion of relation out as a parameter in the dialectica construction, so long as this has enough structure, i.e. so long as you have an algebra (that she called a lineale) to evaluate the relations at. We denote this construction by $\mathsf{Dial}_L(\mathcal{C})$ where L is the lineale controlling the relations coming from the monoidal category $\mathcal{C}$. For example, $\mathsf{Dial}_2(\mathsf{Set})$ is the category of usual dialectica spaces of sets over the Heyting (or Boolean) algebra 2.

This way we can see dialectica categories as a framework of categorical models of various logics, varying the underlying category $\mathcal{C}$ as well as the underlying lineale or algebra of relations L. Depending on which category we start with and which structure we use for the relations in the construction we will obtain different models for different logics.

The underlying category we will choose here is the category Set, but the structure we will define our relations over will be a biclosed poset, defined in the next definition.

Definition 1. *Suppose $(M, \leq, \circ, e)$ is an ordered non-commutative monoid. If there exists a largest $x \in M$ such that $a \circ x \leq b$ for any $a, b \in M$, then we denote x by $a \rightharpoonup b$ and called it the **left-pseudocomplement** of a w.r.t b. Additionally, if there exists a largest $x \in M$ such that $x \circ a \leq b$ for any $a, b \in M$, then we denote x by $b \leftharpoonup a$ and called it the **right-pseudocomplement** of a w.r.t b.*

*A **biclosed poset**, $(M, \leq, \circ, e, \rightharpoonup, \leftharpoonup)$, is an ordered non-commutative monoid, $(M, \leq, \circ, e)$, such that $a \rightharpoonup b$ and $b \leftharpoonup a$ exist for any $a, b \in M$.*

Now using the previous definition we define dialectica Lambek spaces.

Definition 2. *Suppose $(M, \leq, \circ, e, \rightharpoonup, \leftharpoonup)$ is a biclosed poset. Then we define the category of **dialectica Lambek spaces**, $\mathsf{Dial}_M(\mathsf{Set})$, as follows:*

- *objects, or dialectica Lambek spaces, are triples (U, X, α) where U and X are sets, and $\alpha : U \times X \longrightarrow M$ is a generalized relation over M, and*
- *maps that are pairs $(f, F) : (U, X, \alpha) \longrightarrow (V, Y, \beta)$ where $f : U \longrightarrow V$, and $F : Y \longrightarrow X$ are functions such that the weak adjointness condition $\forall u \in U. \forall y \in Y. \alpha(u, F(y)) \leq \beta(f(u), y)$ holds.*

Notice that the biclosed poset is used here as the target of the relations in objects, but also as providing the order relation in the weak adjoint condition on morphisms. This will allow the structure of the biclosed poset to lift up into $\mathsf{Dial}_M(\mathsf{Set})$.

We will show that $\mathsf{Dial}_M(\mathsf{Set})$ is a model of the Lambek Calculus with modalities. First, we show that it is a model of the Lambek Calculus without modalities. Thus, we must show that $\mathsf{Dial}_M(\mathsf{Set})$ is monoidal biclosed.

Definition 3. *Suppose (U, X, α) and (V, Y, β) are two objects of $\mathsf{Dial}_M(\mathsf{Set})$. Then their tensor product is defined as follows:*

$$(U, X, \alpha) \otimes (V, Y, \beta) = (U \times V, (V \to X) \times (U \to Y), \alpha \otimes \beta)$$

where $- \to -$ *is the function space from* Set*, and* $(\alpha \otimes \beta)((u,v),(f,g)) = \alpha(u, f(v)) \circ \beta(g(u), v)$*.*

The identity of the tensor product just defined is $I = (\mathbb{1}, \mathbb{1}, e)$, where $\mathbb{1}$ is the terminal object in Set, and e is the unit of the biclosed poset. It is straightforward to show that the tensor product is functorial, one can define the left and right unitors, and the associator for tensor; see the formalization for the definitions. In addition, all of the usual monoidal diagrams hold [9]. Take note of the fact that this tensor product is indeed non-commutative, because the non-commutative multiplication of the biclosed poset is used to define the relation of the tensor product.

The tensor product has two right adjoints making $\mathsf{Dial}_M(\mathsf{Set})$ biclosed.

Definition 4. *Suppose* (U, X, α) *and* (V, Y, β) *are two objects of* $\mathsf{Dial}_M(\mathsf{Set})$*. Then two internal-homs can be defined as follows:*

$$(U, X, \alpha) \rightharpoonup (V, Y, \beta) = ((U \to V) \times (Y \to X), U \times Y, \alpha \rightharpoonup \beta)$$
$$(V, Y, \beta) \leftharpoonup (U, X, \alpha) = ((U \to V) \times (Y \to X), U \times Y, \alpha \leftharpoonup \beta)$$

These two definitions are functorial, where the first is contravariant in the first argument and covariant in the second, but the second internal-hom is covariant in the first argument and contravariant in the second. The relations in the previous two definitions prevent these two from collapsing into the same object, because of the use of the left and right pseudocomplement. It is straightforward to show that the following bijections hold:

$$\mathsf{Hom}(A \otimes B, C) \cong \mathsf{Hom}(B, A \rightharpoonup C) \qquad \mathsf{Hom}(A \otimes B, C) \cong \mathsf{Hom}(A, C \leftharpoonup B)$$

Therefore, $\mathsf{Dial}_M(\mathsf{Set})$ is biclosed, and we obtain the following result.

