Dialectica Models of the Lambek Calculus Revisited

Valeria de Paiva and Harley Eades III

April 2016

Introduction

This note recalls a Dialectica model of the Lambek Calculus presented by the first author in the Amsterdam Colloquium in 1991. We approach the Lambek Calculus from the perspective of Linear Logic. In that earlier work we took for granted the syntax of the calculus and only discussed the exciting possibilities of new models of Linear Logic like systems.

Twenty five years later we find that the work is still interesting and that it might inform some of the most recent work on the relationship between Categorial Grammars and notions of distributional semantics. But the Amsterdam Colloquium proceedings were never published and not even the author had a copy of the paper. So we have decided to revisit some of the old work, this time using the new tools that have been developed for type theory and sequent proof systems in the time that elapsed. Thus, we implement the calculus in Agda and we use `Ott` [8] to check that we do not have silly mistakes in our term systems. The goal is to see if our new implementations can shed new light on some of the issues that remained open, mostly on the applicability and fit of the original systems to their intended uses.

Historical Overview

The Syntactic Calculus was first introduced by Joachim Lambek in 1958 [7]. Now known as the Lambek Calculus, its motivation was to provide 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 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', pay attention to resources, if desired, but always forget this accounting, if preferred, in the same system. 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 general background on the type-logical approach, there are the monographs of Moortgat, Morril, Carpenter and Steedman. For a shorter introduction, see Moortgat's chapter on the Handbook of Logic in Language [**?**] or the Stanford Encyclopedia Of Philosophy article on Type Logical Grammar.

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 variants of Linear Logic. These systems can be as expressive as Intuitionistic Logic and the claim is that they are more precise i.e. they make finer distinctions. From the beginning it was clear that the Lambek Calculus is the multiplicative fragment of non-commutative Intuitionistic Linear Logic. Hence several interesting questions, considered for Linear Logic, could also be asked of the Lambek Calculus. 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 very 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, building on work of Galatos and others.

We first recall Linear Logic and provide the transformations to show that the Lambek Calculus L really is the multiplicative fragment of non-commutative Intuitionistic Linear Logic. Then we describe the usual String Semantics for the Lambek Calculus L and generalize it, using a categorical perspective in the second section. The third section recalls our Dialectica model for the Lambek Calculus. Finally, in the fourth section we discuss modalities and some untidiness of the Curry-Howard correspondence for the fragments of Linear Logic in question.

1 The Lambek Calculus and Some Extensions

The Lambek Calculus, formerly the Syntactic Calculus L, due to J. Lambek [7], was created to capture the logical structure of sentences. Lambek introduced what we think of as a substructural logic with an operator denoting concate-

$$\frac{}{A \vdash A}\ \textsc{ax} \qquad \frac{}{\cdot \vdash I}\ \textsc{Ur} \qquad \frac{\Gamma_2 \vdash A \qquad \Gamma_1, A, \Gamma_3 \vdash B}{\Gamma_1, \Gamma_2, \Gamma_3 \vdash B}\ \textsc{cut}$$

$$\frac{\Gamma_1, \Gamma_2 \vdash A}{\Gamma_1, I, \Gamma_2 \vdash A}\ \textsc{Ul} \qquad \frac{\Gamma, A, B, \Gamma' \vdash C}{\Gamma, A \otimes B, \Gamma' \vdash C}\ \textsc{Tl}$$

$$\frac{\Gamma_1 \vdash A \qquad \Gamma_2 \vdash B}{\Gamma_1, \Gamma_2 \vdash A \otimes B}\ \textsc{Tr} \qquad \frac{\Gamma_2 \vdash A \qquad \Gamma_1, B, \Gamma_3 \vdash C}{\Gamma_1, A \rightharpoonup B, \Gamma_2, \Gamma_3 \vdash C}\ \textsc{IRl}$$

$$\frac{\Gamma_2 \vdash A \qquad \Gamma_1, B, \Gamma_3 \vdash C}{\Gamma_1, \Gamma_2, B \leftharpoonup A, \Gamma_3 \vdash C}\ \textsc{ILl} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightharpoonup B}\ \textsc{IRr}$$

$$\frac{A, \Gamma \vdash B}{\Gamma \vdash B \leftharpoonup A}\ \textsc{ILr}$$

Figure 1: The Lambek Calculus: L

nation, $A \otimes B$, 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 proceeded by a phrase of type A.