Theorem 1. $\mathsf{Dial}_M(\mathsf{Set})$ *is a sound and complete model for the Lambek Calculus L without modalities.*

We now extend $\mathsf{Dial}_M(\mathsf{Set})$ with two modalities: the usual modality, of-course, denoted $!A$, and the exchange modality denoted κA. However, we must first extended biclosed posets to include an exchange operation.

Definition 5. *A* ***biclosed poset with exchange*** *is a biclosed poset* $(M, \leq, \circ, e, \rightharpoonup, \leftharpoonup)$ *equipped with an unary operation* $\kappa : M \to M$ *satisfying the following:*

$$\begin{array}{rl} \textit{(Compatibility)} & a \leq b \textit{ implies } \kappa a \leq \kappa b \textit{ for all } a, b, c \in M \\ \textit{(Minimality)} & \kappa a \leq a \textit{ for all } a \in M \\ \textit{(Duplication)} & \kappa a \leq \kappa\kappa a \textit{ for all } a \in M \\ \textit{(Left Exchange)} & \kappa a \circ b \leq b \circ \kappa a \textit{ for all } a, b \in M \\ \textit{(Right Exchange)} & a \circ \kappa b \leq \kappa b \circ a \textit{ for all } a, b \in M \end{array}$$

Compatibility results in $\kappa : M \to M$ being a functor in the biclosed poset, and the remainder of the axioms imply that κ is a comonad extending the biclosed poset with left and right exchange.

We can now define the two modalities in $\mathsf{Dial}_M(\mathsf{Set})$ where M is a biclosed poset with exchange; clearly we know $\mathsf{Dial}_M(\mathsf{Set})$ is also a model of the Lambek Calculus without modalities by Theorem 1 because M is a biclosed poset.

Definition 6. *Suppose* (U, X, α) *is an object of* $\mathsf{Dial}_M(\mathsf{Set})$ *where* M *is a biclosed poset with exchange. Then the* ***of-course*** *and* ***exchange*** *modalities can be defined as* $!(U, X, \alpha) = (U, U \to X^*, !\alpha)$ *and* $\kappa(U, X, \alpha) = (U, X, \kappa\alpha)$ *where* X^* *is the free commutative monoid on* X, $(!\alpha)(u, f) = \alpha(u, x_1) \circ \cdots \circ \alpha(u, x_i)$ *for* $f(u) = (x_1, \ldots, x_i)$, *and* $(\kappa\alpha)(u, x) = \kappa(\alpha(u, x))$.

This definition highlights a fundamental difference between the two modalities. The definition of the exchange modality relies on an extension of biclosed posets with essentially the exchange modality in the category of posets. However, the of-course modality is defined by the structure already present in $\mathsf{Dial}_M(\mathsf{Set})$, specifically, the structure of Set.

Both of the modalities have the structure of a comonad. That is, there are monoidal natural transformations $\varepsilon_! : !A \longrightarrow A$, $\varepsilon_\kappa : \kappa A \longrightarrow A$, $\delta_! : !A \longrightarrow !!A$, and $\delta_\kappa : \kappa A \longrightarrow \kappa\kappa A$ which satisfy the appropriate diagrams; see the formalization for the full proofs. Furthermore, these comonads come equipped with arrows $e : !A \longrightarrow I$, $d : !A \longrightarrow !A \otimes !A$, $\beta L : \kappa A \otimes B \longrightarrow B \otimes \kappa A$, and $\beta R : A \otimes \kappa B \longrightarrow \kappa B \otimes A$. Thus, we arrive at the following result.

Theorem 2. *Suppose* M *is a biclosed poset with exchange. Then* $\mathsf{Dial}_M(\mathsf{Set})$ *is a sound and complete model for the Lambek Calculi* $L_!$, L_κ, *and* $L_{!\kappa}$.

5 Type Theory for Lambek Systems

In this section we introduce typed calculi for each of the logics discussed so far. Each type system is based on the term assignment for Intuitionistic Linear Logic introduced in [2]. We show that they are all strongly normalizing and confluent, but we do not give full detailed proofs of each of these properties, because they are straightforward consequences of the proofs of strong normalization and confluence for intuitionistic linear logic. In fact, we will reference Bierman's thesis often within this section. The reader may wish to review Section 3.5 on page 88 of [4].

5.1 The Typed Lambek Calculus: λL

The first system we cover is the Lambek Calculus without modalities. This system can be seen as the initial core of each of the other systems we introduce below, and thus, we will simply extend the results here to three other systems.