The Lambek Calculus can be presented as a non-commutative intuitionistic multiplicative linear logic. The syntax of formulas and contexts of the logic are as follows:

$$\begin{array}{ll} \text{(formulas)} & A, B, C ::= I \mid A \otimes B \mid A \leftharpoonup B \mid A \rightharpoonup B \\ \text{(contexts)} & \Gamma ::= A \mid \Gamma_1, \Gamma_2 \end{array}$$

We denote mapping the modalities over an arbitrary context by $!\Gamma$ and $\kappa\Gamma$. The inference rules are defined in Figure 1.

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.

1.1 Adding Weakening and Contraction to the Lambek Calculus

The linear modality, $!A$, read "of-course A" was first proposed by Girard [3] 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:

$$\frac{\Gamma_1, !A, \Gamma_2, !A, \Gamma_3 \vdash B}{\Gamma_1, !A, \Gamma_2, \Gamma_3 \vdash B}\ \text{C} \qquad \frac{\Gamma_1, \Gamma_2 \vdash B}{\Gamma_1, !A, \Gamma_2 \vdash B}\ \text{W}$$

$$\frac{!\Gamma \vdash B}{!\Gamma \vdash !B}\ \textsc{Br} \qquad \frac{\Gamma_1, A, \Gamma_2 \vdash B}{\Gamma, !A, \Gamma_2 \vdash B}\ \textsc{Bl}$$

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[**?**]. Thus, under the of-course modality the logic becomes non-linear. We will see in Section 4 that these rules define a comonad. We call the extension of L with the of-course modality $\mathrm{L}_!$.

1.2 Adding Exchange to the Lambek Calculus

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:

$$\frac{\kappa\Gamma \vdash B}{\kappa\Gamma \vdash \kappa B}\ \textsc{Er} \qquad \frac{\Gamma_1, A, \Gamma_2 \vdash B}{\Gamma_1, \kappa A, \Gamma_2 \vdash B}\ \textsc{El}$$

$$\frac{\Gamma_1, \kappa A, B, \Gamma_2 \vdash C}{\Gamma_1, B, \kappa A, \Gamma_2 \vdash C}\ \text{E1} \qquad \frac{\Gamma_1, A, \kappa B, \Gamma_2 \vdash C}{\Gamma_1, \kappa B, A, \Gamma_2 \vdash C}\ \text{E2}$$

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.

2 Algebraic Semantics

In Lambek's original paper [7] introducing his calculus L, albeit without modalities, he introduced an algebraic semantics that is now called the String Semantics for L. The semantics begins with a non-empty set of expressions denoted V^+, and then by modeling formulas of L by subsets of V^+ it proceeds by defining operations on these subsets, which correspond to the logical connectives of L. Each

operation is defined as follows (using our notation for the logical connectives):

$$\begin{aligned} A \otimes B &= \{xy \in \mathrm{V}^+ \mid x \in A \text{ and } y \in B\} \\ A \multimap B &= \{x \in \mathrm{V}^+ \mid \text{for all } y \in B, xy \in A\} \\ A \multimap B &= \{y \in \mathrm{V}^+ \mid \text{for all } x \in A, xy \in B\} \end{aligned}$$

Let $\mathcal{N} = \mathcal{P}(\mathrm{V}^+)$ be the powerset of V^+. Then we can view each of the above definitions as binary operations with type $\mathcal{N} \times \mathcal{N} \longrightarrow \mathcal{N}$. In fact, $\mathcal{N}$ has a natural order induced by set containment, and concatenation, $A \otimes B$, gives $\mathcal{N}$ a non-commutative monoidal structure, where the unit $I = \{\epsilon\}$ is the set containing the empty sequence:

(associativity)	$A \otimes (B \otimes C) = (A \otimes B) \otimes C$
(unit)	$A \otimes I = A = I \otimes A$
(non-commutativity)	$A \otimes B \neq B \otimes A$, in general

There happens to be a more general structure underlying the previous semantics. We now make this structure explicit using the tools developed by Hyland and de Paiva [4]. The ordering on $\mathcal{N}$ induces a poset $(\mathcal{N}, \subseteq)$, but even more so, $\mathcal{N}$ is also a monoid $(\mathcal{N}, \otimes, I)$, but that is not all, these two structures are compatible, that is, given $A \subseteq B$ the following hold:

$$\begin{aligned} A \otimes C \subseteq B \otimes C, \text{ for all } C \in \mathcal{N} \\ C \otimes A \subseteq C \otimes B, \text{ for all } C \in \mathcal{N} \end{aligned}$$

Abstracting this structure out yields what is call an ordered non-commutative monoid.

Definition 1. *An **ordered non-commutative monoid**, $(M, \leq, \circ, e)$, is a poset $(M, \leq)$ with a given compatible monoidal structure $(M, \circ, e)$. That is, a set M equipped with a binary relation, $\leq\colon M \times M \longrightarrow \mathsf{2}$, satisfying:*

(reflexivity)	$a \leq a$ *for all* $a \in M$
(transitivity)	$a \leq b$ *and* $b \leq c$, *implies that* $a \leq c$ *for all* $a, b, c \in M$
(antisymmetry)	$a \leq b$ *and* $b \leq a$, *implies that* $a = b$

together with a monoidal multiplication $\circ : M \times M \longrightarrow M$ and a distinguished object $e \in M$ satisfying the following:

(associativity)	$a \circ (b \circ c) = (a \circ b) \circ c$
(identity)	$e \circ a = a = a \circ e$

The structures are compatible in the sense that, if $a \leq b$, then the following hold:

$$\begin{aligned} a \circ c \leq b \circ c \text{ for any } c \in M \\ c \circ a \leq c \circ b \text{ for any } c \in M \end{aligned}$$

It is easy to see that the previous definition accounts for all of the structure we have described so far, and thus, we may conclude that $(\mathcal{N}, \subseteq, \otimes, I)$ is an ordered non-commutative monoid, however, this definition is not able to model the implication operations $A \leftharpoonup B$ and $A \rightharpoonup B$. To do this we need to understand how implication relates to the ordered non-commutative monoid structure. Notice that the following hold:

$$A \otimes (A \rightharpoonup B) \subseteq B$$
$$(A \leftharpoonup B) \otimes A \subseteq B$$

Furthermore, there are no larger objects of $\mathcal{N}$ with these properties. Abstracting this results in the notion of a biclosed poset.

Definition 2. *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 $a \leftharpoonup b$ 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 $a \leftharpoonup b$ exist for any $a, b \in M$.*

At this point we have everything we need to model the Lambek Calculus L without modalities.

Lemma 3. *Any biclosed poset $(M, \leq, \circ, e, \rightharpoonup, \leftharpoonup)$ is a model for the Lambek Calculus L without modalities.*

Proof. First suppose we have an assignment $(-)^0$ which assigns to each formula of L an element of M. Then if $\Gamma \vdash A$ holds we show that $(\Gamma)^0 \leq (A)^0$. This proof can easily be completed by induction on the form $\Gamma \vdash A$. □

3 Categorical Models

Historically the Lambek calculus arose from its intended categorical model, biclosed symmetric monoidal categories, as discussed by Lambek in [6]. Clearly all this happened almost three decades before Girard introduced Linear Logic, hence there were no modalities or exponentials in this setting.

Categorically, one models the of-course modality as a functor endowed with the structure of a comonad $(!, \delta_A, \varepsilon_A)$ with some additional structure.