The syntax for patterns, terms, and contexts are described by the following grammar:

$$\begin{aligned}
&\text{(patterns)}\ p := - \mid x \mid \mathsf{unit} \mid p_1 \otimes p_2 \\
&\text{(terms)}\ \ t := x \mid \mathsf{unit} \mid t_1 \otimes t_2 \mid \lambda_l x : A.t \mid \lambda_r x : A.t \mid \mathsf{app}_l\ t_1\ t_2 \mid \\
&\qquad\qquad \mathsf{app}_r\ t_1\ t_2 \mid \mathsf{let}\ t_1\ \mathsf{be}\ p\ \mathsf{in}\ t_2 \\
&\text{(contexts)}\ \Gamma := \cdot \mid x : A \mid \Gamma_1, \Gamma_2
\end{aligned}$$

Contexts are sequences of pairs of free variables and types. Patterns are only used in the let-expression which is itself used to eliminate logical connectives within the left rules of L. All variables in the pattern of a let-expression are bound. The remainder of the terms are straightforward.

The typing rules can be found in the in Figure 2 and the reduction rules in Figure 3. The typing rules are as one might expect. The reduction rules were extracted from the cut-elimination procedure for L.

We denote the reflexive and transitive closure of the $\rightsquigarrow$ by $\rightsquigarrow^*$. We call a term with no β-redexes a normal form, and we denote normal forms by n. In the interest of space we omit the congruence rules from the definition of the reduction relation; we will do this for each calculi introduced throughout this section. The other typed calculi we introduce below will be extensions of λL, thus, we do not reintroduce these rules each time for readability.

Strong normalization. It is well known that intuitionistic linear logic (ILL) is strongly normalizing, for example, see Bierman's thesis [4] or Benton's beautiful embedding of ILL into system F [3].

It is fairly straightforward to define a reduction preserving embedding of λL into ILL. Intuitionistic linear logic can be obtained from λL by replacing the rules T_IRL, T_ILL, T_IRR, and T_ILR with the following two rules:

$$\frac{\Gamma_2 \vdash t_1 : A \qquad \Gamma_1, x : B, \Gamma_3 \vdash t_2 : C}{\Gamma_1, z : A \multimap B, \Gamma_2, \Gamma_3 \vdash [z\ t_1/x]t_2 : C}\ \text{T\_IL} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A.t : A \multimap B}\ \text{T\_IR}$$

In addition, contexts are considered multisets, and hence, exchange is handled implicitly. Then we can reuse the idea of Benton's embeddings to show type preservation and type reduction.

At this point we define the following embeddings.

Definition 7. *We embed types and terms of λL into ILL as follows:*

Types:

$$I^{\mathsf{e}} = I \qquad (A \rightharpoonup B)^{\mathsf{e}} = A^{\mathsf{e}} \multimap B^{\mathsf{e}}$$
$$(A \otimes B)^{\mathsf{e}} = A^{\mathsf{e}} \otimes B^{\mathsf{e}} \qquad (A \leftharpoonup B)^{\mathsf{e}} = A^{\mathsf{e}} \multimap B^{\mathsf{e}}$$

Terms:

$$x^{\mathsf{e}} = x \qquad (\lambda_l x : A.t)^{\mathsf{e}} = \lambda x : A.t^{\mathsf{e}}$$
$$\mathsf{unit}^{\mathsf{e}} = \mathsf{unit} \qquad (\lambda_r x : A.t)^{\mathsf{e}} = \lambda x : A.t^{\mathsf{e}}$$
$$(t_1 \otimes t_2)^{\mathsf{e}} = t_1{}^{\mathsf{e}} \otimes t_2{}^{\mathsf{e}} \qquad (\mathsf{app}_l\ t_1\ t_2)^{\mathsf{e}} = t_1{}^{\mathsf{e}}\ t_2{}^{\mathsf{e}}$$
$$(\mathsf{let}\ t_1\ \mathsf{be}\ p\ \mathsf{in}\ t_2)^{\mathsf{e}} = \mathsf{let}\ t_1{}^{\mathsf{e}}\ \mathsf{be}\ p\ \mathsf{in}\ t_2{}^{\mathsf{e}} \qquad (\mathsf{app}_r\ t_1\ t_2)^{\mathsf{e}} = t_1{}^{\mathsf{e}}\ t_2{}^{\mathsf{e}}$$

The previous embeddings can be extended to contexts in the straightforward way, and to sequents as follows:

$$(\Gamma \vdash t : A)^{\mathsf{e}} = \Gamma^{\mathsf{e}} \vdash t^{\mathsf{e}} : A^{\mathsf{e}}$$

We can now prove strong normalization using the embedding preserves.

Theorem 3 (Strong Normalization).