That is, there are maps $\delta_A : !A \longrightarrow !!A$ and $\varepsilon_A : !A \longrightarrow A$ subject to a few coherence diagrams, and maps $c_A : !A \longrightarrow !A \otimes !A$ and $w_A : !A \longrightarrow I$. Using this structure we can interpret the rules of the of-course modality. Consider the rule C, and suppose we have a map $\Gamma \otimes (!A \otimes !A) \xrightarrow{f} B$, then we can obtain a new map $\Gamma \otimes !A \xrightarrow{\mathsf{id}_\Gamma \otimes c_A} \Gamma \otimes (!A \otimes !A) \xrightarrow{f} B$. The rule W is similar, but we start with a map $\Gamma \xrightarrow{f} B$ and then we can define the map $\Gamma \otimes !A \xrightarrow{\mathsf{id}_\Gamma \otimes w_A} \Gamma \otimes I \xrightarrow{\cong} \Gamma \xrightarrow{f} B$.

Notice that the previous map exploits the fact that I is the unit for tensor. Now consider the rule BL, and suppose we have a map $!\Gamma \xrightarrow{f} B$, then we may obtain a second map using the fact that of-course is a functor $!\Gamma \xrightarrow{\delta} !!\Gamma \xrightarrow{!f} !B$. Finally, consider the rule BR and suppose we have a map $\Gamma \otimes A \xrightarrow{f} B$, then we can construct the map $\Gamma \otimes !A \xrightarrow{\mathsf{id}_\Gamma \otimes \varepsilon_A} \Gamma \otimes A \xrightarrow{f} B$. This analysis tells us a few things about interpreting logics into categorical models. Sequents, $\Gamma \vdash B$, are interpreted as morphisms, $\Gamma \xrightarrow{f} B$, where Γ is I if it is empty, or it is the tensor product of the interpretations of its formulas. Then interpreting inference rules amounts to starting with the morphisms corresponding to the premises, and then building a map corresponding to the conclusion. The cut-elimination procedure is defined by a set of equations between derivations, and hence, in the model corresponds to equations between morphisms. The various coherence diagrams relating the structure of the model enforce that these equations hold.

We can similarly interpret the rules for the exchange modality. That is, as a functor endowed with the structure of a second comonad, but also with the maps $e_1 : A \otimes \kappa B \longrightarrow \kappa B \otimes A$ and $e_2 : \kappa A \otimes B \longrightarrow B \otimes \kappa A$. Then, each of the inference rules for the exchange modality can easily be interpreted into the model.

4 Dialectica Lambek Spaces

5 Typed Lambek Calculi

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 Full Intuitionistic Linear Logic introduced by Hyland and de Paiva [5]. 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 [2].

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 those other systems.

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

grammar:

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

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 Figure 2 and the reduction rules in Figure 3. The typing rules are as one might expect. Finally, the reduction rules were

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

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

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

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

$$\frac{\Gamma_1 \vdash t_1 : A \qquad \Gamma_2 \vdash t_2 : B}{\Gamma_1, \Gamma_2 \vdash t_1 \otimes t_2 : A \otimes B}\ \text{T\_TR}$$

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

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

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

Figure 2: Typing Rules for The Typed Lambek Calculus: λL

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 $\beta\eta$-redexes a normal form, and we denote normal forms by n. 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 [2] or Benton's beautiful embedding of ILL into system F [1]. It is fairly straightforward to define a reduction preserving embedding of λL – and as we will see the other calculi can be as well – 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.

At this point we define the following embeddings.

Definition 4. *We embed types and terms of λL into ILL as follows:*

Types:

$$\begin{aligned} I^{\mathsf{e}} &= I \\ (A \otimes B)^{\mathsf{e}} &= A^{\mathsf{e}} \otimes B^{\mathsf{e}} \\ (A \rightharpoonup B)^{\mathsf{e}} &= A^{\mathsf{e}} \multimap B^{\mathsf{e}} \\ (A \leftharpoonup B)^{\mathsf{e}} &= A^{\mathsf{e}} \multimap B^{\mathsf{e}} \end{aligned}$$

Terms:

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

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

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

This embedding preserves typing and reduction.