- *If $\Gamma \vdash t : A$ in λL, then $\Gamma^{\mathsf{e}} \vdash t^{\mathsf{e}} : A^{\mathsf{e}}$ in ILL.*
- *If $t_1 \rightsquigarrow t_2$ in λL, then $t_1{}^{\mathsf{e}} \rightsquigarrow t_2{}^{\mathsf{e}}$ in ILL.*
- *If $\Gamma \vdash t : A$, then t is strongly normalizing.*

Proof. The first two cases hold by straightforward induction on the form of the assumed typing or reduction derivation. They then imply the third.

$$\frac{}{x : A \vdash x : A}\ \text{T\_VAR} \qquad \frac{}{\cdot \vdash \mathsf{unit} : I}\ \text{T\_UR}$$

$$\frac{\Gamma_2 \vdash t_1 : A \qquad \Gamma_1, x : A, \Gamma_3 \vdash t_2 : B}{\Gamma_1, \Gamma_2, \Gamma_3 \vdash [t_1/x]t_2 : B}\ \text{T\_CUT}$$

$$\frac{\Gamma_1, \Gamma_2 \vdash t : A}{\Gamma_1, x : I, \Gamma_2 \vdash \mathsf{let}\ x\ \mathsf{be\ unit\ in}\ t : A}\ \text{T\_UL}$$

$$\frac{\Gamma, x : A, y : B, \Gamma' \vdash t : C}{\Gamma, z : A \otimes B, \Gamma' \vdash \mathsf{let}\ z\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ t : C}\ \text{T\_TL} \qquad \frac{\Gamma_1 \vdash t_1 : A \qquad \Gamma_2 \vdash t_2 : B}{\Gamma_1, \Gamma_2 \vdash t_1 \otimes t_2 : A \otimes B}\ \text{T\_TR}$$

$$\frac{\Gamma_2 \vdash t_1 : A \qquad \Gamma_1, x : B, \Gamma_3 \vdash t_2 : C}{\Gamma_1, z : A \multimap B, \Gamma_2, \Gamma_3 \vdash [\mathsf{app}_r\ z\ t_1/x]t_2 : C}\ \text{T\_IRL}$$

$$\frac{\Gamma_2 \vdash t_1 : A \qquad \Gamma_1, x : B, \Gamma_3 \vdash t_2 : C}{\Gamma_1, \Gamma_2, z : B \multimap A, \Gamma_3 \vdash [\mathsf{app}_l\ z\ t_1/x]t_2 : C}\ \text{T\_ILL} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda_r x : A.t : A \multimap B}\ \text{T\_IRR}$$

$$\frac{x : A, \Gamma \vdash t : B}{\Gamma \vdash \lambda_l x : A.t : B \multimap A}\ \text{T\_ILR}$$

Fig. 2. Typing Rules for the Typed Lambek Calculus: λL

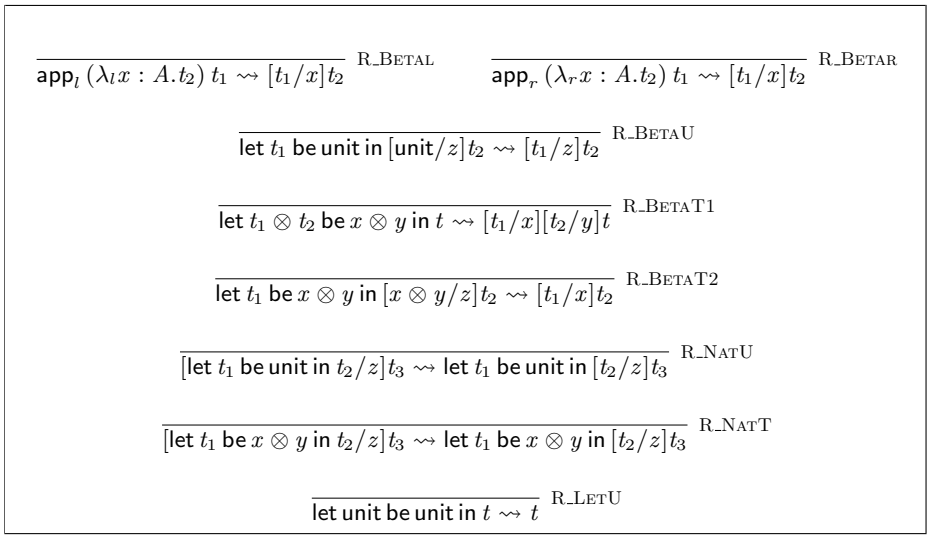
Confluence. The Church-Rosser property is well known to hold for ILL modulo commuting conversions, for example, see Theorem 19 of [4] on page 96. Since λL is essentially a subsystem of ILL, it is straightforward, albeit lengthly, to simply redefine Bierman's candidates and carry out a similar proof as Bierman's (Theorem 19 on page 96 of ibid.).