Lemma 5 (Type Preserving Embedding)**.** *If $\Gamma \vdash t : A$ in λL, then $\Gamma^{\mathsf{e}} \vdash t^{\mathsf{e}} : A^{\mathsf{e}}$ in ILL.*

Proof. This holds by straightforward induction on the form of the assumed typing derivation. □

Lemma 6 (Reduction Preserving Embedding)**.** *If $t_1 \rightsquigarrow t_2$ in λL, then ${t_1}^{\mathsf{e}} \rightsquigarrow {t_2}^{\mathsf{e}}$ in ILL.*

Proof. This holds by straightforward induction on the form of the assumed reduction derivation. □

The previous two results imply that λL is strongly normalizing, because ILL is.

Corollary 7 (Strong Normalization)**.** *If $\Gamma \vdash t : A$, then t is strongly normalizing.*

Proof. Suppose t is not strongly normalizing. Then there exists at least one infinite descending chain starting with t of the form $t \rightsquigarrow t' \rightsquigarrow t'' \rightsquigarrow \cdots$. Lemma 6 implies that there must also be an infinite descending chain of the form $t^{\mathsf{e}} \rightsquigarrow t'^{\mathsf{e}} \rightsquigarrow t''^{\mathsf{e}} \rightsquigarrow \cdots$, but we know this is impossible, because ILL is strongly normalizing. □

Confluence. The Church-Rosser property is well known to hold for ILL modulo commuting conversions, for example, see Theorem 19 of [2] on page 96. Since λL is essentially a subsystem of ILL, it is straightforward, albeit lengthly, to simply redefine Bierman's candidates (Definitions 17, 19, 20, and 21 of Section 5.1 of ibid.), and then carry out a similar proof as Bierman's (Theorem 19 on page 96 of ibid.). However, giving the details here would not be fruitful nor enlightening, and hence, we omit the proof. Thus, we have the following:

Theorem 8 (Confluence)**.** *The reduction relation, $\rightsquigarrow$, modulo the commuting conversions is confluent.*

6 Conclusion

References

[1] P. N. Benton. Strong normalisation for the linear term calculus. *Journal of Functional Programming*, 5:65–80, 1 1995.

[2] G. M. Bierman. *On Intuitionistic Linear Logic*. PhD thesis, Wolfson College, Cambridge, December 1993.

[3] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1 – 101, 1987.

[4] Martin Hyland and Valeria de Paiva. Lineales. *"O que nos faz pensar", Special number in Logic, Cadernos do Dept. de Filosofia da PUC", Pontificial Catholic University of Rio de Janeiro*, 1991.

[5] Martin Hyland and Valeria de Paiva. Full intuitionistic linear logic (extended abstract). *Annals of Pure and Applied Logic*, 64(3):273 – 291, 1993.

[6] J. Lambek. *Categorial Grammars and Natural Language Structures*, chapter Categorial and Categorical Grammars, pages 297–317. Springer Netherlands, Dordrecht, 1988.

[7] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, pages 154–170, 1958.

[8] 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.

$$\frac{}{\lambda_l x : A.\mathsf{app}_l\ t\ x \rightsquigarrow t}\ \text{R\_ETAL} \qquad \frac{}{\lambda_r x : A.\mathsf{app}_r\ t\ x \rightsquigarrow t}\ \text{R\_ETAR}$$

$$\frac{}{\mathsf{app}_l\ (\lambda_l x : A.t_2)\ t_1 \rightsquigarrow [t_1/x]t_2}\ \text{R\_BETAL}$$

$$\frac{}{\mathsf{app}_r\ (\lambda_r x : A.t_2)\ t_1 \rightsquigarrow [t_1/x]t_2}\ \text{R\_BETAR}$$

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

$$\frac{}{\mathsf{let}\ t_1 \otimes t_2\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ t \rightsquigarrow [t_1/x][t_2/y]t}\ \text{R\_BETAT1}$$

$$\frac{}{\mathsf{let}\ t_1\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ [x \otimes y/z]t_2 \rightsquigarrow [t_1/x]t_2}\ \text{R\_BETAT2}$$