Theorem 4 (Confluence). *The reduction relation, $\rightsquigarrow$, modulo the commuting conversions is confluent.*

5.2 The Typed Lambek Calculus: $\lambda\mathbf{L}_!$

The calculus we introduce in this section is an extension of λL with the of-course modality $!A$. This extension follows from ILL exactly. The syntax of types and terms of λL are extended as follows:

$$\begin{array}{lll} \text{(types)} & A & := \cdots \mid !A \\ \text{(terms)} & t & := \cdots \mid \mathsf{copy}\ t'\ \mathsf{as}\ t_1, t_2\ \mathsf{in}\ t \mid \mathsf{discard}\ t'\ \mathsf{in}\ t \mid \mathsf{promote}_!\ t'\ \mathsf{for}\ t''\ \mathsf{in}\ t \mid \\ & & \quad \mathsf{derelict}_!\ t \end{array}$$



$$\frac{}{\mathsf{app}_l\,(\lambda_l x : A.t_2)\,t_1 \rightsquigarrow [t_1/x]t_2}\ \text{R\_BetaL} \qquad \frac{}{\mathsf{app}_r\,(\lambda_r x : A.t_2)\,t_1 \rightsquigarrow [t_1/x]t_2}\ \text{R\_BetaR}$$

$$\frac{}{\mathsf{let}\ t_1\ \mathsf{be\ unit\ in}\ [\mathsf{unit}/z]t_2 \rightsquigarrow [t_1/z]t_2}\ \text{R\_BetaU}$$

$$\frac{}{\mathsf{let}\ t_1 \otimes t_2\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ t \rightsquigarrow [t_1/x][t_2/y]t}\ \text{R\_BetaT1}$$

$$\frac{}{\mathsf{let}\ t_1\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ [x \otimes y/z]t_2 \rightsquigarrow [t_1/x]t_2}\ \text{R\_BetaT2}$$

$$\frac{}{[\mathsf{let}\ t_1\ \mathsf{be\ unit\ in}\ t_2/z]t_3 \rightsquigarrow \mathsf{let}\ t_1\ \mathsf{be\ unit\ in}\ [t_2/z]t_3}\ \text{R\_NatU}$$

$$\frac{}{[\mathsf{let}\ t_1\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ t_2/z]t_3 \rightsquigarrow \mathsf{let}\ t_1\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ [t_2/z]t_3}\ \text{R\_NatT}$$

$$\frac{}{\mathsf{let\ unit\ be\ unit\ in}\ t \rightsquigarrow t}\ \text{R\_LetU}$$

Fig. 3. Rewriting Rules for the Lambek Calculus: λL

The new type and terms are what one might expect, and are the traditional syntax used for the of-course modality. We add the following typing rules to λL:

$$\frac{\Gamma_1, x :!A, \Gamma_2, y :!A, \Gamma_3 \vdash t : B}{\Gamma_1, z :!A, \Gamma_2, \Gamma_3 \vdash \mathsf{copy}\ z\ \mathsf{as}\ x, y\ \mathsf{in}\ t : B}\ \text{T\_C} \qquad \frac{\Gamma_1, \Gamma_2 \vdash t : B}{\Gamma_1, x :!A, \Gamma_2 \vdash \mathsf{discard}\ x\ \mathsf{in}\ t : B}\ \text{T\_W}$$

$$\frac{\overrightarrow{x} :!\Gamma \vdash t : B}{\overrightarrow{y} :!\Gamma \vdash \mathsf{promote}_!\ \overrightarrow{y}\ \mathsf{for}\ \overrightarrow{x}\ \mathsf{in}\ t :!B}\ \text{T\_Br} \qquad \frac{\Gamma_1, x : A, \Gamma_2 \vdash t : B}{\Gamma_1, y :!A, \Gamma_2 \vdash [\mathsf{derelict}_!\ y/x]t : B}\ \text{T\_Bl}$$

Finally, the reduction rules can be found in Figure 4. The equality used in the R_BetaC rule is definitional, meaning, that the rule simply gives the terms on the right side of the equation the name on the left side, and nothing more. This makes the rule easier to read.

Strong normalization. Showing strong normalization for λL$_!$ easily follows by a straightforward extension of the embedding we gave for λL.

Definition 8. *The following is an extension of the embedding of λL into ILL resulting in an embedding of types and terms of λL$_!$ into ILL. First, we define $(!A)^{\mathsf{e}} = !A^{\mathsf{e}}$, then the following defines the embedding of terms:*

$$\begin{aligned}
(\mathsf{copy}\ t'\ \mathsf{as}\ t_1, t_2\ \mathsf{in}\ t)^{\mathsf{e}} &= \mathsf{copy}\ t'^{\mathsf{e}}\ \mathsf{as}\ t_1{}^{\mathsf{e}}, t_2{}^{\mathsf{e}}\ \mathsf{in}\ t^{\mathsf{e}} \\
(\mathsf{discard}\ t'\ \mathsf{in}\ t)^{\mathsf{e}} &= \mathsf{discard}\ t'^{\mathsf{e}}\ \mathsf{in}\ t^{\mathsf{e}} \\
(\mathsf{promote}_!\ t'\ \mathsf{for}\ t''\ \mathsf{in}\ t)^{\mathsf{e}} &= \mathsf{promote}_!\ t'^{\mathsf{e}}\ \mathsf{for}\ t''^{\mathsf{e}}\ \mathsf{in}\ t^{\mathsf{e}} \\
(\mathsf{derelict}_!\ t)^{\mathsf{e}} &= \mathsf{derelict}_!\ t^{\mathsf{e}}
\end{aligned}$$

Just as before this embedding is type preserving and reduction preserving.