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

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

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

$$\frac{t_1 \rightsquigarrow t_2}{\mathsf{let}\ x\ \mathsf{be}\ p\ \mathsf{in}\ t_1 \rightsquigarrow \mathsf{let}\ x\ \mathsf{be}\ p\ \mathsf{in}\ t_2}\ \text{R\_LET} \qquad \frac{t_1 \rightsquigarrow t_1'}{t_1 \otimes t_2 \rightsquigarrow t_1' \otimes t_2}\ \text{R\_T1}$$

$$\frac{t_2 \rightsquigarrow t_2'}{t_1 \otimes t_2 \rightsquigarrow t_1 \otimes t_2'}\ \text{R\_T2} \qquad \frac{t_1 \rightsquigarrow t_1'}{\mathsf{app}_l\ t_1\ t_2 \rightsquigarrow \mathsf{app}_l\ t_1'\ t_2}\ \text{R\_AL1}$$

$$\frac{t_2 \rightsquigarrow t_2'}{\mathsf{app}_l\ t_1\ t_2 \rightsquigarrow \mathsf{app}_l\ t_1\ t_2'}\ \text{R\_AL2} \qquad \frac{t_1 \rightsquigarrow t_1'}{\mathsf{app}_r\ t_1\ t_2 \rightsquigarrow \mathsf{app}_r\ t_1'\ t_2}\ \text{R\_AR1}$$

$$\frac{t_2 \rightsquigarrow t_2'}{\mathsf{app}_r\ t_1\ t_2 \rightsquigarrow \mathsf{app}_r\ t_1\ t_2'}\ \text{R\_AR2} \qquad \frac{t \rightsquigarrow t'}{\lambda_l x : A.t \rightsquigarrow \lambda_l x : A.t'}\ \text{R\_LL}$$

$$\frac{t \rightsquigarrow t'}{\lambda_r x : A.t \rightsquigarrow \lambda_r x : A.t'}\ \text{R\_LR}$$