Theorem 5 (Type and Reduction Preserving Embedding).

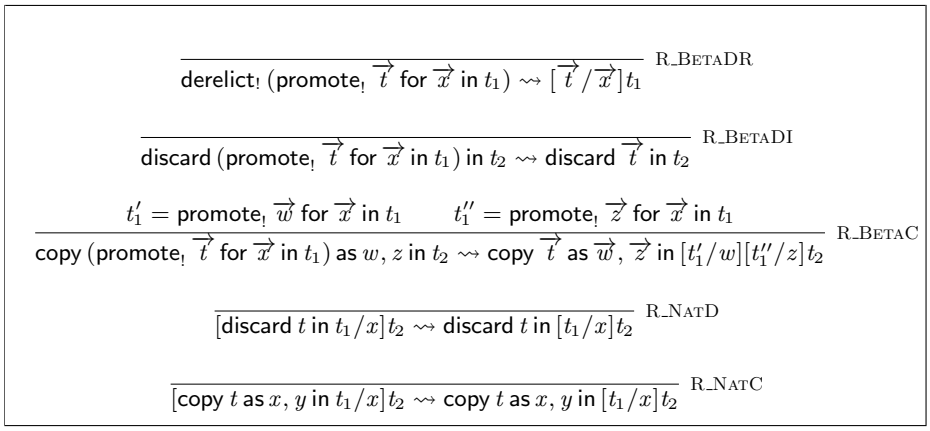


Fig. 4. Rewriting Rules for The Typed Lambek Calculus: $\lambda\mathrm{L}_!$

- *If* $\Gamma \vdash t : A$ *in* $\lambda L_!$*, then* $\Gamma^{\mathsf{e}} \vdash t^{\mathsf{e}} : A^{\mathsf{e}}$ *in ILL.*
- *If* $t_1 \rightsquigarrow t_2$ *in* $\lambda L_!$*, then* ${t_1}^{\mathsf{e}} \rightsquigarrow {t_2}^{\mathsf{e}}$ *in ILL.*
- *If* $\Gamma \vdash t : A$*, then* t *is strongly normalizing.*

Proof. The first two cases hold by straightforward induction on the form of the assumed typing or reduction derivation. They then imply the third.

Confluence. The Church-Rosser property also holds for $\lambda\mathrm{L}_!$, and can be shown by straightforwardly applying a slightly modified version of Bierman's proof [4] just as we did for λL. Thus, we have the following:

Theorem 6 (Confluence). *The reduction relation,* $\rightsquigarrow$*, modulo the commuting conversions is confluent.*

5.3 The Typed Lambek Calculus: $\lambda\mathbf{L}_\kappa$

The next calculus we introduce is also an extension of λL with a modality that adds exchange to $\lambda\mathrm{L}_\kappa$ denoted κA. It is perhaps the most novel of the calculi we have introduced.

The syntax of types and terms of λL are extended as follows:

$$\begin{array}{ll} \text{(types)} & A := \cdots \mid \kappa A \\ \text{(terms)} & t \ := \cdots \mid \mathsf{exchange}_{\mathsf{l}}\ t_1, t_2\ \mathsf{with}\ x, y\ \mathsf{in}\ t_3 \mid \mathsf{exchange}_{\mathsf{r}}\ t_1, t_2\ \mathsf{with}\ x, y\ \mathsf{in}\ t_3 \mid \\ & \qquad \mathsf{promote}_\kappa\ t'\ \mathsf{for}\ t''\ \mathsf{in}\ t \mid \mathsf{derelict}_\kappa\ t \end{array}$$

The syntax for types has been extended to include the exchange modality, and the syntax of terms follow suit. The terms $\mathsf{exchange}_{\mathsf{l}}\ t_1, t_2\ \mathsf{with}\ x, y\ \mathsf{in}\ t_3$ and $\mathsf{exchange}_{\mathsf{r}}\ t_1, t_2\ \mathsf{with}\ x, y\ \mathsf{in}\ t_3$ are used to explicitly track uses of exchange throughout proofs.

We add the following typing rules to λL:

$$\frac{\Gamma_1, x_1 : \kappa A, y_1 : B, \Gamma_2 \vdash t : C}{\Gamma_1, y_2 : B, x_2 : \kappa A, \Gamma_2 \vdash \mathsf{exchange_l}\ y_2, x_2\ \mathsf{with}\ x_1, y_1\ \mathsf{in}\ t : C}\ \text{T\_E1}$$

$$\frac{\Gamma_1, x_1 : A, y_1 : \kappa B, \Gamma_2 \vdash t : C}{\Gamma_1, y_2 : \kappa B, x_2 : A, \Gamma_2 \vdash \mathsf{exchange_r}\ y_2, x_2\ \mathsf{with}\ x_1, y_1\ \mathsf{in}\ t : C}\ \text{T\_E2}$$

$$\frac{\overrightarrow{x} : \kappa\Gamma \vdash t : B}{\overrightarrow{y} : \kappa\Gamma \vdash \mathsf{promote}_\kappa\ \overrightarrow{y}\ \mathsf{for}\ \overrightarrow{x}\ \mathsf{in}\ t : \kappa B}\ \text{T\_ER} \qquad \frac{\Gamma_1, x : A, \Gamma_2 \vdash t : B}{\Gamma_1, y : \kappa A, \Gamma_2 \vdash [\mathsf{derelict}_\kappa\ y/x]t : B}\ \text{T\_EL}$$