Figure 3: Rewriting Rules for The Typed Lambek Calculus: λL

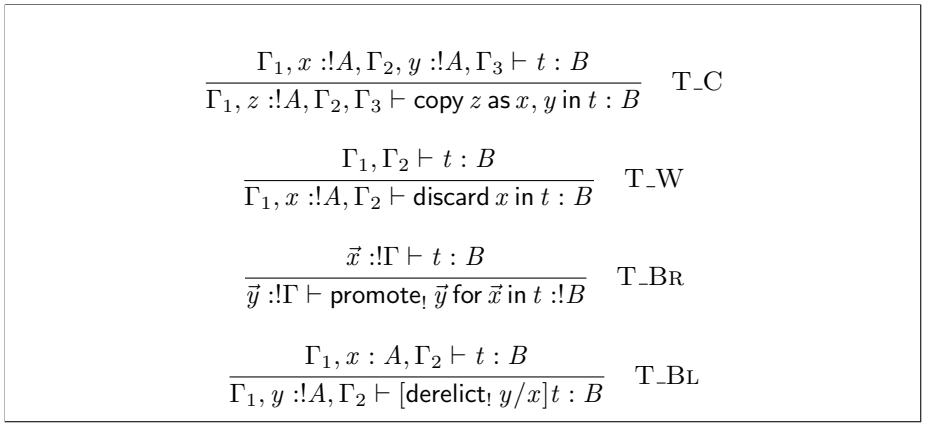

Figure 4: Typing Rules for The Typed Lambda Calculus: $\lambda L_!$

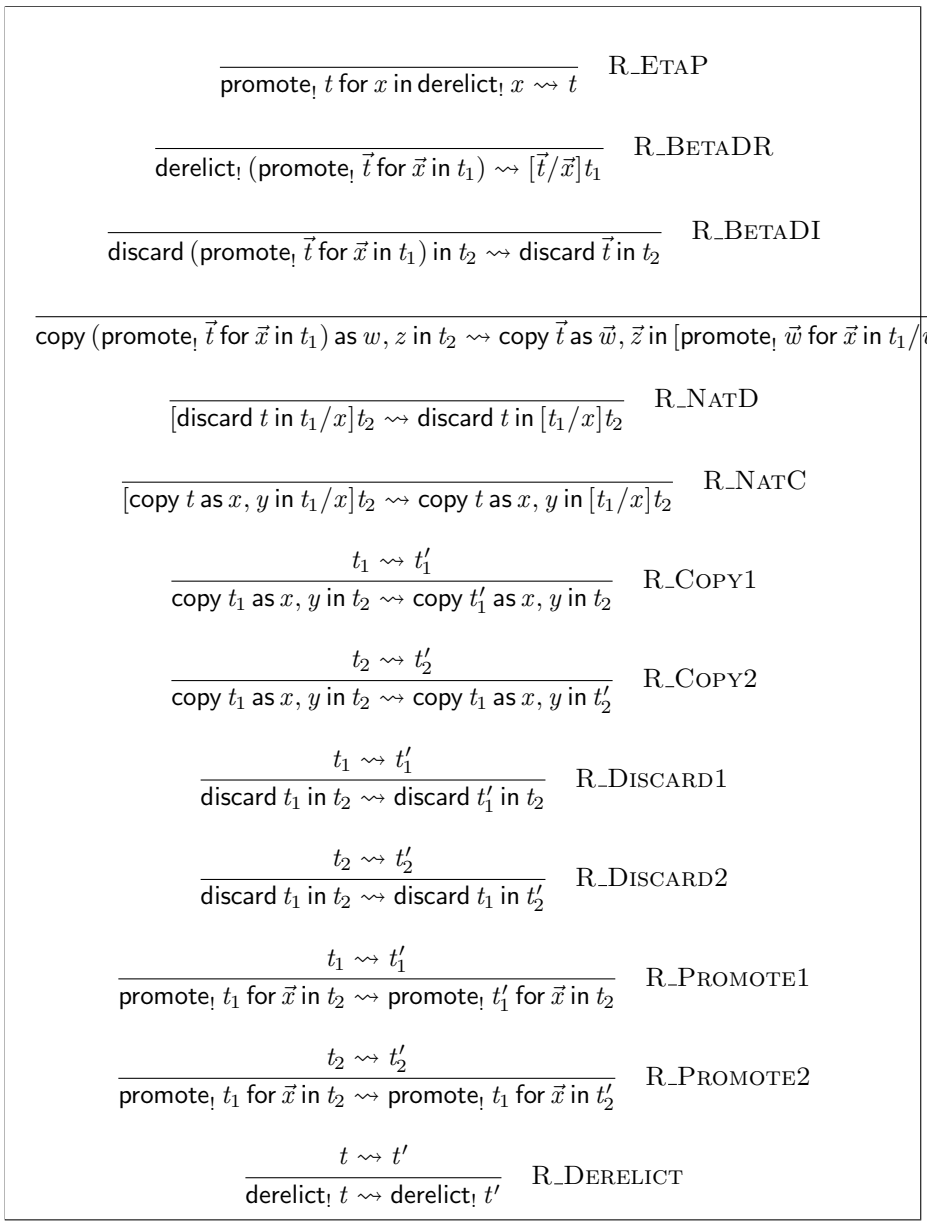

Figure 5: Rewriting Rules for The Typed Lambek Calculus: $\lambda \mathrm{L}_!$

$$\frac{\Gamma_1, x : \kappa A, y : B, \Gamma_2 \vdash t : C}{\Gamma_1, y : B, x : \kappa A, \Gamma_2 \vdash \mathsf{exchange_l}\ x\ \mathsf{with}\ y\ \mathsf{in}\ t : C}\ \text{T\_E1}$$

$$\frac{\Gamma_1, x : A, y : \kappa B, \Gamma_2 \vdash t : C}{\Gamma_1, y : \kappa B, x : A, \Gamma_2 \vdash \mathsf{exchange_r}\ x\ \mathsf{with}\ y\ \mathsf{in}\ t : C}\ \text{T\_E2}$$