The reduction rules are in Figure 5, and are vary similar to the rules from λL$_!$.

$$\frac{}{\mathsf{derelict}_\kappa\ (\mathsf{promote}_\kappa\ \overrightarrow{t}\ \mathsf{for}\ \overrightarrow{x}\ \mathsf{in}\ t_1) \rightsquigarrow [\overrightarrow{t}/\overrightarrow{x}]t_1}\ \text{R\_BetaEDR}$$

$$\frac{}{[\mathsf{exchange_l}\ t_1, t_2\ \mathsf{with}\ x, y\ \mathsf{in}\ t_3/z]t_4 \rightsquigarrow \mathsf{exchange_l}\ t_1, t_2\ \mathsf{with}\ x, y\ \mathsf{in}\ [t_3/z]t_4}\ \text{R\_NatEl}$$

$$\frac{}{[\mathsf{exchange_r}\ t_1, t_2\ \mathsf{with}\ x, y\ \mathsf{in}\ t_3/z]t_4 \rightsquigarrow \mathsf{exchange_r}\ t_1, t_2\ \mathsf{with}\ x, y\ \mathsf{in}\ [t_3/z]t_4}\ \text{R\_NatEr}$$

Fig. 5. Rewriting Rules for The Typed Lambek Calculus: λL_κ

Strong normalization. Similarly, we show that we can embed λL_κ into ILL, but the embedding is a bit more interesting.

Definition 9. *The following is an extension of the embedding of λL into ILL resulting in an embedding of types and terms of λL_κ into ILL. First, we define $(\kappa A)^{\mathsf{e}} = !A^{\mathsf{e}}$, and then the following defines the embedding of terms:*

$$\begin{aligned}
(\mathsf{exchange_l}\ t_1, t_2\ \mathsf{with}\ x, y\ \mathsf{in}\ t_3)^{\mathsf{e}} &= [t_2{}^{\mathsf{e}}/x][t_1{}^{\mathsf{e}}/y]t_3{}^{\mathsf{e}} \\
(\mathsf{exchange_r}\ t_1, t_2\ \mathsf{with}\ x, y\ \mathsf{in}\ t_3)^{\mathsf{e}} &= [t_2{}^{\mathsf{e}}/x][t_1{}^{\mathsf{e}}/y]t_3{}^{\mathsf{e}} \\
(\mathsf{promote}_\kappa\ t'\ \mathsf{for}\ t''\ \mathsf{in}\ t)^{\mathsf{e}} &= \mathsf{promote_!}\ t'^{\mathsf{e}}\ \mathsf{for}\ t''^{\mathsf{e}}\ \mathsf{in}\ t^{\mathsf{e}} \\
(\mathsf{derelict}_\kappa\ t)^{\mathsf{e}} &= \mathsf{derelict_!}\ t^{\mathsf{e}}
\end{aligned}$$

The embedding translates the exchange modality into the of-course modality of ILL. We do this so as to preserve the comonadic structure of the exchange modality. One might think that we could simply translate the exchange modality to the identity, but as Benton showed [3], this would result in an embedding that does not preserve reductions. Furthermore, the left and right exchange terms are translated away completely, but this works because ILL contains exchange in general, and hence, does not need to be tracked explicitly. We now have strong normalization and confluence.

Theorem 7 (Strong Normalization).

– *If $\Gamma \vdash t : A$ in $\lambda L_!$, then $\Gamma^{\mathsf{e}} \vdash t^{\mathsf{e}} : A^{\mathsf{e}}$ in ILL.*

– *If* $t_1 \rightsquigarrow t_2$ *in* $\lambda L_!$*, then* ${t_1}^{\mathsf{e}} \rightsquigarrow {t_2}^{\mathsf{e}}$ *in ILL.*
– *If* $\Gamma \vdash t : A$*, then* t *is strongly normalizing.*
– *The reduction relation,* $\rightsquigarrow$*, modulo the commuting conversions is confluent.*

Proof. The first two cases hold by straightforward induction on the form of the assumed typing or reductions derivation. They then imply the third case.

5.4 The Typed Lambek Calculus: $\lambda L_{!\kappa}$

If we combine all three of the previous typed Lambek Calculi, then we obtain the typed Lambek Calculus $\lambda L_{!\kappa}$. The main characteristics of this system are that it provides the benefits of the non-symmetric adjoint structure of the Lambek Calculus with the ability of having exchange, and the of-course modality, but both are carefully tracked within the proofs.

Strong normalization for this calculus can be proved similarly to the previous calculi by simply merging the embeddings together. Thus, both modalities of $\lambda L_{!\kappa}$ would merge into the of-course modality of ILL. The Church-Rosser property also holds for $\lambda L_{!\kappa}$ by extending the proof of confluence for ILL by Bierman [4] just as we did for the other systems. Thus, we have the following results.

Theorem 8 (Strong Normalization). *If* $\Gamma \vdash t : A$*, then* t *is strongly normalizing.*

Theorem 9 (Confluence). *The reduction relation,* $\rightsquigarrow$*, modulo the commuting conversions is confluent.*

6 Conclusions

We have recalled how to use biclosed posets and sets to construct dialectica-like models of the Lambek Calculus. This construction is admittedly not the easiest one, which is the reason why we use automated tools to verify our definitions, but it has one striking advantage. It shows how to introduce modalities to recover the expressive power of intuitionistic (and a posteriori classical) propositional logic to the system. We know of no other construction of models of Lambek Calculus that does model modalities, not using their syntactic properties. (The traditional view in algebraic semantics is to consider idempotent operators for modalities like !). The categorical semantics here has been described before [10], but the syntactic treatment of the lambda-calculi, in the style of [2] had not been done and there were doubts about its validity, given the results of Jay [16]. We are glad to put this on a firm footing, using another one of Benton's ideas: his embedding of intuitionistic linear logic into system F. Finally, we envisage more work, along the lines of algebraic proof theory, for modalities and non-symmetric type systems.

References

1. Samson Abramsky. Proofs as processes. *Theoretical Computer Science*, 135(1):5 – 9, 1994.
2. Nick Benton, Gavin Bierman, Valeria De Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In *International Conference on Typed Lambda Calculi and Applications*, pages 75–90. Springer Berlin Heidelberg, 1993.
3. P. N. Benton. Strong normalisation for the linear term calculus. *Journal of Functional Programming*, 5:65–80, 1 1995.
4. G. M. Bierman. *On Intuitionistic Linear Logic*. PhD thesis, Wolfson College, Cambridge, December 1993.
5. G.M. Bierman and V.C.V. de Paiva. On an intuitionistic modal logic. *Studia Logica*, 65(3):383–416, 2000.
6. Ana Bove, Peter Dybjer, and Ulf Norell. A brief overview of Agda-a functional language with dependent types. *TPHOLs*, 5674:73–78, 2009.
7. Agata Ciabattoni, Nikolaos Galatos, and Kazushige Terui. Algebraic proof theory for substructural logics: Cut-elimination and completions. *Annals of Pure and Applied Logic*, 163(3):266 – 290, 2012.
8. Bob Coecke, Edward Grefenstette, and Mehrnoosh Sadrzadeh. Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus. *Annals of Pure and Applied Logic*, 164(11):1079–1100, 2013.
9. Valeria de Paiva. *The dialectica categories*. PhD thesis, Computer Laboratory, University of Cambridge, PhD Thesis, 1990. Computer Laboratory, University of Cambridge, PhD Thesis.
10. Valeria de Paiva. A Dialectica model of the Lambek calculus. In *8th Amsterdam Logic Colloquium*, 1991.
11. Valeria De Paiva. Dialectica and chu constructions: Cousins? *Theory and Applications of Categories*, 17(7):127–152, 2007.
12. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1 – 101, 1987.
13. G. Greco and A. Palmigiano. Linear Logic Properly Displayed. *ArXiv e-prints*, November 2016.
14. Kohei Honda and Olivier Laurent. An exact correspondence between a typed pi-calculus and polarised proof-nets. *Theoretical Computer Science*, 411(22):2223 – 2238, 2010.
15. Martin Hyland and Valeria de Paiva. Full intuitionistic linear logic (extended abstract). *Annals of Pure and Applied Logic*, 64(3):273 – 291, 1993.
16. C. Barry Jay. Coherence in category theory and the Church-Rosser property. *Notre Dame J. Formal Logic*, 33(1):140–143, 12 1991.
17. Yves Lafont and Thomas Streicher. Games semantics for linear logic. In *Logic in Computer Science, 1991. LICS'91., Proceedings of Sixth Annual IEEE Symposium on*, pages 43–50. IEEE, 1991.
18. F. Lamarche and C. Retoré. Proof nets for the Lambek calculus – an overview. In *Proceedings of the Third Roma Workshop. Proofs and Linguistic Categories*, pages 241–262, 1996.
19. Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, pages 154–170, 1958.
20. Michael Moortgat. Typelogical grammar. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2014 edition, 2014.
21. Jeff Polakow. *Ordered linear logic and applications*. PhD thesis, Carnegie Mellon University, 2001.

22. Vaughan R. Pratt. Types as processes, via Chu spaces. *Electronic Notes in Theoretical Computer Science*, 7:227 – 247, 1997.
23. P. Sewell, F. Nardelli, S. Owens, G. Peskine, T. Ridge, S. Sarkar, and R. Strnisa. Ott: Effective tool support for the working semanticist. In *Journal of Functional Programming*, volume 20, pages 71–122, 2010.
24. M. E. Szabo. *Algebra of Proofs*. Studies in Logic and the Foundations of Mathematics, Vol. 88, North-Holland, 1978, 1979.