$$\frac{\vec{x} : \kappa\Gamma \vdash t : B}{\vec{y} : \kappa\Gamma \vdash \mathsf{promote}_\kappa\ \vec{y}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ t : \kappa B}\ \text{T\_ER}$$

$$\frac{\Gamma_1, x : A, \Gamma_2 \vdash t : B}{\Gamma_1, y : \kappa A, \Gamma_2 \vdash [\mathsf{derelict}_\kappa\ y/x]t : B}\ \text{T\_EL}$$

Figure 6: Typing Rules for The Typed Lambda Calculus: $\lambda \mathrm{L}_\kappa$

$$\frac{}{\mathsf{promote}_\kappa\ t\ \mathsf{for}\ x\ \mathsf{in}\ \mathsf{derelict}_\kappa\ x \rightsquigarrow t}\ \text{R\_EtaEP}$$

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

$$\frac{}{[\mathsf{exchange}_\mathsf{l}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3/z]t_4 \rightsquigarrow \mathsf{exchange}_\mathsf{l}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ [t_3/x]t_4}\ \text{R\_NatEl}$$

$$\frac{}{[\mathsf{exchange}_\mathsf{r}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3/z]t_4 \rightsquigarrow \mathsf{exchange}_\mathsf{r}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ [t_3/x]t_4}\ \text{R\_NatEr}$$

$$\frac{t_1 \rightsquigarrow t_1'}{\mathsf{exchange}_\mathsf{l}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3 \rightsquigarrow \mathsf{exchange}_\mathsf{l}\ t_1'\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3}\ \text{R\_LEx1}$$

$$\frac{t_2 \rightsquigarrow t_2'}{\mathsf{exchange}_\mathsf{l}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3 \rightsquigarrow \mathsf{exchange}_\mathsf{l}\ t_1\ \mathsf{with}\ t_2'\ \mathsf{in}\ t_3}\ \text{R\_LEx2}$$

$$\frac{t_3 \rightsquigarrow t_3'}{\mathsf{exchange}_\mathsf{l}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3 \rightsquigarrow \mathsf{exchange}_\mathsf{l}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3'}\ \text{R\_LEx3}$$

$$\frac{t_1 \rightsquigarrow t_1'}{\mathsf{exchange}_\mathsf{r}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3 \rightsquigarrow \mathsf{exchange}_\mathsf{r}\ t_1'\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3}\ \text{R\_REx1}$$

$$\frac{t_2 \rightsquigarrow t_2'}{\mathsf{exchange}_\mathsf{r}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3 \rightsquigarrow \mathsf{exchange}_\mathsf{r}\ t_1\ \mathsf{with}\ t_2'\ \mathsf{in}\ t_3}\ \text{R\_REx2}$$

$$\frac{t_3 \rightsquigarrow t_3'}{\mathsf{exchange}_\mathsf{r}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3 \rightsquigarrow \mathsf{exchange}_\mathsf{r}\ t_1\ \mathsf{with}\ t_2\ \mathsf{in}\ t_3'}\ \text{R\_REx3}$$

$$\frac{t_1 \rightsquigarrow t_1'}{\mathsf{promote}_\kappa\ t_1\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ t_2 \rightsquigarrow \mathsf{promote}_\kappa\ t_1'\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ t_2}\ \text{R\_EPromote1}$$

$$\frac{t_2 \rightsquigarrow t_2'}{\mathsf{promote}_\kappa\ t_1\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ t_2 \rightsquigarrow \mathsf{promote}_\kappa\ t_1\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ t_2'}\ \text{R\_EPromote2}$$

$$\frac{t \rightsquigarrow t'}{\mathsf{derelict}_\kappa\ t \rightsquigarrow \mathsf{derelict}_\kappa\ t'}\ \text{R\_EDerelict}$$

Figure 7: Rewriting Rules for The Typed Lambek Calculus: $\lambda \mathrm{L}_\kappa$